МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»

Кафедра інженерії програмного забезпечення

КУРСОВА РОБОТА

(ПОЯСНЮВАЛЬНА ЗАПИСКА)

з дисципліни: «Бази даних:

«База даних інформаційно-пошукової системи: магазин, товари та послуги»

студента II курсу групи ІПЗ-23-5

спеціальності 121 «Інженерія програмного забезпечення»

Шевчук Дмитрій Романович

(прізвище, ім’я та по-батькові)

Керівник Старший викладач кафедри ІПЗ КРАВЧЕНКО Світлана,

Дата захисту: " \_\_\_ " \_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_ р.

Національна шкала \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кількість балів: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Оцінка: ECTS \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Члени комісії \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ І.І. Сугоняк

(підпис) (прізвище та ініціали)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.О. Данильченко

(підпис) (прізвище та ініціали)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В.Н. Ковальчук

(підпис) (прізвище та ініціали)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис) (прізвище та ініціали)

Житомир – 2024

ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»

Факультет інформаційно-комп’ютерних технологій

Кафедра інженерії програмного забезпечення

Освітній рівень: бакалавр

Спеціальність 121 «Інженерія програмного забезпечення»

«ЗАТВЕРДЖУЮ»

Завідувача кафедри ІПЗ

\_\_\_\_\_\_\_\_\_\_\_\_ Тетяна ВАКАЛЮК

«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_2024 р.

ЗАВДАННЯ

НА КУРСОВУ РОБОТУ СТУДЕНТУ

Шевчук Дмитрію Романовичу

Тема роботи: База даних інтернет-магазину книг

керівник роботи: ст. викладач кафедри ІПЗ КРАВЧЕНКО Світлана

1. Строк подання студентом: « 17 » грудня 2024 р.
2. Вихідні дані до роботи: Розробити базу даних інтернет-магазину книг
3. Зміст розрахунково-пояснювальної записки (перелік питань, які підлягають розробці)
   * + 1. Постановка завдання
       2. Аналіз аналогічних розробок
       3. Алгоритми роботи програми

4. Опис роботи програми

5. Програмне дослідження

1. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)

1. Презентація до КР\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

2. Посилання на репозиторій:

<https://git.ztu.edu.ua/ipz/2023-2027/ipz-23-5/shevchuk-dmitriy/2st-year/coursework-2>

1. Консультанти розділів роботи

|  |  |  |  |
| --- | --- | --- | --- |
| Розділ | Прізвище, ініціали та посади консультанта | Підпис, дата | |
| завдання  видав | завдання прийняв |
| 1-3 | КРАВЧЕНКО Світлана | 17.10.2024 | 17.10.2024 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

1. Дата видачі завдання “ 17 ” жовтня 2024 р.

|  |  |  |  |
| --- | --- | --- | --- |
| № з/п | Назва етапів курсового проєктування | Строк виконання етапів роботи | Примітки |
| 1 | Постановка задачі | 07.02.2024 | Виконано |
| 2 | Пошук, огляд та аналіз аналогічних розробок | 10.03.2024 | Виконано |
| 3 | Формулювання технічного завдання | 16.03.2024 | Виконано |
| 4 | Опрацювання літературних джерел | 20.03.2024 | Виконано |
| 5 | Проектування структури | 26.03.2024 | Виконано |
| 6 | Написання програмного коду | 03.05.2024 | Виконано |
| 7 | Відлагодження | 17.05.2024 | Виконано |
| 8 | Написання пояснювальної записки | 22.05.2024 | Виконано |
| 9 | Захист |  |  |

КАЛЕНДАРНИЙ ПЛАН

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Дмитрій ШЕВЧУК

(підпис) (прізвище та ініціали)

Керівник роботи \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Світлана КРАВЧЕНКО

(підпис) (прізвище та ініціали)

РЕФЕРАТ

Курсова робота на тему “База даних інтернет-магазину книг” присвячена розробленню інформаційно-пошукової системи для управління даними інтернет-магазину книг із застосуванням об'єктно-орієнтованого програмування.

Пояснювальна записка складається з трьох розділів, висновків, списку літератури та додатків. Загальний обсяг роботи – 58 сторінок.

У першому розділі обґрунтовано необхідність створення бази даних для інтернет-магазину, проаналізовано існуючі рішення, такі як Amazon та Book Depository, і визначено функціонал для майбутньої системи.

У другому розділі описано проектування програми, зокрема діаграми класів та алгоритми основних компонентів. Розглянуто реалізацію ключових класів, що відповідають за книги, авторів, замовлення та клієнтів.

У третьому розділі подано інструкції щодо роботи з програмою для користувачів та адміністраторів. Також проведено тестування програми.

Пояснювальна записка містить 26 сторінок основного тексту, 32 сторінки додатків, 15 рисунків та список із 8 літературних джерел.

Ключові слова: БАЗА ДАНИХ, ІНТЕРНЕТ-МАГАЗИН КНИГ, ОБ'ЄКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ, SQL, КЕРУВАННЯ ЗАМОВЛЕННЯМИ.

ЗМІСТ

[ВСТУП 9](#_Toc184853950)

[РОЗДІЛ 1. ТЕОРЕТИЧНИЙ АНАЛІЗ ІНФОРМАЦІЙНО-ПОШУКОВОЇ СИСТЕМИ: МАГАЗИН, ТОВАРИ ТА ПОСЛУГИ 13](#_Toc184853951)

[1,1. Аналіз інформаційних потреб та визначення предметної області дослідження 13](#_Toc184853952)

[1,3. Аналіз особливостей процесу формування інформаційних потреб системи 13](#_Toc184853953)

[1,4. Архітектура та методологія реалізації бази даних 14](#_Toc184853954)

[1.5. Інформаційні процеси системи 15](#_Toc184853955)

[1.6. Математичне моделювання та алгоритми обробки даних 16](#_Toc184853956)

[Висновки до розділу 1 16](#_Toc184853957)

[РОЗДІЛ 2 ПРОЕКТУВАННЯ БАЗИ ДАНИХ ІНФОРМАЦІЙНО-ПОШУКОВОЇ СИСТЕМИ МАГАЗИНУ 20](#_Toc184853958)

[2.1 Аналіз інформаційних процесів роботи магазину 20](#_Toc184853959)

[2.2 Аналіз структури та взаємозв’язків даних 22](#_Toc184853975)

[2.3 Проектування бази даних магазину 23](#_Toc184853976)

[2,4 Розробка алгоритмів обробки даних 26](#_Toc184853980)

[Висновки до другого розділу 29](#_Toc184853981)

[РОЗДІЛ 3 РЕАЛІЗАЦІЯ ПІДСИСТЕМИ ОБРОБКИ ДАНИХ ІНФОРМАЦІЙНО-ПОШУКОВА СИСТЕМА: МАГАЗИН, ТОВАРИ ТА ПОСЛУГИ 31](#_Toc184853982)

[3.1 Аналіз інформаційних потреб підсистеми обробки даних 31](#_Toc184853983)

[3.2 Проектування підсистеми обробки даних 32](#_Toc184853984)

[3,3. Алгоритми обробки інформації 33](#_Toc184853985)

[Висновки до третього розділу 37](#_Toc184853986)

[ВИСНОВКИ 38](#_Toc184853987)

[СПИСОК ЛІТЕРАТУРИ 40](#_Toc184853988)

[ДОДАТКИ 41](#_Toc184853989)

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

ІПС – Інформаційно-пошукова система

Рис. – Рисунок.

БД. – База даних

ВСТУП

База даних для інтернет-магазину книг є важливим інструментом для ефективного управління інформацією про книги, авторів, замовлення та клієнтів. У рамках даної курсової роботи розробляється сучасна база даних, яка дозволяє автоматизувати основні процеси електронної торгівлі та забезпечити зручну роботу як для адміністраторів, так і для користувачів магазину.

В умовах зростаючої популярності онлайн-торгівлі база даних є невід'ємною складовою успішного функціонування сучасного інтернет-магазину. Вона забезпечує швидке зберігання, обробку та пошук інформації про книги, управління замовленнями та клієнтськими даними, а також підтримку аналітичних функцій для прийняття обґрунтованих рішень.

Метою даної курсової роботи є проектування та розробка бази даних для інтернет-магазину книг, яка забезпечує:

* Ефективне зберігання та управління інформацією про книги та авторів.
* Облік замовлень і клієнтів.
* Зручний пошук і фільтрацію товарів.
* Масштабованість і можливість розширення функціоналу в майбутньому.

Основними завданнями роботи є:

* Аналіз предметної області та визначення інформаційних потреб системи.
* Проектування структури бази даних із дотриманням принципів нормалізації.
* Реалізація механізмів взаємодії з базою даних за допомогою SQL-запитів і тригерів.
* Розробка програмного інтерфейсу для роботи з базою даних з використанням об'єктно-орієнтованого програмування.

Практична значущість розробки полягає у створенні бази даних, яка може бути використана для автоматизації діяльності книжкових інтернет-магазинів, забезпечуючи швидкість, зручність і надійність роботи.

Результати дослідження також демонструють можливість застосування сучасних методів проектування баз даних для вирішення завдань електронної комерції.

РОЗДІЛ 1. ТЕОРЕТИЧНИЙ АНАЛІЗ ІНФОРМАЦІЙНО-ПОШУКОВОЇ СИСТЕМИ: МАГАЗИН, ТОВАРИ ТА ПОСЛУГИ

1. Аналіз інформаційних потреб та визначення предметної області дослідження
2. **Огляд сучасного стану баз даних у роздрібній торгівлі**  
   Сучасні інтернет-магазини, такі як Amazon, Barnes & Noble та Book Depository, активно використовують потужні бази даних для управління інформацією про книги, авторів, клієнтів та замовлення. Ці системи дозволяють забезпечити швидку обробку даних, зручний пошук товарів та аналіз продажів.
3. **Характеристика основних бізнес-процесів книжкового інтернет-магазину**  
   Основні бізнес-процеси включають облік товарів (книг), управління замовленнями, роботу з клієнтською базою та фінансовий облік.
4. **Ключові інформаційні потоки**:

* Облік книг (номенклатура, автори, кількість у наявності).
* Управління замовленнями (реєстрація, контроль виконання).
* Робота з клієнтами (реєстрація, історія замовлень).
* Фінансовий облік (формування підсумкових звітів).

1. Обґрунтування вибору засобів реалізації
2. **Порівняльний аналіз СУБД**  
   Для створення бази даних було розглянуто сучасні системи управління базами даних, зокрема MySQL, PostgreSQL та Microsoft SQL Server.
3. **Обґрунтування вибору Microsoft SQL Server**  
   Microsoft SQL Server обрано через наступні переваги:

* Надійність: забезпечення цілісності даних.
* Масштабованість: можливість роботи з великими обсягами інформації.
* Продуктивність: оптимізація виконання складних запитів.
* Підтримка збережених процедур, тригерів та функцій.

1. **Переваги T-SQL**  
   T-SQL забезпечує потужні можливості для створення та оптимізації запитів, обробки даних і виконання складних обчислень у базі.
2. Аналіз процесу формування інформаційних потреб системи
3. Основними функціональними завданнями бази даних є:

* Управління асортиментом книг із можливістю миттєвого оновлення даних.
* Комплексний облік замовлень і контроль їх виконання.
* Аналіз клієнтської бази для покращення обслуговування.
* Генерація звітів для підтримки управлінських рішень.

1. Архітектура та методологія реалізації бази даних
   1. **Принципи проектування**
2. Нормалізація даних для мінімізації надлишковості.
3. Забезпечення референтної цілісності.
4. Оптимізація продуктивності інформаційних запитів.
   1. **Етапи проектування**
5. Аналіз предметної області.
6. Визначення сутностей і атрибутів (книги, автори, замовлення, клієнти).
7. Проектування логічної моделі даних.
8. Фізична реалізація бази.
9. Інформаційні процеси систем
10. **Основні процеси:**

* Додавання та актуалізація даних про книги.
* Реєстрація замовлень і контроль їх виконання.
* Обробка та аналіз клієнтських даних.
* Формування аналітичних звітів.

1. **Механізми обробки:**

* Використання збережених процедур для виконання складних операцій.
* Реалізація тригерів для забезпечення цілісності даних.
* Багаторівнева індексація для оптимізації пошуку.и

1. Математичне моделювання та алгоритми обробки даних
2. **Алгоритми обробки даних:**

* Розрахунок загальної суми замовлення.
* Прогнозування обсягів продажів.
* Аналіз популярності книг.

1. **Алгоритми пошуку та фільтрації:**

* Пошук книг за ключовими словами.
* Сортування книг за ціною та популярністю.
* Фільтрація за автором, жанром або іншими атрибутами.

1. Висновки до розділу 1
2. За результатами теоретичного аналізу:

* Визначено інформаційні потреби інтернет-магазину книг.
* Розроблено концептуальну модель бази даних.
* Описано ключові інформаційні процеси та механізми обробки даних.
* Обґрунтовано вибір Microsoft SQL Server як основної платформи для реалізації системи.
* Запропоновано алгоритми для оптимізації пошуку та обробки даних.

РОЗДІЛ 2 ПРОЕКТУВАННЯ БАЗИ ДАНИХ ІНФОРМАЦІЙНО-ПОШУКОВОЇ СИСТЕМИ МАГАЗИНУ

1. Аналіз інформаційних процесів роботи магазину

* Загальна характеристика діяльності магазину
* Опис типу магазину
* Основні напрямки діяльності:
* Продаж товарів
* Облік замовлень
* Управління клієнтською базою
* Формування аналітичних звітів

1. Інформаційні потоки підприємства

* Діаграма інформаційних потоків магазину:



Рис. 2.1 Діаграма процесів клієнтів



Рис. 2.2 Діаграма процесів адміністратора

1. Вимоги до інформаційної системи
   * Забезпечення цілісності даних
   * Можливість масштабування
   * Швидкий пошук та фільтрація товарів
   * Аналітика продажів
   * Захист інформації
2. Аналіз структури та взаємозв’язків даних
3. Характеристика основних сутностей бази даних

**2.2. Аналіз структури та взаємозв’язків даних**

**2.2.1. Характеристика основних сутностей бази даних**

* **Authors (Автори):**
  + Унікальний ідентифікатор автора (author\_id).
  + Ім'я автора (name).
* **Books (Книги):**
  + Унікальний ідентифікатор книги (book\_id).
  + Назва книги (title).
  + Ідентифікатор автора (author\_id), пов'язаний із таблицею Authors.
  + Ціна книги (price).
  + Кількість книг у наявності (stock).
* **Customers (Клієнти):**
  + Унікальний ідентифікатор клієнта (customer\_id).
  + ПІБ клієнта (fullname).
  + Контактні дані клієнта (електронна пошта, email).
* **Orders (Замовлення):**
  + Унікальний номер замовлення (order\_id).
  + Ідентифікатор клієнта (customer\_id), пов'язаний із таблицею Customers.
  + Дата створення замовлення (order\_date).
* **OrderItems (Елементи замовлення):**
  + Номер замовлення (order\_id), пов'язаний із таблицею Orders.
  + Ідентифікатор книги (book\_id), пов'язаний із таблицею Books.
  + Кількість замовлених книг (quantity).

**Зв’язки між сутностями**

1. **Authors (Автори) -> Books (Книги):**  
   Один автор може написати кілька книг (зв’язок "один до багатьох").
2. **Books (Книги) -> OrderItems (Елементи замовлення):**  
   Книга може бути замовлена в кількох замовленнях (зв’язок "один до багатьох").
3. **Customers (Клієнти) -> Orders (Замовлення):**  
   Один клієнт може оформити кілька замовлень (зв’язок "один до багатьох").
4. **Orders (Замовлення) -> OrderItems (Елементи замовлення):**  
   Одне замовлення може містити кілька книг (зв’язок "один до багатьох").

**Опис взаємозв’язків**

* Таблиця Books має зовнішній ключ author\_id, що посилається на Authors.
* Таблиця OrderItems містить зовнішні ключі order\_id і book\_id, що посилаються відповідно на Orders і Books.
* Таблиця Orders має зовнішній ключ customer\_id, що посилається на Customers.

1. Структура таблиць

A screen shot of a computer

Description automatically generated

Рис. 2.1 Таблиця для авторів

A screen shot of a computer code

Description automatically generated

Рис. 2.2 Таблиця для книг

A black screen with orange and blue text

Description automatically generated

Рис. 2.3 Таблиця для клієнтів

A screen shot of a computer code

Description automatically generated

Рис. 2.4 Таблиця для замовлень

A computer screen shot of text

Description automatically generated

Рис. 2.5 Таблиця для елементів замовлення

1. Типи даних та обмеження

* Вибір оптимальних типів
* Первинні та зовнішні ключі
* Обмеження на введення даних

1. Функції аналізу даних

A computer screen with text and numbers

Description automatically generated

Рис. 2.6 Функція для обрахунку загальної суми замовлення

1. Тригери

A screen shot of a computer program

Description automatically generated

Рис 2.7 Тригер на вставку в OrderItems, що оновлює склад

Висновки до другого розділу

**Висновки до другого розділу**

У другому розділі спроектовано базу даних інтернет-магазину книг, включаючи аналіз інформаційних процесів, структури даних і взаємозв’язків між ними.

