**Форма № 17**

Міністерство освіти і науки України

**Національний університет «Запорізька політехніка»**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Кафедра програмних засобів\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(найменування кафедри)

**КУРСОВИЙ ПРОЄКТ**

**(РОБОТА)**

з дисципліни «Об’єктно-орієнтоване програмування»

(назва дисципліни)

на тему: «Цифровий ринок нерухомості»

Студентів 2 курсу КНТ-122 групи

спеціальності 121 Інженерія   
програмного забезпечення

освітня програма (спеціалізація) інженерія програмного забезпечення

\_\_\_\_\_\_\_\_\_\_\_\_Барабаш А. В.\_\_\_\_\_\_\_ \_\_\_\_

(прізвище та ініціали)

\_\_\_\_\_\_\_\_\_\_\_\_Васильєв М. В.\_\_\_\_\_\_\_\_\_\_

(прізвище та ініціали)

\_\_\_\_\_\_\_\_\_\_\_\_Коваленко В. П.\_\_\_\_\_\_\_\_\_

(прізвище та ініціали)

\_\_\_\_\_\_\_\_\_\_\_\_Коганті К. К.\_\_\_\_\_\_\_\_\_\_

(прізвище та ініціали)

\_\_\_\_\_\_\_\_\_\_\_\_Онищенко О. А.\_\_\_\_\_\_\_\_\_

(прізвище та ініціали)

Керівник доцент, к.т.н., Каплієнко Т. І.

(посада, вчене звання, науковий ступінь, прізвище та ініціали)

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

Кількість балів: \_\_\_\_\_\_\_\_\_\_Оцінка: ECTS \_\_\_\_\_

Члени комісії \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_   Каплієнко Т. І.

(підпис) (прізвище та ініціали)

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

(підпис) (прізвище та ініціали)

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

(підпис) (прізвище та ініціали

2023 рік

**Форма № 25**

Міністерство освіти і науки України

**Національний університет «Запорізька політехніка»**

( повне найменування закладу вищої освіти)

Інститут, факультет ІІРЕ, ФКНТ\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кафедра програмних засобів\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ступінь вищої освіти      бакалавр**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Спеціальність 121 Інженерія програмного забезпечення**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

(код і найменування)

Освітня програма (спеціалізація) Інженерія програмного забезпечення**\_\_\_\_\_\_\_**

(назва освітньої програми (спеціалізації)

**ЗАТВЕРДЖУЮ**

**Завідувач кафедри ПЗ, д.т.н, проф.**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_С.О. Субботін**

“\_\_\_\_” \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_\_року

З А В Д А Н Н Я

**НА КУРСОВИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)**

Барабаш А. В., Васильєв М. В., Коваленко В. П., Коганті К. К., Онищенко О. А.

(прізвище, ім’я, по батькові)

1. Тема проєкту (роботи) Цифровий ринок нерухомості

керівник проєкту (роботи)\_Каплієнко Тетяна Ігорівна, к.т.н., доцент ,

( прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від

2. Строк подання студентом проєкту (роботи)\_05 грудня 2023 року\_\_\_\_\_\_\_\_\_\_

3. Вихідні дані до проєкту (роботи) створити застосунок Цифрового ринку нерухомості для спрощення процесу пошуку та придбання нерухомості\_\_\_\_\_

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області; 2. Аналіз програмних засобів; 3. Основні рішення з реалізації компонентів системи; 4. Керівництво програміста; 5. Керівництво користувача; 6. Додатки.     
5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)\_\_

Слайди презентації\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**\_**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

6. Консультанти розділів проєкту (роботи)

|  |  |  |  |
| --- | --- | --- | --- |
| Розділ | Прізвище, ініціали та посада  консультанта | Підпис, дата | |
| завдання видав | прийняв  виконане завдання |
| 1-5 Основна частина | Каплієнко Т.І., доцент |  |  |
|  |  |  |  |
|  |  |  |  |

7. Дата видачі завдання\_\_14 вересня 2023 р.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**КАЛЕНДАРНИЙ ПЛАН**

|  |  |  |  |
| --- | --- | --- | --- |
| №  з/п | Назва етапів курсового  проєкту (роботи) | Строк виконання етапів проєкту  ( роботи ) | Примітка |
| 1. | Аналіз індивідуального завдання. | 1 тиждень |  |
| 2. | Аналіз програмних засобів, що будуть використовуватись в роботі. | 2 тиждень |  |
| 3. | Аналіз структур даних, що необхідно використати в курсової роботі. | 3 тиждень |  |
| 4. | Затвердження завдання | 4 тиждень |  |
| 5. | Вивчення можливостей програмної реалізації структур даних та інтерфейсу користувача. | 5-9 тиждень |  |
| 6. | Аналіз вимог до апаратних засобів | 9 тиждень |  |
| 7. | Розробка програмного забезпечення | 9-13 тиждень |  |
| 8. | Проміжний контроль | 10 тиждень | Розділи 1-2 ПЗ |
| 9. | Оформлення, відповідних пунктів пояснювальної записки. | 10-14 тиждень | Розділи 1-5 ПЗ |
| 10. | Захист курсової роботи. | 15 тиждень |  |
|  |  |  |  |
|  |  |  |  |

**Студент \_\_\_\_\_\_\_\_\_ \_\_\_** Барабаш А. В.**\_\_\_\_**

( підпис ) (прізвище та ініціали)

**Студент \_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_** Васильєв М. В.**\_\_\_\_\_**

( підпис ) (прізвище та ініціали)

**Студент \_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_** Коваленко В. П.**\_\_\_\_\_**

( підпис ) (прізвище та ініціали)

**Студент \_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_** Коганті К. К.**\_\_\_\_\_**

( підпис ) (прізвище та ініціали)

**Студент \_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_** Онищенко О. А.**\_\_\_\_\_**

( підпис ) (прізвище та ініціали)

**Керівник проєкту (роботи) \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_** Каплієнко Т.І.**\_\_\_\_**

( підпис ) (прізвище та ініціали)

реферат

Проєкт "Цифровий ринок нерухомості" - це комплексний застосунок, розроблений з використанням C# Winforms .NET. Застосунок призначений для полегшення процесу купівлі-продажу нерухомості в цифровому середовищі. Це зручна платформа, яка дозволяє користувачам переглядати різні оголошення про нерухомість, додавати власні оголошення та керувати інформацією свого профілю. Застосунок також включає функції для обробки автентифікації та авторизації користувачів, та реалізацію алгоритмів пошуку та сортування для покращення користувацького досвіду. Проєкт має на меті спростити процес здійснення операцій з нерухомістю, зробивши його більш доступним та ефективним як для покупців, так і для продавців.

Цифрова нерухомість, цифровий ринок, C# Winforms, Windows застосунок, курсовий проєкт, курсова робота, цифровий ринок нерухомості.

зміст

[реферат 4](#_Toc152182581)

[зміст 5](#_Toc152182582)

[перелік умовних позначень, символів, одиниць, скорочень і термінів 8](#_Toc152182583)

[вступ 9](#_Toc152182584)

[1 Аналіз предметної області 11](#_Toc152182585)

[1.1 Аналіз цифрової трансформації в галузі нерухомості, як основи предметної області 11](#_Toc152182586)

[1.2 Огляд існуючих методів вирішення завдання 12](#_Toc152182587)

[1.2 Огляд існуючих програм та сервісів для цифровізації ринку нерухомості 12](#_Toc152182588)

[1.2.1 Передмова 12](#_Toc152182589)

[1.2.2 Система «Zillow» 12](#_Toc152182590)

[1.2.3 Система «Realtor.com» 13](#_Toc152182591)

[1.2.4 Система «Airbnb» 15](#_Toc152182592)

[1.3 Висновки з першого розділу та постановка задачі 16](#_Toc152182593)

[2 Аналіз програмних засобів 18](#_Toc152182594)

[2.1 Огляд особливостей мови програмування 18](#_Toc152182595)

[2.2 Огляд особливостей обраного компілятору 18](#_Toc152182596)

[2.3 Огляд класової ієрархії 19](#_Toc152182597)

[2.4 Висновки з розділу 23](#_Toc152182598)

[3 основні рішення з реалізації компонентів системи 25](#_Toc152182599)

[3.1 Основні рішення щодо розроблених класів 25](#_Toc152182600)

[3.2 Основні розроблені алгоритми 37](#_Toc152182601)

[3.3 Основні рішення щодо розробки інтерфейсу 37](#_Toc152182602)

[3.4 Основні рішення щодо роботи з даними 38](#_Toc152182603)

[3.5 Обробка виключних ситуацій 39](#_Toc152182604)

[4 керівництво програміста 50](#_Toc152182605)

[4.1 Призначення та умови застосування програми 50](#_Toc152182606)

[4.1.1 Призначення програми 50](#_Toc152182607)

[4.1.2 Функції програми 50](#_Toc152182608)

[4.1.3 Умови застосування програми 50](#_Toc152182609)

[4.2 Характеристика програми 51](#_Toc152182610)

[4.2.1 Структура програми 51](#_Toc152182611)

[4.3 Звертання до програми 53](#_Toc152182612)

[4.4 Вхідні та вихідні дані 54](#_Toc152182613)

[4.4.1 Вхідні дані 54](#_Toc152182614)

[4.4.2 Вихідні дані 54](#_Toc152182615)

[4.5 Повідомлення 54](#_Toc152182616)

[5 керівництво користувача 56](#_Toc152182617)

[5.1 Призначення програми 56](#_Toc152182618)

[5.2 Умови виконання програми 56](#_Toc152182619)

[5.2.1 Апаратні вимоги програми 56](#_Toc152182620)

[5.2.2 Вимоги до користувача 57](#_Toc152182621)

[5.3 Виконання програми 57](#_Toc152182622)

[5.3.1 Запуск програми 57](#_Toc152182623)

[5.3.2 Виконання роботи з програмою 57](#_Toc152182624)

[5.4 Повідомлення користувачу 58](#_Toc152182625)

[5.5 Довідка програми 65](#_Toc152182626)

[Висновки 66](#_Toc152182627)

[ПЕРЕЛІК ПОСИЛАНЬ 68](#_Toc152182628)

[Додаток А Діаграма класів 69](#_Toc152182629)

[ЯКЩО Є 69](#_Toc152182630)

[Додаток Б Код Програми 70](#_Toc152182631)

[Б1 – FormMenu 70](#_Toc152182632)

[Б2 - FormAddedListing 74](#_Toc152182633)

[Б3 – FormAddListing 74](#_Toc152182634)

[Б4 – FormBoughtListing 77](#_Toc152182635)

[Б5 – FormBuyListing 78](#_Toc152182636)

[Б6 – FormCodeConfirmation 80](#_Toc152182637)

[Б7 – FormDevelopers 81](#_Toc152182638)

[Б8 – FormHelp 82](#_Toc152182639)

[Б9 – FormLogin 82](#_Toc152182640)

[Б10 – FormMarket 86](#_Toc152182641)

[Б11 – FormProfile 92](#_Toc152182642)

[Б12 – FormRestore 100](#_Toc152182643)

[Б13 – FormSignUp 102](#_Toc152182644)

[Б14 – FormWelcome 107](#_Toc152182645)

[Б15 – Advertisement 108](#_Toc152182646)

[Б16 – AlgorithmManager 108](#_Toc152182647)

[Б17 – ExceptionManager 114](#_Toc152182648)

[Б18 – FileHandler 115](#_Toc152182649)

[Б19 – GlobalData 119](#_Toc152182650)

[Б20 – LoginManager 119](#_Toc152182651)

[Б21 – User 122](#_Toc152182652)

[Додаток В Слайди презентації 124](#_Toc152182653)

перелік умовних позначень, символів, одиниць, скорочень і термінів

ООП - об’єктно-орієнтоване програмування.

SQL – Structured Query Language.

CSV – Comma-Separated Values

JSON – JavaScript Object Notation

вступ

"Цифровий ринок нерухомості" - це цифрова платформа, покликана спростити процес купівлі-продажу нерухомості. В епоху цифрової трансформації для галузей стає все більш важливим адаптуватися і розвиватися, щоб відповідати мінливим потребам і очікуванням споживачів. Зокрема, галузь торгівлі нерухомістю повільно цифровізується, і багато транзакцій все ще відбуваються традиційними, часто неефективними методами.

Потреба в цифровій платформі для операцій з нерухомістю очевидна. Традиційні методи купівлі-продажу нерухомості можуть забирати багато часу, бути складними і схильними до помилок. Вони часто вимагають фізичної присутності, що не завжди можливо або зручно. Крім того, їм бракує прозорості та ефективності, які можуть запропонувати цифрові платформи.

Проєкт "Цифровий ринок нерухомості" має на меті вирішити ці проблеми шляхом створення цифрової платформи для операцій з нерухомістю. Ця платформа розроблена таким чином, щоб бути зручною, ефективною та прозорою, що робить її цінним інструментом як для покупців, так і для продавців. Вона пропонує широкий спектр функцій, включаючи можливість переглядати різні оголошення про нерухомість, додавати нові оголошення, керувати інформацією про профіль користувача, а також здійснювати автентифікацію та авторизацію користувачів.

Незважаючи на те, що вже існують цифрові платформи для операцій з нерухомістю, проєкт "Цифровий ринок нерухомості" вирізняється своїми унікальними можливостями та дизайном, орієнтованим на користувача. Він включає в себе передові технології та методології, такі як алгоритми пошуку та сортування, для покращення користувацького досвіду. Продукт також містить надійну систему обробки виключних ситуацій для забезпечення стабільності та надійності роботи застосунку.

Насамкінець, проєкт "Цифровий ринок нерухомості" є значним кроком на шляху до діджиталізації ринку нерухомості. Він є свідченням потенціалу технологій у трансформації традиційних галузей і підвищенні їхньої зручності та ефективності. Оскільки цифровий ландшафт невпинно розвивається, проєкт "Цифровий ринок нерухомості" готовий стати лідером у революційних змінах в індустрії нерухомості.

1 Аналіз предметної області

Індустрія нерухомості - одна з найбільш традиційних галузей економіки, багато процесів у якій досі відбуваються класичними, часто неефективними методами. Поява цифрових технологій спричинила хвилю змін у галузі, що призвело до розвитку цифрових платформ для операцій з нерухомістю. Однак, незважаючи на досягнутий прогрес, галузь нерухомості ще далека від повної цифровізації. Існує значна потреба в цифрових платформах, які можуть спростити процес купівлі-продажу нерухомості, зробивши його більш ефективним і зручним для користувачів.

1.1 Аналіз цифрової трансформації в галузі нерухомості, як основи предметної області

Цифрова трансформація в галузі нерухомості - це складний процес, який передбачає інтеграцію цифрових технологій у традиційні методи роботи з нерухомістю. Ця трансформація зумовлена необхідністю підвищити ефективність, зменшити витрати та покращити клієнтський досвід. Вона передбачає використання передових технологій, таких як штучний інтелект, машинне навчання та блокчейн, для автоматизації процесів, підвищення точності та забезпечення прозорості.

Одним із ключових напрямків цифрової трансформації є розробка цифрових платформ для операцій з нерухомістю. Ці платформи забезпечують віртуальний простір, де покупці та продавці можуть переглядати списки об'єктів нерухомості, додавати нові оголошення та здійснювати транзакції. Вони пропонують широкий спектр функцій, включаючи алгоритми пошуку та сортування, автентифікацію та авторизацію користувачів, а також управління профілем користувача.

Однак, незважаючи на успіхи цифрової трансформації, все ще існує багато проблем, які потребують вирішення. Однією з головних проблем є відсутність стандартизації цифрових платформ для операцій з нерухомістю. Відсутність стандартизації може призвести до неефективності та неузгодженості, що ускладнює навігацію на платформі як для покупців, так і для продавців.

1.2 Огляд існуючих методів вирішення завдання

У відповідь на ці виклики було розроблено кілька цифрових платформ для операцій з нерухомістю. Ці платформи пропонують різні рішення проблем, з якими стикається галузь нерухомості. Однак кожна платформа має свій власний набір функцій і можливостей, що призводить до відсутності уніфікації.

**1.2 Огляд існуючих програм та сервісів для цифровізації ринку нерухомості**

1.2.1 Передмова

У цьому розділі ми розглянемо три існуючі цифрові платформи для операцій з нерухомістю: Zillow, Realtor.com та Airbnb. Ці платформи пропонують широкий спектр можливостей і функцій, і вони широко використовуються покупцями і продавцями. Однак вони також мають свої переваги та недоліки.

1.2.2 Система «Zillow»

Zillow - це широко відома цифрова платформа для операцій з нерухомістю. Вона надає всеосяжну базу даних оголошень про нерухомість, алгоритми пошуку та сортування, автентифікацію та авторизацію користувачів, а також управління профілем користувача.

Переваги:

- Повна база даних об'єктів нерухомості;

- Просунуті алгоритми пошуку та сортування;

- Зручний інтерфейс;

- Детальні описи та фотографії об'єктів нерухомості;

- Автентифікація та авторизація користувачів.

Недоліки:

- Відсутність прозорості в процесі укладання угоди;

- Складність у розумінні ціноутворення на нерухомість;

- Складний інтерфейс для деяких користувачів.

Роботу програми продемонстровано на рисунку 1.1.

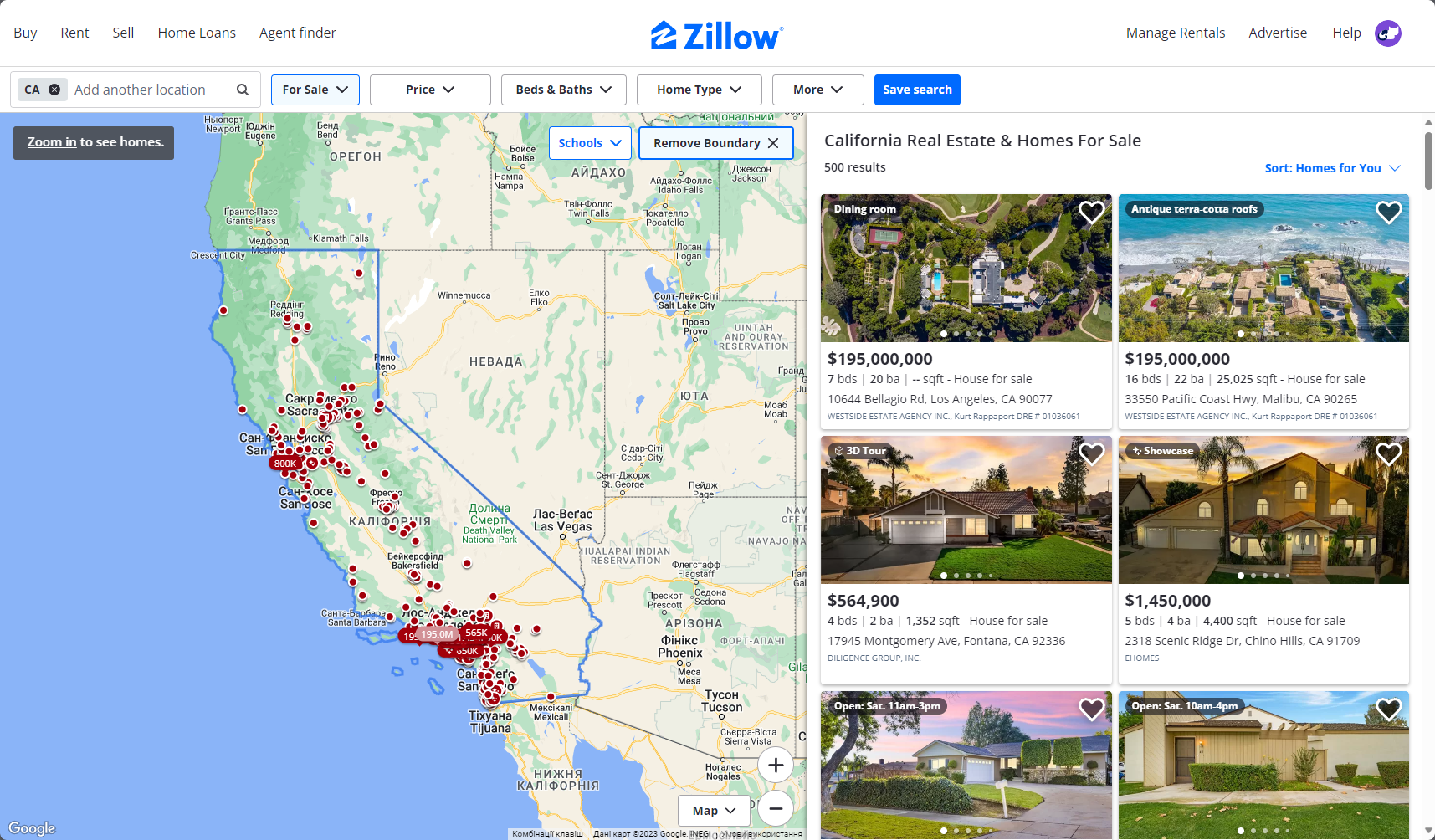


Рисунок 1.1 – Робота програми «Zillow»

**1.2.3** Система «Realtor.com»

Realtor.com - популярна цифрова платформа для операцій з нерухомістю. Вона пропонує широкий спектр функцій, включаючи повну базу даних оголошень про нерухомість, алгоритми пошуку та сортування, автентифікацію та авторизацію користувачів, а також управління профілем користувача.

Переваги:

- Повна база даних об'єктів нерухомості;

- Просунуті алгоритми пошуку та сортування;

- Автентифікація та авторизація користувачів;

- Детальні описи та фотографії об'єктів нерухомості.

Недоліки:

- Складний інтерфейс;

- Відсутність зручності для користувача;

- Складність у розумінні ціноутворення на нерухомість.

Роботу програми продемонстровано на рисунку 1.2.