1. Визначено ключові сутності бази даних (автори, книги, клієнти, замовлення, елементи замовлення) та зв’язки між ними.
2. Розроблено структуру таблиць із врахуванням референтної цілісності, використання зовнішніх ключів та оптимальних типів даних.
3. Описано функції для аналізу даних і тригери для підтримки автоматизації основних бізнес-процесів.

РОЗДІЛ 3 РЕАЛІЗАЦІЯ ПІДСИСТЕМИ ОБРОБКИ ДАНИХ ІНФОРМАЦІЙНО-ПОШУКОВА СИСТЕМА: МАГАЗИН, ТОВАРИ ТА ПОСЛУГИ

1. Аналіз інформаційних потреб підсистеми обробки даних
2. Характеристика інформаційних процесів

У сучасних умовах розвит роздрібної торгівлі та електронної комерції виникає гостра необхідність у створенні ефективних інформаційних систем управління даними. Розроблена інформаційно-пошукова система призначена забезпечити комплексне управління процесами обліку товарів, клієнтів, замовлень та транзакцій.

Основні інформаційні процеси системи включають:

* Облік та каталогізацію товарів
* Реєстрацію та моніторинг замовлень
* Управління клієнтською базою
* Фінансовий облік та аналітику продажів
* Формування звітності

1. Вимоги до підсистеми обробки даниз

До підсистеми висуваються наступні ключові вимоги:

* + - 1. Функціональність:
* Повний спектр операцій з обліку товарів
* Підтримка складних запитів та аналітичних операцій
* Забезпечення цілісності та несуперечності даних
  + - 1. Продуктивність
* Швидкодія обробки запитів
* Ефективне використання індексації
* Мінімізаця навантаження на систему
  + - 1. Безпека
* Захист від некоректних операцій
* Підтримка транзакційної цілісності
* Логування помилок та критичних подій

1. Функціональне призначення підсистеми

Підсистема реалізує наступні ключові функції:

* Введення та оновлення інформації про товари
* Реєстрація замовлень
* Формування аналітичних звітів
* Управління клієнтською базою
* Моніторинг фінансових транзакцій

1. Проектування підсистеми обробки даних
2. Методологія проектування

При проектування бази даних використано реляційну модель з дотримання принципів нормалізації.

Система базується на Microsoft SQL Server, що забезпечує надійність та масштабованість.

Основні принципи проектування:

* Нормалізація до 3-ї нормальної форми
* Використання зовнішніших ключів для підтримки цілісності
* Створення індексів для оптимізації продуктивності

1. Архітектурні рішення

Архітектура бази даних включає наступні ключові компоненти:

1. Таблиці даних:

* Products: Каталог товарів
* Orders: Облік замовлень
* Clients: Інформація про кдієнтів
* Transactions: Фінансові транзакції
* Shops: Інформація про магазинів
* Users: Система користувачів
* Reviews: Огляди та відгуки

1. Додаткові об’єкти бази даних:

* Індекси для прискорення пошуку
* Зовнішні ключі для підтримки цілісності
* Тригери для автоматизації процесів
* Збережені процедури для складних операцій

1. Інтеграційні механізми

Інтеграційні механізми забезпечують:

* Автоматичне оновлення статусів замовлень
* Каскадні оновлення пов’язаних записів
* Валідація даних на рівні бази даних
* Підтримку транзакційної цілісності

1. Алгоритми обробки інформації
   * + 1. Методи пошуку та фільтрації

Реалізовано функцію fn\_SearchProducts (Рис. 3.1) для гнучкого пошуку товарів з підтримкою:

* Фільтрації за категорією
* Обмеження діапазону цін
* Пошуку за назвою магазину

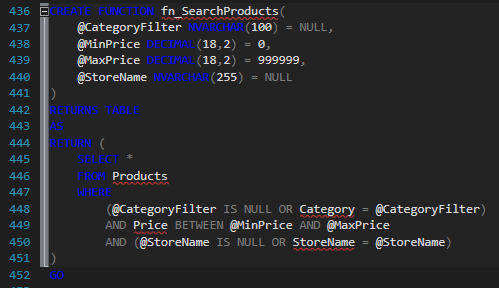
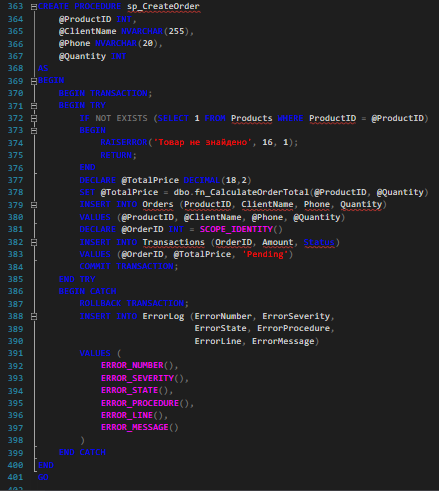


Рис. 3.1 Функція fn\_SearchProducts

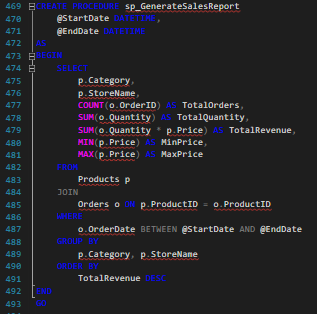
* + - 1. Алгоритми оброблення запитів

Розроблено спеціалізовані алгоритми:

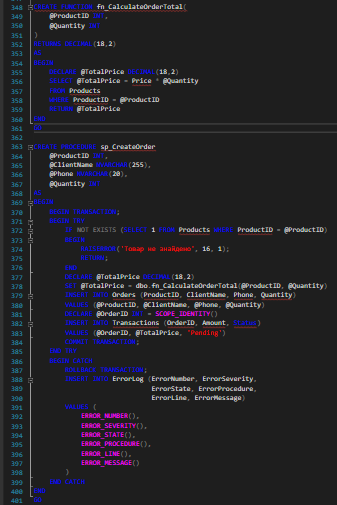
* sp\_CreateOrder (Рис. 3.2): Створення замовлення з перевіркою наявності товару



* Рис. 3.2 Функція sp\_CreateOrder
* sp\_GenerateSalesReport (Рис. 3.3): Формування звіту про продажі



* Рис. 3.3 Функція sp\_GenerateSalesReport
* fn\_CalculateOrderTotal (Рис. 3.4): Розрахунок вартості замовлення



* Рис. 3.4 Функція fn\_CalculateOrderTotal
  + - 1. Оптимізація інформаційного пошуку

Оптимізація реалізована через:

* Створення складних індексів (Рис. 3.5):
* idx\_Product\_Category
* idx\_Orders\_ProductID
* idx\_Product\_PriceCategory
* Підтримка ефективних запитів
* Мінімізація навантаження на систем

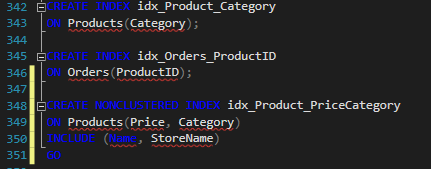


Рис. 3.5 Складні індекси для оптимізації

1. Інформаційне забезпечення системи
2. Структура інформаційних масивів

* Структура даних включає:
* Нормалізовані таблиці
* Зв’язки типу “один-до-багатьох”
* Підтримка для різних типів даних

1. Методи накопичення та зберігання даних

* Реалізовано механізми:
* Автоматичне штамп-поле часу ( DEFAULT GETDATE() )
* Збереження історії транзакцій
* Логування помилок в ErrorLog

1. Управління інформаційними ресурсами

* Передбачено
* Контроль цілісності даних
* Механізми резервного копіювання
* Розмежування прав доступу

1. Особливості реалізації підсистеми
2. Технологічні аспекті розробки

* Технології:
* Microsoft SQL Server
* Реляційна модель бази даних
* Транзакційна система
* T-SQL

1. Програмні засоби реалізації

* Використані засоби:
* СУБД MS SQL Server
* Середовище розробки SQL Server Management Studio
* Мова T-SQL для маніпуляцій з даними

1. Тестування та налагодження підсистеми

* Тестування включало:
* Перевірку цілісності даних
* Тестування збережених процедур
* Навантажувальне тестування
* Перевірка тригерів та функцій

Висновки до третього розділу

У розділі представлно комплексне проектування інформаційно-пошукової системи з урахуванням сучасних вимог до баз даних. Розроблена система забезпечує ефективне управління даними, високу продуктивність та надійність обробки інформації.

* Ключові переваги:
* Гнучка архітектура
* Висока продуктивність
* Підтримка складних аналітичних операцій
* Надійність та масштабованість

ВИСНОВКИ

У межах даної курсової роботи розроблено комплексну інформаційно-пошукову систему (ІПС) для управління електронним магазином на базі Microsoft SQL Server з використанням реляційної моделі даних та мови T-SQL.

Основні результати дослідження:

1. Проведено ґрунтовний аналіз предметної області електронної комерції та визначено ключові вимоги до інформаційної системи магазину.
2. Розроблено концептуальну модель бази даних, що включає взаємопов'язані таблиці:

* Products (каталог товарів)
* Orders (облік замовлень)
* Clients (інформація про клієнтів)
* Transactions (фінансові операції)
* Users (система користувачів)
* Reviews (огляди та відгуки)
* Реалізовано ефективні механізми обробки даних:
* Збережені процедури для складних операцій
* Тригери для автоматизації процесів
* Функції пошуку та фільтрації товарів
* Алгоритми генерації аналітичних звітів

1. Забезпечено високу продуктивність системи через:

* Нормалізацію даних до 3-ої нормальної форми
* Створення оптимізованих індексів
* Підтримку транзакційної цілісності
* Ефективні механізми пошуку та фільтрації

1. Розроблена база даних відповідає сучасним вимогам електронної комерції:

* Масштабованість
* Гнучкість архітектури
* Підтримка складних аналітичних операцій
* Надійність зберігання та обробки даних

Практичне значення роботи полягає в створенні універсального рішення для автоматизації діяльності торговельних підприємств різного масштабу - від невеликих інтернет-магазинів до потужних роздрібних мереж.

Перспективи подальших досліджень включають:

* Розширення функціоналу системи
* Впровадження машинного навчання для аналітики
* Інтеграцію з системами управління ресурсами підприємства
* Розробку додаткових модулів звітності та прогнозування

# СПИСОК ЛІТЕРАТУРИ

1. Дейт К. Дж. Введення в системи баз даних / Пер. з англ. – К.: Вільямс, 2019. – 1312 с.
2. Озкарагоз М. Проектування реляційних баз даних / Пер. з англ. – М.: Діалектика, 2018. – 512 с.
3. Фаулер М. Архітектура корпоративних додатків / Пер. з англ. – М.: Вільямс, 2019. – 544 с.
4. Мартін Р. Чистий код: створення, аналіз і рефакторинг / Пер. з англ. – СПб.: Пітер, 2020. – 464 с.
5. Флоренс Н. Мистецтво проектування баз даних / Пер. з англ. – К.: Діалектика, 2019. – 672 с.
6. Вігерс К., Біттер Д. Розробка вимог до програмного забезпечення / Пер. з англ. – М.: Російська редакція, 2019. – 736 с.
7. Саймон А. SQL. Повний посібник / Пер. з англ. – М.: Вільямс, 2020. – 848 с.
8. Кімберлінг М. UML. Уніфікована мова моделювання / Пер. з англ. – СПб.: Пітер, 2018. – 288 с.
9. Шварц Б. Високоефективні бази даних / Пер. з англ. – М.: Діалектика, 2018. – 960 с.
10. Фаулер М. NoSQL: нова методологія розробки нереляційних баз даних / Пер. з англ. – М.: Вільямс, 2019. – 192 с.
11. Вальверде Р. Проектування корпоративних інформаційних систем / Пер. з англ. – К.: Діалектика, 2019. – 368 с.
12. Страчан А. Хмарні обчислення та бази даних / Пер. з англ. – М.: ДМК Прес, 2021. – 256 с.
13. Ольсон Д. Сховища даних: теорія та практика / Пер. з англ. – СПб.: Пітер, 2018. – 512 с.

ДОДАТКИ

Додаток А.

Код форм

А.1. Форма адміністратора (Admin\_menu)

using System;

using System.Collections.Generic;

using System.Data;

using System.Windows.Forms;

using DatabaseDevelopers;

using System.Data.SqlClient;

namespace ItemSearchPro

{

public partial class Admin\_menu : Form

{

private string connectionString = "Data Source=localhost;Initial Catalog=InformationSearchSystem;Integrated Security=True";

public Admin\_menu()

{

InitializeComponent();

InitializeMenu();

this.WindowState = FormWindowState.Maximized;

}

private void Authorisation\_for\_developers\_Load(object sender, EventArgs e)

{

listView\_Show.View = View.Details;

listView\_Show.GridLines = true;

listView\_Show.FullRowSelect = true;

LoadOrdersFromDatabase();

listView\_Show.Items.Clear();

}

private void Button\_Show\_all\_products\_Click(object sender, EventArgs e)

{

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = "SELECT Category, Name, Price, StoreName, Description, Address FROM Products";

SqlCommand command = new SqlCommand(query, connection);

SqlDataReader reader = command.ExecuteReader();

List<string[]> dataRows = new List<string[]>();

while (reader.Read())

{

dataRows.Add(new string[]

{

reader["Name"].ToString(),

reader["Category"].ToString(),

reader["Price"].ToString(),

reader["StoreName"].ToString(),

reader["Description"].ToString(),

reader["Address"].ToString()

});

}

if (dataRows.Count > 0)

{

listView\_Show.Columns.Clear();

listView\_Show.Columns.Add("Категорія", 150);

listView\_Show.Columns.Add("Назва", 200);

listView\_Show.Columns.Add("Ціна", 100);

listView\_Show.Columns.Add("Магазин", 200);

listView\_Show.Columns.Add("Опис", 300);

listView\_Show.Columns.Add("Адреса", 350);

UpdateSearchResults(dataRows);

}

else

{

MessageBox.Show("Товарів не знайдено", "Повідомлення");

listView\_Show.Items.Clear();

}

}

}

catch (Exception ex)

{

MessageBox.Show($"Помилка пошуку в базі даних: {ex.Message}", "Помилка");

}

}

private void UpdateSearchResults(List<string[]> dataRows)

{

listView\_Show.Items.Clear();

foreach (string[] rowData in dataRows)

{

ListViewItem item = new ListViewItem(new string[]

{

rowData[1],

rowData[0],

rowData[2] + " ₴",

rowData[3],

rowData[4],

rowData[5]

});

listView\_Show.Items.Add(item);

}

}

private void Button\_Delete\_the\_product\_Click(object sender, EventArgs e)

{

if (listView\_Show.SelectedItems.Count > 0)

{

ListViewItem selectedItem = listView\_Show.SelectedItems[0];

string selectedProductName = selectedItem.SubItems[1].Text;

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string deleteQuery = "DELETE FROM Products WHERE Name = @ProductName";

SqlCommand command = new SqlCommand(deleteQuery, connection);

command.Parameters.AddWithValue("@ProductName", selectedProductName);

int rowsAffected = command.ExecuteNonQuery();

if (rowsAffected > 0)

{

MessageBox.Show("Товар успішно видалений з бази даних.", "Успіх");

listView\_Show.Items.Remove(selectedItem);

}

else

{

MessageBox.Show("Не вдалося знайти товар для видалення.", "Помилка");

}

}

}

catch (Exception ex)

{

MessageBox.Show($"Помилка видалення товару з бази даних: {ex.Message}", "Помилка");

}

}

else

{

MessageBox.Show("Виберіть товар для видалення.", "Попередження");

}

}

private void Button\_Add\_product\_Click(object sender, EventArgs e)

{

string category = TextBox\_Category.Text.Trim();

string name = TextBox\_Name.Text.Trim();

string priceText = TextBox\_Price.Text.Replace("₴", "").Trim();

decimal price;

if (!decimal.TryParse(priceText, out price))

{

MessageBox.Show("Некоректна ціна.", "Помилка");

return;

}

string storeName = TextBox\_Store\_Name.Text.Trim();

string description = TextBox\_Description.Text.Trim();

string address = TextBox\_Address.Text.Trim();

if (string.IsNullOrEmpty(category) || string.IsNullOrEmpty(name) || string.IsNullOrEmpty(storeName))

{

MessageBox.Show("Заповніть обов'язкові поля: Категорія, Назва, Назва магазину.", "Попередження");

return;

}

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string insertQuery = @"

INSERT INTO Products (Category, Name, Price, StoreName, Description, Address)

VALUES (@Category, @Name, @Price, @StoreName, @Description, @Address)";

SqlCommand command = new SqlCommand(insertQuery, connection);

command.Parameters.AddWithValue("@Category", category);

command.Parameters.AddWithValue("@Name", name);

command.Parameters.AddWithValue("@Price", price);

command.Parameters.AddWithValue("@StoreName", storeName);

command.Parameters.AddWithValue("@Description", description);

command.Parameters.AddWithValue("@Address", address);

command.ExecuteNonQuery();