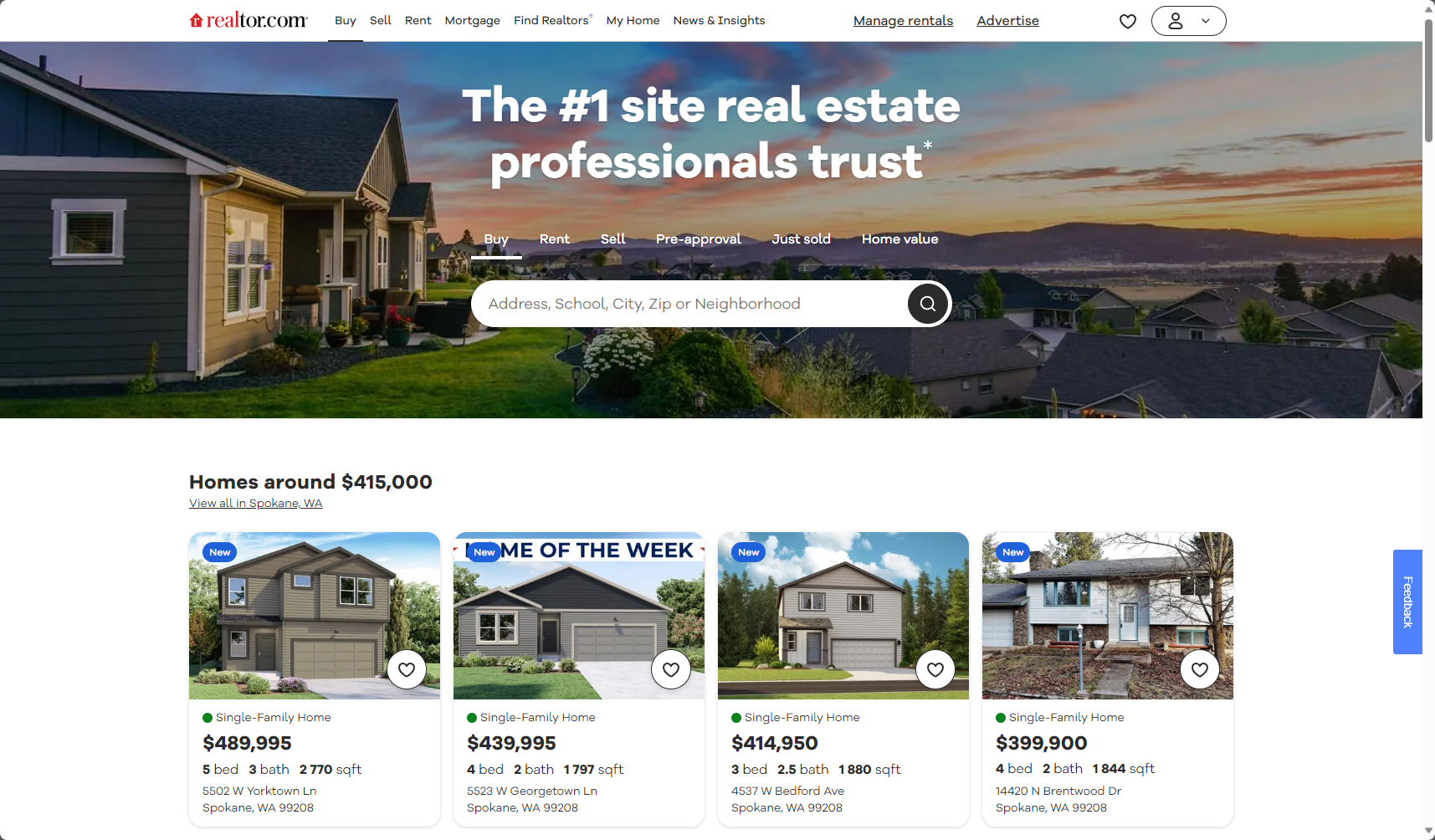


Рисунок 1.2 – Робота програми “Realtor.com”

1.2.4 Система «Airbnb»

Airbnb - це цифрова платформа для операцій з нерухомістю, яка фокусується на короткостроковій оренді. Вона пропонує широкий спектр функцій, включаючи всеосяжну базу даних оголошень про нерухомість, алгоритми пошуку та сортування, автентифікацію та авторизацію користувачів, а також управління профілем користувача.

Переваги:

- Повна база даних об'єктів нерухомості;

- Просунуті алгоритми пошуку та сортування;

- Автентифікація та авторизація користувачів;

- Детальні описи та фотографії об'єктів нерухомості.

Недоліки:

- Відсутність прозорості в процесі укладання угоди;

- Складність у розумінні ціноутворення на нерухомість;

- Складний інтерфейс для деяких користувачів.

Роботу програми продемонстровано на рисунку 1.3.

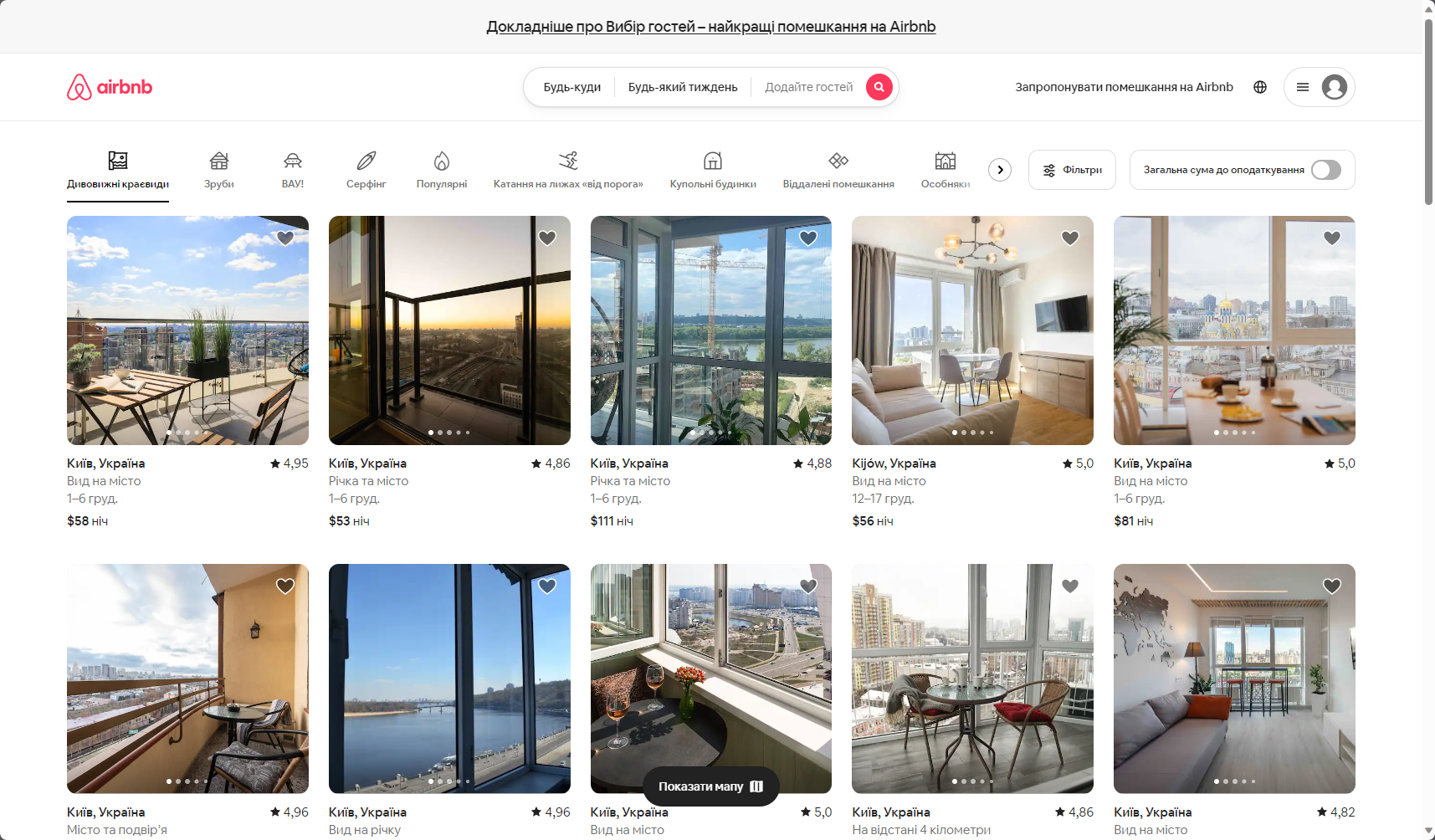


Рисунок 1.3 – Робота програми “Airbnb”

1.3 Висновки з першого розділу та постановка задачі

З нашого аналізу можна зробити висновок, що, незважаючи на те, що існують цифрові платформи для операцій з нерухомістю, всі вони мають свої переваги та недоліки. Найважливішою перевагою цих платформ є широкий спектр можливостей і функцій, які полегшують покупцям і продавцям проведення транзакцій. Однак їхнім головним недоліком є відсутність стандартизації та прозорості, що може призвести до неефективності та неузгодженості.

Виходячи з цих висновків, завданням нашого проєкту є розробка застосунку, який вирішить ці проблеми. Програма буде пропонувати широкий спектр функцій, включаючи повну базу даних оголошень про нерухомість, алгоритми пошуку та сортування, автентифікацію та авторизацію користувачів, а також управління профілем користувача. Він також забезпечить прозорість процесу укладання угод, що полегшить покупцям і продавцям розуміння цін на нерухомість і проведення транзакцій. Застосунок також міститиме функцію обробки виняткових ситуацій, що має вирішальне значення для підтримання стабільності та надійності роботи програми.

На завершення, проєкт "Цифровий ринок нерухомості" є значним кроком на шляху до цифровізації ринку нерухомості. Він є свідченням потенціалу технологій у трансформації традиційних галузей і робить їх більш зручними та ефективними. Оскільки цифровий ландшафт продовжує розвиватися, проєкт "Цифровий ринок нерухомості" готовий очолити революцію в індустрії нерухомості.

2 Аналіз програмних засобів

У цьому розділі подано детальний аналіз процесів розробки програмного забезпечення з акцентом на мові програмування та компіляторі, ієрархії класів та бібліотеках, що використовуються у проєкті. Він має на меті забезпечити комплексне розуміння процесу розробки програмного забезпечення та ролі різних елементів програмування в рамках проєкту.

2.1 Огляд особливостей мови програмування

C# (вимовляється як C Sharp) - це сучасна об'єктно-орієнтована мова програмування, розроблена компанією Microsoft. Вона є частиною платформи .NET, яка являє собою фреймворк для створення застосунків, що працюють на платформі Windows. C# є суворо типізованою мовою, що означає, що тип змінної перевіряється під час компіляції. Ця особливість допомагає запобігти багатьом поширеним помилкам програмування.

C# також підтримує різноманітні парадигми програмування, включаючи процедурне, об'єктно-орієнтоване та функціональне програмування. Вона надає багатий набір функцій, включаючи очистку сміття, приведення типів та обробку винятків. Мова також підтримує асинхронне програмування, що дозволяє ефективно виконувати завдання, які можуть зайняти багато часу.

Інформацію здебільшого взято з наступного джерела – [Джерело](https://learn.microsoft.com/en-us/dotnet/csharp/)

2.2 Огляд особливостей обраного компілятору

Компілятор C# є частиною платформи .NET і використовується для перетворення коду мовою C# у проміжну мову Microsoft Intermediate Language (MSIL), яка потім може бути виконана середовищем виконання .NET. Компілятор надає різноманітні функції, включаючи перевірку помилок, оптимізацію та генерацію коду. Він також підтримує різноманітні опції компіляції, за допомогою яких можна керувати його поведінкою.

2.3 Огляд класової ієрархії

Ієрархія класів у проєкті "Цифровий ринок нерухомості" в основному базується на фреймворку .NET. У рамках проєкту використовується декілька вбудованих класів з фреймворку .NET, які організовано у декількох просторах імен. Нижче наведено структуру класів, що використовуються проєктом.

**Простір імен System**

Простір імен System містить основоположні та базові класи, які визначають загальновживані типи даних значень та посилань, події та обробники подій, інтерфейси, атрибути та обробки винятків.

**Простір імен System.Collections.Generic**

Простір імен System.Collections.Generic містить інтерфейси та класи, що визначають загальні колекції, які дозволяють користувачам створювати суворо типізовані контейнери, що забезпечують кращу безпеку типів та продуктивність, ніж нестандартні суворо типізовані колекції.

**Простір імен System.ComponentModel**

Простір імен System.ComponentModel надає класи, які використовуються для реалізації поведінки компонентів та елементів керування під час виконання та проєктування.

**Простір імен System.Data**

Простір імен System.Data містить класи, які представляють архітектуру ADO.NET, включаючи класи, які працюють з базами даних, наборами даних, таблицями даних тощо.

**Простір імен System.Drawing**

Простір імен System.Drawing надає доступ до графічної функціональності GDI+.

**Простір імен System.Linq**

Простір імен System.Linq надає класи та інтерфейси, які підтримують запити, що використовують Language-Integrated Query (LINQ).

**Простір імен System.Text**

Простір імен System.Text містить класи, що представляють кодування символів ASCII та Unicode; абстрактні базові класи для перетворення блоків символів у різні кодування та у зворотній бік; а також допоміжний клас, який маніпулює та форматує об'єкти String без створення проміжних об'єктів String.

**Простір імен System.Threading.Tasks**

Простір імен System.Threading.Tasks надає типи, які спрощують роботу з написання паралельного та асинхронного коду.

**Простір імен System.Windows.Forms**

Простір імен System.Windows.Forms містить класи для створення Windows-застосунків, які використовують всі переваги багатого користувацького інтерфейсу, доступного в операційній системі Microsoft Windows.

**Простір імен System.Diagnostics.Eventing.Reader**

Простір імен System.Diagnostics.Eventing.Reader містить класи, які використовуються для читання журналів подій та файлів журналів трасування подій.

**Простір імен System.Xml**

Простір імен System.Xml містить класи для роботи з XML документами та даними.

**Простір імен System.Xml.Linq**

Простір імен System.Xml.Linq містить класи для роботи з XML-даними у форматі LINQ до XML-дерев.

**Простір імен System.Xml.Schema**

Простір імен System.Xml.Schema містить класи для роботи з XML-схемами.

**Простір імен System.Reflection**

Простір імен System.Reflection містить класи, які отримують інформацію про збірки, модулі, члени, параметри та інші сутності у керованому коді шляхом перегляду їх метаданих.

**Простір імен System.Linq.Expressions**

Простір імен System.Linq.Expressions надає класи, які представляють код у вигляді структур даних, якими можна маніпулювати та виконувати.

**Простір імен System.Windows.Forms.VisualStyles.VisualStyleElement**

Простір імен System.Windows.Forms.VisualStyles.VisualStyleElement надає класи, які представляють елементи візуального стилю.

**Простір імен System.Globalization**

Простір імен System.Globalization містить класи, які визначають інформацію, пов'язану з культурними особливостями, зокрема мову, країну/регіон, календарі та національні особливості форматування дат і сортування рядків.

**Простір імен System.Security.Cryptography**

Простір імен System.Security.Cryptography містить класи, які реалізують криптографічні алгоритми, зокрема набір оболонок для некерованих криптографічних бібліотек.

**Простір імен System.Net.Mail**

Простір імен System.Net.Mail надає класи для надсилання пошти та обробки поштових подій.

**Простір імен Newtonsoft.Json**

Простір імен Newtonsoft.Json містить класи для серіалізації та десеріалізації об'єктів у та з JSON.

**Простір імен System.Configuration**

Простір імен System.Configuration містить класи для роботи з конфігураційними файлами для клієнтських застосунків.

**Простір імен System.IO**

Простір імен System.IO містить класи для читання та запису в синхронні та асинхронні потоки, для маніпулювання файлами та каталогами, а також для обробки виключних ситуацій, які можуть виникнути під час роботи з файлами.

2.4 Висновки з розділу

Отже, проєкт "Цифровий ринок нерухомості" розроблено з використанням мови C# та платформи .NET, яка забезпечує надійне та багатофункціональне середовище для розробки програмного забезпечення. Проєкт використовує декілька вбудованих класів з фреймворку .NET та сторонніх бібліотек для реалізації необхідного функціоналу. Вибір C# та .NET обумовлений їх надійністю, універсальністю та широкою підтримкою об'єктно-орієнтованого програмування.

3 основні рішення з реалізації компонентів системи

У цьому розділі будуть розглянуті рішення, прийняті під час розробки компонентів системи. Основна увага буде приділена класам, які були розроблені командою, і буде надано загальний огляд рішень, прийнятих щодо реалізації цих класів.

3.1 Основні рішення щодо розроблених класів

Класи, розроблені для цього проєкту, були покликані полегшити різні функціональні можливості застосунку. Вони були обрані на основі їхньої здатності виконувати конкретні завдання та сумісності із загальною архітектурою застосунку. Класи були розроблені так, щоб працювати в гармонії один з одним, утворюючи цілісну систему, яка дозволяє застосунку функціонувати як єдине ціле.

**Клас FormAddedListing**

Клас FormAddedListing - це форма, яка відображає деталі оголошення про нерухомість, доданого користувачем. Він відповідає за відображення назви, опису, адреси, ціни та зображення оголошення. Клас також відображає дату останнього перегляду оголошення.

Клас має конструктор, який приймає об'єкт Advertisement як параметр. Цей об'єкт містить деталі оголошення, які потім відображаються на формі. Клас також включає об'єкт toolTipDescription, який використовується для відображення повного опису оголошення, коли користувач наводить курсор на скорочений опис.

Клас також включає блок try-catch, який обробляє будь-які винятки, що можуть виникнути під час завантаження форми. Якщо виникає виняток, він перехоплюється і передається методу ExceptionManager.HandleException, який відображає користувачеві повідомлення про помилку.

**Клас FormAddListing**

Клас FormAddListing - це форма, яка дозволяє користувачам додавати нові оголошення про нерухомість. Вона містить поля для введення назви, опису, адреси та ціни оголошення, а також можливість завантаження зображення. Форма також містить кнопку для додавання оголошення та кнопку для скасування операції.

Клас має приватне поле selectedImagePath, в якому зберігається шлях до вибраного зображення. Це поле використовується для зберігання зображення, яке користувач обирає для завантаження.

Конструктор класу ініціалізує форму і встановлює початковий текст мітки lblImageName "Виберіть зображення".

Метод btnCancel\_Click закриває форму при натисканні на кнопку скасування.

Метод btnAdd\_Click відповідає за додавання нового оголошення. Спочатку він перевіряє, чи всі поля заповнені і чи ціна не дорівнює нулю. Якщо будь-яка з цих умов не виконана, він виводить повідомлення про помилку і повертається. Якщо всі умови виконано, система просить користувача підтвердити операцію. Якщо користувач підтверджує, він створює новий об'єкт Advertisement з введеними даними, додає його до списку GlobalData.AvailableListings та до списку LoginManager.CurrentUser.AddedListings. Після цього відображається повідомлення про успішне завершення і форма закривається.

Метод txtDescription\_TextChanged оновлює поле lblDescriptionCharactersCount поточною кількістю символів у полі опису. Він також змінює колір мітки на основі кількості символів.

Метод btnAddImage\_Click дозволяє користувачеві вибрати зображення для завантаження. Він відкриває діалогове вікно, яке дозволяє користувачеві вибрати файл зображення. Якщо вибрано зображення, він зберігає шлях до нього у полі selectedImagePath і відображає назву зображення у мітці lblImageName. Якщо зображення не вибрано, він очищає поле selectedImagePath і встановлює мітку lblImageName у значення "Виберіть зображення".

Клас також включає блок try-catch у кожному методі, який обробляє будь-які винятки, що можуть виникнути під час виконання методу. Якщо виникає виняток, він перехоплюється і передається методу ExceptionManager.HandleException, який виводить користувачеві повідомлення про помилку.

**Клас FormBoughtListing**

Клас FormBoughtListing - це форма, яка відображає деталі оголошення про нерухомість, яке було придбано користувачем. Він відповідає за відображення назви, опису, адреси, ціни та зображення оголошення. Він також відображає ім'я продавця та ім'я поточного користувача.

Клас має конструктор, який приймає об'єкт Advertisement як параметр. Цей об'єкт містить деталі оголошення, які потім відображаються на формі. Клас також включає об'єкт toolTipDescription, який використовується для відображення повного опису оголошення, коли користувач наводить курсор на скорочений опис.

Клас також включає блок try-catch, який обробляє будь-які винятки, що можуть виникнути під час завантаження форми. Якщо виникає виняток, він перехоплюється і передається методу ExceptionManager.HandleException, який виводить користувачеві повідомлення про помилку.

**Клас FormBuyListing**

Клас FormBuyListing - це форма, яка дозволяє користувачам купувати оголошення про нерухомість. Вона надає користувачеві поля для перегляду назви, опису, адреси, ціни та зображення оголошення, а також імені продавця. Форма також містить кнопку для покупки оголошення та кнопку для закриття форми.

Клас має приватне поле listing, в якому зберігається об'єкт Advertisement, що представляє оголошення, яке купується. Він також має приватне поле formMarket, в якому зберігається об'єкт FormMarket, що представляє форму ринку.

Конструктор класу ініціалізує форму і встановлює початковий текст різних міток у відповідні властивості об'єкта listing. Він також встановлює зображення елемента керування pictureBoxListingImage до зображення об'єкта listing.

Метод btnClose\_Click закриває форму при натисканні на кнопку закриття.

Метод btnBuy\_Click відповідає за покупку оголошення. Спочатку він перевіряє, чи користувач увійшов в систему. Якщо користувач не зареєстрований або не авторизований, він виводить повідомлення про помилку і завершує роботу. Якщо користувач авторизований, він перевіряє, чи є він продавцем оголошення. Якщо користувач є продавцем, відображається повідомлення про помилку і відбувається завершення роботи. Якщо користувач не є продавцем, він додає об'єкт listing до списку BoughtListings об'єкта CurrentUser і видаляє об'єкт listing зі списку AvailableListings об'єкта FormMarket. Після цього він закриває форму.

Клас також включає блок try-catch у кожному методі, який обробляє будь-які винятки, що можуть виникнути під час виконання методу. Якщо виникає виняток, він перехоплюється і передається методу ExceptionManager.HandleException, який виводить користувачеві повідомлення про помилку.

**Клас FormDevelopers**

Клас FormDevelopers - це форма, яка відображає інформацію про розробників застосунку. Вона містить кнопки, які дозволяють користувачеві переглянути репозиторій проєкту на GitHub та ліцензію проєкту.

Конструктор класу ініціалізує форму і встановлює властивість Text форми в порожній рядок, а властивість ControlBox форми в значення false. Це приховує рядок заголовка і блок управління форми, роблячи її схожою на діалогове вікно.

Метод styledButtonGithubLink\_Click відповідає за відкриття репозиторію проєкту на GitHub у веб-браузері користувача за замовчуванням. Він використовує метод System.Diagnostics.Process.Start для відкриття URL репозиторію GitHub.

Метод styledButtonLicense\_Click відповідає за відкриття ліцензії проєкту у браузері користувача за замовчуванням. Він використовує метод System.Diagnostics.Process.Start для відкриття URL-адреси ліцензії.

Клас також включає блок try-catch у кожному методі, який обробляє будь-які винятки, що можуть виникнути під час виконання методу. Якщо виникає виняток, він перехоплюється і передається методу ExceptionManager.HandleException, який виводить користувачеві повідомлення про помилку.

**Клас FormHelp**

Клас FormHelp - це форма, яка надає довідку для програми. Вона містить декілька підказок, які надають інформацію про різні розділи програми.

Конструктор класу ініціалізує форму і встановлює властивість Text форми у порожній рядок, а властивість ControlBox форми у значення false. Це приховує рядок заголовка і блок управління форми, роблячи її схожою на діалогове вікно.

Форма містить декілька текстових блоків, які надають інформацію про різні розділи програми. Кожен блок містить заголовок і абзац, який описує відповідний розділ. Заголовки та абзаци відформатовані з використанням різних шрифтів і розмірів, щоб їх можна було легко розрізнити.

Форма також містить елемент управління MainTitle, який використовується для відображення заголовка форми. Елемент управління MainTitle є користувацьким елементом управління, який використовується в усьому застосунку для відображення заголовка кожної форми.

Форма не містить кнопок або інших інтерактивних елементів управління. Вона призначена лише для ознайомлення з інформацією, яка надається користувачеві.

**Клас FormLogin**

Клас FormLogin - це форма, яка дозволяє користувачам увійти до програми. Вона містить поля для введення імені користувача та пароля, а також кнопки для входу, реєстрації та перемикання видимості пароля.

Конструктор класу ініціалізує форму і встановлює властивість PasswordChar текстового поля txtPassword у значення '\*', що приховує введений пароль.

Статичне поле LoginAttempts використовується для відстеження кількості невдалих спроб входу.

Метод ValidateLogin використовується для перевірки облікових даних користувача. Він перевіряє, чи збігаються введені ім'я користувача та пароль з будь-яким користувачем у списку GlobalData.Users. Якщо збіг знайдено, повертається відповідний об'єкт User. Якщо збігу не знайдено, повертається null.

Метод btnLogin\_Click відповідає за обробку процесу входу в систему. Спочатку він перевіряє, чи поля імені користувача та пароля не порожні і чи відповідають вони вимогам до довжини. Якщо вони не відповідають вимогам, він виводить повідомлення про помилку і інкрементує поле LoginAttempts. Якщо кількість спроб входу перевищує 3, програма виводить повідомлення і скасовує процес входу. Якщо ім'я користувача та пароль відповідають вимогам, викликається метод ValidateLogin для перевірки облікових даних. Якщо перевірка пройшла успішно, він встановлює поле CurrentUser класу LoginManager на користувача, що пройшов перевірку, додає списки користувача до списку AvailableListings класу GlobalData і встановлює поле IsLoggedIn класу LoginManager у значення true.

Метод btnSignUp\_Click відповідає за обробку процесу реєстрації. Він відкриває форму FormSignUp, де користувач може ввести свої дані для створення нового облікового запису. Якщо реєстрація пройшла успішно, він додає нового користувача до списку Users класу GlobalData, встановлює поле CurrentUser класу LoginManager на нового користувача і встановлює поле IsLoggedIn класу LoginManager на true.

Метод btnTogglePasswordVisibility\_Click відповідає за перемикання видимості пароля. Він змінює властивість PasswordChar текстового поля txtPassword між '\*' і '\0', що приховує і показує введений пароль відповідно.

Клас також містить блок try-catch у кожному методі, який обробляє будь-які винятки, що можуть виникнути під час виконання методу. Якщо виникає виняток, він перехоплюється і передається методу ExceptionManager.HandleException, який виводить користувачеві повідомлення про помилку.

**Клас FormMarket**

Клас FormMarket - це форма, яка відображає список оголошень про продаж нерухомості. Він забезпечує представлення таблиці даних, яка відображає перелік оголошень, поле пошуку, яке дозволяє користувачеві шукати оголошення, і випадаючі меню, які дозволяють користувачеві сортувати оголошення за назвою, продавцем або ціною.

Конструктор класу ініціалізує форму і встановлює властивість DataSource таблиці даних dataGridViewListings на список AvailableListings класу GlobalData. Вона також встановлює шрифт і відступи для заголовків стовпців і комірок подання сітки даних, а також встановлює початковий вибраний індекс спадних списків сортування на 0.

Методи AddListing та RemoveListing використовуються для додавання та видалення списків зі списку AvailableListings класу GlobalData. Вони оновлюють властивість DataSource таблиці даних dataGridViewListings для відображення змін у списку AvailableListings.

Метод btnSearch\_Click відповідає за процес пошуку. Він фільтрує список AvailableListings на основі пошукового запиту і встановлює властивість DataSource таблиці даних dataGridViewListings на відфільтрований список.

Метод txtSearch\_Enter відповідає за обробку події натискання клавіші Enter у текстовому полі txtSearch. Він очищає текст підказки та змінює колір переднього плану на чорний, коли вводиться текстовий рядок.

Метод dataGridViewListings\_CellDoubleClick відповідає за обробку події подвійного кліку по комірці таблиці даних dataGridViewListings. Він відкриває форму FormBuyListing при подвійному кліку по комірці.

Метод txtSearch\_KeyDown відповідає за обробку події натискання клавіші у текстовому полі txtSearch. Він викликає метод btnSearch\_Click при натисканні клавіші Enter.

Методи reshowHint та hideHint використовуються для показу та приховування підказок у випадаючих меню.

Методи dropdownSortByName\_DropDown, dropdownSortBySeller\_DropDown і dropdownSortByPrice\_DropDown відповідають за обробку події випадаючого меню. Вони приховують підказку у відповідному випадаючому меню при його відкритті.

Методи dropdownSortByName\_DropDownClosed, dropdownSortBySeller\_DropDownClosed і dropdownSortByPrice\_DropDownClosed відповідають за обробку події закриття випадаючого меню. Вони повторно показують підказку у відповідному випадаючому меню, якщо не вибрано жодного елемента.

Методи dropdownSortByName\_SelectedIndexChanged, dropdownSortByPrice\_SelectedIndexChanged і dropdownSortBySeller\_SelectedIndexChanged відповідають за обробку події зміни індексу випадаючого меню. Вони сортують список AvailableListings на основі вибраного елемента відповідного випадаючого меню і повторно показують підказки в інших випадаючих меню.

Клас також включає блок try-catch у кожному методі, який обробляє будь-які винятки, що можуть виникнути під час виконання методу. Якщо виникає виняток, він перехоплюється і передається методу ExceptionManager.HandleException, який виводить користувачеві повідомлення про помилку.

**Клас FormProfile**

Клас FormProfile - це форма, яка відображає профіль поточного користувача. Він надає користувачеві поля для перегляду та редагування його імені користувача, електронної пошти та пароля, а також відображення таблиці даних для перегляду оголошень, які користувач купив або додав.

Конструктор класу ініціалізує форму і встановлює властивість DataSource таблиці даних dataGridViewBoughtListings і dataGridViewAddedListings на списки BoughtListings і AddedListings об'єкта CurrentUser класу LoginManager. Він також встановлює шрифт і відступи заголовків стовпців і комірок таблиці даних, а також встановлює початкові значення текстових полів txtUsername, txtEmail і txtPassword на ім'я, електронну пошту і пароль об'єкта CurrentUser класу LoginManager відповідно.

Метод OpenChildForm використовується для відкриття дочірньої форми на панелі pnlMainContainer форми FormProfile. Він приховує поточну активну форму, додає нову форму до панелі pnlMainContainer і встановлює нову форму як активну.

Метод btnTogglePasswordVisibility\_Click відповідає за перемикання видимості пароля. Він змінює властивість PasswordChar текстового поля txtPassword між '\*' і '\0', що приховує і показує введений пароль відповідно.

Метод btnSaveChanges\_Click відповідає за процес збереження змін. Спочатку він перевіряє, чи відповідає нове ім'я користувача, адреса електронної пошти та пароль встановленим вимогам. Якщо вони не відповідають вимогам, він виводить повідомлення про помилку. Якщо вони відповідають усім вимогам, він оновлює об'єкт CurrentUser класу LoginManager новим іменем користувача, електронною поштою та паролем, а також скидає тип відображення поля txtPassword.

Методи btnRefreshBoughtListings\_Click та btnClearBoughtListings\_Click відповідають за обробку кнопок оновлення та очищення куплених списків. Вони оновлюють властивість DataSource таблиці даних dataGridViewBoughtListings для відображення змін у списку BoughtListings об'єкта CurrentUser класу LoginManager.

Методи dataGridViewBoughtListings\_CellDoubleClick та dataGridViewAddedListings\_CellDoubleClick відповідають за обробку події подвійного кліку на комірці таблиці даних dataGridViewBoughtListings та dataGridViewAddedListings. Вони відкривають форму FormBoughtListings і FormAddedListings при подвійному клацанні на відповідній комірці.

Метод btnAddListing\_Click відповідає за обробку кнопки додавання оголошення. Він відкриває форму FormAddListing.

Метод deleteAdvertisementButton\_Click відповідає за обробку кнопки видалення оголошення. Він видаляє вибране оголошення зі списку AvailableListings класу GlobalData та списку AddedListings об'єкта CurrentUser класу LoginManager.

Клас також містить блок try-catch у кожному методі, який обробляє будь-які винятки, що можуть виникнути під час виконання методу. Якщо виникає виняток, він перехоплюється і передається методу ExceptionManager.HandleException, який виводить користувачеві повідомлення про помилку.

**Клас FormSignUp**

Клас FormSignUp - це форма, яка дозволяє користувачам створювати нові облікові записи. Вона містить поля для введення імені користувача, пароля та електронної пошти, а також кнопки для входу та реєстрації.

Конструктор класу ініціалізує форму і встановлює властивість PasswordChar текстових полів txtPassword і txtPasswordConfirm у значення '\*', що приховує введений пароль.

Методи DoesUsernameExist і DoesEmailExist використовуються для перевірки того, чи існує введене ім'я користувача та адреса електронної пошти у списку Users класу GlobalData.

Метод btnSignUp\_Click відповідає за обробку процесу реєстрації. Спочатку він перевіряє, чи відповідають введені ім'я користувача, пароль, адреса електронної пошти та підтвердження пароля встановленим вимогам. Якщо вони не відповідають вимогам, він додає повідомлення про помилку до списку errors. Якщо вони відповідають вимогам, програма генерує код підтвердження і надсилає його на введену адресу електронної пошти. Потім відкривається форма InputBox, де користувач може ввести код підтвердження. Якщо введений код збігається зі згенерованим, він встановлює властивості Username, Password та Email класу FormSignUp на введені ім'я користувача, гешований пароль та адресу електронної пошти, а також встановлює властивість DialogResult класу FormSignUp у значення OK.

Методи btnTogglePasswordVisibility\_Click та btnTogglePasswordConfirmVisibility\_Click відповідають за перемикання видимості пароля та підтвердження пароля. Вони змінюють властивість PasswordChar текстових полів txtPassword і txtPasswordConfirm між '\*' і '\0', що приховує і показує введений пароль відповідно.

Клас також включає блок try-catch у кожному методі, який обробляє будь-які винятки, що можуть виникнути під час виконання методу. Якщо виникає виняток, він перехоплюється і передається до методу ExceptionManager.HandleException, який відображає користувачеві повідомлення про помилку.

**Клас FormWelcome**

Клас FormWelcome - це форма, яка відображається, коли користувач вперше відкриває програму. Вона містить привітальне повідомлення, вікно з картинкою та випадкову підказку.

Конструктор класу ініціалізує форму і встановлює властивість Text порожнім рядком, а властивість ControlBox - значенням false. Це приховує рядок заголовка і поле керування форми, роблячи її схожою на діалогове вікно. Також викликається метод DisplayRandomTip для відображення випадкової підказки.

Метод DisplayRandomTip використовується для відображення випадкової підказки. Він вибирає випадкову підказку зі списку tips і встановлює властивість Text мітки labelRandomTip до вибраної підказки.

Метод InitializeComponent використовується для ініціалізації компонентів форми. Він створює та налаштовує компоненти logoPictureBox, labelWelcomeMessage та labelRandomTip. Компонент logoPictureBox - це вікно, у якому відображається зображення. labelWelcomeMessage - це мітка, яка відображає привітальне повідомлення. labelRandomTip - мітка, яка відображає випадкову підказку.

Клас також містить блок try-catch у методі DisplayRandomTip, який обробляє будь-які винятки, що можуть виникнути під час виконання методу. Якщо виникає виняток, він перехоплюється і передається до методу ExceptionManager.HandleException, який відображає користувачеві повідомлення про помилку.

3.2 Основні розроблені алгоритми

Для алгоритмів відведений спеціальний файл AlgorithmManager.cs в якому знаходяться три такі класи: СurrencyСonverter, ShowMap, SortManager. Це статичні класи, тобто до них можна отримати доступ без створення екземплярів, що дуже зручно, коли потрібно використовувати алгоритми в різних частинах застосунку.

Клас СurrencyСonverter містить один публічний статичний метод GetDollarRate.

Метод GetDollarRate призначений для визначення курсу долара відносно гривні з метою відображення ціни нерухомості в двох валютах. Метод отримує інформацію про необхідну валюту на поточну дату у форматі XML від Національного банку України. Після цього він конвертує курс валюти у тип double та повертає його значення. Метод GetDollarRate викликається у тих місцях, де необхідно відобразити ціну нерухомості.

Клас ShowMap містить один публічний статичний метод GoToMap.

Метод GoToMap дозволяє перейти до Google Maps за вказаною адресою. Він отримує один параметр – рядок address, який містить інформацію про адресу нерухомості. Метод відкриває Google Maps у браузері, використовуючи вказану адресу.

Клас SortManager містить 6 приватних статичних методів: QuickSort, Sort, Partition, QuickSortDescending, SortDescending, PartitionDescending та один публічний статичний метод SortListings. Клас потрібен для сортування нерухомості за такими полями: ціна, назва, продавець.

Метод SortListings за допомогою параметру sortType типу string визначає, за яким полем треба сортувати список нерухомості. Також, за допомогою параметру dropdown типу ComboBox визначає, який напрям сортування вибран. Далі викликається метод QuickSort, в який передається список нерухомості та поле, за яким треба сортувати.

В методі QuickSort викликається метод Sort куди передається список нерухомості, поле за яким потрібно сортувати та межі сортування від 0 до list.Count - 1 оскільки треба відсортувати весь список.

Метод Sort в ньому вибирається опорний елемент викликом методу Partition, який визначає і повертає індекс опорного елемента. Після визначення опорного елемента, метод рекурсивно викликає себе для сортування лівої та правої частини відносно опорного елемента. Sort(list, left, pivotIndex - 1, type) сортує ліву частину від опорного елемента. Sort(list, pivotIndex + 1, right, type) сортує праву частину від опорного елемента. Цей процес повторюється рекурсивно поки весь список не буде вірно відсортований.

Метод Partition – це ключова частина алгоритму швидкого сортування. Він вибирає опорний елемент та переставляє елементи так, що всі елементи менше опорного знаходяться ліворуч, а всі більші праворуч, також за допомогою розгалуження в ньому визначається за яким полем порівнювати нерухомість.

Методи: QuickSortDescending, SortDescending, PartitionDescending мають такий же алгоритм роботи як QuickSort, Sort, Partition з єдиною відмінністю сортування в них відбувається за спаданням.

Нижче наведено код файлу AlgorithmManager.cs:

using System;

using System.Collections.Generic;

using System.Globalization;

using System.Linq;

using System.Net;

using System.Security.Policy;

using System.Windows.Forms;

using System.Xml;

namespace code.Classes

{

public static class СurrencyСonverter

{

public static double GetDollarRate()

{

try

{

string link = @"https://bank.gov.ua/NBUStatService/v1/statdirectory/exchange?valcode=USD";

WebClient wc = new WebClient();

string xmlString = wc.DownloadString(link);

XmlDocument xmlDoc = new XmlDocument();

xmlDoc.LoadXml(xmlString);

XmlNode rateNode = xmlDoc.SelectSingleNode("/exchange/currency/rate");

string usdRate = rateNode.InnerText;

double rate = Convert.ToDouble(usdRate, CultureInfo.InvariantCulture);

return rate;

}

catch (Exception ex)

{

ExceptionManager.HandleException(ex, "Не вдалося отримати курс долара. Спробуйте пізніше", "Помилка отримання курсу");

return 0;

}

}

}

public static class ShowMap

{

public static void GoToMap(string adress)

{

try

{

string url = $"https://www.google.com/maps/place/{adress}";

System.Diagnostics.Process.Start(url);

}

catch(Exception ex)

{

ExceptionManager.HandleException(ex, "Не вдалося перейти на мапу. Спробуйте пізніше", "Помилка переходу на мапу");

}

}

}

public static class SortManager

{

private static void QuickSort(string type, List<Advertisement> list)

{

Sort(list, 0, list.Count - 1, type);

}

private static void Sort(List<Advertisement> list, int left, int right, string type)

{

if (left < right)

{

int pivotIndex = Partition(list, left, right, type);

Sort(list, left, pivotIndex - 1, type);

Sort(list, pivotIndex + 1, right, type);

}

}

private static int Partition(List<Advertisement> list, int left, int right, string type)

{

if (type == "Ціна")

{

int pivot = list[right].Price;

int i = left - 1;

Advertisement temp;

for (int j = left; j < right; j++)

{

if (list[j].Price < pivot)

{

i++;

temp = list[i];

list[i] = list[j];

list[j] = temp;

}

}

temp = list[i + 1];

list[i + 1] = list[right];

list[right] = temp;

return i + 1;

}

if (type == "Назва")

{

string pivot = list[right].Name;

int i = left - 1;

Advertisement temp;

for (int j = left; j < right; j++)

{

if ((string.Compare(list[j].Name, pivot) < 0))

{

i++;

temp = list[i];

list[i] = list[j];

list[j] = temp;

}

}

temp = list[i + 1];

list[i + 1] = list[right];

list[right] = temp;

return i + 1;

}

if (type == "Продавець")

{

string pivot = list[right].Seller;

int i = left - 1;

Advertisement temp;

for (int j = left; j < right; j++)

{

if ((string.Compare(list[j].Seller, pivot) < 0))

{

i++;

temp = list[i];

list[i] = list[j];

list[j] = temp;

}

}

temp = list[i + 1];

list[i + 1] = list[right];

list[right] = temp;

return i + 1;

}

return 0;

}

private static void QuickSortDescending(string type, List<Advertisement> list)

{

SortDescending(list, 0, list.Count - 1, type);

}

private static void SortDescending(List<Advertisement> list, int left, int right, string type)

{

if (left < right)

{

int pivotIndex = PartitionDescending(list, left, right, type);

SortDescending(list, left, pivotIndex - 1, type);

SortDescending(list, pivotIndex + 1, right, type);

}

}

private static int PartitionDescending(List<Advertisement> list, int left, int right, string type)

{

if (type == "Ціна")

{

int pivot = list[right].Price;

int i = left - 1;

Advertisement temp;

for (int j = left; j < right; j++)

{

if (list[j].Price > pivot)

{

i++;

temp = list[i];

list[i] = list[j];

list[j] = temp;

}

}

temp = list[i + 1];

list[i + 1] = list[right];

list[right] = temp;

return i + 1;

}

if (type == "Назва")

{

string pivot = list[right].Name;

int i = left - 1;

Advertisement temp;

for (int j = left; j < right; j++)

{

if ((string.Compare(list[j].Name, pivot) > 0))

{

i++;

temp = list[i];

list[i] = list[j];

list[j] = temp;

}

}

temp = list[i + 1];

list[i + 1] = list[right];

list[right] = temp;

return i + 1;

}

if (type == "Продавець")

{

string pivot = list[right].Seller;

int i = left - 1;

Advertisement temp;

for (int j = left; j < right; j++)

{

if ((string.Compare(list[j].Seller, pivot) > 0))

{

i++;

temp = list[i];

list[i] = list[j];

list[j] = temp;

}

}

temp = list[i + 1];

list[i + 1] = list[right];

list[right] = temp;

return i + 1;

}

return 0;

}

public static void SortListings(ComboBox dropdown, string sortType, DataGridView dataGridViewListings)

{

var nameHeading = "Назва";

var sellerHeading = "Продавець";

var priceHeading = "Ціна";

var listingsCopy = GlobalData.AvailableListings.ToList();

var selectedItem = dropdown.SelectedIndex;

if (sortType == "byName")

{

if (selectedItem == 0)

{

QuickSort(nameHeading, listingsCopy);

nameHeading = "Назва »";

}

else

{

QuickSortDescending(nameHeading, listingsCopy);

nameHeading = "Назва «";

}

}

else if (sortType == "bySeller")

{

if (selectedItem == 0)

{

QuickSort(sellerHeading, listingsCopy);

sellerHeading = "Продавець »";

}

else

{

QuickSortDescending(sellerHeading, listingsCopy);

sellerHeading = "Продавець «";

}

}

else if (sortType == "byPrice")

{

if (selectedItem == 0)

{

QuickSort(priceHeading, listingsCopy);

priceHeading = "Ціна »";

}

else

{

QuickSortDescending(priceHeading, listingsCopy);

priceHeading = "Ціна «";

}

}

dataGridViewListings.DataSource = listingsCopy;

dataGridViewListings.Columns[0].HeaderText = nameHeading;

dataGridViewListings.Columns[3].HeaderText = sellerHeading;

dataGridViewListings.Columns[4].HeaderText = priceHeading;

}

}

}

3.3 Основні рішення щодо розробки інтерфейсу

Розробка нашого інтерфейсу була кропітким процесом, в якому основна увага приділялася користувацькому досвіду та зручності використання. Ми вирішили використовувати Visual Studio та C# WinForms .NET для цього завдання, оскільки ці технології пропонують надійну платформу для створення десктопних застосунків для Windows. Використання WinForms .NET дозволило нам використати можливості C#, а також забезпечити зручний інтерфейс для проєктування та розробки нашого застосунку.

Інтерфейс був розроблений з чіткою візуальною ієрархією, що спрямовує увагу користувача на найбільш важливі частини програми. Цього було досягнуто завдяки стратегічному використанню кольорів, розміру та товщини тексту, а також інтервалів між рядками. Для розміщення меню ми обрали бічну панель у темних кольорах, що створює чіткий контраст з основною зоною контенту, яка виконана у світлих тонах. Такий вибір дизайну був зроблений для того, щоб виділити основний контент і зменшити візуальне навантаження.

Кнопки в нашому інтерфейсі були стилізовані під фірмовий колір - насичений темно-синій з білим жирним текстом. Таке поєднання було обрано через його високу контрастність, завдяки чому кнопки легко ідентифікуються та виділяються на фоні. Це важливий аспект зручності використання, оскільки він полегшує користувачам взаємодію з інтерфейсом.

Ми також використовували табличні подання даних, які були стилізовані у фірмовому синьому кольорі для заголовків та аквамариновому кольорі для обраного елемента. Ці компоненти були побудовані без границь, щоб зробити інтерфейс чистішим і легшим для очей. Такий підхід не тільки підвищує естетичну привабливість інтерфейсу, але й покращує навігацію.

На додаток до цих елементів, ми використовували різні властивості користувацьких елементів управління в Visual Studio для подальшої кастомізації нашого інтерфейсу. Це дозволило нам пристосувати зовнішній вигляд нашого застосунку до конкретних потреб та вподобань, а також забезпечити його цілісність та логічність.

На закінчення, наш процес розробки інтерфейсу був спільним зусиллям, зосередженим на створенні чистого, витонченого користувацького інтерфейсу з сильним акцентом на користувацькому досвіді та зручності використання. Ми використали можливості Visual Studio та C# WinForms .NET у поєднанні зі стратегічним використанням кольорів, розміру та ваги тексту, а також інтервалів для створення візуально привабливого та зручного для користувача застосунку.

3.4 Основні рішення щодо роботи з даними

Робота з даними в програмі реалізована через файли формату CSV та JSON, які знаходяться в директорії DataBase. Робота с JSON здійснюється за допомогою бібліотеки Newtonsoft.Json. В нас є два основних файла, в яких зберігається вся основна інформація щодо нерухомості і користувачів. При реєстрації нового акаунту генерується два нових файла, в які заноситься інформація щодо дій с нерухомістю поточного користувача (куплена, виставлена нерухомість):

advertisment.csv – файл, який містить всю інформацію про нерухомість, задіяну в програмі. Дані зберігаються у вигляді таблиці, розділені знаками ”|”. Кожний рядок файлу представляє окрему нерухомість, та має такі дані як назва, опис, ціна, адреса, продавець, шлях до директорії з фото.

users.json – файл, який містить всю інформацію щодо зареєстрованих користувачів. Представляє себе у вигляді масиву об’єктів, кожен з яких – зареєстрований користувач. Об’єкти містять таку інформацію як UserId, ім’я, електронну пошту, пароль, шлях до згенерованого файлу з купленою нерухомістю та шлях до згенерованого файлу з доданою нерухомістю.

user\_02f7b03f-1553-414c-8835-d30ef42818cb\_AddedListings.csv – згенерований файл за певним користувачем, який містить інформацію про додану нерухомість. Дані зберігаються в такому ж вигляді, як і в advertisment.csv.

user\_02f7b03f-1553-414c-8835-d30ef42818cb\_BoughtListings.csv - згенерований файл за певним користувачем, який містить інформацію про куплену нерухомість. Дані зберігаються в такому ж вигляді, як і в advertisment.csv.

За роботу з даними відповідає клас FileHandler, який містить такі класи та методи:

UserObj – клас для роботи з файлом формату JSON. Містить такі поля як UserId, Name, Email, Password, BoughtListingsFilePath, AddedListingsFilePath.

saveAdToCSV - метод для запису даних до CSV файлу. Приймає два праметри: list writeFromList та рядок filepath. Якщо при виклику функції шлях файлу не передали, то дані будуть записані у файл advertisment.csv. В іншому випадку дані будуть записані у файл за відповідним шляхом. Метод відкриває файловий потік та записує у файл дані кожного об’єкту з list’а.

readAdFromCSV – метод для зчитування даних з CSV файлу. Приймає один параметр: рядок filepath. Якщо при виклику функції шлях файлу не передали, то дані будуть зчитані з файлу advertisment.csv. Метод зчитує всі дані з файлу за шляхом у масив рядків, розділяє їх за роздільним знаком та вносить дані кожного об’єкту за індексом у list readToList, після чого повертає його.