MessageBox.Show("Елемент успішно додано до бази даних.", "Успіх");

TextBox\_Category.Clear();

TextBox\_Name.Clear();

TextBox\_Price.Clear();

TextBox\_Store\_Name.Clear();

TextBox\_Description.Clear();

TextBox\_Address.Clear();

}

}

catch (Exception ex)

{

MessageBox.Show($"Помилка додавання товару в базу даних: {ex.Message}", "Помилка");

}

}

private void InitializeMenu()

{

MenuStrip menuStrip = new MenuStrip();

this.Controls.Add(menuStrip);

ToolStripMenuItem ordersMenu = new ToolStripMenuItem("Замовлення");

ToolStripMenuItem showOrdersMenuItem = new ToolStripMenuItem("Показати замовлення");

showOrdersMenuItem.Click += ShowOrdersMenuItem\_Click;

ordersMenu.DropDownItems.Add(showOrdersMenuItem);

ToolStripMenuItem acceptOrderMenuItem = new ToolStripMenuItem("Прийняти замовлення");

acceptOrderMenuItem.Click += AcceptOrderMenuItem\_Click;

ordersMenu.DropDownItems.Add(acceptOrderMenuItem);

ToolStripMenuItem ShowAcceptedOrdersMenuItem = new ToolStripMenuItem("Всі прийняті замовлення");

ShowAcceptedOrdersMenuItem.Click += ShowAcceptedOrdersMenuItem\_Click;

ordersMenu.DropDownItems.Add(ShowAcceptedOrdersMenuItem);

ToolStripMenuItem cancelOrderMenuItem = new ToolStripMenuItem("Скасувати замовлення");

cancelOrderMenuItem.Click += CancelOrderMenuItem\_Click;

ordersMenu.DropDownItems.Add(cancelOrderMenuItem);

menuStrip.Items.Add(ordersMenu);

ToolStripMenuItem developerMenu = new ToolStripMenuItem("Швидкий перехід на інші форми");

menuStrip.Items.Add(developerMenu);

ToolStripMenuItem welcomeFormMenuItem = new ToolStripMenuItem("Форма ласково просимо");

welcomeFormMenuItem.Click += WelcomeFormMenuItem\_Click;

developerMenu.DropDownItems.Add(welcomeFormMenuItem);

ToolStripMenuItem registrationFormMenuItem = new ToolStripMenuItem("Форма реєстрації");

registrationFormMenuItem.Click += RegistrationFormMenuItem\_Click;

developerMenu.DropDownItems.Add(registrationFormMenuItem);

ToolStripMenuItem authorizationFormMenuItem = new ToolStripMenuItem("Форма авторизації");

authorizationFormMenuItem.Click += AuthorizationFormMenuItem\_Click;

developerMenu.DropDownItems.Add(authorizationFormMenuItem);

ToolStripMenuItem information\_and\_search\_system\_for\_guestsFormMenuItem = new ToolStripMenuItem("Пошукова форма для гостя");

information\_and\_search\_system\_for\_guestsFormMenuItem.Click += Information\_and\_search\_system\_for\_guestsFormMenuItem\_Click;

developerMenu.DropDownItems.Add(information\_and\_search\_system\_for\_guestsFormMenuItem);

ToolStripMenuItem information\_and\_search\_systemFormMenuItem = new ToolStripMenuItem("Пошукова форма для зарегістрированого користувача");

information\_and\_search\_systemFormMenuItem.Click += Information\_and\_search\_systemFormMenuItem\_Click;

developerMenu.DropDownItems.Add(information\_and\_search\_systemFormMenuItem);

ToolStripMenuItem password\_recoveryFormMenuItem = new ToolStripMenuItem("Форма для відновлення пароля");

password\_recoveryFormMenuItem.Click += Password\_recoveryFormMenuItem\_Click;

developerMenu.DropDownItems.Add(password\_recoveryFormMenuItem);

this.MainMenuStrip = menuStrip;

Button buttonAccept = new Button();

buttonAccept.Text = "Прийняти замовлення";

buttonAccept.Click += Button\_Accept\_Click;

buttonAccept.Location = new System.Drawing.Point(20, 250);

Button buttonCancel = new Button();

buttonCancel.Text = "Скасувати замовлення";

buttonCancel.Click += Button\_Cancel\_Click;

buttonCancel.Location = new System.Drawing.Point(150, 250);

this.Controls.Add(buttonAccept);

this.Controls.Add(buttonCancel);

ToolStripMenuItem analystMenu = new ToolStripMenuItem("Аналітика");

menuStrip.Items.Add(analystMenu);

ToolStripMenuItem salesAnalyticsMenuItem = new ToolStripMenuItem("Аналітика Продажів");

salesAnalyticsMenuItem.Click += ShowSalesAnalyticsMenuItem\_Click;

analystMenu.DropDownItems.Add(salesAnalyticsMenuItem);

ToolStripMenuItem reportsMenu = new ToolStripMenuItem("Звіти");

ToolStripMenuItem salesReportItem = new ToolStripMenuItem("Звіт про Продажі");

salesReportItem.Click += (s, e) => ExecuteAndDisplayReport("SalesReport");

reportsMenu.DropDownItems.Add(salesReportItem);

ToolStripMenuItem clientReportItem = new ToolStripMenuItem("Звіт про Клієнтів");

clientReportItem.Click += (s, e) => ExecuteAndDisplayReport("ClientReport");

reportsMenu.DropDownItems.Add(clientReportItem);

ToolStripMenuItem discountReportItem = new ToolStripMenuItem("Звіт про Знижки");

discountReportItem.Click += (s, e) => ExecuteAndDisplayReport("DiscountReport");

reportsMenu.DropDownItems.Add(discountReportItem);

menuStrip.Items.Add(reportsMenu);

}

private void Information\_and\_search\_systemFormMenuItem\_Click(object sender, EventArgs e)

{

Information\_and\_search\_system information\_and\_search\_system = new Information\_and\_search\_system();

information\_and\_search\_system.Show();

}

private void Information\_and\_search\_system\_for\_guestsFormMenuItem\_Click(object sender, EventArgs e)

{

Information\_and\_search\_system\_for\_guests information\_And\_Search\_System\_For\_Guests = new Information\_and\_search\_system\_for\_guests();

information\_And\_Search\_System\_For\_Guests.Show();

}

private void Password\_recoveryFormMenuItem\_Click(object sender, EventArgs e)

{

Password\_recovery password\_recoveryForm = new Password\_recovery();

password\_recoveryForm.Show();

}

private void WelcomeFormMenuItem\_Click(object sender, EventArgs e)

{

Welcome\_to\_ItemSearchPro welcomForm = new Welcome\_to\_ItemSearchPro();

welcomForm.Show();

}

private void AuthorizationFormMenuItem\_Click(object sender, EventArgs e)

{

Authorization\_for\_ItemSearchPro authorizationForm = new Authorization\_for\_ItemSearchPro();

authorizationForm.Show();

}

private void RegistrationFormMenuItem\_Click(object sender, EventArgs e)

{

Registration registrationForm = new Registration();

registrationForm.Show();

}

private void AcceptOrderMenuItem\_Click(object sender, EventArgs e)

{

if (listView\_Show.SelectedItems.Count > 0)

{

ListViewItem selectedOrder = listView\_Show.SelectedItems[0];

if (selectedOrder != null && selectedOrder.SubItems.Count >= 4)

{

int productId = Convert.ToInt32(selectedOrder.SubItems[0].Text);

string clientName = selectedOrder.SubItems[2].Text;

string phone = selectedOrder.SubItems[3].Text;

int quantity = Convert.ToInt32(selectedOrder.SubItems[4].Text);

int orderId = Convert.ToInt32(selectedOrder.SubItems[1].Text);

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string deleteTransactionsQuery = "DELETE FROM Transactions WHERE OrderID = @OrderID";

using (SqlCommand deleteCommand = new SqlCommand(deleteTransactionsQuery, connection))

{

deleteCommand.Parameters.AddWithValue("@OrderID", orderId);

deleteCommand.ExecuteNonQuery();

}

string moveOrderQuery = @"

INSERT INTO AcceptedOrders (ProductID, ClientName, Phone, Quantity, AcceptDate)

VALUES (@ProductID, @ClientName, @Phone, @Quantity, GETDATE());

-- Step 3: Delete the order from Orders

DELETE FROM Orders WHERE OrderID = @OrderID;

";

using (SqlCommand moveCommand = new SqlCommand(moveOrderQuery, connection))

{

moveCommand.Parameters.AddWithValue("@ProductID", productId);

moveCommand.Parameters.AddWithValue("@ClientName", clientName);

moveCommand.Parameters.AddWithValue("@Phone", phone);

moveCommand.Parameters.AddWithValue("@Quantity", quantity);

moveCommand.Parameters.AddWithValue("@OrderID", orderId);

moveCommand.ExecuteNonQuery();

}

listView\_Show.Items.Remove(selectedOrder);

MessageBox.Show("Замовлення прийнято і перенесено до AcceptedOrders.", "Успішно", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

}

else

{

MessageBox.Show("Вибраний товар, а не саме замовлення!", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

else

{

MessageBox.Show("Будь ласка, виберіть замовлення для прийняття.", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void ShowAcceptedOrdersMenuItem\_Click(object sender, EventArgs e)

{

try

{

LoadAcceptedOrdersFromDatabase();

}

catch (Exception ex)

{

MessageBox.Show($"Помилка при завантаженні замовлень: {ex.Message}", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void LoadAcceptedOrdersFromDatabase()

{

listView\_Show.Items.Clear();

listView\_Show.Columns.Clear();

listView\_Show.Columns.Add("ПродуктID", 100);

listView\_Show.Columns.Add("Ім'я клієнта", 250);

listView\_Show.Columns.Add("Телефон", 150);

listView\_Show.Columns.Add("Кількість", 100);

listView\_Show.Columns.Add("Дата прийняття", 180);

try

{

using (SqlConnection conn = new SqlConnection(connectionString))

{

conn.Open();

string query = "SELECT ProductID, ClientName, Phone, Quantity, AcceptDate FROM AcceptedOrders";

SqlCommand cmd = new SqlCommand(query, conn);

SqlDataReader reader = cmd.ExecuteReader();

while (reader.Read())

{

int productId = Convert.ToInt32(reader["ProductID"]);

string clientName = reader["ClientName"].ToString();

string phone = reader["Phone"].ToString();

int quantity = Convert.ToInt32(reader["Quantity"]);

DateTime acceptDate = Convert.ToDateTime(reader["AcceptDate"]);

ListViewItem item = new ListViewItem(new string[]

{

productId.ToString(),

clientName,

phone,

quantity.ToString(),

acceptDate.ToString("yyyy-MM-dd HH:mm:ss")

});

listView\_Show.Items.Add(item);

}

reader.Close();

}

}

catch (Exception ex)

{

MessageBox.Show($"Помилка при підключенні до бази даних: {ex.Message}", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void CancelOrderMenuItem\_Click(object sender, EventArgs e)

{

if (listView\_Show.SelectedItems.Count > 0)

{

ListViewItem selectedItem = listView\_Show.SelectedItems[0];

string productId = selectedItem.SubItems[0].Text;

string clientName = selectedItem.SubItems[2].Text;

DialogResult result = MessageBox.Show("Ви дійсно бажаєте скасувати це замовлення?", "Підтвердження", MessageBoxButtons.YesNo, MessageBoxIcon.Question);

if (result == DialogResult.Yes)

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string deleteOrderQuery = @"

DELETE FROM Orders

WHERE ProductID = @ProductID AND ClientName = @ClientName; -- Використовуємо ProductID

";

using (SqlCommand command = new SqlCommand(deleteOrderQuery, connection))

{

command.Parameters.AddWithValue("@ProductID", productId);

command.Parameters.AddWithValue("@ClientName", clientName);

command.ExecuteNonQuery();

}

listView\_Show.Items.Remove(selectedItem);

MessageBox.Show("Замовлення було скасоване.", "Успішно", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

}

}

}

private void ShowOrdersMenuItem\_Click(object sender, EventArgs e)

{

LoadOrdersFromDatabase();

}

private void LoadOrdersFromDatabase()

{

listView\_Show.Items.Clear();

listView\_Show.Columns.Clear();

listView\_Show.Columns.Add("ПродуктID", 100);

listView\_Show.Columns.Add("ПорядокID", 100);

listView\_Show.Columns.Add("Ім'я клієнта", 250);

listView\_Show.Columns.Add("Телефон", 150);

listView\_Show.Columns.Add("Кількість", 100);

using (SqlConnection conn = new SqlConnection(connectionString))

{

conn.Open();

string query = "SELECT OrderID, ProductID, ClientName, Phone, Quantity FROM Orders";

SqlCommand cmd = new SqlCommand(query, conn);

SqlDataReader reader = cmd.ExecuteReader();

while (reader.Read())

{

int orderId = Convert.ToInt32(reader["OrderID"]);

int productId = Convert.ToInt32(reader["ProductID"]);

string clientName = reader["ClientName"].ToString();

string phone = reader["Phone"].ToString();

int quantity = Convert.ToInt32(reader["Quantity"]);

ListViewItem item = new ListViewItem(new string[] { productId.ToString(), orderId.ToString(), clientName, phone, quantity.ToString() });

listView\_Show.Items.Add(item);

}

reader.Close();

}

}

private void Button\_Accept\_Click(object sender, EventArgs e)

{

MoveSelectedOrderToAccepted();

}

private void MoveSelectedOrderToAccepted()

{

if (listView\_Show.SelectedItems.Count > 0)

{

ListViewItem selectedOrder = listView\_Show.SelectedItems[0];

string productName = selectedOrder.SubItems[0].Text;

decimal price = decimal.Parse(selectedOrder.SubItems[1].Text.Replace(" ₴", ""));

string clientName = selectedOrder.SubItems[2].Text;

string phone = selectedOrder.SubItems[3].Text;

using (SqlConnection conn = new SqlConnection(connectionString))

{

conn.Open();

string insertAcceptedQuery = "INSERT INTO AcceptedOrders (ProductName, Price, ClientName, Phone) " +

"VALUES (@ProductName, @Price, @ClientName, @Phone)";

SqlCommand cmdInsert = new SqlCommand(insertAcceptedQuery, conn);

cmdInsert.Parameters.AddWithValue("@ProductName", productName);

cmdInsert.Parameters.AddWithValue("@Price", price);

cmdInsert.Parameters.AddWithValue("@ClientName", clientName);

cmdInsert.Parameters.AddWithValue("@Phone", phone);

cmdInsert.ExecuteNonQuery();

string deleteOrderQuery = "DELETE FROM Orders WHERE ProductName = @ProductName AND ClientName = @ClientName";

SqlCommand cmdDelete = new SqlCommand(deleteOrderQuery, conn);

cmdDelete.Parameters.AddWithValue("@ProductName", productName);

cmdDelete.Parameters.AddWithValue("@ClientName", clientName);

cmdDelete.ExecuteNonQuery();

MessageBox.Show("Замовлення прийнято.", "Успіх", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

listView\_Show.Items.Remove(selectedOrder);

}

}

private void Button\_Cancel\_Click(object sender, EventArgs e)

{

RemoveSelectedOrder();

}

private void RemoveSelectedOrder()

{

if (listView\_Show.SelectedItems.Count > 0)

{

ListViewItem selectedItem = listView\_Show.SelectedItems[0];

string productName = selectedItem.SubItems[0].Text;

string clientName = selectedItem.SubItems[2].Text;

DialogResult result = MessageBox.Show("Ви дійсно бажаєте скасувати це замовлення?", "Підтвердження", MessageBoxButtons.YesNo, MessageBoxIcon.Question);

if (result == DialogResult.Yes)

{

using (SqlConnection conn = new SqlConnection(connectionString))

{

conn.Open();

string deleteOrderQuery = "DELETE FROM Orders WHERE ProductName = @ProductName AND ClientName = @ClientName";

SqlCommand cmdDelete = new SqlCommand(deleteOrderQuery, conn);

cmdDelete.Parameters.AddWithValue("@ProductName", productName);

cmdDelete.Parameters.AddWithValue("@ClientName", clientName);

cmdDelete.ExecuteNonQuery();

}

listView\_Show.Items.Remove(selectedItem);

}

}

}

private void ExecuteAndDisplayReport(string reportType)

{

listView\_Show.Items.Clear();

listView\_Show.Columns.Clear();

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

SqlCommand cmd = null;

switch (reportType)

{

case "SalesReport":

cmd = new SqlCommand("sp\_AdvancedSalesReport", connection);

SetupSalesReportColumns();

break;

case "ClientReport":

cmd = new SqlCommand("sp\_ClientActivityReport", connection);

SetupClientReportColumns();

break;

case "DiscountReport":

cmd = new SqlCommand("sp\_DiscountEffectivenessReport", connection);

SetupDiscountReportColumns();

break;

}

cmd.CommandType = CommandType.StoredProcedure;

using (SqlDataReader reader = cmd.ExecuteReader())

{

while (reader.Read())