UserFileWriter - метод для запису даних User. Приймає один параметр: list writeFromList. Викликається інший метод DeleteFilesExcept для видалення зайвих даних перед записом, після чого створює новий list writeList типу UserObj. Далі в циклі створюється новий об’єкт user, який заповнюється даними об’єкту з list’а writeFromList. Також викликаються методи saveAdToCSV для генерування нових файлів за унікальною назвою по UserId, в які записуються дані щодо нерухомості пов’язані з даним користувачем. В кінці кожної ітерації циклу здійснюється зберігання записаного об’єкту в list writeList. Далі всі дані з writeList записуються в рядок за допомогою методу серіалізації з бібліотеки Newtonsoft.Json й записуються у файл users.json.

UserFileReader - метод для зчитування даних User. Створюється новий list readList типу UserObj, в який після процесу десеріалізації записуються дані з файлу users.json. Також створюється list users. Далі в циклі йде присвоєння даних конкретного за ітерацією об’єкта з readList в новий створений об’єкт user, після чого він записується в раніше створений list users, який після спрацювання циклу повертається.

DeleteFilesExcept - метод для видалення зайвих даних перед записом додаткових файлів. Приймає два параметри: рядки directoryPath та fileToKeep. Метод створює масив рядків files в який за допомогою метода Directory.GetFiles записуються шляхи до всіх файлів в директорії directoryPath, далі створюється масив рядків filesToDelete в який записуються всі шляхи окрім fileToKeep, після чого в масиві здійснюється видалення зайвих файлів з директорії за шляхами з filesToDelete.

3.5 Обробка виключних ситуацій

Обробкою виключних ситуацій займається спеціальний клас ExceptionManager. Цей клас призначений для обробки винятків, їх обліку та відображення відповідних повідомлень користувачеві. Це статичний клас, тобто до нього можна отримати доступ без створення екземпляра, що дуже зручно, коли нам потрібно обробляти винятки в різних частинах нашого застосунку.

Клас ExceptionManager містить три публічні статичні методи: HandleException, Confirm та ShowInfo.

Метод HandleException призначений для обробки виключень. Він отримує три параметри: об'єкт Exception, рядок body і рядок heading. Об'єкт Exception містить деталі виключення, що виникло. Рядки body та heading використовуються для відображення користувачеві вікна з деталями виключення. Метод записує повідомлення про виняток у консоль, а потім виводить вікно з повідомленням про виняток, тілом і заголовком.

Метод Confirm використовується для відображення користувачеві діалогового вікна підтвердження. Він отримує два параметри: рядок body і рядок heading. Метод виводить вікно повідомлення з тілом і заголовком, і дає користувачеві можливість підтвердити або скасувати дію. Метод повертає булеве значення, яке вказує, чи підтвердив користувач дію.

Метод ShowInfo використовується для відображення інформаційного повідомлення користувачеві. Він отримує два параметри: рядок body і рядок heading. Метод виводить вікно повідомлення з тілом і заголовком.

Нижче наведено місця, де оброблено виключення з використанням відповідного класу:

// FormMenu - Line 54

            catch (Exception ex)

            {

                // If an exception occurs, handle it and display an error message

                ExceptionManager.HandleException(ex, "Не вдалося завантажити дані. Спробуйте пізніше", "Помилка завантаження даних");

            }

// FormMenu - Line 105

            catch (Exception ex)

            {

                // Handle the exception and display an error message

                ExceptionManager.HandleException(ex, "Не вдалося активувати кнопку. Спробуйте пізніше", "Помилка активації");

            }

// FormMenu - Line 133

            catch (Exception ex)

            {

                // Handle any exceptions that occur during the button disabling process

                ExceptionManager.HandleException(ex, "Не вдалося деактивувати кнопку. Спробуйте пізніше", "Помилка деактивації");

            }

// FormMenu - Line 174

            catch (Exception ex)

            {

                // Handle any exception that occurs during form opening

                ExceptionManager.HandleException(ex, "Не вдалося відкрити форму. Спробуйте пізніше", "Помилка відкриття");

            }

// AlgorithmManager - Line 27

            catch (Exception ex)

            {

                ExceptionManager.HandleException(ex, "Не вдалося отримати курс долара. Спробуйте пізніше", "Помилка отримання курсу");

                return 0;

            }

// FileHandler - Line 39

            catch (Exception ex)

            {

                ExceptionManager.HandleException(ex, "Не вдалося записати дані в файл.", "Помилка запису даних");

            }

// FileHandler - Line 78

            catch (Exception ex)

            {

                ExceptionManager.HandleException(ex, "Не вдалося зчитати дані з файлу.", "Помилка зчитування даних");

                return null;

            }

// FileHandler - Line 126

            catch (Exception ex)

            {

                ExceptionManager.HandleException(ex, "Не вдалося записати дані в файл.", "Помилка запису даних");

            }

// FileHandler - Line 162

            catch (Exception ex)

            {

                ExceptionManager.HandleException(ex, "Не вдалося зчитати дані з файлу.", "Помилка зчитування даних");

                return null;

            }

// FileHandler - Line 182

            catch (Exception ex)

            {

                ExceptionManager.HandleException(ex, "Не вдалося видалити дані.", "Помилка видалення даних");

            }

// FormAddedListing - Line 27

            catch (Exception ex)

            {

                ExceptionManager.HandleException(ex, "Не вдалося завантажити деталі оголошення. Спробуйде закрити розділ і повторити спробу", "Помилка завантаження даних");

            }

// FormAddListing - Line 29

            try

            {

                if (string.IsNullOrWhiteSpace(txtName.Text) ||

                    string.IsNullOrWhiteSpace(txtDescription.Text) ||

                    string.IsNullOrWhiteSpace(txtAddress.Text) ||

                    numericUpDownPrice.Value == 0 ||

                    fileDialogImageUpload.FileName == string.Empty)

                {

                    ExceptionManager.HandleException(new Exception("Заповніть всі поля"), "Будь ласка, заповніть всі поля і переконайтеся, що ціна не дорівнює нулю", "Валідація полів");

                    return;

                }

                bool isConfirmed = ExceptionManager.Confirm("Ви впевнені що хочете додати це оголошення?", "Підтвердження додавання");

                if (isConfirmed)

                {

                    destinationPath = destinationDirectory + "/" + txtName.Text + ".jpg";

                    Advertisement listing = new Advertisement

                    {

                        Name = txtName.Text,

                        Description = txtDescription.Text,

                        Address = txtAddress.Text,

                        Price = Convert.ToInt32(numericUpDownPrice.Value),

                        Seller = LoginManager.CurrentUser.Name,

                        ImagePath = destinationPath

                    };

                    GlobalData.AvailableListings.Add(listing);

                    LoginManager.CurrentUser.AddedListings.Add(listing);

                    ExceptionManager.ShowInfo("Оголошення успішно додано", "Успіх");

                    Close();

                }

            }

            catch (Exception ex)

            {

                ExceptionManager.HandleException(ex, "Не вдалося додати оголошення. Спробуйте пізніше", "Помилка додавання");

            }

// FormAddListing - Line 88

            catch (Exception ex)

            {

                ExceptionManager.HandleException(ex, "Не вдалося оновити кількість символів. Спробуйте пізніше", "Помилка перевірки довжини опису");

            }

// FormAddListing - Line 128

            catch (Exception ex)

            {

                ExceptionManager.HandleException(ex, "Не вдалося завантажити фото. Спробуйте пізніше", "Помилка завантаження фото");

            }

// FormBoughtListing - Line 27

            catch (Exception ex)

            {

                ExceptionManager.HandleException(ex, "Не вдалося завантажити деталі оголошення. Спробуйте пізніше", "Помилка завантаження даних");

            }

// FormBuyListing - Line 31

            catch (Exception ex)

            {

                ExceptionManager.HandleException(ex, "Не вдалося завантажити деталі оголошення. Спробуйте пізніше", "Помилка завантаження даних");

            }

// FormBuyListing - Line 43

            try

            {

                if (LoginManager.IsLoggedIn)

                {

                    if (LoginManager.CurrentUser.Name == listing.Seller)

                    {

                        ExceptionManager.HandleException(new Exception("Неможливо купити свою власну нерухомість."), "Ви не можете купувати свою нерухомість.", "Помилка купівлі");

                        return;

                    }

                    LoginManager.CurrentUser.BoughtListings.Add(listing);

                    formMarket.RemoveListing(listing);

                }

                else

                {

                    ExceptionManager.HandleException(new Exception("Авторизуйтесь перш ніж купувати нерухомість."), "Ви повинні авторизуватися, щоб купити нерухомість. Будь ласка, увійдіть у свій акаунт і спробуйте ще раз", "Помилка купівлі");

                    return;

                }

                Close();

            }

            catch (Exception ex)

            {

                ExceptionManager.HandleException(ex, "Не вдалося купити нерухомість. Спробуйте пізніше", "Помилка купівлі");

            }

// FormDevelopers - Line 23

            catch (Exception ex)

            {

                ExceptionManager.HandleException(ex, "Не вдалося відкрити сторінку в браузері. Спробуйте ще раз пізніше", "Помилка відкриття");

            }

// FormDevelopers - Line 35

            catch (Exception ex)

            {

                ExceptionManager.HandleException(ex, "Не вдалося відкрити сторінку в браузері. Спробуйте ще раз пізніше", "Помилка відкриття");

            }

// FormLogin - Line 26

            catch (Exception ex)

            {

                ExceptionManager.HandleException(ex, "Не вдалося авторизуватися. Спробуйте ще раз пізніше", "Помилка авторизації");

                return null;

            }

// FormLogin - Line 55

            if (errors.Count > 0)

            {

                ExceptionManager.HandleException(new Exception(string.Join("\n", errors)), string.Join("\n", errors), "Помилка авторизації");

                LoginAttempts++;

                if (LoginAttempts >= 3)

                {

                    ExceptionManager.ShowInfo("Ви зробили надто багато невдалих спроб входу за короткий час. Спробуйте пізніше.", "Помилка входу");

                    this.DialogResult = DialogResult.Cancel;

                }

                return;

            }

// FormLogin - Line 69

            try

            {

                var user = ValidateLogin(username, password);

                if (user == null)

                {

                    ExceptionManager.HandleException(new Exception("Невірний логін або пароль."), "Невірний логін або пароль.", "Помилка авторизації");

                    LoginAttempts++;

                    return;

                }

                this.DialogResult = DialogResult.OK;

                LoginManager.CurrentUser = user;

                LoginManager.IsLoggedIn = true;

            }

            catch (Exception ex)

            {

                ExceptionManager.HandleException(ex, "Виникла помилка під час входу", "Помилка входу");

            }

// FormLogin - Line 117

            catch (Exception ex)

            {

                ExceptionManager.HandleException(ex, "Виникла помилка під час реєстрації", "Помилка реєстрації");

            }

// FormLogin - Line 138

            catch (Exception ex)

            {

                ExceptionManager.HandleException(ex, "Виникла помилка при перемиканні видимості пароля", "Помилка видимості пароля");

            }

// FormLogin - Line 159

            catch (Exception ex)

            {

                ExceptionManager.HandleException(ex, "Виникла помилка під час зміни паролю", "Помилка зміни паролю");

            }

// FormMarket - Line 41

            catch (Exception ex)

            {

                ExceptionManager.HandleException(ex, "Не вдалося додати об'єкт. Спробуйте ще раз пізніше", "Помилка додавання об'єкту");

            }

// FormMarket - Line 56

            catch (Exception ex)

            {

                ExceptionManager.HandleException(ex, "Не вдалося видалити об'єкт. Спробуйте ще раз пізніше", "Помилка видалення об'єкту");

            }

// FormMarket - Line 71

            catch (Exception ex)

            {

                ExceptionManager.HandleException(ex, "Помилка при пошуку оголошення. Спробуйте ще раз пізніше", "Помилка пошуку об'єкту");

            }

// FormMarket - Line 93

            catch (Exception ex)

            {

                ExceptionManager.HandleException(ex, "Не вдалося відкрити форму. Спробуйте ще раз пізніше", "Помилка відкриття форми");

            }

// FormMarket - Line 123

            catch (Exception ex)

            {

                ExceptionManager.HandleException(ex, "Не вдалося показати підказку.", "Помилка показу підказки");

            }

// FormMarket - Line 139

            catch (Exception ex)

            {

                ExceptionManager.HandleException(ex, "Не вдалося приховати підказку.", "Помилка приховання підказки");

            }

// FormMarket - Line 199

            catch (Exception ex)

            {

                ExceptionManager.HandleException(ex, "Не вдалося відсортування оголошення", "Помилка сортування");

            }

// FormMarket - Line 220

            catch (Exception ex)

            {

                ExceptionManager.HandleException(ex, "Не вдалося відсортування оголошення", "Помилка сортування");

            }

// FormMarket - Line 241

            catch (Exception ex)

            {

                ExceptionManager.HandleException(ex, "Не вдалося відсортування оголошення", "Помилка сортування");

            }

// FormProfile - Line 37

            catch (Exception ex)

            {

                ExceptionManager.HandleException(ex, "Не вдалося авторизуватися. Спробуйте ще раз пізніше", "Помилка авторизації");

            }

// FormProfile - Line 93

            catch (Exception ex)

            {

                ExceptionManager.HandleException(ex, "Не вдалося відкрити форму. Спробуйте ще раз пізніше", "Помилка відкриття форми");

            }

// FormProfile - Line 120

            if (newUsername == currentUsername && newEmail == currentEmail && newPassword.Length == 0)

            {

                ExceptionManager.ShowInfo("Ви не внесли жодних змін.", "Внесіть зміни і спробуйте ще раз");

            }

            else if (!string.IsNullOrEmpty(newUsername) && !string.IsNullOrEmpty(newPassword) && !string.IsNullOrEmpty(newEmail))

            {

                bool isConfirmed = ExceptionManager.Confirm("Ви впевнені що хочете змінити свої облікові дані?", "Підтвердження зміни");

// FormProfile - Line 216

                    if (errors.Count > 0)

                    {

                        ExceptionManager.HandleException(new Exception(string.Join("\n", errors)), string.Join("\n", errors), "Помилка реєстрації");

                        return;

                    }

                    ExceptionManager.ShowInfo("Ви успішно змінили свої облікові дані!", "Зміни внесено успішно");

// FormProfile - Line 243

            catch (Exception ex)

            {

                ExceptionManager.HandleException(ex, "Не вдалося оновити список куплених оголошень. Спробуйте ще раз пізніше", "Помилка оновлення списку оголошень");

            }

// FormProfile - Line 259

            catch (Exception ex)

            {

                ExceptionManager.HandleException(ex, "Не вдалося очистити список куплених оголошень. Спробуйте ще раз пізніше", "Помилка очищення списку оголошень");

            }

// FormProfile - Line 273

            catch (Exception ex)

            {

                ExceptionManager.HandleException(ex, "Не вдалося відкрити форму. Спробуйте ще раз пізніше", "Помилка відкриття форми");

            }

// FormProfile - Line 287

            catch (Exception ex)

            {

                ExceptionManager.HandleException(ex, "Не вдалося відкрити форму. Спробуйте ще раз пізніше", "Помилка відкриття форми");

            }

// FormProfile - Line 300

            catch (Exception ex)

            {

                ExceptionManager.HandleException(ex, "Не вдалося відкрити форму. Спробуйте ще раз пізніше", "Помилка відкриття форми");

            }

// FormProfile - Line 329

                bool isConfirmed = ExceptionManager.Confirm("Ви впевнені що хочете видалити це оголошення?", "Підтвердження видалення");

// FormProfile - Line 347

            catch (Exception ex)

            {

                ExceptionManager.HandleException(ex, "Не вдалося видалити оголошення. Спробуйте пізніше", "Помилка видалення");

            }

// FormRestore - Line 90

            if (errors.Count > 0)

            {

                ExceptionManager.HandleException(new Exception(string.Join("\n", errors)), string.Join("\n", errors), "Помилка реєстрації");

                return;

            }

// FormSignUp - Line 131

            if (errors.Count > 0)

            {

                ExceptionManager.HandleException(new Exception(string.Join("\n", errors)), string.Join("\n", errors), "Помилка реєстрації");

                return;

            }

// FormSignUp - Line 158

            catch (Exception ex)

            {

                ExceptionManager.HandleException(ex, "Виникла помилка при перемиканні видимості пароля", "Помилка видимості пароля");

            }

// FormSignUp - Line 184

            catch (Exception ex)

            {

                ExceptionManager.HandleException(ex, "Виникла помилка при перемиканні видимості підтвердження пароля", "Помилка видимості підтвердження пароля");

            }

// FormSignUp - Line 41

            catch (Exception ex)

            {

                ExceptionManager.HandleException(ex, "Виникла помилка під час відображення випадкової підказки. Спробуйте змінити розділ і повторити спробу", "Не вдалося відобразити випадкову підказку.");

            }

4 керівництво програміста

Цей розділ призначений для інших розробників, які мають доступ до вихідного коду нашої програми. Він надасть їм детальні інструкції щодо встановлення та використання програми, а також всебічне розуміння її структури та функціоналу. Ми заглибимося в призначення програми, її функції, умови використання, загальний опис, структуру та методи доступу. Ми також обговоримо вхідні та вихідні дані, а також типи повідомлень, які програміст може очікувати побачити при роботі з нашою системою.

4.1 Призначення та умови застосування програми

4.1.1 Призначення програми

Метою нашої програми є надання зручного інтерфейсу для управління цифровими даними ринку нерухомості. Вона дозволяє користувачам вводити, переглядати та керувати даними, пов'язаними з різними об'єктами нерухомості, такими як їх місцезнаходження, вартість, розмір та інші відповідні деталі.

4.1.2 Функції програми

Наша програма призначена для виконання різноманітних функцій. До них відносяться додавання, перегляд, купівля та видалення оголошень на ринку. Вона також включає в себе функції пошуку і сортування оголошень, а також збереження інформації в різних форматах для подальшого аналізу.

4.1.3 Умови застосування програми

Використання нашої програми регулюється певними умовами та положеннями. Користувачі зобов'язані дотримуватися цих умов під час використання програми. Ці умови включають, але не обмежуються належним використанням програми, захистом даних, повагою до конфіденційності та прав інтелектуальної власності. Для отримання більш детальної та повної інформації про умови використання можна ознайомитися з [ліцензією](https://github.com/amaiboy/digital-real-estate-market-course-project/blob/main/LICENSE) нашого застосунку.

4.2 Характеристика програми

Наша програма є комплексним інструментом для управління цифровими даними ринку нерухомості. Вона розроблена таким чином, щоб бути зручною та ефективною, аби користувачі могли легко вводити, переглядати та керувати даними, пов'язаними з різними об'єктами нерухомості.

4.2.1 Структура програми

Структура нашої програми складається з декількох каталогів і файлів, кожен з яких слугує певному призначенню.

Каталог assets містить підкаталоги для зображень та інших статичних файлів, що використовуються у програмі. Сюди входить список зображень та інших піктограм, які використовуються у програмі.

Каталог code - це місце, де знаходиться основна логіка нашого застосунку. Він містить декілька підкаталогів:

- Classes: Цей каталог містить декілька файлів .cs, кожен з яких представляє клас нашого застосунку. Ці класи виконують різні функції, такі як управління оголошеннями (Advertisement.cs), управління алгоритмами (AlgorithmManager.cs), управління винятками (ExceptionManager.cs), управління файловими операціями (FileHandler. cs), зберігання глобальних даних (GlobalData.cs), управління переліками (Listing.cs), управління логіном користувача (LoginManager.cs), управління властивостями (Property.cs), управління пошуком (SearchManager.cs) та управління даними користувача (User.cs);

- CustomControls: Цей каталог містить кастомні класи елементів керування, які використовуються для створення кастомних елементів інтерфейсу у нашому застосунку. До них відносяться MainTitle.cs для головного заголовка сторінки, StyledButton.cs для стилізованих кнопок і StyledTable.cs для стилізованих таблиць;

- Database: Цей каталог містить файли даних, які використовуються нашим застосунком. До них відносяться properties.json для зберігання даних про нерухомість і users.csv для зберігання даних про користувачів.

Каталог Forms містить класи форм для нашого застосунку. Кожен клас форми представляє окремий екран у нашому застосунку, наприклад, FormAddedListing.cs для екрану, де користувач може додати нове оголошення, FormAddListing.cs для екрану, де користувач може додати оголошення, FormBoughtListing.cs для екрану, де користувач може переглянути куплені оголошення, і так далі.

Файл Program.cs є точкою входу в наш застосунок. Він відповідає за запуск програми та ініціалізацію головної форми.

Файл FormMenu.cs - це клас для головного меню нашого застосунку. Він містить код меню та обробники подій для пунктів меню.

Така структура дозволяє ефективно виконувати завдання, а також легко підтримувати та оновлювати програму. Кожен клас і файл відповідає за певну функціональність, що робить код більш зрозумілим і простим в управлінні.

Більш детальна файлова структура нашого проєкту наведена нижче:

.

├── assets

│   └── images

│       └── ...зображення оголошень та інші іконки

├── docs

│   └── ...документація проєкту і всі дотичні файли

└── code

    ├── Classes

    │   ├── Advertisement.cs

    │   ├── AlgorithmManager.cs

    │   ├── ExceptionManager.cs

    │   ├── FileHandler.cs

    │   ├── GlobalData.cs

    │   ├── Listing.cs

    │   ├── LoginManager.cs

    │   ├── Property.cs

    │   ├── SearchManager.cs

    │   └── User.cs

    ├── CustomControls

    │   ├── MainTitle.cs

    │   ├── StyledButton.cs

    │   └── StyledTable.cs

    ├── DataBase

    │   ├── properties.json

    │   └── users.csv

    ├── Forms

    │   ├── FormAddedListing.cs, FormAddedListing.Designer.cs

    │   ├── FormAddListing.cs, FormAddListing.Designer.cs

    │   ├── FormBoughtListing.cs, FormBoughtListing.Designer.cs

    │   ├── FormBuyListing.cs, FormBuyListing.Designer.cs

    │   ├── FormDevelopers.cs, FormDevelopers.Designer.cs

    │   ├── FormHelp.cs, FormHelp.Designer.cs

    │   ├── FormLogin.cs, FormLogin.Designer.cs

    │   ├── FormMarket.cs, FormMarket.Designer.cs

    │   ├── FormProfile.cs, FormProfile.Designer.cs

    │   ├── FormSignUp.cs, FormSignUp.Designer.cs

    │   └── FormWelcome.cs, FormWelcome.Designer.cs

    ├── FormMenu.cs, FormMenu.Designer.cs

    └── Program.cs

4.3 Звертання до програми

Доступ до нашої програми можна отримати шляхом клонування вихідного коду з нашого [репозиторію](https://github.com/amaiboy/digital-real-estate-market-course-project/blob/main/LICENSE) та відкриття його у сумісному інтегрованому середовищі розробки (IDE), такому як Visual Studio. Після відкриття програми в IDE, її можна запустити безпосередньо звідти.

4.4 Вхідні та вихідні дані

4.4.1 Вхідні дані

Для коректної роботи програми потрібні певні файли. До них відносяться файл бази даних, що містить дані про ринок нерухомості, і конфігураційний файл, що визначає деталі підключення до бази даних. Файл бази даних повинен бути у форматі, сумісному з рушієм бази даних, що використовується програмою.

4.4.2 Вихідні дані

Програма надає програмісту різні типи вихідних даних. До них відносяться повідомлення про помилки, повідомлення про підтвердження та інформаційні повідомлення. За обробку цих повідомлень відповідає клас ExceptionManager. Він надає методи для виведення користувачеві повідомлень про помилки, підтверджень та інформаційних повідомлень.