{

string[] rowData = new string[reader.FieldCount];

for (int i = 0; i < reader.FieldCount; i++)

{

rowData[i] = reader[i].ToString();

}

listView\_Show.Items.Add(new ListViewItem(rowData));

}

}

}

}

private void SetupSalesReportColumns()

{

listView\_Show.Columns.Add("", 210);

listView\_Show.Columns.Add("Категорія", 150);

listView\_Show.Columns.Add("Назва Продукту", 250);

listView\_Show.Columns.Add("Всього Замовлень", 155);

listView\_Show.Columns.Add("Загальна Кількість", 155);

listView\_Show.Columns.Add("Загальний Дохід", 145);

listView\_Show.Columns.Add("Прогноз Кількості", 150);

listView\_Show.Columns.Add("Прогноз Доходу", 130);

listView\_Show.Columns.Add("Рейтинг категорії", 190);

}

private void SetupClientReportColumns()

{

listView\_Show.Columns.Add("", 210);

listView\_Show.Columns.Add("Ім'я Клієнта", 150);

listView\_Show.Columns.Add("Всього Замовлень", 180);

listView\_Show.Columns.Add("Куплено Товарів", 165);

listView\_Show.Columns.Add("Витрачено", 120);

listView\_Show.Columns.Add("Остання Покупка", 170);

listView\_Show.Columns.Add("Днів з Першої Покупки", 175);

listView\_Show.Columns.Add("Категорія Клієнта", 150);

}

private void SetupDiscountReportColumns()

{

listView\_Show.Columns.Add("", 210);

listView\_Show.Columns.Add("Категорія", 120);

listView\_Show.Columns.Add("Назва Продукту", 190);

listView\_Show.Columns.Add("Всього Замовлень", 180);

listView\_Show.Columns.Add("Загальна Кількість", 155);

listView\_Show.Columns.Add("Загальний Дохід", 145);

listView\_Show.Columns.Add("Середня Вартість Замовлення", 250);

}

private void ShowSalesAnalyticsMenuItem\_Click(object sender, EventArgs e)

{

listView\_Show.Items.Clear();

listView\_Show.Columns.Clear();

listView\_Show.Columns.Add("Категорія", 150);

listView\_Show.Columns.Add("Назва Продукту", 250);

listView\_Show.Columns.Add("Всього Замовлень", 150);

listView\_Show.Columns.Add("Загальна Кількість", 150);

listView\_Show.Columns.Add("Загальний Дохід", 150);

listView\_Show.Columns.Add("Середня Ціна", 150);

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = "SELECT \* FROM vw\_SalesAnalytics";

SqlCommand command = new SqlCommand(query, connection);

SqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

ListViewItem item = new ListViewItem(new string[]

{

reader["Category"].ToString(),

reader["ProductName"].ToString(),

reader["TotalOrders"].ToString(),

reader["TotalQuantitySold"].ToString(),

reader["TotalRevenue"].ToString() + " ₴",

reader["AveragePrice"].ToString() + " ₴"

});

listView\_Show.Items.Add(item);

}

}

}

catch (Exception ex)

{

MessageBox.Show($"Помилка при завантаженні аналітики продажів: {ex.Message}", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

}

А.2. Форма авторизації (Authorization of ItemSearchPro)

using UserHandlerAuthorization;

namespace ItemSearchPro

{

public partial class Authorization\_for\_ItemSearchPro : Form

{

private UserHandler userHandler;

private DeveloperFormHandler developerHandler;

public Authorization\_for\_ItemSearchPro()

{

InitializeComponent();

string connectionString = "Data Source=localhost;Initial Catalog=InformationSearchSystem;Integrated Security=True";

this.userHandler = new UserHandler(connectionString);

this.developerHandler = new DeveloperFormHandler();

}

private void Button\_Authorization\_Click(object sender, EventArgs e)

{

string usernameOrEmail = TextBox\_Username\_or\_email.Text.Trim();

string password = TextBox\_Password.Text;

bool userExists = userHandler.UserExists(usernameOrEmail);

if (!userExists)

{

MessageBox.Show("Користувача з введеним ім'ям користувача або email не знайдено.", "Помилка авторизації");

return;

}

bool isAuthenticated = userHandler.AuthenticateUser(usernameOrEmail, password);

if (isAuthenticated)

{

ShowModeSelectionDialog();

MessageBox.Show("Авторизація пройшла успішно!");

}

else

{

MessageBox.Show("Не вдалося авторизуватися. Будь ласка, перевірте свої дані.", "Помилка авторизації");

}

}

private void ShowModeSelectionDialog()

{

string result = Microsoft.VisualBasic.Interaction.InputBox(

"Виберіть режим роботи: (Вкажіть тільки цифру)\n\n1. Частковий режим (лише пошук товарів та послуг)\n2. Повний режим (пошук та замовлення товарів та послуг)",

"Вибір режиму",

"");

if (!string.IsNullOrEmpty(result))

{

Information\_and\_search\_system nextForm;

if (result == "1")

{

nextForm = new Information\_and\_search\_system(false);

}

else if (result == "2")

{

nextForm = new Information\_and\_search\_system(true);

}

else

{

MessageBox.Show("Невірний вибір режиму.", "Помилка");

return;

}

nextForm.Show();

this.Hide();

}

}

private void linkLabel\_ForgotPassword\_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)

{

Password\_recovery passwordRecoveryForm = new Password\_recovery();

this.Hide();

passwordRecoveryForm.ShowDialog();

}

private void linkLabel\_Return\_to\_the\_home\_page\_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)

{

Welcome\_to\_ItemSearchPro homeForm = new Welcome\_to\_ItemSearchPro();

homeForm.Show();

this.Close();

}

private void Authorization\_for\_ItemSearchPro\_Load(object sender, EventArgs e)

{

}

private void Button\_Guest\_Click(object sender, EventArgs e)

{

DialogResult result = MessageBox.Show("Ви впевнені, що хочете перейти у режим гостя?", "Підтвердження переходу", MessageBoxButtons.YesNo, MessageBoxIcon.Question);

if (result == DialogResult.Yes)

{

Information\_and\_search\_system\_for\_guests home = new Information\_and\_search\_system\_for\_guests();

home.WindowState = FormWindowState.Maximized;

home.Show();

this.Close();

}

}

}

}

А.4. Форма інформаційно-пошукова система для гостей (Information and search system for guests)

using ItemSearchPro\_guests;

using System;

using System.Collections.Generic;

using System.Windows.Forms;

namespace ItemSearchPro

{

public partial class Information\_and\_search\_system\_for\_guests : Form

{

private string connectionString = "Data Source=localhost;Initial Catalog=InformationSearchSystem;Integrated Security=True";

private DatabaseManagerGuests databaseManager;

private List<ProductManagerGuests> cart;

private List<ProductManagerGuests> filteredProducts;

private void Information\_and\_search\_system\_for\_guests\_Load(object sender, EventArgs e)

{

listView\_Translation\_result.Items.Clear();

listView\_Translation\_result.View = View.Details;

listView\_Translation\_result.GridLines = true;

listView\_Translation\_result.FullRowSelect = true;

listView\_Translation\_result.Columns.Add("Ім'я", 330);

listView\_Translation\_result.Columns.Add("Ціна", 137);

LoadProductsFromDatabase();

listView\_Translation\_result.Items.Clear();

}

public Information\_and\_search\_system\_for\_guests()

{

InitializeComponent();

databaseManager = new DatabaseManagerGuests();

cart = new List<ProductManagerGuests>();

filteredProducts = new List<ProductManagerGuests>();

this.MaximizeBox = false;

}

private void LoadProductsFromDatabase()

{

List<string> allResults = databaseManager.SearchNamesInDatabase("");

UpdateSearchResults(allResults);

}

private void Button\_Search\_Click(object sender, EventArgs e)

{

string keyword = TextBox\_Search\_field.Text.Trim();

List<string> searchResults = databaseManager.SearchNamesInDatabase(keyword);

UpdateSearchResults(searchResults);

}

private void Button\_More\_details\_Click(object sender, EventArgs e)

{

if (listView\_Translation\_result.SelectedItems.Count > 0)

{

string selectedProductName = listView\_Translation\_result.SelectedItems[0].Text;

ProductManagerGuests productDetails = databaseManager.GetProductByNameFromDatabase(selectedProductName);

if (productDetails != null)

{

string detailsMessage =

$"Магазин: {productDetails.ShopName}\n" +

$"Категорія: {productDetails.Category}\n" +

$"Назва: {productDetails.Name}\n" +

$"Опис: {productDetails.Description}\n" +

$"Ціна: {productDetails.Price}\n" +

$"Адреса: {productDetails.Address}\n";

MessageBox.Show(detailsMessage, "Детальна інформація про продукт");

}

else

{

MessageBox.Show("Інформація про продукт не знайдена", "Помилка");

}

}

else

{

MessageBox.Show("Виберіть продукт для відображення деталей", "Попередження");

}

}

private void UpdateSearchResults(List<string> results)

{

listView\_Translation\_result.Items.Clear();

foreach (string result in results)

{

ProductManagerGuests product = databaseManager.GetProductByNameFromDatabase(result);

if (product != null)

{

ListViewItem item = new ListViewItem(new string[] { product.Name, product.Price.ToString("C") });

listView\_Translation\_result.Items.Add(item);

}

else

{

ListViewItem item = new ListViewItem(result);

listView\_Translation\_result.Items.Add(item);

}

}

}

private void linkLabel\_Return\_to\_the\_home\_page\_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)

{

DialogResult result = MessageBox.Show("Оберіть дію:\nТак - Повернутися на початкову сторінку\nНі - Залишитися на поточній сторінці\nСкасувати - Закрити програму", "Повернення на початкову сторінку", MessageBoxButtons.YesNoCancel, MessageBoxIcon.Question);

if (result == DialogResult.Yes)

{

Welcome\_to\_ItemSearchPro homeForm = new Welcome\_to\_ItemSearchPro();

homeForm.Show();

this.Close();

}

else if (result == DialogResult.No)

{

}

else if (result == DialogResult.Cancel)

{

DialogResult exitResult = MessageBox.Show("Ви впевнені, що бажаєте вийти з програми?", "Підтвердження виходу", MessageBoxButtons.YesNo, MessageBoxIcon.Question);

if (exitResult == DialogResult.Yes)

{

Application.Exit();

}

}

}

private void listView\_Translation\_result\_SelectedIndexChanged(object sender, EventArgs e)

{

}

}

}

А.5. Форма інформаційно-пошукова система (Information and search system)

using OfficeOpenXml;

using System.Text.RegularExpressions;

using YourLibraryNamespace;

using Information\_and\_search\_system;

using System.Data.SqlClient;

namespace ItemSearchPro

{

public partial class Information\_and\_search\_system : Form

{

private string connectionString = "Data Source=localhost;Initial Catalog=InformationSearchSystem;Integrated Security=True";

private SearchEngine searchEngine;

private List<Product> cart;

private List<Product> filteredProducts;

private List<Product> products;

private Validator validator;

private bool isFullMode;

private List<Shop> shops;

public Information\_and\_search\_system()

{

InitializeComponent();

}

private void Form1\_Load(object sender, EventArgs e)

{

listView\_Basket.SelectedIndexChanged += listView\_Basket\_SelectedIndexChanged;

listView\_Translation\_result.MouseDoubleClick += listView\_Translation\_result\_SelectedIndexChanged;

listView\_Shop\_Results.View = View.Details;

listView\_Shop\_Results.GridLines = true;

listView\_Shop\_Results.FullRowSelect = true;

LoadOrdersFromDatabase();

LoadProductsShopFromDatabase();

LoadShopsFromDatabase();

listView\_Translation\_result.View = View.Details;

listView\_Translation\_result.GridLines = true;

listView\_Translation\_result.FullRowSelect = true;

listView\_Basket.View = View.Details;

listView\_Basket.GridLines = true;

listView\_Basket.FullRowSelect = true;

listView\_Basket.Columns.Add("Ім'я", 300);

listView\_Basket.Columns.Add("Ціна", 155);

listView\_Translation\_result.Columns.Add("Ім'я", 280);

listView\_Translation\_result.Columns.Add("Ціна", 130);

LoadProductsFromDatabase();

listView\_Translation\_result.Items.Clear();

listView\_Basket.Items.Clear();

listView\_Shop\_Results.Items.Clear();

}

public Information\_and\_search\_system(bool isRegisteredUser)

{

InitializeComponent();

searchEngine = new SearchEngine();

cart = new List<Product>();

filteredProducts = new List<Product>();

validator = new Validator();

isFullMode = isRegisteredUser;

SetUIVisibility();

listView\_Basket.SelectedIndexChanged += listView\_Basket\_SelectedIndexChanged;

listView\_Basket.MouseDoubleClick += listView\_Basket\_MouseDoubleClick;

Button\_Cancel\_added\_product.Click += Button\_Cancel\_added\_product\_Click;

label\_Customer\_name.Visible = false;

label\_Phone\_number.Visible = false;

label\_TotalPrice.Visible = false;

TextBox\_Customer\_name.Visible = false;

TextBox\_Phone\_number.Visible = false;

Button\_Order.Visible = false;

Button\_Cancel\_added\_product.Visible = false;

listView\_Basket.Visible = false;

Button\_Cancel\_added\_product.Enabled = false;

listView\_Translation\_result.Items.Clear();

listView\_Basket.Items.Clear();

listView\_Shop\_Results.Items.Clear();

}

private void pictureBox\_Order\_Click(object sender, EventArgs e)

{

if (label\_Customer\_name.Visible)

{

label\_Customer\_name.Visible = false;

label\_Phone\_number.Visible = false;

label\_TotalPrice.Visible = false;

TextBox\_Customer\_name.Visible = false;

TextBox\_Phone\_number.Visible = false;

Button\_Order.Visible = false;

Button\_Cancel\_added\_product.Visible = false;

listView\_Basket.Visible = false;

}

else

{

label\_Customer\_name.Visible = true;

label\_Phone\_number.Visible = true;

label\_TotalPrice.Visible = true;

TextBox\_Customer\_name.Visible = true;

TextBox\_Phone\_number.Visible = true;

Button\_Order.Visible = true;

Button\_Cancel\_added\_product.Visible = true;

listView\_Basket.Visible = true;

}

}

private void SetUIVisibility()

{

label\_Customer\_name.Visible = isFullMode;

label\_Phone\_number.Visible = isFullMode;

label\_TotalPrice.Visible = isFullMode;

TextBox\_Customer\_name.Visible = isFullMode;

TextBox\_Phone\_number.Visible = isFullMode;

Button\_Order.Visible = isFullMode;

Button\_Cancel\_added\_product.Visible = isFullMode;

Button\_Add\_to\_purchase.Visible = isFullMode;

listView\_Basket.Visible = isFullMode;

pictureBox\_Order.Visible = isFullMode;

}

private void LoadOrdersFromDatabase()

{

try

{

listView\_Shop\_Results.Items.Clear();

listView\_Shop\_Results.Columns.Clear();

listView\_Shop\_Results.Columns.Add("ПродуктID", 100);

listView\_Shop\_Results.Columns.Add("ПорядокID", 100);

listView\_Shop\_Results.Columns.Add("Ім'я клієнта", 250);

listView\_Shop\_Results.Columns.Add("Телефон", 150);

listView\_Shop\_Results.Columns.Add("Кількість", 100);

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = "SELECT OrderID, ProductID, ClientName, Phone, Quantity FROM Orders ORDER BY OrderID DESC";

using (SqlCommand command = new SqlCommand(query, connection))

using (SqlDataReader reader = command.ExecuteReader())

{

while (reader.Read())

{

ListViewItem item = new ListViewItem(new string[]

{

reader["ProductID"].ToString(),

reader["OrderID"].ToString(),

reader["ClientName"].ToString(),

reader["Phone"].ToString(),

reader["Quantity"].ToString()

});

listView\_Shop\_Results.Items.Add(item);

}

}

}

}

catch (Exception ex)

{

MessageBox.Show($"Помилка завантаження замовлень: {ex.Message}", "Помилка");

}

}

private void LoadProductsFromDatabase()

{

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = "SELECT Name, Price FROM Products";

using (SqlCommand command = new SqlCommand(query, connection))

using (SqlDataReader reader = command.ExecuteReader())

{

listView\_Translation\_result.Items.Clear();

while (reader.Read())

{

string name = reader["Name"].ToString();

decimal price = Convert.ToDecimal(reader["Price"]);

ListViewItem item = new ListViewItem(new string[] { name, price.ToString("C") });

listView\_Translation\_result.Items.Add(item);

}

}

}

}

catch (Exception ex)

{

MessageBox.Show($"Помилка завантаження товарів: {ex.Message}", "Помилка");

}

}

private void LoadShopsFromDatabase()

{

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = @"SELECT ShopID, Name, Address, WorkingHours, ContactInfo, Reviews

FROM Shops

ORDER BY Name";

shops = new List<Shop>();

using (SqlCommand command = new SqlCommand(query, connection))

using (SqlDataReader reader = command.ExecuteReader())

{

while (reader.Read())

{

shops.Add(new Shop

{

ShopID = reader.GetInt32(reader.GetOrdinal("ShopID")),

Name = reader["Name"].ToString(),

Address = reader["Address"].ToString(),

WorkingHours = reader["WorkingHours"].ToString(),

ContactInfo = reader["ContactInfo"].ToString(),

Reviews = reader["Reviews"].ToString()

});

}

}

}

}

catch (Exception ex)

{

MessageBox.Show($"Помилка завантаження магазинів: {ex.Message}", "Помилка");

}

}

private void Button\_Search\_Click(object sender, EventArgs e)

{

if (!radioButton\_SearchProducts.Checked && !radioButton\_SearchShops.Checked)

{

MessageBox.Show("Будь ласка, виберіть тип пошуку: товари або магазини.", "Увага");

return;

}

if (!TestDatabaseConnection())

{

MessageBox.Show("Помилка підключення до бази даних. Перевірте з'єднання.", "Помилка");

return;

}

try

{

string keyword = TextBox\_Search\_field.Text.Trim();

if (radioButton\_SearchProducts.Checked)

{

SearchProducts(keyword);

}

else if (radioButton\_SearchShops.Checked)

{

SearchShops(keyword);

}

}

catch (Exception ex)

{

MessageBox.Show($"Помилка пошуку: {ex.Message}", "Помилка");

}

}

private void SearchProducts(string keyword)

{

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query;

SqlCommand command;

if (string.IsNullOrWhiteSpace(keyword))

{

query = @"SELECT Name, Price

FROM Products

ORDER BY Name";

command = new SqlCommand(query, connection);

}

else

{

query = @"SELECT Name, Price

FROM Products

WHERE Name LIKE @Keyword + '%'

ORDER BY Name";

command = new SqlCommand(query, connection);

command.Parameters.AddWithValue("@Keyword", keyword);

}

using (command)

using (SqlDataReader reader = command.ExecuteReader())

{

listView\_Translation\_result.Items.Clear();

while (reader.Read())

{

ListViewItem item = new ListViewItem(reader["Name"].ToString());

item.SubItems.Add(Convert.ToDecimal(reader["Price"]).ToString("C"));

listView\_Translation\_result.Items.Add(item);

}

if (listView\_Translation\_result.Items.Count == 0)

{

MessageBox.Show("Товарів не знайдено", "Інформація",

MessageBoxButtons.OK, MessageBoxIcon.Information);

}

}

}

}

catch (Exception ex)

{

MessageBox.Show($"Помилка при пошуку товарів: {ex.Message}", "Помилка",

MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void SearchShops(string keyword)

{

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query;

SqlCommand command;

if (string.IsNullOrWhiteSpace(keyword))

{

query = @"SELECT Name, Address, WorkingHours, ContactInfo, Reviews

FROM Shops

ORDER BY Name";

command = new SqlCommand(query, connection);

}

else

{

query = @"SELECT Name, Address, WorkingHours, ContactInfo, Reviews

FROM Shops

WHERE Name LIKE @Keyword + '%'

ORDER BY Name";

command = new SqlCommand(query, connection);

command.Parameters.AddWithValue("@Keyword", keyword);

}

using (command)

using (SqlDataReader reader = command.ExecuteReader())

{

listView\_Shop\_Results.Items.Clear();

listView\_Shop\_Results.Columns.Clear();

listView\_Shop\_Results.Columns.Add("Назва магазину", 200);

listView\_Shop\_Results.Columns.Add("Адреса магазина", 300);

listView\_Shop\_Results.Columns.Add("Години роботи", 150);

listView\_Shop\_Results.Columns.Add("Контактні дані", 200);

listView\_Shop\_Results.Columns.Add("Репутація", 250);

while (reader.Read())

{

ListViewItem item = new ListViewItem(reader["Name"].ToString());

item.SubItems.Add(reader["Address"].ToString());

item.SubItems.Add(reader["WorkingHours"].ToString());

item.SubItems.Add(reader["ContactInfo"].ToString());

item.SubItems.Add(reader["Reviews"].ToString());

listView\_Shop\_Results.Items.Add(item);

}

if (listView\_Shop\_Results.Items.Count == 0)

{

MessageBox.Show("Магазинів не знайдено", "Інформація",

MessageBoxButtons.OK, MessageBoxIcon.Information);

}

}

}

}

catch (Exception ex)

{

MessageBox.Show($"Помилка при пошуку магазинів: {ex.Message}", "Помилка",

MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private bool TestDatabaseConnection()

{

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = "SELECT COUNT(\*) FROM Shops";

using (SqlCommand command = new SqlCommand(query, connection))

{

int count = (int)command.ExecuteScalar();

return true;

}

}

}

catch (Exception)

{

return false;

}

}

private void UpdateShopListView(SqlDataReader reader)

{

listView\_Shop\_Results.Items.Clear();

while (reader.Read())

{

ListViewItem item = new ListViewItem(reader["Name"].ToString());

item.SubItems.Add(reader["Address"].ToString());

item.SubItems.Add(reader["WorkingHours"].ToString());

item.SubItems.Add(reader["ContactInfo"].ToString());

item.SubItems.Add(reader["Reviews"].ToString());

listView\_Shop\_Results.Items.Add(item);

}

}

private void UpdateSearchResults(List<ProductManager> results)

{

listView\_Translation\_result.Items.Clear();

foreach (ProductManager result in results)

{

ListViewItem item = new ListViewItem(result.Name);

item.SubItems.Add(result.Price.ToString("C"));

listView\_Translation\_result.Items.Add(item);

}

}

private void UpdateShopSearchResults(List<Shop> results)

{

listView\_Shop\_Results.Items.Clear();

listView\_Shop\_Results.Columns.Clear();

listView\_Shop\_Results.Columns.Add("Назва магазину", 200);

listView\_Shop\_Results.Columns.Add("Адреса магазина", 500);

listView\_Shop\_Results.Columns.Add("Години роботи", 250);

listView\_Shop\_Results.Columns.Add("Контактні дані", 400);

listView\_Shop\_Results.Columns.Add("Репутація", 100);

foreach (Shop shop in results)

{

ListViewItem item = new ListViewItem(shop.Name);

item.SubItems.Add(shop.Address);

item.SubItems.Add(shop.WorkingHours);

item.SubItems.Add(shop.ContactInfo);

item.SubItems.Add(shop.Reviews);

listView\_Shop\_Results.Items.Add(item);

}

}

private void LoadProductsShopFromDatabase()

{

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = @"SELECT p.ProductID, p.Name, p.Price, p.Description, p.Category,

s.Name as ShopName, s.Address

FROM Products p

LEFT JOIN Shops s ON p.ShopID = s.ShopID

ORDER BY p.Name";

using (SqlCommand command = new SqlCommand(query, connection))

using (SqlDataReader reader = command.ExecuteReader())

{

products = new List<Product>();

listView\_Translation\_result.Items.Clear();

while (reader.Read())

{

var product = new Product

{

ProductID = reader.GetInt32(reader.GetOrdinal("ProductID")),

Name = reader["Name"].ToString(),

Price = Convert.ToDecimal(reader["Price"]),

Description = reader["Description"].ToString(),

Category = reader["Category"].ToString(),

ShopName = reader["ShopName"].ToString(),

Adress = reader["Address"].ToString()

};

products.Add(product);

var item = new ListViewItem(product.Name) { Tag = product };

item.SubItems.Add(product.Price.ToString("C"));

listView\_Translation\_result.Items.Add(item);

}

}

}

}

catch (Exception ex)

{

MessageBox.Show($"Помилка завантаження товарів: {ex.Message}", "Помилка");

}

}

private void Button\_More\_details\_Click(object sender, EventArgs e)

{

if (listView\_Translation\_result.SelectedItems.Count > 0)

{

string selectedProductName = listView\_Translation\_result.SelectedItems[0].Text;

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = @"SELECT p.\*, s.Name as ShopName

FROM Products p

LEFT JOIN Shops s ON p.ShopID = s.ShopID

WHERE p.Name = @ProductName";

using (SqlCommand command = new SqlCommand(query, connection))

{

command.Parameters.AddWithValue("@ProductName", selectedProductName);

using (SqlDataReader reader = command.ExecuteReader())

{

if (reader.Read())

{

string detailsMessage =

$"Магазин: {reader["StoreName"]}\n" +

$"Категорія: {reader["Category"]}\n" +

$"Назва: {reader["Name"]}\n" +

$"Опис: {reader["Description"]}\n" +

$"Ціна: {Convert.ToDecimal(reader["Price"]):C}\n" +

$"Адреса: {reader["Address"]}\n";

MessageBox.Show(detailsMessage, "Детальна інформація про продукт");

}

else

{

MessageBox.Show("Інформація про продукт не знайдена", "Помилка");

}

}

}

}

}

catch (Exception ex)

{

MessageBox.Show($"Помилка отримання деталей продукту: {ex.Message}", "Помилка");

}

}

else

{

MessageBox.Show("Виберіть продукт для відображення деталей", "Попередження");

}

}

private void Button\_Add\_to\_purchase\_Click(object sender, EventArgs e)

{

List<ListViewItem> itemsToAdd = new List<ListViewItem>();

foreach (ListViewItem selectedItem in listView\_Translation\_result.SelectedItems)

{

string selectedProductName = selectedItem.Text;

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = @"SELECT p.\*, s.Name as ShopName

FROM Products p

LEFT JOIN Shops s ON p.ShopID = s.ShopID

WHERE p.Name = @ProductName";

using (SqlCommand command = new SqlCommand(query, connection))

{

command.Parameters.AddWithValue("@ProductName", selectedProductName);

using (SqlDataReader reader = command.ExecuteReader())

{

if (reader.Read())

{

string productName = reader["Name"].ToString();

decimal price = Convert.ToDecimal(reader["Price"]);

ListViewItem newBasketItem = new ListViewItem(productName);

newBasketItem.SubItems.Add(price.ToString("C"));

newBasketItem.Tag = new Product

{

Name = productName,

Price = price,

Description = reader["Description"].ToString(),

Category = reader["Category"].ToString(),

ShopName = reader["ShopName"].ToString(),

Adress = reader["Address"].ToString()

};

itemsToAdd.Add(newBasketItem);

MessageBox.Show($"Додано в кошик: {productName} - {price:C}");

}

else

{

MessageBox.Show($"Продукт '{selectedProductName}' не знайдено в базі даних.", "Помилка");

}

}

}

}

}

catch (Exception ex)

{

MessageBox.Show($"Помилка додавання продукту до кошика: {ex.Message}", "Помилка");

}

}

if (itemsToAdd.Any())

{

listView\_Basket.Items.AddRange(itemsToAdd.ToArray());

UpdateCartDisplay();

UpdateTotalPrice();

}

}

private void UpdateCartDisplay()

{

listView\_Basket.BeginUpdate();

// Для очистити і перезаповнити таблицю

// listView\_Basket.Items.Clear();

foreach (Product product in cart)

{

bool itemExists = false;

foreach (ListViewItem item in listView\_Basket.Items)

{

if (item.Text == product.Name)

{

itemExists = true;

break;

}

}

if (!itemExists)

{

ListViewItem item = new ListViewItem(product.Name);

item.SubItems.Add(product.Price.ToString("C"));

listView\_Basket.Items.Add(item);

}

}

listView\_Basket.EndUpdate();

}

private void UpdateTotalPrice()

{

decimal totalPrice = 0;

foreach (ListViewItem item in listView\_Basket.Items)

{

if (item.Tag is Product product)

{

totalPrice += product.Price;

}

else

{

string priceText = item.SubItems[1].Text.Replace("₴", "").Trim();

if (decimal.TryParse(priceText, out decimal price))

{

totalPrice += price;

}

}

}

label\_TotalPrice.Text = $"Загальна ціна: {totalPrice:C}";

}

private void Button\_Order\_Click(object sender, EventArgs e)

{

if (listView\_Basket.Items.Count == 0)

{

MessageBox.Show("Корзина порожня. Додайте товари перед оформленням замовлення.", "Попередження");

return;

}

if (string.IsNullOrWhiteSpace(TextBox\_Customer\_name.Text))

{

MessageBox.Show("Будь ласка, заповніть всі обов'язкові поля (ім'я, номер телефону).", "Помилка");

return;

}

string customerName = TextBox\_Customer\_name.Text.Trim();

string customerPhone = TextBox\_Phone\_number.Text.Trim();

if (!IsValidPhoneNumber(customerPhone))

{

MessageBox.Show("Неправильний формат номера телефону", "Помилка");

return;

}

MessageBox.Show("Ваше замовлення успішно оформлено!", "Підтвердження замовлення");

List<Product> order = new List<Product>();

foreach (ListViewItem item in listView\_Basket.Items)

{

string productName = item.Text;

decimal productPrice = decimal.Parse(item.SubItems[1].Text.Replace("₴", "").Replace(" ", ""));

Product product = GetProductDetails(productName, productPrice);

order.Add(product);

}

SaveOrderToDatabase(order, customerName, customerPhone);

TextBox\_Customer\_name.Clear();

TextBox\_Phone\_number.Clear();

listView\_Basket.Items.Clear();

UpdateTotalPrice();

}

private bool IsValidPhoneNumber(string phoneNumber)

{

string pattern = @"^(\+38)?(0\d{9})$";

Regex regex = new Regex(pattern);

return regex.IsMatch(phoneNumber);

}

private Product GetProductDetails(string productName, decimal productPrice)

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = @"SELECT Name, Price, Description, Category, ShopName, Address

FROM Products

WHERE Name = @Name AND Price = @Price";

using (SqlCommand command = new SqlCommand(query, connection))

{

command.Parameters.AddWithValue("@Name", productName);

command.Parameters.AddWithValue("@Price", productPrice);

using (SqlDataReader reader = command.ExecuteReader())

{

if (reader.Read())

{

return new Product

{

Name = reader["Name"].ToString(),

Price = Convert.ToDecimal(reader["Price"]),

Description = reader["Description"].ToString(),

Category = reader["Category"].ToString(),

ShopName = reader["ShopName"].ToString(),

Adress = reader["Address"].ToString()

};

}

}

}

}

return null;

}

private void SaveOrderToDatabase(List<Product> order, string customerName, string customerPhone)

{

if (order == null || !order.Any())

{

MessageBox.Show("Помилка: порожнє замовлення", "Помилка");

return;

}

using (SqlConnection connection = new SqlConnection(connectionString))

{

try

{

connection.Open();

using (SqlTransaction transaction = connection.BeginTransaction())

{

try

{

string query = @"INSERT INTO Orders (ProductID, ClientName, Phone, Quantity, OrderDate)

VALUES (@ProductID, @ClientName, @Phone, @Quantity, @OrderDate)";

using (SqlCommand command = new SqlCommand(query, connection, transaction))

{

foreach (Product product in order.Where(p => p != null))

{

int productId = GetProductId(product.Name, connection, transaction);

command.Parameters.Clear();

command.Parameters.AddWithValue("@ProductID", productId);

command.Parameters.AddWithValue("@ClientName", customerName);

command.Parameters.AddWithValue("@Phone", customerPhone);

command.Parameters.AddWithValue("@Quantity", 1);

command.Parameters.AddWithValue("@OrderDate", DateTime.Now);

command.ExecuteNonQuery();

}

}

transaction.Commit();

MessageBox.Show("Замовлення успішно збережено!", "Успіх");

}

catch (Exception)

{

transaction.Rollback();

throw;

}

}

}

catch (Exception ex)

{

MessageBox.Show($"Помилка збереження замовлення: {ex.Message}", "Помилка");

}

}

}

private int GetProductId(string productName, SqlConnection connection, SqlTransaction transaction)

{

string query = "SELECT ProductID FROM Products WHERE Name = @Name";

using (SqlCommand command = new SqlCommand(query, connection, transaction))

{

command.Parameters.AddWithValue("@Name", productName);

object result = command.ExecuteScalar();

if (result != null && result != DBNull.Value)

{

return Convert.ToInt32(result);

}

throw new Exception($"Продукт не знайдено: {productName}");

}

}

private void listView\_Translation\_result\_SelectedIndexChanged(object sender, EventArgs e)

{

}

public class SearchEngine

{

private Dictionary<string, List<Product>> categoriesAndProducts;

public SearchEngine()

{

categoriesAndProducts = new Dictionary<string, List<Product>>();

}

}

private void listView\_Basket\_MouseDoubleClick(object sender, MouseEventArgs e)

{

if (listView\_Basket.SelectedItems.Count > 0)

{

ListViewItem selectedItem = listView\_Basket.SelectedItems[0];

string selectedProductName = selectedItem.Text;

Product selectedProduct = cart.FirstOrDefault(product => product.Name == selectedProductName);

if (selectedProduct != null)

{

cart.Remove(selectedProduct);

UpdateCartDisplay();

UpdateTotalPrice();

}

}

}

private void listView\_Basket\_SelectedIndexChanged(object sender, EventArgs e)

{

Button\_Cancel\_added\_product.Enabled = (listView\_Basket.SelectedItems.Count > 0);

}

private void linkLabel\_Return\_to\_the\_home\_page\_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)