4.5 Повідомлення

Наша програма надає програмісту різноманітні інформаційні повідомлення. До них відносяться повідомлення про підтвердження, коли дія користувача є успішною, повідомлення про помилки, коли виникає виняток, та інформаційні повідомлення, що надають детальну інформацію про роботу програми. Наприклад, повідомлення про підтвердження може з'явитися після успішного додавання нового об'єкта до бази даних. Повідомлення про помилку може з'явитися, якщо виникла проблема з підключенням до бази даних. Інформаційне повідомлення може з'явитися, коли програма успішно виконує купівлю нерухомості.

Більш детальний список можливих повідомлень наведено нижче:

- Підтвердження успішного додавання оголошення на ринок;

- Попередження про надто велику кількість спроб входу до акаунту за короткий період часу;

- Попередження про відсутність наявних змін при спробі змінити облікові дані у профілі;

- Підтвердження успішної зміни облікових даних у профілі;

- Попередження про надсилання тимчасового паролю на електронну пошту при спробі відновити обліковий запис.

5 керівництво користувача

Цей розділ містить вичерпний посібник з користування програмою "Цифровий ринок нерухомості". Він покликаний допомогти користувачам зрозуміти призначення програми, умови її виконання, процес виконання програми, а також повідомлення, з якими користувач може зіткнутися під час використання програми.

5.1 Призначення програми

Застосунок "Цифровий ринок нерухомості" - це комплексна платформа, призначена для полегшення купівлі-продажу об'єктів нерухомості. Вона надає користувачам зручний інтерфейс для пошуку, перегляду та купівлі об'єктів нерухомості. Застосунок також дозволяє користувачам керувати своїми оголошеннями, переглядати свій профіль та взаємодіяти з усіма іншими секціями застосунку.

5.2 Умови виконання програми

5.2.1 Апаратні вимоги програми

Застосунок "Цифровий ринок нерухомості" розроблений таким чином, щоб бути сумісним з широким спектром апаратних конфігурацій. Однак для забезпечення оптимальної продуктивності рекомендується запускати програму на комп'ютері з наступними мінімальними вимогами:

- Процесор з тактовою частотою 1,6 ГГц або вище;

- 2 ГБ оперативної пам'яті;

- 2 ГБ вільного місця на диску.

5.2.2 Вимоги до користувача

Застосунок "Цифровий ринок нерухомості" призначений для користувачів, які зацікавлені в купівлі, продажу або управлінні об'єктами нерухомості. Користувачі повинні мати базове розуміння того, як користуватися комп'ютерним програмним забезпеченням, і вміти орієнтуватися в інтерфейсі користувача.

5.3 Виконання програми

5.3.1 Запуск програми

Щоб запустити програму "Цифровий ринок нерухомості", виконайте наступні кроки:

1. Перейдіть до директорії, в яку встановлено застосунок;
2. Відкрийте наступний шлях, відносно до батьківської теки: code/bin/Debug/code.exe

2. Двічі клацніть на виконуваному файлі, щоб запустити застосунок;

3. Після запуску програми перед вами відкриється головне меню.

5.3.2 Виконання роботи з програмою

Застосунок "Цифровий ринок нерухомості" складається з декількох розділів, кожен з яких має свій набір функцій. Ось короткий огляд:

- Головне меню: Це перший екран, який ви бачите при відкритті програми. Він надає доступ до всіх основних функцій застосунку. Разом із ним відкривається вітальна форма, що показує привітальне повідомлення та випадкову підказку щодо користування програмою;

- Ринок: У цьому розділі відображається список доступних об'єктів нерухомості на продаж. Користувачі можуть переглядати оголошення, сортувати їх за різними критеріями та переглядати детальну інформацію про кожну нерухомість;

- Профіль: У цьому розділі відображається особиста інформація користувача та його оголошення, як придбані, так і виставлені. Користувачі можуть оновлювати свою особисту інформацію та керувати своїми оголошеннями з цього розділу;

- Логін/Реєстрація: Цей розділ дозволяє користувачам увійти до свого облікового запису або створити новий обліковий запис, якщо у них його ще немає;

- Допомога: Цей розділ містить загальну інформацію про програму та її можливості;

- Про розробників: Цей розділ надає загальну інформація про розробників, розподіл відповідальностей та посилання на GitHub сторінку і ліцензію проєкту.

Більш детальні відомості з користування програмою у вигляді знімків з екрану наведено нижче:

5.4 Повідомлення користувачу

Під час використання застосунку "Цифровий ринок нерухомості" користувач може зіткнутися з різними повідомленнями в залежності від своїх дій. Ці повідомлення призначені для забезпечення зворотного зв'язку та надання користувачеві рекомендацій щодо ефективного використання програми.

Деякі з повідомлень наведено нижче у вигляді знімків з екрану:

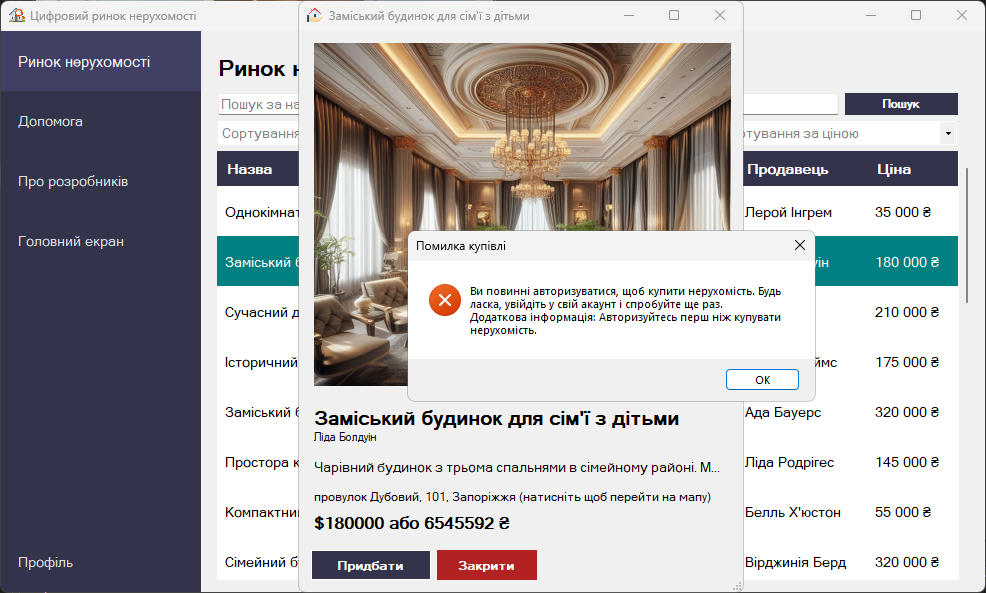


Рисунок 4.1 – Помилка при спробі купівлі нерухомості неавторизованим користувачем



Рисунок 4.2 – Помилка при введені невірних облікових даних



Рисунок 4.3 – Помилка при спробі ввести надто короткий пароль

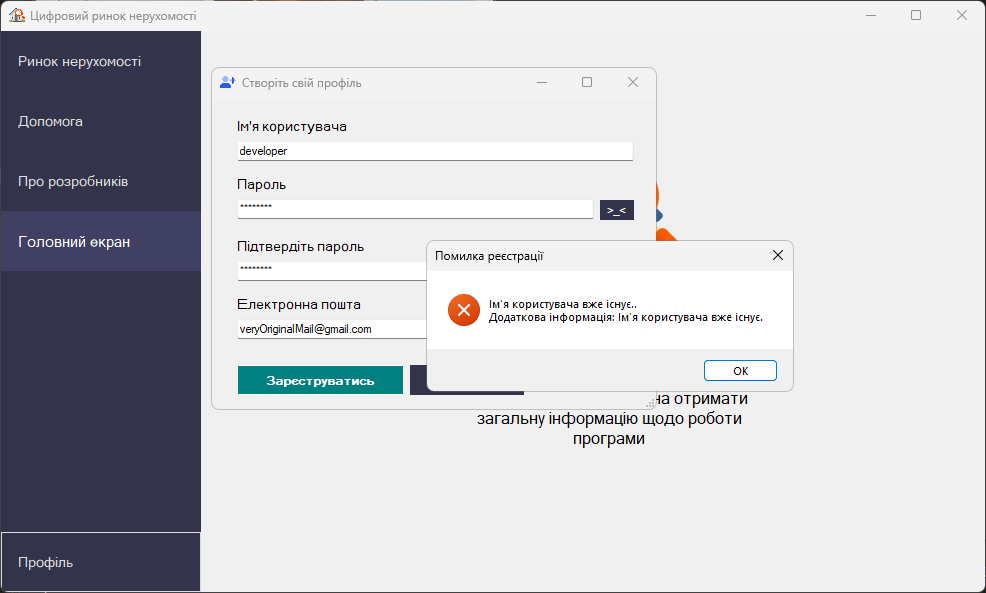


Рисунок 4.4 – Помилка при спробі зареєструвати обліоквий запис із вже існуючим ім’ям користувача

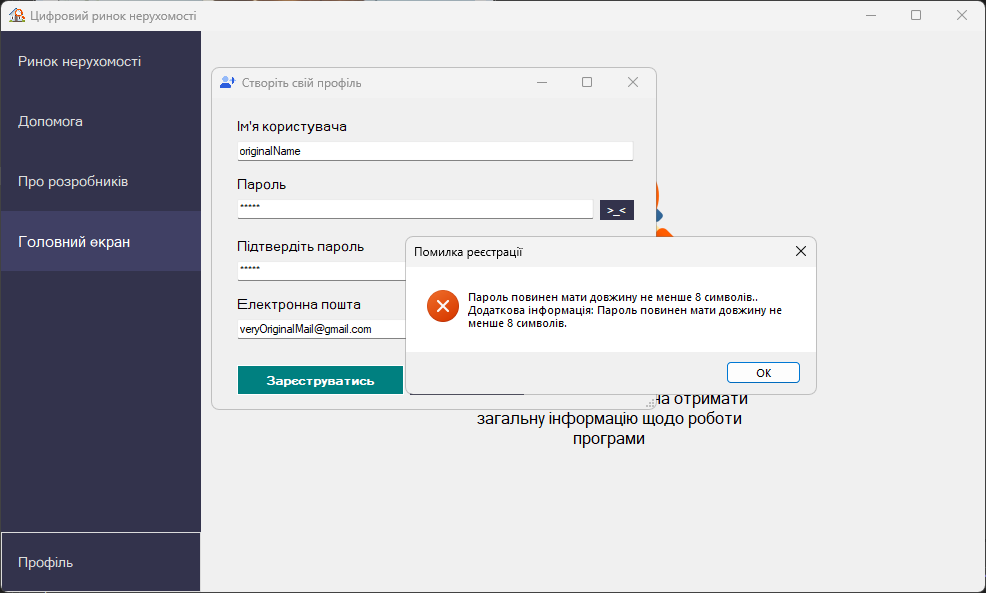


Рисунок 4.5 – Помилка при спробі реєстрації з надто коротким паролем

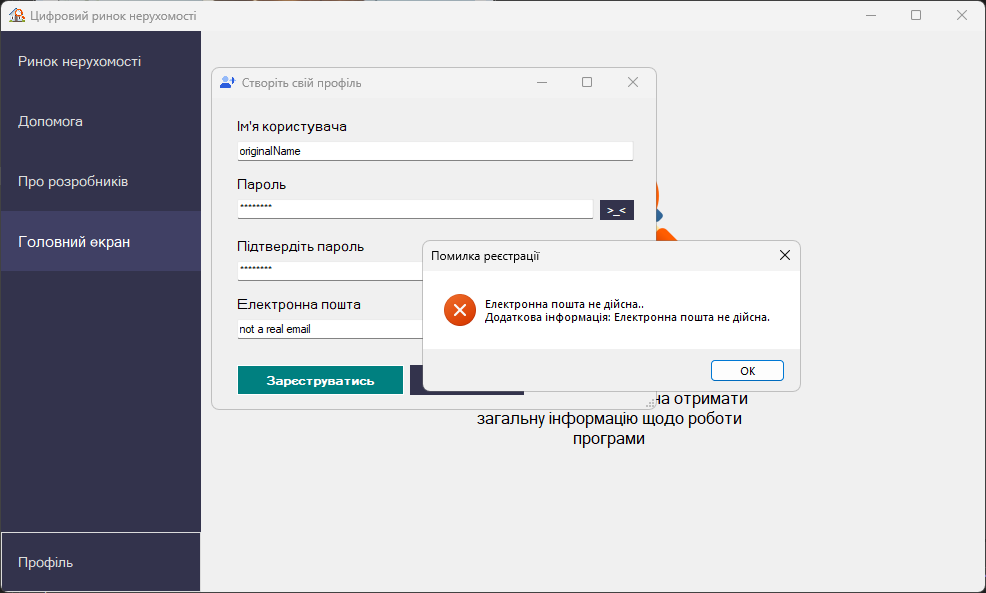


Рисунок 4.6 – Помилка при спробі зареєструватися із невірною електронною поштою

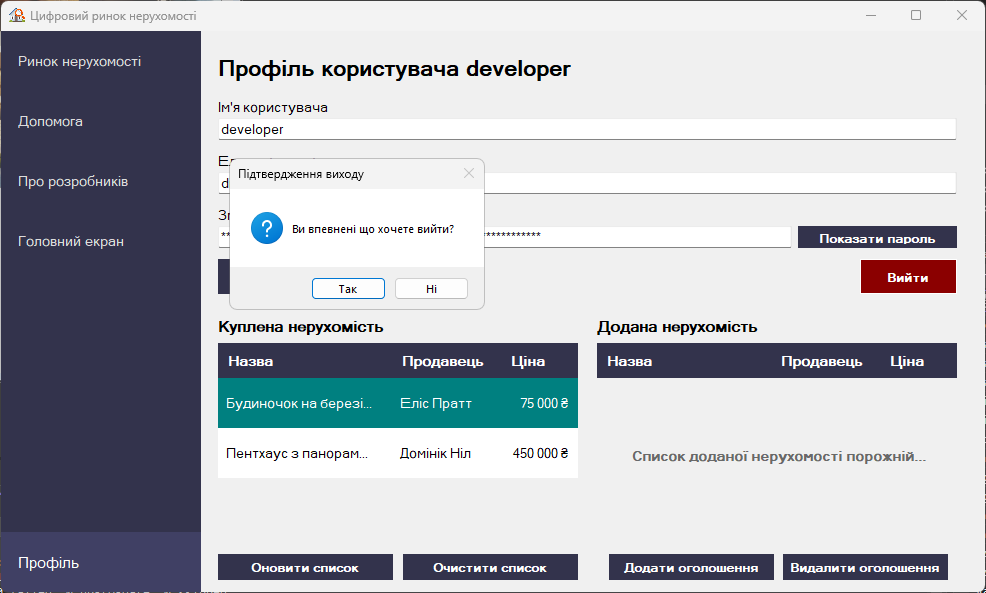


Рисунок 4.7 – Підтвердження виходу з облікового запису

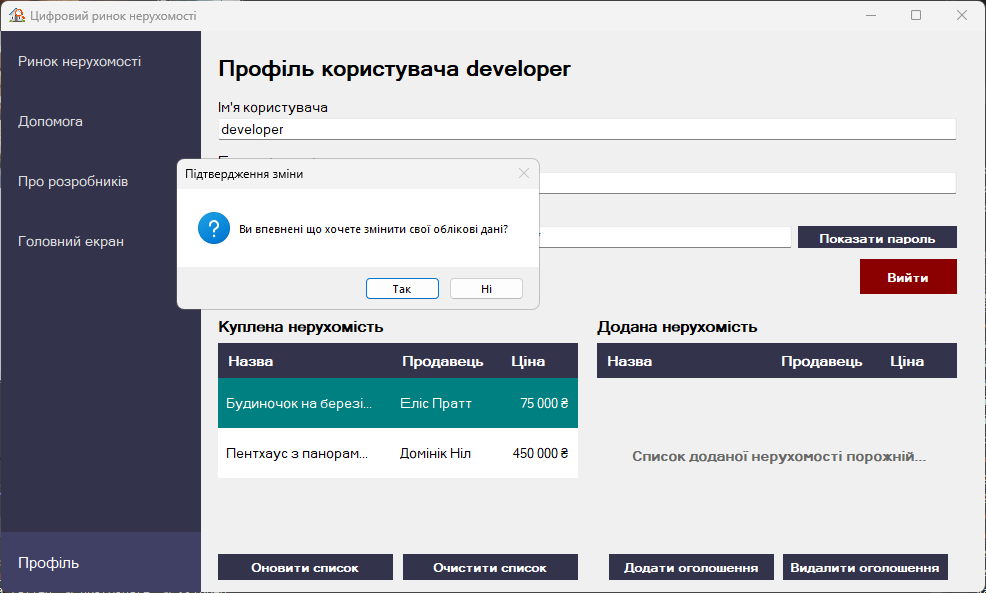
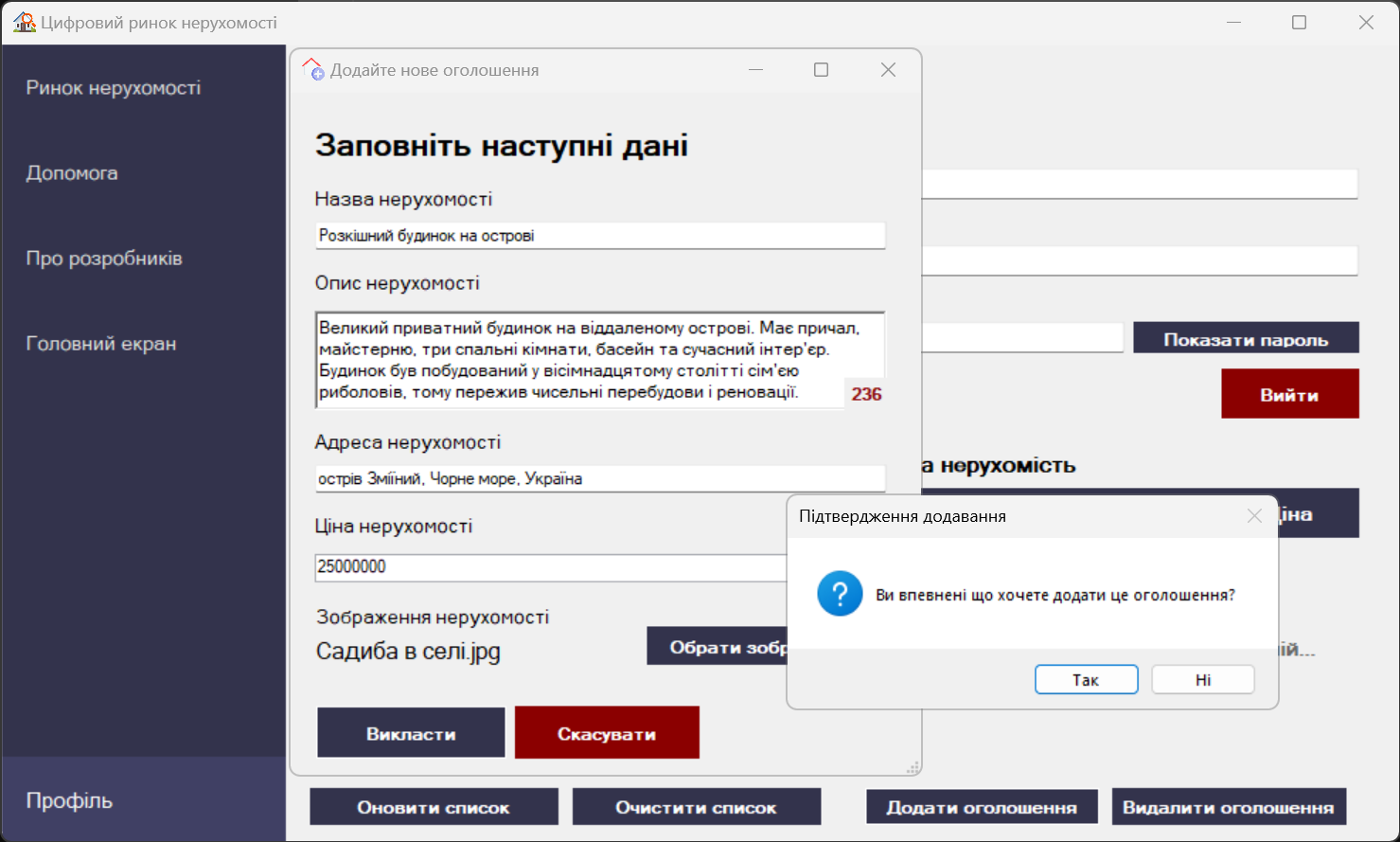


Рисунок 4.8 – Підтвердження збережння змін облікових даних профіля

  
Рисунок 4.9 – Підтвердження викладення нерухомості на ринок

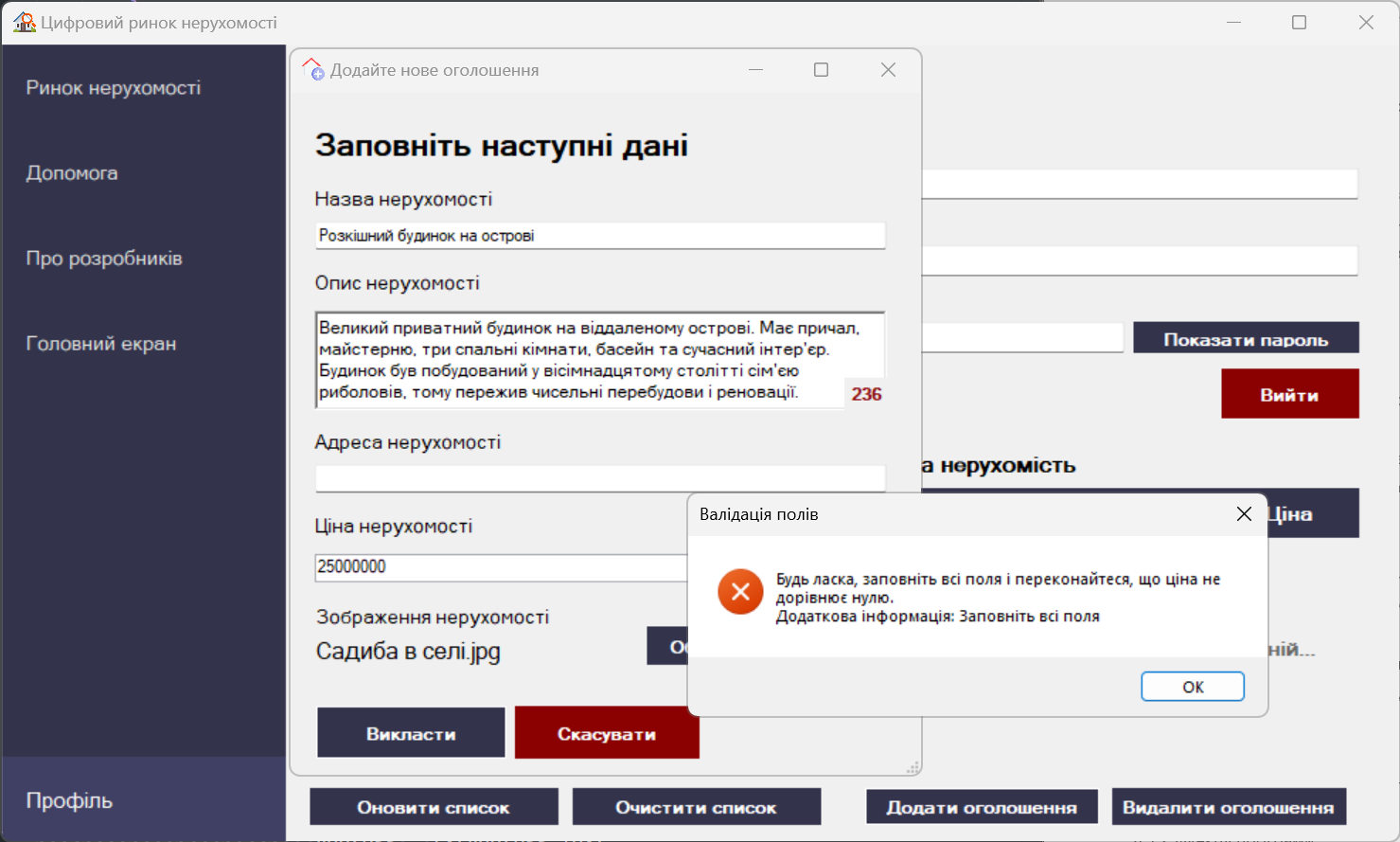


Рисунок 4.10 – Помилка при спробі додати нерухомість з незаповненим полем або декількома полями