{

DialogResult result = MessageBox.Show("Оберіть дію:\nТак - Повернутися на початкову сторінку\nНі - Залишитися на поточній сторінці\nСкасувати - Закрити програму", "Повернення на початкову сторінку", MessageBoxButtons.YesNoCancel, MessageBoxIcon.Question);

if (result == DialogResult.Yes)

{

Welcome\_to\_ItemSearchPro homeForm = new Welcome\_to\_ItemSearchPro();

homeForm.Show();

this.Close();

}

else if (result == DialogResult.No)

{

}

else if (result == DialogResult.Cancel)

{

DialogResult exitResult = MessageBox.Show("Ви впевнені, що бажаєте вийти з програми?", "Підтвердження виходу", MessageBoxButtons.YesNo, MessageBoxIcon.Question);

if (exitResult == DialogResult.Yes)

{

Application.Exit();

}

}

}

private void Button\_Cancel\_added\_product\_Click(object sender, EventArgs e)

{

if (listView\_Basket.SelectedItems.Count > 0)

{

foreach (ListViewItem selectedItem in listView\_Basket.SelectedItems)

{

listView\_Basket.Items.Remove(selectedItem);

string selectedProductName = selectedItem.Text;

Product selectedProduct = cart.FirstOrDefault(product => product.Name == selectedProductName);

if (selectedProduct != null)

{

cart.Remove(selectedProduct);

}

}

UpdateTotalPrice();

}

}

}

}

А.6. Форма для відновлення пароля (Password recovery)

using System;

using System.Windows.Forms;

using Class\_Password\_recovery;

namespace ItemSearchPro

{

public partial class Password\_recovery : Form

{

public Password\_recovery()

{

InitializeComponent();

HidePasswordFields();

}

private void HidePasswordFields()

{

TextBox\_NewPassword.Visible = false;

TextBox\_ConfirmPassword.Visible = false;

label\_NewPassword.Visible = false;

label\_ConfirmPassword.Visible = false;

Button\_Reset\_Password.Visible = false;

}

private void ShowPasswordFields()

{

TextBox\_NewPassword.Visible = true;

TextBox\_ConfirmPassword.Visible = true;

label\_NewPassword.Visible = true;

label\_ConfirmPassword.Visible = true;

Button\_Reset\_Password.Visible = true;

TextBox\_Restore.Visible = false;

Button\_Verification.Visible = false;

label\_Restore.Visible = false;

}

private void Button\_Reset\_Password\_Click(object sender, EventArgs e)

{

string emailOrUsername = TextBox\_Restore.Text.Trim();

string newPassword = TextBox\_NewPassword.Text;

string newConfirmPassword = TextBox\_ConfirmPassword.Text;

bool userExists = RegistrationRecoverySQLManager.CheckUserExists(emailOrUsername);

if (userExists)

{

if (newPassword != newConfirmPassword)

{

MessageBox.Show("Підтвердження пароля не співпадає з новим паролем.", "Помилка");

return;

}

if (newPassword.Length < 6 || newPassword.Length > 20)

{

MessageBox.Show("Пароль повинен містити від 6 до 20 символів.", "Помилка");

return;

}

RegistrationRecoverySQLManager.UpdateUserPassword(emailOrUsername, newPassword, newConfirmPassword);

MessageBox.Show("Ваш пароль успішно змінено!", "Успіх");

HidePasswordFields();

ShowLoginForm();

}

else

{

MessageBox.Show("Користувача з такою електронною поштою або логіном не існує", "Помилка");

}

}

private void Button\_Verification\_Click(object sender, EventArgs e)

{

string emailOrUsername = TextBox\_Restore.Text.Trim();

bool userExists = RegistrationRecoverySQLManager.CheckUserExists(emailOrUsername);

if (userExists)

{

MessageBox.Show("Можете відновити пароль. Введіть новий пароль.", "Успіх");

ShowPasswordFields();

}

else

{

MessageBox.Show("Користувача з такою електронною поштою або логіном не існує", "Помилка");

}

}

private void ShowLoginForm()

{

this.Hide();

Authorization\_for\_ItemSearchPro loginForm = new Authorization\_for\_ItemSearchPro();

loginForm.Closed += (s, args) => this.Close();

loginForm.Show();

}

private void linkLabel\_Result\_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)

{

this.Close();

ShowLoginForm();

}

private void linkLabel\_Return\_to\_the\_home\_page\_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)

{

this.Close();

Welcome\_to\_ItemSearchPro authorizationForm = new Welcome\_to\_ItemSearchPro();

authorizationForm.Show();

}

}

}

А.7. Форма для реєстрації (Registration)

using YourLibraryNamespace;

using OfficeOpenXml;

using System.Text.RegularExpressions;

using System.Data.SqlClient;

namespace ItemSearchPro

{

public partial class Registration : Form

{

private bool formClosingConfirmed = false;

private bool isPasswordVisible = false;

private Validator validator;

private Dictionary<string, string> countryCodes;

private string connectionString = "Data Source=localhost;Initial Catalog=InformationSearchSystem;Integrated Security=True";

public Registration()

{

InitializeComponent();

this.FormClosing += Registration\_FormClosing;

validator = new Validator();

SetTextBoxPlaceholders();

ExcelPackage.LicenseContext = LicenseContext.NonCommercial;

pictureBox\_Password.Image = Image.FromFile(@"C:\Users\User\Pictures\Око закрите.png");

}

private void Регістрація\_Load(object sender, EventArgs e)

{

}

private void Registration\_FormClosing(object sender, FormClosingEventArgs e)

{

}

private void Button\_Sign\_up\_Click(object sender, EventArgs e)

{

string firstName = textBox\_First\_Name.Text.Trim();

string lastName = textBox\_Last\_Name.Text.Trim();

string country = textBox\_Country.Text.Trim();

string phoneNumber = textBox\_Phone\_Number.Text.Trim();

string email = textBox\_Email.Text.Trim();

string username = textBox\_Username.Text.Trim();

string password = textBox\_Password.Text;

string confirmPassword = textBox\_Confirm\_Password.Text;

Validator validator = new Validator();

if (!IsValidFirstName(firstName))

{

MessageBox.Show("Некоректне ім'я", "Помилка");

return;

}

if (!IsValidLastName(lastName))

{

MessageBox.Show("Некоректне прізвище", "Помилка");

return;

}

if (!IsValidCountry(country))

{

MessageBox.Show("Некоректна назва країни", "Помилка");

return;

}

if (!IsValidPhoneNumber(phoneNumber))

{

MessageBox.Show("Неправильний формат номера телефону", "Помилка");

return;

}

if (!validator.IsValidEmail(email))

{

MessageBox.Show("Некоректна електронна пошта", "Помилка");

return;

}

if (!IsValidUsername(username))

{

MessageBox.Show("Ім'я користувача повинно складатися з 4-20 символів (літер, цифр та підкреслення)", "Помилка");

return;

}

if (!IsValidPassword(password))

{

MessageBox.Show("Пароль повинен містити мінімум 6 символів, цифру, маленьку літеру, велику літеру та спеціальний символ (@#\_$%^&+=)", "Помилка");

return;

}

if (password != confirmPassword)

{

MessageBox.Show("Паролі не співпадають", "Помилка");

return;

}

if (string.IsNullOrEmpty(firstName) || string.IsNullOrEmpty(lastName) || string.IsNullOrEmpty(phoneNumber) ||

string.IsNullOrEmpty(email) || string.IsNullOrEmpty(username) || string.IsNullOrEmpty(password) ||

string.IsNullOrEmpty(confirmPassword))

{

MessageBox.Show("Будь ласка, заповніть всі поля", "Помилка");

return;

}

if (IsUsernameTaken(username))

{

MessageBox.Show("Такий логін вже зайнятий", "Помилка");

return;

}

if (IsEmailTaken(email))

{

MessageBox.Show("Ця адреса електронної пошти вже зайнята", "Помилка");

return;

}

YourLibraryNamespace.User newUser = new YourLibraryNamespace.User

{

FirstName = firstName,

LastName = lastName,

Country = country,

PhoneNumber = phoneNumber,

Email = email,

Username = username,

Password = password,

СonfirmPassword = confirmPassword

};

RegistrationDatabaseManager.SaveToDatabase(newUser);

MessageBox.Show("Реєстрація пройшла успішно!", "Підтвердження");

ClearFields();

Authorization\_for\_ItemSearchPro loginForm = new Authorization\_for\_ItemSearchPro();

loginForm.Show();

this.DialogResult = DialogResult.OK;

this.Close();

}

private bool IsUsernameTaken(string username)

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = "SELECT COUNT(\*) FROM Users WHERE Username = @Username";

using (SqlCommand command = new SqlCommand(query, connection))

{

command.Parameters.AddWithValue("@Username", username);

int count = (int)command.ExecuteScalar();

return count > 0;

}

}

}

private bool IsEmailTaken(string email)

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = "SELECT COUNT(\*) FROM Users WHERE Email = @Email";

using (SqlCommand command = new SqlCommand(query, connection))

{

command.Parameters.AddWithValue("@Email", email);

int count = (int)command.ExecuteScalar();

return count > 0;

}

}

}

private void ClearFields()

{

textBox\_First\_Name.Clear();

textBox\_Last\_Name.Clear();

textBox\_Phone\_Number.Clear();

textBox\_Email.Clear();

textBox\_Username.Clear();

textBox\_Password.Clear();

textBox\_Confirm\_Password.Clear();

}

private void SetTextBoxPlaceholders()

{

textBox\_First\_Name.PlaceholderText = "Введіть ім'я";

textBox\_Last\_Name.PlaceholderText = "Введіть прізвище";

textBox\_Country.PlaceholderText = "Введіть країну";

textBox\_Phone\_Number.PlaceholderText = "0984616746";

textBox\_Email.PlaceholderText = "Введіть електронну пошту";

textBox\_Username.PlaceholderText = "Введіть ім'я користувача";

textBox\_Password.PlaceholderText = "Введіть пароль";

textBox\_Confirm\_Password.PlaceholderText = "Підтвердіть пароль";

}

private void textBox\_Password\_TextChanged(object sender, EventArgs e)

{

if (!isPasswordVisible)

{

textBox\_Password.UseSystemPasswordChar = true;

textBox\_Confirm\_Password.UseSystemPasswordChar = true;

}

}

private void textBox\_Confirm\_Password\_TextChanged(object sender, EventArgs e)

{

if (!isPasswordVisible)

{

textBox\_Confirm\_Password.UseSystemPasswordChar = true;

}

}

private void pictureBox\_Password\_Click(object sender, EventArgs e)

{

isPasswordVisible = !isPasswordVisible;

if (isPasswordVisible)

{

textBox\_Password.UseSystemPasswordChar = false;

textBox\_Confirm\_Password.UseSystemPasswordChar = false;

pictureBox\_Password.Image = Image.FromFile(@"C:\Users\User\Pictures\Око відкрите.png");

}

else

{

textBox\_Password.UseSystemPasswordChar = true;

textBox\_Confirm\_Password.UseSystemPasswordChar = true;

pictureBox\_Password.Image = Image.FromFile(@"C:\Users\User\Pictures\Око закрите.png");

}

}

private void linkLabel\_Authorization\_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)

{

this.Close();

Authorization\_for\_ItemSearchPro authorizationForm = new Authorization\_for\_ItemSearchPro();

authorizationForm.Show();

}

private void linkLabel\_Return\_to\_the\_home\_page\_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)

{

this.Close();

Welcome\_to\_ItemSearchPro authorizationForm = new Welcome\_to\_ItemSearchPro();

authorizationForm.Show();

}

private bool IsValidFirstName(string firstName)

{

string pattern = @"^[a-zA-Zа-яА-ЯіІїЇ\s'-]+$";

Regex regex = new Regex(pattern);

return regex.IsMatch(firstName);

}

private bool IsValidLastName(string lastName)

{

string pattern = @"^[a-zA-Zа-яА-ЯіІїЇ\s'-]+$";

Regex regex = new Regex(pattern);

return regex.IsMatch(lastName);

}

private bool IsValidCountry(string country)

{

string pattern = @"^[a-zA-Zа-яА-ЯіІїЇ\s'-]+$";

Regex regex = new Regex(pattern);

return regex.IsMatch(country);

}

private bool IsValidPhoneNumber(string phoneNumber)

{

string pattern = @"^(\+?38)?(0\d{9})|((\d{2,3})\s?\d{3}\s?\d{2}\s?\d{2})$";

Regex regex = new Regex(pattern);

return regex.IsMatch(phoneNumber);

}

private bool IsValidUsername(string username)

{

string pattern = @"^[\w]{4,20}$";

Regex regex = new Regex(pattern);

return regex.IsMatch(username);

}

private bool IsValidPassword(string password)

{

string pattern = @"^(?=.\*\d)(?=.\*[a-z])(?=.\*[A-Z])(?=.\*[@\_#$%^&+=]).{6,}$";

Regex regex = new Regex(pattern);

return regex.IsMatch(password);

}

}

}

А.8. Вітальна форма (Welcome to ItemSearchPro)

using Welcom;

namespace ItemSearchPro

{

public partial class Welcome\_to\_ItemSearchPro : Form

{

private const string DeveloperLogin = "admin";

private const string DeveloperPassword = "password";

private bool isDeveloperMode = false;

public Welcome\_to\_ItemSearchPro()

{

InitializeComponent();

this.MaximizeBox = false;

this.FormBorderStyle = FormBorderStyle.FixedSingle;

this.KeyPreview = true;

this.KeyDown += Welcome\_to\_ItemSearchPro\_KeyDown;

}

private void ShowDeveloperMenu()

{

Form loginForm = new Form();

loginForm.Text = "Вхід для адміністратора";

loginForm.StartPosition = FormStartPosition.CenterParent;

loginForm.FormBorderStyle = FormBorderStyle.FixedSingle;

loginForm.MaximizeBox = false;

loginForm.MinimizeBox = false;

loginForm.Width = 300;

loginForm.Height = 150;

Label loginLabel = new Label();

loginLabel.Text = "Логін:";

loginLabel.Location = new Point(30, 10);

loginForm.Controls.Add(loginLabel);

TextBox loginTextBox = new TextBox();

loginTextBox.Location = new Point(130, 10);

loginTextBox.Width = 120;

loginForm.Controls.Add(loginTextBox);

Label passwordLabel = new Label();

passwordLabel.Text = "Пароль:";

passwordLabel.Location = new Point(30, 40);

loginForm.Controls.Add(passwordLabel);

TextBox passwordTextBox = new TextBox();

passwordTextBox.PasswordChar = '\*';

passwordTextBox.Location = new Point(130, 40);

passwordTextBox.Width = 120;

loginForm.Controls.Add(passwordTextBox);

Button loginButton = new Button();

loginButton.Text = "Увійти";

loginButton.Location = new Point(150, 70);

loginButton.Click += (sender, args) =>

{

string enteredLogin = loginTextBox.Text;

string enteredPassword = passwordTextBox.Text;

if (enteredLogin == DeveloperLogin && enteredPassword == DeveloperPassword)

{

MessageBox.Show("Вхід успішний!");

loginForm.Close();

Admin\_menu developerForm = new Admin\_menu();

developerForm.Show();

this.Hide();

}

else

{

MessageBox.Show("Неправильний логін або пароль!", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

};

loginForm.Controls.Add(loginButton);

loginForm.ShowDialog();

}

private void Welcome\_to\_ItemSearchPro\_KeyDown(object sender, KeyEventArgs e)

{

if (e.Control && e.Shift && e.KeyCode == Keys.D)

{

if (!isDeveloperMode)

{

DialogResult result = MessageBox.Show("Ви дійсно хочете перейти в режим адміністратора?", "Підтвердження", MessageBoxButtons.YesNo, MessageBoxIcon.Question);

if (result == DialogResult.Yes)

{

ShowDeveloperMenu();

}

}

else

{

ShowDeveloperMenu();

MessageBox.Show("Вхід пройшов успішно!", "Успіх", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

}

}

private void Button\_Sign\_up\_Click(object sender, EventArgs e)

{

Registration registrationForm = new Registration();

registrationForm.Show();

this.Hide();

}

private void Button\_Authorize\_Click(object sender, EventArgs e)

{

Authorization\_for\_ItemSearchPro authorizationForm = new Authorization\_for\_ItemSearchPro();

authorizationForm.Show();

this.Hide();

}

private void AdminModeButton\_Click(object sender, EventArgs e)

{

Admin\_menu adminForm = new Admin\_menu();

adminForm.Show();

this.Hide();

}

private void SellerModeButton\_Click(object sender, EventArgs e)

{

Admin\_menu sellerForm = new Admin\_menu();

sellerForm.Show();

this.Hide();

}

private void linkLabel\_Viber\_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)

{

ViberLinkHandler.OpenViberChat("+380986057400");

}

private void linkLabel\_Instagram\_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)

{

Welcom.Welcome.OpenInstagramProfile("https://www.instagram.com/victor\_87469?igsh=MXRnZDVmcDEwaDVmbQ==");

}

private void linkLabel\_Viber\_Manager\_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)

{

ViberLinkHandler.OpenViberChat("+380986575119");

}

private void linkLabel\_Instagram\_Manager\_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)

{

Welcom.Welcome.OpenInstagramProfile("https://www.instagram.com/hardware\_machine?utm\_source=ig\_web\_button\_share\_sheet&igsh=ZDNlZDc0MzIxNw==");

}

private void linkLabel\_Viber\_Developer\_and\_Manager\_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)

{

ViberLinkHandler.OpenViberChat("+380986057400");

}

private void linkLabel\_Instagram\_Developer\_and\_Manager\_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)

{

Welcom.Welcome.OpenInstagramProfile("https://www.instagram.com/victor\_87469?igsh=MXRnZDVmcDEwaDVmbQ==");

}

private void Welcome\_to\_ItemSearchPro\_Load(object sender, EventArgs e)

{

}

private void linkLabel\_Viber\_Administrator\_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)

{

ViberLinkHandler.OpenViberChat("+380689676513");

}

private void linkLabel\_Instagram\_Administrator\_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)

{

Welcom.Welcome.OpenInstagramProfile("https://www.instagram.com/young\_fox\_alya/#");

}

private void linkLabel\_Developer\_Number\_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)

{

string phoneNumber = "+380986057400";

DialogResult result = MessageBox.Show("Ви дійсно хочете зателефонувати до керівника?", "Підтвердження дзвінка", MessageBoxButtons.YesNo, MessageBoxIcon.Question);

if (result == DialogResult.Yes)

{

PhoneLinkHandler.OpenPhoneNumber(phoneNumber);

}

}

}

}

Додаток Б

Класи додатку

Б.1. Клас DatabaseManagerAdmin

Опис класу: Клас DatabaseManagerAdmin забезпечує адміністраторське управління базою даних, зокрема можливості для завантаження, додавання та видалення продуктів у базі даних. Також цей клас реалізує метод для створення таблиці продуктів.

Код класу:

using System;

using System.Collections.Generic;

using System.Data;

using System.Data.SqlClient;

using System.Windows.Forms;

namespace DatabaseDevelopers

{

public class DatabaseManagerAdmin

{

private string connectionString = "Data Source=localhost;Initial Catalog=SmartphoneStores;Integrated Security=True";

public class Product

{

public int ProductID { get; set; }

public string Category { get; set; }

public string Name { get; set; }

public decimal Price { get; set; }

public string StoreName { get; set; }

public string Description { get; set; }

public string Address { get; set; }

}

public List<Product> LoadProductsFromDatabase()

{

List<Product> products = new List<Product>();

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = "SELECT Category, Name, Price, StoreName, Description, Address FROM Products";

SqlCommand command = new SqlCommand(query, connection);

using (SqlDataReader reader = command.ExecuteReader())

{

while (reader.Read())

{

Product product = new Product

{

Category = reader["Category"].ToString(),

Name = reader["Name"].ToString(),

Price = Convert.ToDecimal(reader["Price"]),

StoreName = reader["StoreName"].ToString(),

Description = reader["Description"].ToString(),

Address = reader["Address"].ToString()

};

products.Add(product);

}

}

}

return products;

}

public void DeleteProductByName(string productName)

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = "DELETE FROM Products WHERE Name = @Name";

SqlCommand command = new SqlCommand(query, connection);

command.Parameters.AddWithValue("@Name", productName);

int rowsAffected = command.ExecuteNonQuery();

if (rowsAffected > 0)

{

MessageBox.Show("Продукт успішно видалено.");

}

else

{

MessageBox.Show("Продукт не знайдено.");

}

}

}

public void AddProductToDatabase(string category, string name, decimal price, string storeName, string description, string address)

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = "INSERT INTO Products (Category, Name, Price, StoreName, Description, Address) " +

"VALUES (@Category, @Name, @Price, @StoreName, @Description, @Address)";

SqlCommand command = new SqlCommand(query, connection);

command.Parameters.AddWithValue("@Category", category);

command.Parameters.AddWithValue("@Name", name);

command.Parameters.AddWithValue("@Price", price);

command.Parameters.AddWithValue("@StoreName", storeName);

command.Parameters.AddWithValue("@Description", description);

command.Parameters.AddWithValue("@Address", address);

command.ExecuteNonQuery();

MessageBox.Show("Продукт успішно додано.");

}

}

public int GetNextAvailableRow()

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = "SELECT COUNT(\*) FROM Products";

SqlCommand command = new SqlCommand(query, connection);

int rowCount = (int)command.ExecuteScalar();

return rowCount + 1;

}

}

public void CreateProductsTable()

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = @"

CREATE TABLE Products (

ProductID INT PRIMARY KEY IDENTITY(1,1),

Category NVARCHAR(50),

Name NVARCHAR(100),

Price DECIMAL(18, 2),

StoreName NVARCHAR(100),

Description NVARCHAR(MAX),

Address NVARCHAR(200)

)";

SqlCommand command = new SqlCommand(query, connection);

command.ExecuteNonQuery();

}

}

}

}

Б.2. Клас DeveloperFormHandler

Опис класу: Клас DeveloperFormHandler відповідає за автентифікацію розробника та відкриття форми для адміністратора. Клас реалізує метод для перевірки пароля та забезпечує доступ до адміністративної форми після успішної автентифікації.

Код класу:

using ItemSearchPro;

namespace UserHandlerAuthorization

{

public class DeveloperFormHandler

{

public bool AuthenticateDeveloper(string enteredPassword, string expectedPassword)

{

return SecureCompare(enteredPassword, expectedPassword);

}

private bool SecureCompare(string entered, string expected)

{

return string.Equals(entered, expected, StringComparison.OrdinalIgnoreCase);

}

public void OpenDeveloperForm()

{

Admin\_menu developerForm = new Admin\_menu();

developerForm.WindowState = FormWindowState.Normal;

developerForm.Show();

}

}

}

Б.3. Клас UserHandler

**Опис класу:** Клас UserHandler реалізує методи для роботи з користувачами, зокрема для перевірки існування користувача в базі даних, а також для автентифікації користувачів за допомогою їх імені користувача або електронної пошти та паролю.

Код класу:

using System.IO;

using OfficeOpenXml;

using System.Data.SqlClient;

namespace UserHandlerAuthorization

{

public class UserHandler

{

private string connectionString;

public UserHandler(string connectionString)

{

this.connectionString = connectionString;

}

public bool UserExists(string usernameOrEmail)

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = "SELECT COUNT(1) FROM Users WHERE Username = @usernameOrEmail OR Email = @usernameOrEmail";

using (SqlCommand command = new SqlCommand(query, connection))

{

command.Parameters.AddWithValue("@usernameOrEmail", usernameOrEmail);

int count = Convert.ToInt32(command.ExecuteScalar());

return count > 0;

}

}

}

public bool AuthenticateUser(string usernameOrEmail, string password)

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = "SELECT PasswordHash FROM Users WHERE (Username = @usernameOrEmail OR Email = @usernameOrEmail)";

using (SqlCommand command = new SqlCommand(query, connection))

{

command.Parameters.AddWithValue("@usernameOrEmail", usernameOrEmail);

object result = command.ExecuteScalar();

if (result != null)

{

string storedPasswordHash = result.ToString();

return storedPasswordHash.Equals(password, StringComparison.OrdinalIgnoreCase);

}

return false;

}

}

}

}

}

Б.4. Клас ProductManagerGuests

Опис класу: Клас ProductManagerGuests представляє собою модель продукту для гостей, яка включає властивості товару, такі як назва, адреса, ціна, опис, категорія і назва магазину. Клас використовується для зберігання і відображення інформації про продукти в системі для гостей.

Код класу:

using OfficeOpenXml;

namespace ItemSearchPro\_guests

{

public class ProductManagerGuests

{

public string Name { get; set; }

public string Address { get; set; }

public decimal Price { get; set; }

public string Description { get; set; }

public string Category { get; set; }

public string ShopName { get; set; }

}

}

Б.5. Клас DatabaseManagerGuests

Опис класу: Клас DatabaseManagerGuests відповідає за взаємодію з базою даних для пошуку товарів та отримання їх детальної інформації для гостей. Він включає методи для пошуку товарів за ключовими словами та отримання детальної інформації про товар за його назвою.

Код класу:

using System;

using System.Collections.Generic;

using System.Data;

using System.Data.SqlClient;

using System.Windows.Forms;

namespace ItemSearchPro\_guests

{

public class DatabaseManagerGuests

{

private string connectionString = "Data Source=localhost;Initial Catalog=InformationSearchSystem;Integrated Security=True";

public List<string> SearchNamesInDatabase(string keyword)

{

List<string> results = new List<string>();

string query = "SELECT Name FROM Products WHERE Name LIKE @keyword OR Description LIKE @keyword";

using (SqlConnection connection = new SqlConnection(connectionString))

{

using (SqlCommand command = new SqlCommand(query, connection))

{

command.Parameters.AddWithValue("@keyword", "%" + keyword + "%");

try

{

connection.Open();

SqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

results.Add(reader["Name"].ToString());

}

}

catch (Exception ex)

{

MessageBox.Show("Помилка підключення до бази даних: " + ex.Message, "Помилка.");

}

}

}

return results;

}

public ProductManagerGuests GetProductByNameFromDatabase(string productName)

{

string query = "SELECT \* FROM Products WHERE Name = @productName";

using (SqlConnection connection = new SqlConnection(connectionString))

{

using (SqlCommand command = new SqlCommand(query, connection))

{

command.Parameters.AddWithValue("@productName", productName);

try

{

connection.Open();

SqlDataReader reader = command.ExecuteReader();

if (reader.Read())

{

return new ProductManagerGuests

{

Name = reader["Name"].ToString(),

Price = Convert.ToDecimal(reader["Price"]),

Description = reader["Description"].ToString(),

Category = reader["Category"].ToString(),

ShopName = reader["StoreName"].ToString(),

Address = reader["Address"].ToString()

};

}

}

catch (Exception ex)

{

MessageBox.Show("Помилка при отриманні товару з бази даних: " + ex.Message, "Error");

}

}

}

return null;

}

}

}

Б.6. Клас Product

Опис класу: Клас Product являє собою модель продукту, що містить основні властивості, які описують товар в інформаційно-пошуковій системі. Це включає ідентифікатор товару, назву, ціну, опис, категорію, магазин, адресу, а також можливий ідентифікатор магазину.

Код класу:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Information\_and\_search\_system

{

public class Product

{

public int ProductID { get; set; }

public string Name { get; set; }

public decimal Price { get; set; }

public string Description { get; set; }

public string Category { get; set; }

public string ShopName { get; set; }

public string Adress { get; set; }

public int? ShopID { get; set; }

}

}

Б.7. Клас ProductManager

Опис класу: Клас ProductManager відповідає за взаємодію з базою даних для отримання детальної інформації про товар та доступу до категорій продуктів. Клас містить методи для отримання продуктів за їхньою назвою і для отримання доступних категорій.

Код класу:

using System;

using System.Collections.Generic;

using System.Data.SqlClient;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Information\_and\_search\_system

{

public class ProductManager

{

public string Name { get; set; }

public decimal Price { get; set; }

public string Description { get; set; }

public string Category { get; set; }

public string ShopName { get; set; }

public string Address { get; set; }

private string connectionString = "Data Source=localhost;Initial Catalog=InformationSearchSystem;Integrated Security=True";

public ProductManager GetProductByNameFromDatabase(string selectedProductName)

{

string query = "SELECT \* FROM Products WHERE Name = @productName";

using (SqlConnection connection = new SqlConnection(connectionString))

{

SqlCommand command = new SqlCommand(query, connection);

command.Parameters.AddWithValue("@productName", selectedProductName);

connection.Open();

SqlDataReader reader = command.ExecuteReader();

if (reader.Read())

{

return new ProductManager

{

Name = reader["Name"].ToString(),

Price = Convert.ToDecimal(reader["Price"]),

Description = reader["Description"].ToString(),

Category = reader["Category"].ToString(),

ShopName = reader["ShopName"].ToString(),

Address = reader["Address"].ToString()

};

}

else

{

MessageBox.Show("Товар не знайдено", "Помилка");

return null;

}

}

}

public List<string> GetCategoriesFromDatabase()

{

List<string> categories = new List<string>();

string query = "SELECT DISTINCT Category FROM Products";

using (SqlConnection connection = new SqlConnection(connectionString))

{

SqlCommand command = new SqlCommand(query, connection);

connection.Open();

SqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

categories.Add(reader["Category"].ToString());

}

}

return categories;

}

}

}

Б.8. Клас SearchEngine

Опис класу: Клас SearchEngine реалізує пошук товарів у базі даних по категорії. Він отримує список всіх товарів, що належать до певної категорії, та зберігає їх у вигляді списку об'єктів ProductManager.

Код класу:

using System;

using System.Collections.Generic;

using System.Data.SqlClient;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Information\_and\_search\_system

{

public class SearchEngine

{

private string connectionString = "Data Source=localhost;Initial Catalog=InformationSearchSystem;Integrated Security=True";

public List<ProductManager> GetProductsInCategory(string category)

{

List<ProductManager> products = new List<ProductManager>();

string query = "SELECT Name, Price, Description, Category, ShopName, Address FROM Products WHERE Category = @Category";

using (SqlConnection connection = new SqlConnection(connectionString))

{

SqlCommand command = new SqlCommand(query, connection);

command.Parameters.AddWithValue("@Category", category);

connection.Open();

SqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

products.Add(new ProductManager

{

Name = reader["Name"].ToString(),

Price = Convert.ToDecimal(reader["Price"]),

Description = reader["Description"].ToString(),

Category = reader["Category"].ToString(),

ShopName = reader["ShopName"].ToString(),

Address = reader["Address"].ToString()

});

}

}

return products;

}

}

}

Б.9. Клас Shop

Опис класу: Клас Shop містить основну інформацію про магазин, включаючи ідентифікатор магазину, назву, адресу, години роботи, контактну інформацію та відгуки. Це дозволяє відображати детальну інформацію про магазин в системі.

Код класу:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Information\_and\_search\_system

{

public class Shop

{

public int ShopID { get; set; }

public string Name { get; set; }

public string Address { get; set; }

public string WorkingHours { get; set; }

public string ContactInfo { get; set; }

public string Reviews { get; set; }

}

}

Б.10. Клас RegistrationRecoverySQLManager

Клас відповідає за взаємодію з базою даних для відновлення пароля користувача. Він включає методи для перевірки наявності користувача за логіном або електронною поштою, а також для оновлення пароля користувача. Паролі зберігаються у вигляді хешованих значень для забезпечення безпеки

Код класу:

using System;

using System.Data.SqlClient;

namespace Class\_Password\_recovery

{

public class RegistrationRecoverySQLManager

{

private static readonly string connectionString = "Data Source=localhost;Initial Catalog=InformationSearchSystem;Integrated Security=True";

public static bool CheckUserExists(string loginOrEmail)

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = "SELECT COUNT(\*) FROM Users WHERE Username = @loginOrEmail OR Email = @loginOrEmail";

using (SqlCommand command = new SqlCommand(query, connection))

{

command.Parameters.AddWithValue("@loginOrEmail", loginOrEmail);

int count = (int)command.ExecuteScalar();

return count > 0;

}

}

}

public static void UpdateUserPassword(string emailOrUsername, string newPassword, string newConfirmPassword)

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = "UPDATE Users SET PasswordHash = @newPassword, PasswordRepeat = @newConfirmPassword WHERE Username = @emailOrUsername OR Email = @emailOrUsername";

using (SqlCommand command = new SqlCommand(query, connection))

{

command.Parameters.AddWithValue("@newPassword", newPassword);

command.Parameters.AddWithValue("@newConfirmPassword", newConfirmPassword);

command.Parameters.AddWithValue("@emailOrUsername", emailOrUsername);

command.ExecuteNonQuery();

}

}

}

}

}

Б.11. Клас Registration\_User

Опис: Клас представляє користувача системи. Містить властивості, що зберігають персональні дані користувача, такі як ім'я, прізвище, країна, телефонний номер, електронна пошта, ім'я користувача, пароль та підтвердження пароля. Цей клас використовує атрибут Key для позначення ідентифікатора користувача як первинного ключа в базі даних.

Код класу:

using System.ComponentModel.DataAnnotations;

using ItemSearchPro;

namespace YourLibraryNamespace

{

public class User

{

[Key]

public int Id { get; set; }

public string FirstName { get; set; }

public string LastName { get; set; }

public string Country { get; set; }

public string PhoneNumber { get; set; }

public string Email { get; set; }

public string Username { get; set; }

public string Password { get; set; }

public string СonfirmPassword { get; set; }

}

}

Б.12. Клас Registration\_Validator:

Опис: Клас відповідає за валідацію введених даних користувача. Зокрема, метод IsValidEmail перевіряє правильність введеної електронної пошти, а також перевіряє її домен на відповідність з переліком підтримуваних доменів. Якщо домен електронної пошти є одним з підтримуваних, метод повертає true, в іншому випадку false

Код класу:

using System.Net.Mail;

namespace YourLibraryNamespace

{

public class Validator

{

public bool IsValidEmail(string email)

{

try

{

var addr = new MailAddress(email);

string[] supportedDomains = { "gmail.com", "outlook.com", "yahoo.com", "hotmail.com", "icloud.com", "aol.com", "protonmail.com", "student.ztu.edu.ua" };

string domain = addr.Host.ToLower();

if (supportedDomains.Contains(domain))

{

return true;

}

else

{

return false;

}

}

catch

{

return false;

}

}

}

}

Б.13. Клас RegistrationDatabaseManager

Опис: Клас відповідає за взаємодію з базою даних для реєстрації нових користувачів. Використовує SQL-запити для збереження даних користувача (включаючи ім'я, прізвище, електронну пошту, номер телефону, ім'я користувача, пароль та підтвердження пароля) у таблиці Users. Паролі зберігаються у вигляді хешів для безпеки.