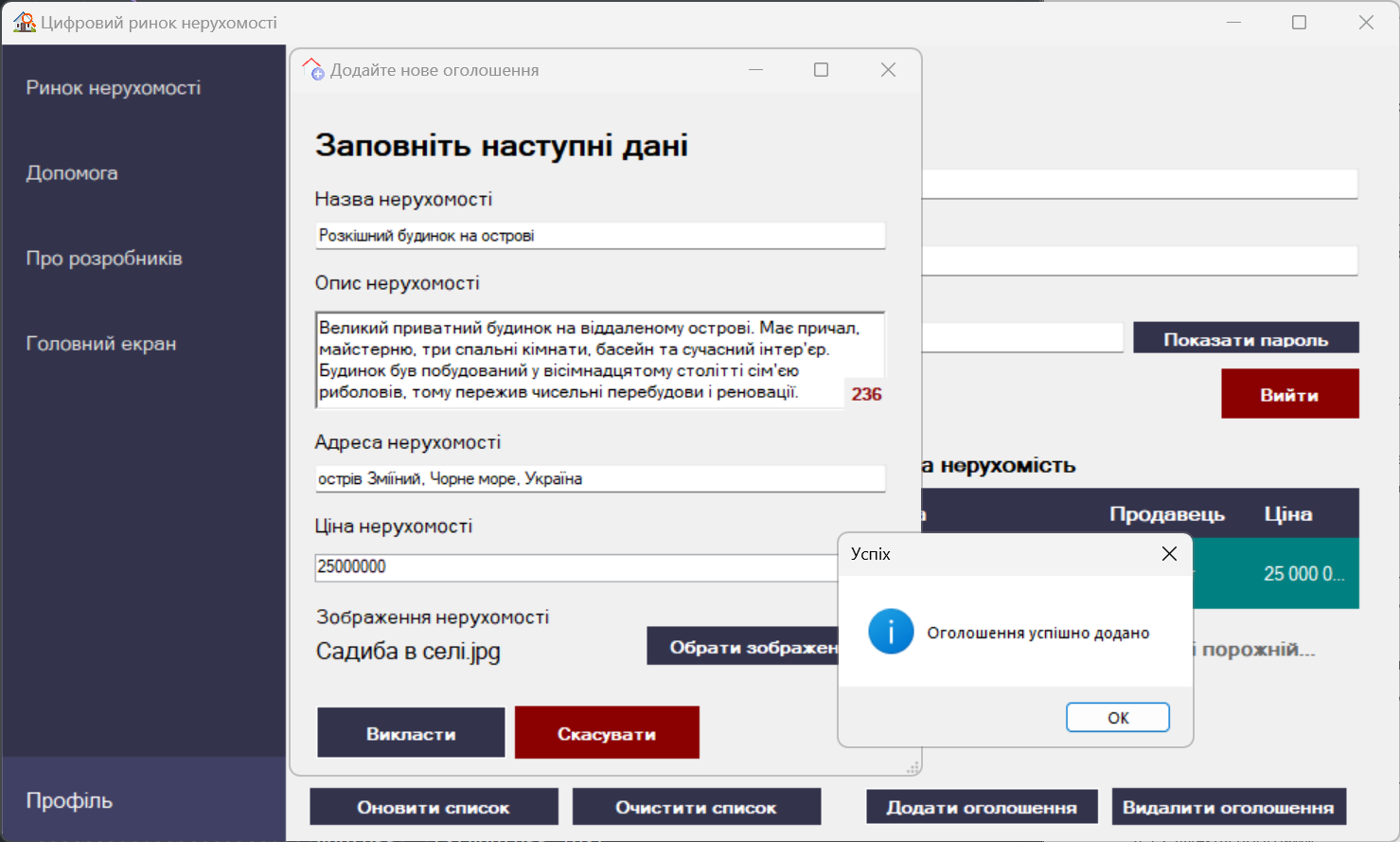


Рисунок 4.11 – Повідомлення підтвердження викладення нерухомості на ринок

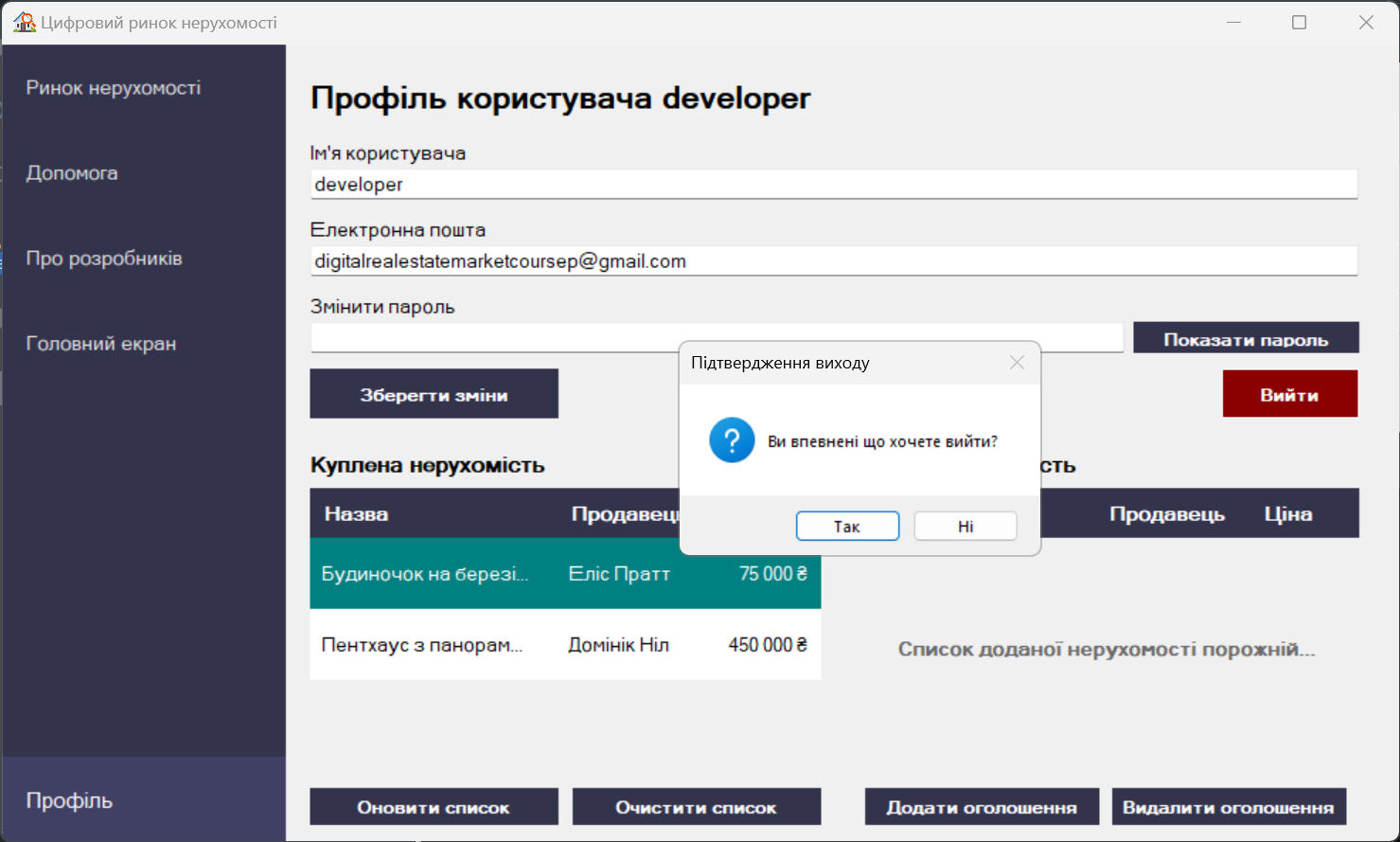


Рисунок 4.12 – Підтверження виходу з профілю

5.5 Довідка програми

Для отримання більш детальної інформації про застосунок "Цифровий ринок нерухомості" та його можливості можна звернутися до розділу "Довідка" програми. Цей розділ містить вичерпну документацію про функціональні можливості програми та способи їх використання.

Висновки

Проєкт "Цифровий ринок нерухомості" - це всебічний та збагачуючий досвід, який дозволив нам застосувати та розширити наші знання в різних сферах розробки програмного забезпечення. Ми використовували різноманітні технології та методології, в першу чергу зосередившись на C# Winforms .NET для розробки інтерфейсу та внутрішньої логіки застосунку.

Проєкт став значною можливістю для навчання для кожного з нас, оскільки нам довелося розуміти та реалізовувати складні алгоритми, обробляти винятки та ефективно управляти даними. Нам також довелося розробити надійну архітектуру системи, створивши різноманітні класи та кастомні елементи керування, щоб забезпечити безперебійну роботу для користувачів.

Проєкт був особливо корисним з точки зору розуміння тонкощів роботи з Winforms .NET. Ми отримали глибоке розуміння того, як створювати та керувати формами, елементами управління та обробниками подій, як взаємодіяти з базами даних та зовнішніми файлами.

Ми також дізналися про важливість командної роботи та управління проєктами. Ми змогли ефективно розподілити обов'язки між собою, і кожен учасник команди зосередився на конкретному аспекті роботи над проєктом. Такий розподіл обов'язків забезпечив нам ефективну роботу та уникнення потенційних конфліктів або затримок у виконанні робіт.

З точки зору корисності, проєкт дав нам практичне розуміння того, як функціонують цифрові ринки нерухомості. Ми змогли створити систему, яка імітує процес купівлі-продажу нерухомості, що може бути корисним для всіх, хто цікавиться галуззю нерухомості.

Загалом, проєкт "Цифровий ринок нерухомості" став цінним навчальним досвідом, який дозволив нам застосувати та розширити наші знання у сфері розробки програмного забезпечення. Навички та знання, отримані під час цього проєкту, безперечно стануть у пригоді в нашій подальшій роботі.

ПЕРЕЛІК ПОСИЛАНЬ

1) C# Tutorial - Connect MySQL Database ~ DataGrid View ~ Data Search ~ Bunifu Framework [Electronic resource]. – Access mode: <https://youtu.be/aRlTp4V9jjo>

Додаток А Діаграма класів

ЯКЩО Є

Додаток Б Код Програми

Б1 – FormMenu

using System;

using System.IO;

using System.Collections.Generic;

using System.ComponentModel;

using System.Drawing;

using System.Windows.Forms;

using code.Classes;

namespace code

{

    public partial class FormMenu : Form

    {

        private Button currentButton;

        private Form activeForm;

        BindingList<Advertisement> predefinedListings = new BindingList<Advertisement>();

        List<User> users = new List<User>();

        public FormMenu()

        {

            predefinedListings = FileHandler.readAdFromCSV();

            users = FileHandler.UserFileReader();

            InitializeComponent();

            OpenChildForm(new Forms.FormWelcome(), null);

            try

            {

                LoginManager.CurrentUser = new User();

                LoginManager.IsLoggedIn = false;

                GlobalData.AvailableListings = predefinedListings;

                GlobalData.Users = users;

                var imageFiles = Directory.GetFiles(@"../../../assets/images/real-estate-pictures/", "\*.jpg");

                SetImagesForListings(predefinedListings, imageFiles);

                foreach (var user in users)

                {

                    SetImagesForListings(user.BoughtListings, imageFiles);

                    SetImagesForListings(user.AddedListings, imageFiles);

                }

            }

            catch (Exception ex)

            {

                ExceptionManager.HandleException(ex, "Не вдалося завантажити дані. Спробуйте пізніше", "Помилка завантаження даних");

            }

        }

        // Метод для синхронізації шляхів до фото

        private void SetImagesForListings(BindingList<Advertisement> listings, string[] imageFiles)

        {

            for (int i = 0; i < listings.Count; i++)

            {

                for (int j = 0; j < imageFiles.Length; j++)

                {

                    if (@"../../../assets/images/real-estate-pictures/" + listings[i].Name + ".jpg" == imageFiles[j])

                    {

                        var ad = listings[i];

                        ad.ImagePath = imageFiles[j];

                    }

                }

            }

        }

        private void ActivateButton(object btnSender)

        {

            try

            {

                if (btnSender != null)

                {

                    if (currentButton != (Button)btnSender)

                    {

                        DisableButton();

                        currentButton = (Button)btnSender;

                        currentButton.BackColor = Color.FromArgb(64, 64, 100);

                        currentButton.ForeColor = Color.White;

                        currentButton.Font = new System.Drawing.Font("Microsoft Sans Serif", 11F, System.Drawing.FontStyle.Regular);

                    }

                }

            }

            catch (Exception ex)

            {

                ExceptionManager.HandleException(ex, "Не вдалося активувати кнопку. Спробуйте пізніше", "Помилка активації");

            }

        }

        private void DisableButton()

        {

            try

            {

                foreach (Control control in panelMenu.Controls)

                {

                    if (control is Button button)

                    {

                        button.BackColor = Color.FromArgb(51, 51, 76);

                        button.ForeColor = Color.Gainsboro;

                        button.Font = new Font("Microsoft Sans Serif", 10F, FontStyle.Regular);

                    }

                }

            }

            catch (Exception ex)

            {

                ExceptionManager.HandleException(ex, "Не вдалося деактивувати кнопку. Спробуйте пізніше", "Помилка деактивації");

            }

        }

        private void OpenChildForm(Form newForm, object buttonSender)

        {

            try

            {

                if (activeForm != null)

                {

                    activeForm.Close();

                }

                ActivateButton(buttonSender);

                activeForm = newForm;

                newForm.TopLevel = false;

                newForm.FormBorderStyle = FormBorderStyle.None;

                newForm.Dock = DockStyle.Fill;

                this.panelFormContainer.Controls.Add(newForm);

                this.panelFormContainer.Tag = newForm;

                newForm.BringToFront();

                newForm.Show();

            }

            catch (Exception ex)

            {

                ExceptionManager.HandleException(ex, "Не вдалося відкрити форму. Спробуйте пізніше", "Помилка відкриття");

            }

        }

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

        {

            OpenChildForm(new Forms.FormMarket(), sender);

        }

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

        {

            OpenChildForm(new Forms.FormHelp(), sender);

        }

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

        {

            OpenChildForm(new Forms.FormDevelopers(), sender);

        }

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

        {

            try

            {

                Forms.FormLogin login = new Forms.FormLogin();

                if (!LoginManager.IsLoggedIn)

                {

                    if (login.ShowDialog() == DialogResult.OK)

                    {

                        LoginManager.IsLoggedIn = true;

                        OpenChildForm(new Forms.FormProfile(), sender);

                    }

                    else

                    {

                        OpenChildForm(new Forms.FormWelcome(), sender);

                        return;

                    }

                }

                else

                {

                    OpenChildForm(new Forms.FormProfile(), sender);

                }

            }

            catch (Exception ex)

            {

                ExceptionManager.HandleException(ex, "Не вдалося авторизуватися. Спробуйте ще раз пізніше", "Помилка авторизації");

            }

        }

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

        {

            OpenChildForm(new Forms.FormWelcome(), sender);

        }

    }

}

Б2 - FormAddedListing

using System;

using System.Drawing;

using System.Reflection;

using System.Windows.Forms;

using code.Classes;

namespace code.Forms

{

    public partial class FormAddedListing : Form

    {

        public FormAddedListing(Advertisement listing)

        {

            InitializeComponent();

            try

            {

                this.Text = listing.Name;

                lblName.Text = listing.Name;

                lblSeller.Text = $"Це ваше оголошення";

                string truncatedDescription = listing.Description.Length > 60 ? listing.Description.Substring(0, 57) + "..." : listing.Description;

                this.lblDescription.Text = truncatedDescription;

                this.toolTipDescription.SetToolTip(this.lblDescription, listing.Description);

                lblAddress.Text = listing.Address;

                lblPrice.Text = $"${listing.Price} або {listing.Price \* Classes.СurrencyСonverter.GetDollarRate()} ₴";

                lblDateLastViewed.Text = $"Останній раз переглянуто {DateTime.Now:dd.MM.yyyy}";

                pictureBoxListingImage.Image = Image.FromFile(listing.ImagePath);

            }

            catch (Exception ex)

            {

                ExceptionManager.HandleException(ex, "Не вдалося завантажити деталі оголошення. Спробуйде закрити розділ і повторити спробу", "Помилка завантаження даних");

            }

        }

    }

}

Б3 – FormAddListing

using code.Classes;

using System;

using System.Drawing;

using System.IO;

using System.Windows.Forms;

namespace code.Forms

{

    public partial class FormAddListing : Form

    {

        private string selectedImagePath = string.Empty;

        private string destinationPath;

        private string destinationDirectory = @"../../../assets/images/real-estate-pictures";

        public FormAddListing()

        {

            InitializeComponent();

            lblImageName.Text = "Виберіть зображення";

        }

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

        {

            Close();

        }

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

        {

            try

            {

                if (string.IsNullOrWhiteSpace(txtName.Text) ||

                    string.IsNullOrWhiteSpace(txtDescription.Text) ||

                    string.IsNullOrWhiteSpace(txtAddress.Text) ||

                    numericUpDownPrice.Value == 0 ||

                    fileDialogImageUpload.FileName == string.Empty)

                {

                    ExceptionManager.HandleException(new Exception("Заповніть всі поля"), "Будь ласка, заповніть всі поля і переконайтеся, що ціна не дорівнює нулю", "Валідація полів");

                    return;

                }

                bool isConfirmed = ExceptionManager.Confirm("Ви впевнені що хочете додати це оголошення?", "Підтвердження додавання");

                if (isConfirmed)

                {

                    destinationPath = destinationDirectory + "/" + txtName.Text + ".jpg";

                    Advertisement listing = new Advertisement

                    {

                        Name = txtName.Text,

                        Description = txtDescription.Text,

                        Address = txtAddress.Text,

                        Price = Convert.ToInt32(numericUpDownPrice.Value),

                        Seller = LoginManager.CurrentUser.Name,

                        ImagePath = destinationPath

                    };

                    GlobalData.AvailableListings.Add(listing);

                    LoginManager.CurrentUser.AddedListings.Add(listing);

                    ExceptionManager.ShowInfo("Оголошення успішно додано", "Успіх");

                    Close();

                }

            }

            catch (Exception ex)

            {

                ExceptionManager.HandleException(ex, "Не вдалося додати оголошення. Спробуйте пізніше", "Помилка додавання");

            }

        }

        private void txtDescription\_TextChanged(object sender, EventArgs e)

        {

            try

            {

                var characterCount = txtDescription.Text.Length;

                lblDescriptionCharactersCount.Text = characterCount.ToString();

                if (characterCount == 250)

                {

                    lblDescriptionCharactersCount.ForeColor = Color.FromArgb(97, 9, 9);

                }

                else if (characterCount > 230)

                {

                    lblDescriptionCharactersCount.ForeColor = Color.FromArgb(144, 14, 14);

                }

                else

                {

                    lblDescriptionCharactersCount.ForeColor = Color.Black;

                }

            }

            catch (Exception ex)

            {

                ExceptionManager.HandleException(ex, "Не вдалося оновити кількість символів. Спробуйте пізніше", "Помилка перевірки довжини опису");

            }

        }

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

        {

            try

            {

                OpenFileDialog fileDialogImageUpload = new OpenFileDialog();

                fileDialogImageUpload.Filter = "Image Files|\*.jpg;\*.jpeg;\*.png;";

                if (fileDialogImageUpload.ShowDialog() == DialogResult.OK)

                {

                    selectedImagePath = fileDialogImageUpload.FileName;

                    string fileName = Path.GetFileName(selectedImagePath);

                    int maxLength = 26;

                    if (fileName.Length > maxLength)

                    {

                        fileName = fileName.Substring(0, maxLength - 3) + "...";

                    }

                    lblImageName.Text = fileName;

                    string destinationDirectory = @"../../../assets/images/real-estate-pictures";

                    string newFileName = txtName.Text + ".jpg";

                    string destinationPath = Path.Combine(destinationDirectory, newFileName);

                    File.Copy(selectedImagePath, destinationPath, true);

                }

                else

                {

                    selectedImagePath = string.Empty;

                    lblImageName.Text = "Виберіть зображення";

                }

            }

            catch (Exception ex)

            {

                ExceptionManager.HandleException(ex, "Не вдалося завантажити фото. Спробуйте пізніше", "Помилка завантаження фото");

            }

        }

    }

}

Б4 – FormBoughtListing

using System;

using System.Drawing;

using System.Windows.Forms;

using code.Classes;

namespace code.Forms

{

    public partial class FormBoughtListing : Form

    {

        public FormBoughtListing(Advertisement listing)

        {

            InitializeComponent();

            try

            {

                this.Text = listing.Name;

                lblName.Text = listing.Name;

                lblSeller.Text = $"Куплено від: {listing.Seller}";

                string truncatedDescription = listing.Description.Length > 60 ? listing.Description.Substring(0, 57) + "..." : listing.Description;

                this.lblDescription.Text = truncatedDescription;

                this.toolTipDescription.SetToolTip(this.lblDescription, listing.Description);

                lblAddress.Text = listing.Address;

                lblPrice.Text = $"${listing.Price} або {listing.Price \* Classes.СurrencyСonverter.GetDollarRate()} ₴";

                lblOwner.Text = $"Власник: {LoginManager.CurrentUser.Name}";

                pictureBoxListingImage.Image = Image.FromFile(listing.ImagePath);

            }

            catch (Exception ex)

            {

                ExceptionManager.HandleException(ex, "Не вдалося завантажити деталі оголошення. Спробуйте пізніше", "Помилка завантаження даних");

            }

        }

    }

}

Б5 – FormBuyListing

using System;

using System.Drawing;

using System.Windows.Forms;

using code.Classes;

namespace code.Forms

{

    public partial class FormBuyListing : Form

    {

        private readonly Advertisement listing;

        private readonly FormMarket formMarket;

        public FormBuyListing(Advertisement listing, FormMarket formMarket)

        {

            InitializeComponent();

            try

            {

                this.formMarket = formMarket;

                this.listing = listing;

                this.Text = listing.Name;

                this.lblName.Text = listing.Name;

                string truncatedDescription = listing.Description.Length > 60 ? listing.Description.Substring(0, 57) + "..." : listing.Description;

                this.lblDescription.Text = truncatedDescription;

                this.toolTipDescription.SetToolTip(this.lblDescription, listing.Description);

                this.lblAddress.Text = $"{listing.Address} (натисніть щоб перейти на мапу)";

                this.lblPrice.Text = $"${listing.Price} або {listing.Price \* Classes.СurrencyСonverter.GetDollarRate()} ₴";

                this.lblSeller.Text = listing.Seller;

                this.pictureBoxListingImage.Image = Image.FromFile(listing.ImagePath);

            }

            catch (Exception ex)

            {

                ExceptionManager.HandleException(ex, "Не вдалося завантажити деталі оголошення. Спробуйте пізніше", "Помилка завантаження даних");

            }

        }

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

        {

            Close();

        }

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

        {

            try

            {

                if (LoginManager.IsLoggedIn)

                {

                    if (LoginManager.CurrentUser.Name == listing.Seller)

                    {

                        ExceptionManager.HandleException(new Exception("Неможливо купити свою власну нерухомість."), "Ви не можете купувати свою нерухомість.", "Помилка купівлі");

                        return;

                    }

                    LoginManager.CurrentUser.BoughtListings.Add(listing);

                    formMarket.RemoveListing(listing);

                }

                else

                {

                    ExceptionManager.HandleException(new Exception("Авторизуйтесь перш ніж купувати нерухомість."), "Ви повинні авторизуватися, щоб купити нерухомість. Будь ласка, увійдіть у свій акаунт і спробуйте ще раз", "Помилка купівлі");

                    return;

                }

                Close();

            }

            catch (Exception ex)

            {

                ExceptionManager.HandleException(ex, "Не вдалося купити нерухомість. Спробуйте пізніше", "Помилка купівлі");

            }

        }

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

        {

            Classes.ShowMap.GoToMap(listing.Address);

        }

    }

}

Б6 – FormCodeConfirmation

using System;

using System.Windows.Forms;

namespace code.Forms

{

    public partial class FormCodeConfirmation : Form

    {

        public FormCodeConfirmation()

        {

            InitializeComponent();

            this.Text = string.Empty;

            this.ControlBox = false;

        }

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

        {

            DialogResult = DialogResult.OK;

            Close();

        }

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

        {

            DialogResult = DialogResult.Cancel;

            Close();

        }

        public string getVerifyCode()

        {

            return inputTextBox.Text;

        }

    }

}

Б7 – FormDevelopers

using System;

using System.Windows.Forms;

using code.Classes;

namespace code.Forms

{

    public partial class FormDevelopers : Form

    {

        public FormDevelopers()

        {

            InitializeComponent();

            this.Text = string.Empty;

            this.ControlBox = false;

        }

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

        {

            try

            {

                System.Diagnostics.Process.Start("https://github.com/amaiboy/digital-real-estate-market-course-project");

            }

            catch (Exception ex)

            {

                ExceptionManager.HandleException(ex, "Не вдалося відкрити сторінку в браузері. Спробуйте ще раз пізніше", "Помилка відкриття");

            }

        }

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

        {

            try

            {

                System.Diagnostics.Process.Start("https://github.com/amaiboy/digital-real-estate-market-course-project/blob/dev/LICENSE");

            }

            catch (Exception ex)

            {

                ExceptionManager.HandleException(ex, "Не вдалося відкрити сторінку в браузері. Спробуйте ще раз пізніше", "Помилка відкриття");

            }

        }

    }

}

Б8 – FormHelp

using System.Windows.Forms;

namespace code.Forms

{

    public partial class FormHelp : Form

    {

        public FormHelp()

        {

            InitializeComponent();

            this.Text = string.Empty;

            this.ControlBox = false;

        }

    }

}

Б9 – FormLogin

using code.Classes;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Windows.Forms;

namespace code.Forms

{

    public partial class FormLogin : Form

    {

        public FormLogin()

        {

            InitializeComponent();

            txtPassword.PasswordChar = '\*';

        }

        public static int LoginAttempts = 0;

        public User ValidateLogin(string username, string password)

        {

            try

            {

                return GlobalData.Users.FirstOrDefault(user => user.Name == username && LoginManager.verifyPassword(password, user.Password));

            }

            catch (Exception ex)

            {

                ExceptionManager.HandleException(ex, "Не вдалося авторизуватися. Спробуйте ще раз пізніше", "Помилка авторизації");

                return null;

            }

        }

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

        {

            var username = txtUsername.Text;

            var password = txtPassword.Text;

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

            if (string.IsNullOrWhiteSpace(username) || string.IsNullOrWhiteSpace(password))

            {

                errors.Add("Ім'я користувача або пароль не можуть бути порожніми.");

            }

            if (username.Length < 3 || username.Length > 16)

            {

                errors.Add("Ім'я користувача має містити від 3 до 16 символів.");

            }

            if (password.Length < 8)

            {

                errors.Add("Пароль має містити від 8 до 16 символів.");

            }

            if (errors.Count > 0)

            {

                ExceptionManager.HandleException(new Exception(string.Join("\n", errors)), string.Join("\n", errors), "Помилка авторизації");

                LoginAttempts++;

                if (LoginAttempts >= 3)

                {

                    ExceptionManager.ShowInfo("Ви зробили надто багато невдалих спроб входу за короткий час. Спробуйте пізніше.", "Помилка входу");

                    this.DialogResult = DialogResult.Cancel;

                }

                return;

            }

            try

            {

                var user = ValidateLogin(username, password);

                if (user == null)

                {

                    ExceptionManager.HandleException(new Exception("Невірний логін або пароль."), "Невірний логін або пароль.", "Помилка авторизації");

                    LoginAttempts++;

                    return;

                }

                this.DialogResult = DialogResult.OK;

                LoginManager.CurrentUser = user;

                LoginManager.IsLoggedIn = true;

            }

            catch (Exception ex)

            {

                ExceptionManager.HandleException(ex, "Виникла помилка під час входу", "Помилка входу");

            }

        }

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

        {

            try

            {

                FormSignUp signUpForm = new FormSignUp();

                this.Hide();

                if (signUpForm.ShowDialog() == DialogResult.OK)

                {

                    var username = signUpForm.Username;

                    var password = signUpForm.Password;

                    var email = signUpForm.Email;

                    User user = new User(username, email, password);

                    GlobalData.Users.Add(user);

                    LoginManager.CurrentUser = user;

                    LoginManager.IsLoggedIn = true;

                    this.DialogResult = DialogResult.OK;

                }

                else

                {

                    this.DialogResult = DialogResult.Cancel;

                }

            }

            catch (Exception ex)

            {

                ExceptionManager.HandleException(ex, "Виникла помилка під час реєстрації", "Помилка реєстрації");

            }

        }

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

        {

            try

            {

                if (txtPassword.PasswordChar == '\*')

                {

                    txtPassword.PasswordChar = '\0';

                    btnTogglePasswordVisibility.Text = "0\_0";

                }

                else

                {

                    txtPassword.PasswordChar = '\*';

                    btnTogglePasswordVisibility.Text = ">\_<";

                }

            }

            catch (Exception ex)

            {

                ExceptionManager.HandleException(ex, "Виникла помилка при перемиканні видимості пароля", "Помилка видимості пароля");

            }

        }

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

        {

            try

            {

                FormRestore restoreForm = new FormRestore();

                this.Hide();

                if (restoreForm.ShowDialog() == DialogResult.OK)

                {

                    this.DialogResult = DialogResult.OK;

                }

                else

                {

                    this.DialogResult = DialogResult.Cancel;

                }

            }

            catch (Exception ex)

            {

                ExceptionManager.HandleException(ex, "Виникла помилка під час зміни паролю", "Помилка зміни паролю");

            }

        }

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

        {

            Close();

        }

    }

}

Б10 – FormMarket

using System;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Windows.Forms;

using code.Classes;

namespace code.Forms

{

    public partial class FormMarket : Form

    {

        public FormMarket()

        {

            InitializeComponent();

            this.Text = string.Empty;

            this.ControlBox = false;

            dataGridViewListings.DataSource = GlobalData.AvailableListings;

            dataGridViewListings.ColumnHeadersDefaultCellStyle.Font = new Font("Microsoft Sans Serif", 10F, System.Drawing.FontStyle.Bold);

            Padding cellPadding = new Padding(5, 5, 5, 5);

            dataGridViewListings.ColumnHeadersDefaultCellStyle.Padding = cellPadding;

            foreach (DataGridViewColumn column in dataGridViewListings.Columns)

            {

                column.DefaultCellStyle.Padding = cellPadding;

            }

            dropdownSortByName.SelectedIndex = 0;

            dropdownSortBySeller.SelectedIndex = 0;

            dropdownSortByPrice.SelectedIndex = 0;

        }

        public void AddListing(Advertisement listing)

        {

            try

            {

                var updatedListings = GlobalData.AvailableListings;

                updatedListings.Add(listing);

                GlobalData.AvailableListings = updatedListings;

                dataGridViewListings.DataSource = GlobalData.AvailableListings;

            }

            catch (Exception ex)

            {

                ExceptionManager.HandleException(ex, "Не вдалося додати об'єкт. Спробуйте ще раз пізніше", "Помилка додавання об'єкту");

            }

        }

        public void RemoveListing(Advertisement listing)

        {

            try

            {

                var updatedListings = GlobalData.AvailableListings;

                updatedListings.Remove(listing);

                GlobalData.AvailableListings = updatedListings;

                dataGridViewListings.DataSource = GlobalData.AvailableListings;

            }

            catch (Exception ex)

            {

                ExceptionManager.HandleException(ex, "Не вдалося видалити об'єкт. Спробуйте ще раз пізніше", "Помилка видалення об'єкту");

            }

        }

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

        {

            var searchQuery = txtSearch.Text;

            try

            {

                var filteredList = GlobalData.AvailableListings.Where(l => l.Name.Contains(searchQuery) || l.Description.Contains(searchQuery));

                dataGridViewListings.DataSource = filteredList.ToList();

            }

            catch (Exception ex)

            {

                ExceptionManager.HandleException(ex, "Помилка при пошуку оголошення. Спробуйте ще раз пізніше", "Помилка пошуку об'єкту");

            }

        }

        private void txtSearch\_Enter(object sender, EventArgs e)

        {

            if (txtSearch.Text == "Пошук за назвою чи описом...")

            {

                txtSearch.Text = string.Empty;

                txtSearch.ForeColor = System.Drawing.Color.Black;

            }

        }

        private void dataGridViewListings\_CellDoubleClick(object sender, DataGridViewCellEventArgs e)

        {

            try

            {

                var listing = (Advertisement)dataGridViewListings.SelectedRows[0].DataBoundItem;

                FormBuyListing listingForm = new FormBuyListing(listing, this);

                listingForm.ShowDialog();

            }

            catch (Exception ex)

            {

                ExceptionManager.HandleException(ex, "Не вдалося відкрити форму. Спробуйте ще раз пізніше", "Помилка відкриття форми");

            }

        }

        private void txtSearch\_KeyDown(object sender, KeyEventArgs e)

        {

            if (e.KeyCode == Keys.Enter)

            {

                // Call the same method or perform the same operation as btnSearch does

                btnSearch\_Click(sender, e);

                // Prevent the beep sound

                e.SuppressKeyPress = true;

                e.Handled = true;

            }

        }

        private void reshowHint(ComboBox dropdown, string hint)

        {

            try

            {

                if (!dropdown.Items.Contains(hint))

                {

                    dropdown.Items.Insert(0, hint);

                    dropdown.ForeColor = System.Drawing.Color.Gray;

                    dropdown.SelectedIndex = 0;

                }

            }

            catch (Exception ex)

            {

                ExceptionManager.HandleException(ex, "Не вдалося показати підказку.", "Помилка показу підказки");

            }

        }

        private void hideHint(ComboBox dropdown, string hint)

        {

            try

            {

                if (dropdown.Items.Contains(hint))

                {

                    dropdown.Items.Remove(hint);

                    dropdown.ForeColor = System.Drawing.Color.Black;

                }

            }

            catch (Exception ex)

            {

                ExceptionManager.HandleException(ex, "Не вдалося приховати підказку.", "Помилка приховання підказки");

            }

        }

        private void dropdownSortByName\_DropDown(object sender, EventArgs e)

        {

            hideHint(dropdownSortByName, "Сортування за назвою");

        }

        private void dropdownSortBySeller\_DropDown(object sender, EventArgs e)

        {

            hideHint(dropdownSortBySeller, "Сортування за продавцем");

        }

        private void dropdownSortByPrice\_DropDown(object sender, EventArgs e)

        {

            hideHint(dropdownSortByPrice, "Сортування за ціною");

        }

        private void dropdownSortByName\_DropDownClosed(object sender, EventArgs e)

        {

            if (dropdownSortByName.SelectedIndex == -1)

            {

                reshowHint(dropdownSortByName, "Сортування за назвою");

            }

        }

        private void dropdownSortBySeller\_DropDownClosed(object sender, EventArgs e)

        {

            if (dropdownSortBySeller.SelectedIndex == -1)

            {

                reshowHint(dropdownSortBySeller, "Сортування за продавцем");

            }

        }

        private void dropdownSortByPrice\_DropDownClosed(object sender, EventArgs e)

        {

            if (dropdownSortByPrice.SelectedIndex == -1)

            {

                reshowHint(dropdownSortByPrice, "Сортування за ціною");

            }

        }

        private void dropdownSortByName\_SelectedIndexChanged(object sender, EventArgs e)

        {

            try

            {

                if (dropdownSortByName.SelectedItem.ToString() != "Сортування за назвою")

                {

                    Classes.AlgorithmManager.SortListings(dropdownSortByName, "byName", dataGridViewListings);

                    reshowHint(dropdownSortByPrice, "Сортування за ціною");

                    dropdownSortByPrice.SelectedIndex = 0;

                    reshowHint(dropdownSortBySeller, "Сортування за продавцем");

                    dropdownSortBySeller.SelectedIndex = 0;

                }

            }

            catch (Exception ex)

            {

                ExceptionManager.HandleException(ex, "Не вдалося відсортування оголошення", "Помилка сортування");

            }

        }

        private void dropdownSortByPrice\_SelectedIndexChanged(object sender, EventArgs e)

        {

            try

            {

                if (dropdownSortByPrice.SelectedItem.ToString() != "Сортування за ціною")

                {

                    Classes.AlgorithmManager.SortListings(dropdownSortByPrice, "byPrice", dataGridViewListings);

                    reshowHint(dropdownSortByName, "Сортування за назвою");

                    dropdownSortByName.SelectedIndex = 0;

                    reshowHint(dropdownSortBySeller, "Сортування за продавцем");

                    dropdownSortBySeller.SelectedIndex = 0;

                }

            }

            catch (Exception ex)

            {

                ExceptionManager.HandleException(ex, "Не вдалося відсортування оголошення", "Помилка сортування");

            }

        }

        private void dropdownSortBySeller\_SelectedIndexChanged(object sender, EventArgs e)

        {

            try

            {

                if (dropdownSortBySeller.SelectedItem.ToString() != "Сортування за продавцем")

                {

                    Classes.AlgorithmManager.SortListings(dropdownSortBySeller, "bySeller", dataGridViewListings);

                    reshowHint(dropdownSortByPrice, "Сортування за ціною");

                    dropdownSortByPrice.SelectedIndex = 0;

                    reshowHint(dropdownSortByName, "Сортування за назвою");

                    dropdownSortByName.SelectedIndex = 0;

                }

            }

            catch (Exception ex)

            {

                ExceptionManager.HandleException(ex, "Не вдалося відсортування оголошення", "Помилка сортування");

            }

        }

    }

}

Б11 – FormProfile

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Drawing;

using System.Windows.Forms;

using code.Classes;

using System.IO;

namespace code.Forms

{

    public partial class FormProfile : Form

    {

        private Form activeForm = null;

        public FormProfile()

        {

            InitializeComponent();

            this.Text = string.Empty;

            this.ControlBox = false;

            this.activeForm = this;

            txtUsername.Text = LoginManager.CurrentUser.Name;

            txtEmail.Text = LoginManager.CurrentUser.Email;

            txtPassword.Text = "";

            txtPassword.PasswordChar = '\*';

            formTitle.Text = $"Профіль користувача {LoginManager.CurrentUser.Name}";

            var headingFont = new Font("Microsoft Sans Serif", 10F, System.Drawing.FontStyle.Bold);

            Padding cellPadding = new Padding(5, 5, 5, 5);

            dataGridViewBoughtListings.DataSource = LoginManager.CurrentUser.BoughtListings;

            dataGridViewBoughtListings.ColumnHeadersDefaultCellStyle.Font = headingFont;

            dataGridViewBoughtListings.ColumnHeadersDefaultCellStyle.Padding = cellPadding;

            foreach (DataGridViewColumn column in dataGridViewBoughtListings.Columns)

            {

                column.DefaultCellStyle.Padding = cellPadding;

            }

            dataGridViewAddedListings.DataSource = LoginManager.CurrentUser.AddedListings;

            dataGridViewAddedListings.ColumnHeadersDefaultCellStyle.Font = headingFont;

            dataGridViewAddedListings.ColumnHeadersDefaultCellStyle.Padding = cellPadding;

            foreach (DataGridViewColumn column in dataGridViewAddedListings.Columns)

            {

                column.DefaultCellStyle.Padding = cellPadding;

            }

            if (dataGridViewBoughtListings.Rows.Count != 0)

            {

                lblEmptyBoughtListings.Visible = false;

            }

            if (dataGridViewAddedListings.Rows.Count != 0)

            {

                lblEmptyAddedListings.Visible = false;

            }

        }

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

        {

            if (txtPassword.PasswordChar == '\*')

            {

                txtPassword.PasswordChar = '\0';

            }

            else

            {

                txtPassword.PasswordChar = '\*';

            }

        }

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

        {

            var newUsername = txtUsername.Text;

            var newPassword = txtPassword.Text;

            var newEmail = txtEmail.Text;

            var currentUsername = LoginManager.CurrentUser.Name;

            var currentPassword = LoginManager.CurrentUser.Password;

            var currentEmail = LoginManager.CurrentUser.Email;

            if (newUsername == currentUsername && newEmail == currentEmail && newPassword.Length == 0)

            {

                ExceptionManager.ShowInfo("Ви не внесли жодних змін.", "Внесіть зміни і спробуйте ще раз");

            }

            else if (!string.IsNullOrEmpty(newUsername) && !string.IsNullOrEmpty(newEmail))

            {

                bool isConfirmed = ExceptionManager.Confirm("Ви впевнені що хочете змінити свої облікові дані?", "Підтвердження зміни");

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

                if (isConfirmed)

                {

                    if (newUsername != LoginManager.CurrentUser.Name)

                    {

                        foreach (var user in GlobalData.Users)

                        {

                            if (user.Name == newUsername)

                            {

                                errors.Add("Таке ім'я користувача вже існує.");

                                break;

                            }

                        }

                    }

                    if (newEmail != LoginManager.CurrentUser.Email)

                    {

                        foreach (var user in GlobalData.Users)

                        {

                            if (user.Email == newEmail)

                            {

                                errors.Add("Така пошта вже використовується.");

                                break;

                            }

                        }

                    }

                    if (newPassword.Length != 0 && LoginManager.hashPassword(newPassword) == LoginManager.CurrentUser.Password)

                    {

                        errors.Add("Пароль має бути різним.");

                    }

                    if (newUsername.Length < 3 || newUsername.Length > 16)

                    {

                        errors.Add("Ім'я користувача має містити від 3 до 16 символів.");

                    }

                    if (newPassword.Length != 0 && newPassword.Length < 8)

                    {

                        errors.Add("Пароль повинен мати довжину не менше 8 символів.");

                    }

                    if (newEmail != LoginManager.CurrentUser.Email)

                    {

                        try

                        {

                            if (!string.IsNullOrWhiteSpace(newEmail) && newEmail != LoginManager.CurrentUser.Email)

                            {

                                var validEmail = new System.Net.Mail.MailAddress(newEmail);

                            }

                            else

                            {

                                throw new FormatException();

                            }

                        }

                        catch (FormatException)

                        {

                            errors.Add("Електронна пошта не дійсна.");

                        }

                    }

                    if (errors.Count == 0)

                    {

                        try

                        {

                            string verificationCode = LoginManager.generateVerificationCode();

                            Console.WriteLine(verificationCode);

                            \_ = LoginManager.sendVerificationCodeToEmail(newEmail, verificationCode);

                            FormCodeConfirmation InputForm = new FormCodeConfirmation();

                            DialogResult result = InputForm.ShowDialog();

                            if (result == DialogResult.OK)

                            {

                                string userEnteredVerificationCode = InputForm.getVerifyCode();

                                if (userEnteredVerificationCode == verificationCode)

                                {

                                }

                                else

                                {

                                    throw new Exception();

                                }

                            }

                            else

                            {

                                throw new Exception();

                            }

                        }

                        catch (Exception)

                        {

                            errors.Add("Електронна пошта не веріфікована.");

                        }

                    }

                    if (errors.Count > 0)

                    {

                        ExceptionManager.HandleException(new Exception(string.Join("\n", errors)), string.Join("\n", errors), "Помилка реєстрації");

                        return;

                    }

                    if (LoginManager.CurrentUser.Name != newUsername)

                    {

                        for (int i = 0; i < LoginManager.CurrentUser.AddedListings.Count; i++)

                        {

                            LoginManager.CurrentUser.AddedListings[i].Seller = newUsername;

                        }

                        foreach (var ad in GlobalData.AvailableListings)

                        {

                            if (ad.Seller == LoginManager.CurrentUser.Name)

                            {

                                ad.Seller = newUsername;

                            }

                        }

                        LoginManager.CurrentUser.Name = newUsername;

                    }

                    if (newPassword.Length != 0)

                    {

                        LoginManager.CurrentUser.Password = LoginManager.hashPassword(newPassword);

                    }

                    LoginManager.CurrentUser.Email = newEmail;

                    txtPassword.Text = "";

                    ExceptionManager.ShowInfo("Ви успішно змінили свої облікові дані!", "Зміни внесено успішно");

                }

            }

        }

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

        {

            try

            {

                dataGridViewBoughtListings.DataSource = null;

                dataGridViewBoughtListings.DataSource = LoginManager.CurrentUser.BoughtListings;

                if (dataGridViewBoughtListings.Rows.Count != 0)

                {

                    lblEmptyBoughtListings.Visible = false;

                }

            }

            catch (Exception ex)

            {

                ExceptionManager.HandleException(ex, "Не вдалося оновити список куплених оголошень. Спробуйте ще раз пізніше", "Помилка оновлення списку оголошень");

            }

        }

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

        {

            try

            {

                dataGridViewBoughtListings.DataSource = null;

                LoginManager.CurrentUser.BoughtListings.Clear();

                dataGridViewBoughtListings.DataSource = LoginManager.CurrentUser.BoughtListings;

                lblEmptyBoughtListings.Visible = true;

            }

            catch (Exception ex)

            {

                ExceptionManager.HandleException(ex, "Не вдалося очистити список куплених оголошень. Спробуйте ще раз пізніше", "Помилка очищення списку оголошень");

            }

        }

        private void dataGridViewBoughtListings\_CellDoubleClick(object sender, DataGridViewCellEventArgs e)

        {

            try

            {

                var selectedListing = (Advertisement)dataGridViewBoughtListings.SelectedRows[0].DataBoundItem;

                FormBoughtListing listingForm = new FormBoughtListing(selectedListing);

                listingForm.ShowDialog();

            }

            catch (Exception ex)

            {

                ExceptionManager.HandleException(ex, "Не вдалося відкрити форму. Спробуйте ще раз пізніше", "Помилка відкриття форми");

            }

        }

        private void dataGridViewAddedListings\_CellDoubleClick(object sender, DataGridViewCellEventArgs e)

        {

            try

            {

                var selectedListing = (Advertisement)dataGridViewAddedListings.SelectedRows[0].DataBoundItem;

                FormAddedListing listingForm = new FormAddedListing(selectedListing);

                listingForm.ShowDialog();

            }

            catch (Exception ex)

            {

                ExceptionManager.HandleException(ex, "Не вдалося відкрити форму. Спробуйте ще раз пізніше", "Помилка відкриття форми");

            }

        }

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

        {

            try

            {

                FormAddListing formAddListing = new FormAddListing();

                formAddListing.ShowDialog();

            }

            catch (Exception ex)

            {

                ExceptionManager.HandleException(ex, "Не вдалося відкрити форму. Спробуйте ще раз пізніше", "Помилка відкриття форми");

            }

        }

        static void RemoveElement(BindingList<Advertisement> list, Advertisement elementToRemove)

        {

            int index = -1;

            for (int i = 0; i < list.Count; i++)