Код класу:

using System;

using System.Data.SqlClient;

namespace YourLibraryNamespace

{

public static class RegistrationDatabaseManager

{

private static string connectionString = "Data Source=localhost;Initial Catalog=InformationSearchSystem;Integrated Security=True";

public static void SaveToDatabase(User newUser)

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = "INSERT INTO Users (FirstName, LastName, Country, PhoneNumber, Email, Username, PasswordHash, PasswordRepeat) " +

"VALUES (@FirstName, @LastName, @Country, @PhoneNumber, @Email, @Username, @PasswordHash, @PasswordRepeat)";

using (SqlCommand command = new SqlCommand(query, connection))

{

command.Parameters.AddWithValue("@FirstName", newUser.FirstName);

command.Parameters.AddWithValue("@LastName", newUser.LastName);

command.Parameters.AddWithValue("@Country", newUser.Country);

command.Parameters.AddWithValue("@PhoneNumber", newUser.PhoneNumber);

command.Parameters.AddWithValue("@Email", newUser.Email);

command.Parameters.AddWithValue("@Username", newUser.Username);

command.Parameters.AddWithValue("@PasswordHash", newUser.Password);

command.Parameters.AddWithValue("@PasswordRepeat", newUser.СonfirmPassword);

command.ExecuteNonQuery();

}

}

}

}

}

Б.14. Клас Welcome

Опис: Клас Welcome є нащадком абстрактного класу LinkHandler. Цей клас містить статичний метод OpenInstagramProfile, який дозволяє відкривати Instagram профіль за допомогою URL. Метод використовує можливості класу LinkHandler для відкриття переданого URL через стандартний механізм операційної системи.

Код класу:

using System;

using System.Collections.Generic;

using System.Diagnostics;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Welcom

{

public abstract class LinkHandler

{

protected static void OpenUrl(string url)

{

try

{

Process.Start(new ProcessStartInfo

{

FileName = url,

UseShellExecute = true

});

}

catch (Exception ex)

{

Console.WriteLine($"Не вдалося відкрити URL: {ex.Message}");

}

}

}

public class ViberLinkHandler : LinkHandler

{

public static void OpenViberChat(string viberNumber)

{

string viberUrl = $"viber://chat?number={viberNumber}";

OpenUrl(viberUrl);

}

}

public class Welcome : LinkHandler

{

public static void OpenInstagramProfile(string instagramUrl)

{

OpenUrl(instagramUrl);

}

}

public class PhoneLinkHandler : LinkHandler

{

public static void OpenPhoneNumber(string phoneNumber)

{

string mobileUrl = $"tel:{phoneNumber}";

OpenUrl(mobileUrl);

}

}

}

Додаток Б

Робота форм інформаційно-пошукової системи: магазин, товари та послуги

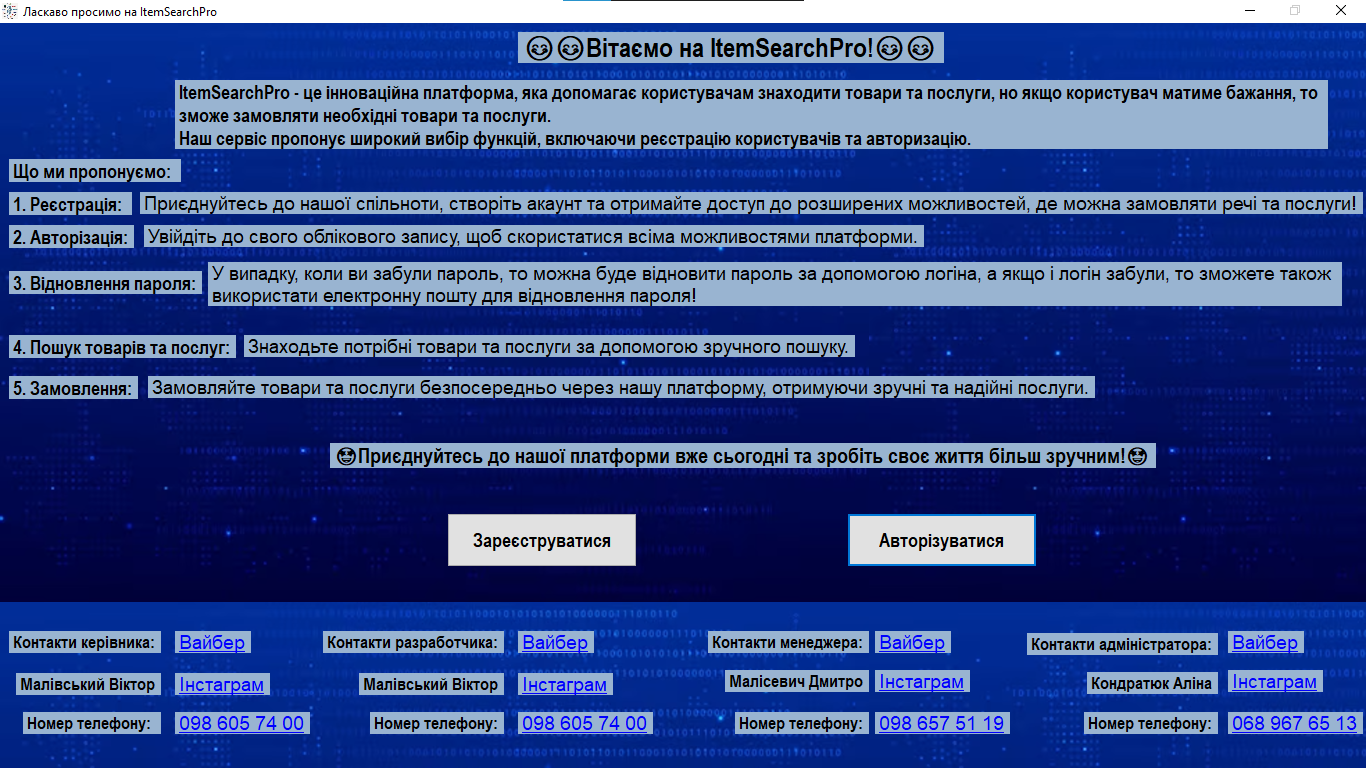


Рисунок Б.1. Вітальна форма

* Функціональні елементи:
* Кнопка “Авторизація”
* Кнопка “Реєстрація”
* Посилання на контакти підтримки
* Комбінація клавіш для виклику форми входу

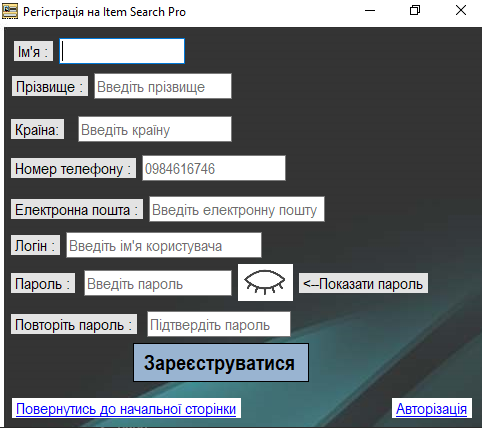


Рисунок Б.2. Форма реєстрації

* Поля форми:
* Логін
* Унікальність
* Мінімум 4 символи
* Заборона спеціальних символів
* Email
* Перевірка формату
* Унікальність в системі
* Опція підтвердження email
* Пароль
* Мінімальна довжина 8 символів
* Комбінація літер, цифр, спецсимволів
* Показник складності паролю
* Кнопка реєстрації:
* Активується після валідації всіх полів
* Надсилає дані на сервер
* Створює обліковий запис

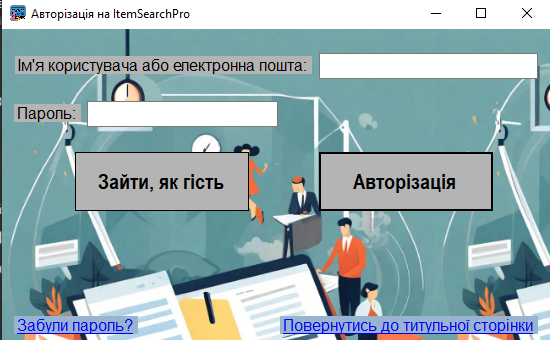


Рисунок Б.4. Форма Авторизації

* Поля форми:
* Логін/Email
* Підтримка email або Username
* Автозаповнення
* Перевірка існування облікового запису
* Пароль
* Приховане введення
* Механізм відновлення паролю
* Обмеження спроб входу

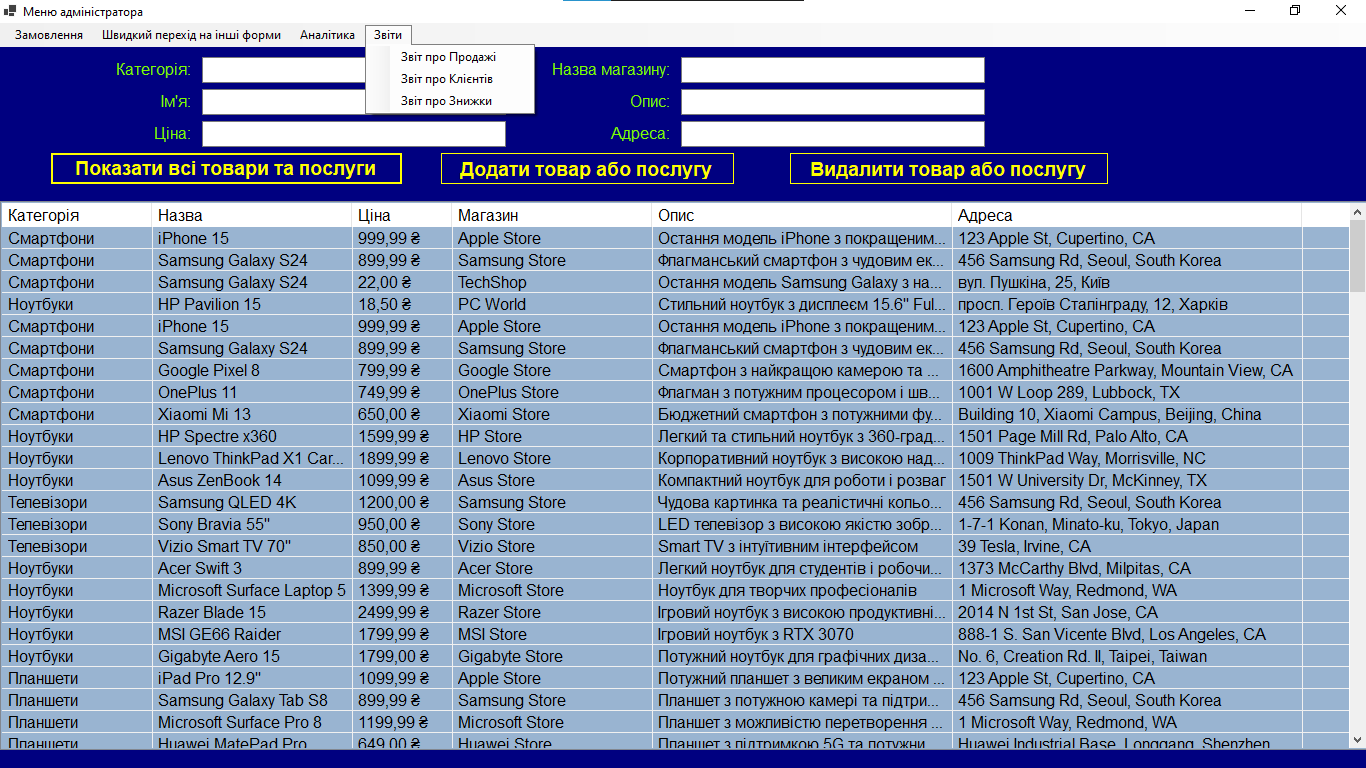


Рисунок Б.5.Форма Адміністратора

* Розділи:
* Управління товарами
* Додавання
* Редагування
* Видалення
* Миттєве оновлення бази
* Звіти
* Продажі за період
* Аналітика клієнтів
* Статистика знижок
* Замовлення
* Статуси замовлень
* Фільтрація
* Швидке реагування
* Експорт звітів

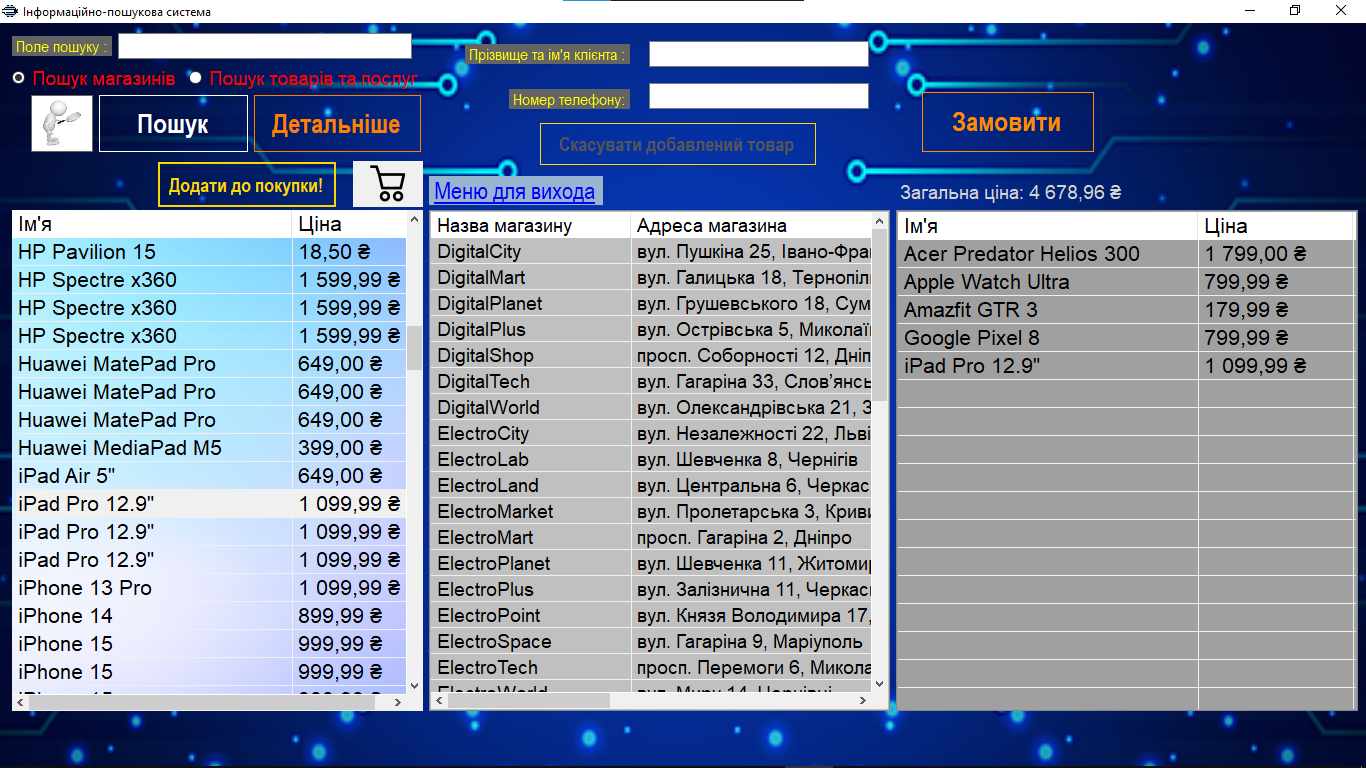


Рисунок Б.6. Форма для пошуку та замовлення

* Верхній блок
* Поле вибору режиму (Пошук магазину чи товарів)
* Пошукове поле з автозаповненням
* Кнопка пошуку
* Пошук магазинів
* Таблиця з даними товарів (назва, ціна, наявність)
* Фільтри для пошуку (за назвою, категорією, ціною, виробником)
* Кнопка “Додати до кошика” для обраних товарів
* Кошик
* Список обраних товарів
* Загальна сума замовлення
* Форма введення контактних даних
* Кнопка ‘Оформити замовлення’

Таким чином, всі ключові функції системи - пошук магазинів та товарів, перегляд кошика та оформлення замовлення - доступні в рамках єдиного інтерфейсу. Це дозволяє користувачам ефективно взаємодіяти з системою, не переходячи між різними сторінками.