            {

                if (list[i].Name == elementToRemove.Name && list[i].Description == elementToRemove.Description && list[i].Price == elementToRemove.Price && list[i].Address == elementToRemove.Address && list[i].Seller == elementToRemove.Seller && list[i].ImagePath == elementToRemove.ImagePath)

                {

                    index = i;

                    break;

                }

            }

            // If the element is found, remove it

            if (index != -1)

            {

                list.RemoveAt(index);

            }

        }

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

        {

            try

            {

                bool isConfirmed = ExceptionManager.Confirm("Ви впевнені що хочете видалити це оголошення?", "Підтвердження видалення");

                if (dataGridViewAddedListings.SelectedRows.Count > 0 && isConfirmed)

                {

                    var selectedListing = dataGridViewAddedListings.SelectedRows[0];

                    var selectedListingAdvertisment = (Advertisement)selectedListing.DataBoundItem; // Исправлено здесь

                    dataGridViewAddedListings.Rows.Remove(selectedListing);

                    // видалення з глобальної колекції

                    //GlobalData.AvailableListings.Remove(selectedListingAdvertisment);

                    RemoveElement(GlobalData.AvailableListings, selectedListingAdvertisment);

                    //Видалення фото с каталогу

                    File.Delete(selectedListingAdvertisment.ImagePath);

                    // видалення з колекції поточного користувача

                    LoginManager.CurrentUser.AddedListings.Remove(selectedListingAdvertisment);

                }

            }

            catch (Exception ex)

            {

                ExceptionManager.HandleException(ex, "Не вдалося видалити оголошення. Спробуйте пізніше", "Помилка видалення");

            }

        }

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

        {

            bool isConfirmed = ExceptionManager.Confirm("Ви впевнені що хочете вийти?", "Підтвердження виходу");

            try

            {

                if (isConfirmed)

                {

                    LoginManager.IsLoggedIn = false;

                    Application.Restart();

                }

            }

            catch (Exception ex)

            {

                ExceptionManager.HandleException(ex, "Не вдалося вийти.", "Помилка виходу");

            }

        }

    }

}

Б12 – FormRestore

using code.Classes;

using System;

using System.Collections.Generic;

using System.Windows;

using System.Windows.Forms;

namespace code.Forms

{

    public partial class FormRestore : Form

    {

        public FormRestore()

        {

            InitializeComponent();

        }

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

        {

            string name = inputName.Text;

            string email = inputEmail.Text;

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

            bool isFinded = false;

            if (name.Length != 0 && email.Length != 0)

            {

                foreach (var user in GlobalData.Users)

                {

                    if (user.Email == email && user.Name == name)

                    {

                        isFinded = true;

                        try

                        {

                            string verificationCode = LoginManager.generateVerificationCode();

                            Console.WriteLine(verificationCode);

                            \_ = LoginManager.sendVerificationCodeToEmail(email, verificationCode);

                            FormCodeConfirmation InputForm = new FormCodeConfirmation();

                            DialogResult result = InputForm.ShowDialog();

                            if (result == DialogResult.OK)

                            {

                                string userEnteredVerificationCode = InputForm.getVerifyCode();

                                if (userEnteredVerificationCode == verificationCode)

                                {

                                    ExceptionManager.ShowInfo("Тимчасовий пароль було надіслано на вашу електронну пошту!", "Увага");

                                    string newPassword = LoginManager.generateNewPassword();

                                    user.Password = LoginManager.hashPassword(newPassword);

                                    \_ = LoginManager.sendNewPasswordToEmail(email, newPassword);

                                    FormLogin login = new FormLogin();

                                    this.Hide();

                                    if (login.ShowDialog() == DialogResult.OK)

                                    {

                                        this.DialogResult = DialogResult.OK;

                                    }

                                    else

                                    {

                                        this.DialogResult = DialogResult.Cancel;

                                    }

                                }

                                else

                                {

                                    throw new Exception();

                                }

                            }

                            else

                            {

                                throw new Exception();

                            }

                        }

                        catch (Exception)

                        {

                            errors.Add("Щоб змінити пароль, підтвердіть електронну пошту.");

                        }

                        break;

                    }

                }

                if (!isFinded)

                {

                    errors.Add("Користувача с такими даними не знайдено. Спробуйте ще раз.");

                }

            }

            else

            {

                errors.Add("Значення не можуть бути порожніми!");

            }

            if (errors.Count > 0)

            {

                ExceptionManager.HandleException(new Exception(string.Join("\n", errors)), string.Join("\n", errors), "Помилка реєстрації");

                return;

            }

        }

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

        {

            FormLogin login = new FormLogin();

            this.Hide();

            if (login.ShowDialog() == DialogResult.OK)

            {

                this.DialogResult = DialogResult.OK;

                LoginManager.IsLoggedIn = true;

            }

            else

            {

                this.DialogResult = DialogResult.Cancel;

            }

        }

    }

}

Б13 – FormSignUp

using System;

using System.Collections.Generic;

using System.Linq;

using System.Windows.Forms;

using code.Classes;

namespace code.Forms

{

    public partial class FormSignUp : Form

    {

        public string Username

        {

            get; set;

        }

        public string Password

        {

            get; set;

        }

        public string Email

        {

            get; set;

        }

        public FormSignUp()

        {

            InitializeComponent();

            txtPassword.PasswordChar = '\*';

            txtPasswordConfirm.PasswordChar = '\*';

        }

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

        {

            FormLogin login = new FormLogin();

            this.Hide();

            login.ShowDialog();

        }

        public bool DoesUsernameExist(string username)

        {

            return GlobalData.Users.Any(user => user.Name == username);

        }

        public bool DoesEmailExist(string email)

        {

            return GlobalData.Users.Any(user => user.Email == email);

        }

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

        {

            var username = txtUsername.Text;

            var password = txtPassword.Text;

            var passwordConfirm = txtPasswordConfirm.Text;

            var email = txtEmail.Text;

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

            if (string.IsNullOrWhiteSpace(username) || string.IsNullOrWhiteSpace(password) || string.IsNullOrWhiteSpace(email) || string.IsNullOrWhiteSpace(passwordConfirm))

            {

                errors.Add("Ім'я користувача, пароль, підтвердження паролю або електронна пошта не можуть бути порожніми.");

            }

            if (DoesUsernameExist(username))

            {

                errors.Add("Ім'я користувача вже існує.");

            }

            if (DoesEmailExist(email))

            {

                errors.Add("Електронна пошта вже існує.");

            }

            if (username.Length < 3 || username.Length > 16)

            {

                errors.Add("Ім'я користувача має містити від 3 до 16 символів.");

            }

            if (password.Length < 8)

            {

                errors.Add("Пароль повинен мати довжину не менше 8 символів.");

            }

            if (password != passwordConfirm)

            {

                errors.Add("Пароль не співпадає з підтвердженням паролю.");

            }

            try

            {

                if (!string.IsNullOrWhiteSpace(email))

                {

                    var validEmail = new System.Net.Mail.MailAddress(email);

                }

                else

                {

                    throw new FormatException();

                }

            }

            catch (FormatException)

            {

                errors.Add("Електронна пошта не дійсна.");

            }

            if (errors.Count == 0)

            {

                try

                {

                    string verificationCode = LoginManager.generateVerificationCode();

                    Console.WriteLine(verificationCode);

                    \_ = LoginManager.sendVerificationCodeToEmail(email, verificationCode);

                    FormCodeConfirmation InputForm = new FormCodeConfirmation();

                    DialogResult result = InputForm.ShowDialog();

                    if (result == DialogResult.OK)

                    {

                        string userEnteredVerificationCode = InputForm.getVerifyCode();

                        if (userEnteredVerificationCode == verificationCode)

                        {

                        }

                        else

                        {

                            throw new Exception();

                        }

                    }

                    else

                    {

                        throw new Exception();

                    }

                }

                catch (Exception)

                {

                    errors.Add("Електронна пошта не веріфікована.");

                }

            }

            if (errors.Count > 0)

            {

                ExceptionManager.HandleException(new Exception(string.Join("\n", errors)), string.Join("\n", errors), "Помилка реєстрації");

                return;

            }

            this.Username = username;

            this.Password = LoginManager.hashPassword(password);

            this.Email = email;

            this.DialogResult = DialogResult.OK;

        }

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

        {

            try

            {

                if (txtPassword.PasswordChar == '\*')

                {

                    txtPassword.PasswordChar = '\0';

                    btnTogglePasswordVisibility.Text = "0\_0";

                }

                else

                {

                    txtPassword.PasswordChar = '\*';

                    btnTogglePasswordVisibility.Text = ">\_<";

                }

            }

            catch (Exception ex)

            {

                ExceptionManager.HandleException(ex, "Виникла помилка при перемиканні видимості пароля", "Помилка видимості пароля");

            }

        }

        private void txtEmail\_TextChanged(object sender, EventArgs e)

        {

        }

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

        {

            try

            {

                if (txtPasswordConfirm.PasswordChar == '\*')

                {

                    txtPasswordConfirm.PasswordChar = '\0';

                    btnTogglePasswordConfirmVisibility.Text = "0\_0";

                }

                else

                {

                    txtPasswordConfirm.PasswordChar = '\*';

                    btnTogglePasswordConfirmVisibility.Text = ">\_<";

                }

            }

            catch (Exception ex)

            {

                ExceptionManager.HandleException(ex, "Виникла помилка при перемиканні видимості підтвердження пароля", "Помилка видимості підтвердження пароля");

            }

        }

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

        {

            Close();

        }

    }

}

Б14 – FormWelcome

using System;

using System.Collections.Generic;

using System.Windows.Forms;

using code.Classes;

namespace code.Forms

{

    public partial class FormWelcome : Form

    {

        private readonly List<string> tips = new List<string> {

            "У розділі Профіль можна ознайомитись зі всіма своїми публікаціями",

            "У розділі Профіль можна переглянути всі придбані нерухомості",

            "У розділі Профіль є можливість змінювати персональні дані",

            "У розділі Ринок можна додавати свої публікації",

            "У розділі Ринок можливо купувати бажану нерухомість",

            "У розділі Ринок можна сортувати та фільтрувати публікації",

            "У розділі Ринок можна виконувати пошук бажаної нерухомості",

            "У розділі Допомога можна отримати загальну інформацію щодо роботи програми",

            "У розділі Про розробників можна дізнатися більше про розробників проєкту",

        };

        private readonly Random rnd = new Random();

        public FormWelcome()

        {

            InitializeComponent();

            this.Text = string.Empty;

            this.ControlBox = false;

            DisplayRandomTip();

        }

        private void DisplayRandomTip()

        {

            try

            {

                var randomTipIndex = rnd.Next(tips.Count);

                var randomTip = tips[randomTipIndex];

                labelRandomTip.Text = randomTip;

            }

            catch (Exception ex)

            {

                ExceptionManager.HandleException(ex, "Виникла помилка під час відображення випадкової підказки. Спробуйте змінити розділ і повторити спробу", "Не вдалося відобразити випадкову підказку.");

            }

        }

    }

}

Б15 – Advertisement

namespace code.Classes

{

    // клас оголошення

    public class Advertisement

    {

        public string Name

        {

            get; set;

        }

        public string Description

        {

            get; set;

        }

        public int Price

        {

            get; set;

        }

        public string Address

        {

            get; set;

        }

        public string Seller

        {

            get; set;

        }

        public string ImagePath

        {

            get; set;

        }

    }

}

Б16 – AlgorithmManager

using System;

using System.Collections.Generic;

using System.Globalization;

using System.Linq;

using System.Net;

using System.Security.Policy;

using System.Windows.Forms;

using System.Xml;

namespace code.Classes

{

    public static class СurrencyСonverter

    {

        public static double GetDollarRate()

        {

            try

            {

                string link = @"https://bank.gov.ua/NBUStatService/v1/statdirectory/exchange?valcode=USD";

                WebClient wc = new WebClient();

                string xmlString = wc.DownloadString(link);

                XmlDocument xmlDoc = new XmlDocument();

                xmlDoc.LoadXml(xmlString);

                XmlNode rateNode = xmlDoc.SelectSingleNode("/exchange/currency/rate");

                string usdRate = rateNode.InnerText;

                double rate = Convert.ToDouble(usdRate, CultureInfo.InvariantCulture);

                return rate;

            }

            catch (Exception ex)

            {

                ExceptionManager.HandleException(ex, "Не вдалося отримати курс долара. Спробуйте пізніше", "Помилка отримання курсу");

                return 0;

            }

        }

    }

    public static class ShowMap

    {

        public static void GoToMap(string adress)

        {

            try

            {

                string url = $"https://www.google.com/maps/place/{adress}";

                System.Diagnostics.Process.Start(url);

            }

            catch(Exception ex)

            {

                ExceptionManager.HandleException(ex, "Не вдалося перейти на мапу. Спробуйте пізніше", "Помилка переходу на мапу");

            }

        }

    }

    // клас, що відповідає за керування функціями сортування

    public static class AlgorithmManager

    {

        private static void QuickSort(string type, List<Advertisement> list)

        {

            Sort(list, 0, list.Count - 1, type);

        }

        private static void Sort(List<Advertisement> list, int left, int right, string type)

        {

            if (left < right)

            {

                int pivotIndex = Partition(list, left, right, type);

                Sort(list, left, pivotIndex - 1, type);

                Sort(list, pivotIndex + 1, right, type);

            }

        }

        private static int Partition(List<Advertisement> list, int left, int right, string type)

        {

            if (type == "Ціна")

            {

                int pivot = list[right].Price;

                int i = left - 1;

                Advertisement temp;

                for (int j = left; j < right; j++)

                {

                    if (list[j].Price < pivot)

                    {

                        i++;

                        temp = list[i];

                        list[i] = list[j];

                        list[j] = temp;

                    }

                }

                temp = list[i + 1];

                list[i + 1] = list[right];

                list[right] = temp;

                return i + 1;

            }

            if (type == "Назва")

            {

                string pivot = list[right].Name;

                int i = left - 1;

                Advertisement temp;

                for (int j = left; j < right; j++)

                {

                    if ((string.Compare(list[j].Name, pivot) < 0))

                    {

                        i++;

                        temp = list[i];

                        list[i] = list[j];

                        list[j] = temp;

                    }

                }

                temp = list[i + 1];

                list[i + 1] = list[right];

                list[right] = temp;

                return i + 1;

            }

            if (type == "Продавець")

            {

                string pivot = list[right].Seller;

                int i = left - 1;

                Advertisement temp;

                for (int j = left; j < right; j++)

                {

                    if ((string.Compare(list[j].Seller, pivot) < 0))

                    {

                        i++;

                        temp = list[i];

                        list[i] = list[j];

                        list[j] = temp;

                    }

                }

                temp = list[i + 1];

                list[i + 1] = list[right];

                list[right] = temp;

                return i + 1;

            }

            return 0;

        }

        private static void QuickSortDescending(string type, List<Advertisement> list)

        {

            SortDescending(list, 0, list.Count - 1, type);

        }

        private static void SortDescending(List<Advertisement> list, int left, int right, string type)

        {

            if (left < right)

            {

                int pivotIndex = PartitionDescending(list, left, right, type);

                SortDescending(list, left, pivotIndex - 1, type);

                SortDescending(list, pivotIndex + 1, right, type);

            }

        }

        private static int PartitionDescending(List<Advertisement> list, int left, int right, string type)

        {

            if (type == "Ціна")

            {

                int pivot = list[right].Price;

                int i = left - 1;

                Advertisement temp;

                for (int j = left; j < right; j++)

                {

                    if (list[j].Price > pivot)

                    {

                        i++;

                        temp = list[i];

                        list[i] = list[j];

                        list[j] = temp;

                    }

                }

                temp = list[i + 1];

                list[i + 1] = list[right];

                list[right] = temp;

                return i + 1;

            }

            if (type == "Назва")

            {

                string pivot = list[right].Name;

                int i = left - 1;

                Advertisement temp;

                for (int j = left; j < right; j++)

                {

                    if ((string.Compare(list[j].Name, pivot) > 0))

                    {

                        i++;

                        temp = list[i];

                        list[i] = list[j];

                        list[j] = temp;

                    }

                }

                temp = list[i + 1];

                list[i + 1] = list[right];

                list[right] = temp;

                return i + 1;

            }

            if (type == "Продавець")

            {

                string pivot = list[right].Seller;

                int i = left - 1;

                Advertisement temp;

                for (int j = left; j < right; j++)

                {

                    if ((string.Compare(list[j].Seller, pivot) > 0))

                    {

                        i++;

                        temp = list[i];

                        list[i] = list[j];

                        list[j] = temp;

                    }

                }

                temp = list[i + 1];

                list[i + 1] = list[right];

                list[right] = temp;

                return i + 1;

            }

            return 0;

        }

        public static void SortListings(ComboBox dropdown, string sortType, DataGridView dataGridViewListings)

        {

            var nameHeading = "Назва";

            var sellerHeading = "Продавець";

            var priceHeading = "Ціна";

            var listingsCopy = GlobalData.AvailableListings.ToList();

            var selectedItem = dropdown.SelectedIndex;

            if (sortType == "byName")

            {

                if (selectedItem == 0)

                {

                    QuickSort(nameHeading, listingsCopy);

                    nameHeading = "Назва »";

                }

                else

                {

                    QuickSortDescending(nameHeading, listingsCopy);

                    nameHeading = "Назва «";

                }

            }

            else if (sortType == "bySeller")

            {

                if (selectedItem == 0)

                {

                    QuickSort(sellerHeading, listingsCopy);

                    sellerHeading = "Продавець »";

                }

                else

                {

                    QuickSortDescending(sellerHeading, listingsCopy);

                    sellerHeading = "Продавець «";

                }

            }

            else if (sortType == "byPrice")

            {

                if (selectedItem == 0)

                {

                    QuickSort(priceHeading, listingsCopy);

                    priceHeading = "Ціна »";

                }

                else

                {

                    QuickSortDescending(priceHeading, listingsCopy);

                    priceHeading = "Ціна «";

                }

            }

            dataGridViewListings.DataSource = listingsCopy;

            dataGridViewListings.Columns[0].HeaderText = nameHeading;

            dataGridViewListings.Columns[3].HeaderText = sellerHeading;

            dataGridViewListings.Columns[4].HeaderText = priceHeading;

        }

    }

}

Б17 – ExceptionManager

using System;

using System.Windows.Forms;

namespace code.Classes

{

    // клас, що виконує логування, обробку винятків

    public static class ExceptionManager

    {

        public static void HandleException(Exception ex, string body, string heading)

        {

            Console.WriteLine(ex.Message);

            MessageBox.Show($"{body}.\nДодаткова інформація: {ex.Message}", heading, MessageBoxButtons.OK, MessageBoxIcon.Error);

        }

        public static bool Confirm(string body, string heading)

        {

            DialogResult result = MessageBox.Show(body, heading, MessageBoxButtons.YesNo, MessageBoxIcon.Question);

            return result == DialogResult.Yes;

        }

        public static void ShowInfo(string body, string heading)

        {

            MessageBox.Show(body, heading, MessageBoxButtons.OK, MessageBoxIcon.Information);

        }

    }

}

Б18 – FileHandler

using System;

using System.Collections.Generic;

using System.Linq;

using System.IO;

using Newtonsoft.Json;

using System.ComponentModel;

using System.Reflection;

namespace code.Classes

{

    // клас, що відповідає за считування/зберігання даних з файлів

    public static class FileHandler

    {

        // метод для запису даних до CSV файлу з

        // BindingList<Advertisement>

        public static void saveAdToCSV(BindingList<Advertisement> writeFromList, string filepath = null)

        {

            if (filepath == null)

            {

                string currentDirectory = Directory.GetCurrentDirectory();

                filepath = currentDirectory.Substring(0, currentDirectory.Length - 9) + "DataBase\\advertisment.csv";

            }

            try

            {

                using (StreamWriter writer = new StreamWriter(filepath))

                {

                    foreach (var item in writeFromList)

                    {

                        string CSV\_line = $"{item.Name}|{item.Description}|{item.Price}|{item.Address}|{item.Seller}|{item.ImagePath}";

                        writer.WriteLine(CSV\_line);

                    }

                }

            }

            catch (Exception ex)

            {

                ExceptionManager.HandleException(ex, "Не вдалося записати дані в файл.", "Помилка запису даних");

            }

        }

        // метод для зчитування даних до BindingList<Advertisement>

        // з CSV файлу

        public static BindingList<Advertisement> readAdFromCSV(string filepath = null)

        {

            if (filepath == null)

            {

                string currentDirectory = Directory.GetCurrentDirectory();

                filepath = currentDirectory.Substring(0, currentDirectory.Length - 9) + "DataBase\\advertisment.csv";

            }

            BindingList<Advertisement> readToList = new BindingList<Advertisement>();

            try

            {

                string[] CSV\_lines = File.ReadAllLines(filepath);

                for (int i = 0; i < CSV\_lines.Length; i++)

                {

                    string[] CSV\_data = CSV\_lines[i].Split('|');

                    Advertisement advertisement = new Advertisement();

                    advertisement.Name = CSV\_data[0];

                    advertisement.Description = CSV\_data[1];

                    advertisement.Price = int.Parse(CSV\_data[2]);

                    advertisement.Address = CSV\_data[3];

                    advertisement.Seller = CSV\_data[4];

                    advertisement.ImagePath = CSV\_data[5];

                    readToList.Add(advertisement);

                }

                return readToList;

            }

            catch (Exception ex)

            {

                ExceptionManager.HandleException(ex, "Не вдалося зчитати дані з файлу.", "Помилка зчитування даних");

                return null;

            }

        }

        // клас для роботи з JSON

        public class UserObj

        {

            public string UserId { get; set; }

            public string Name { get; set; }

            public string Email { get; set; }

            public string Password { get; set; }

            public string BoughtListingsFilePath { get; set; }

            public string AddedListingsFilePath { get; set; }

        }

        // метод для запису даних User

        public static void UserFileWriter(List<User> writeFromList)

        {

            string currentDirectory = Directory.GetCurrentDirectory();

            string databaseDirectory = currentDirectory.Substring(0, currentDirectory.Length - 9) + "DataBase";

            string filePath = currentDirectory.Substring(0, currentDirectory.Length - 9) + "DataBase\\users.json";

            string fileToSave = currentDirectory.Substring(0, currentDirectory.Length - 9) + "DataBase\\advertisment.csv";

            DeleteFilesExcept(databaseDirectory, fileToSave);

            try

            {

                List<UserObj> writeList = new List<UserObj>();

                foreach (var item in writeFromList)

                {

                    UserObj user = new UserObj();

                    user.UserId = item.UserId;

                    user.Name = item.Name;

                    user.Email = item.Email;

                    user.Password = item.Password;

                    user.BoughtListingsFilePath = Path.Combine(databaseDirectory, $"user\_{user.UserId}\_BoughtListings.csv");

                    saveAdToCSV(item.BoughtListings, user.BoughtListingsFilePath);

                    user.AddedListingsFilePath = Path.Combine(databaseDirectory, $"user\_{user.UserId}\_AddedListings.csv");

                    saveAdToCSV(item.AddedListings, user.AddedListingsFilePath);

                    writeList.Add(user);

                }

                string json = JsonConvert.SerializeObject(writeList, Formatting.Indented);

                File.WriteAllText(filePath, json);

            }

            catch (Exception ex)

            {

                ExceptionManager.HandleException(ex, "Не вдалося записати дані в файл.", "Помилка запису даних");

            }

        }

        // метод для зчитування даних User

        public static List<User> UserFileReader()

        {

            string currentDirectory = Directory.GetCurrentDirectory();

            string databaseDirectory = currentDirectory.Substring(0, currentDirectory.Length - 9) + "DataBase";

            string filePath = currentDirectory.Substring(0, currentDirectory.Length - 9) + "DataBase\\users.json";

            try

            {

                List<UserObj> readList = new List<UserObj>();

                string json = File.ReadAllText(filePath);

                readList = JsonConvert.DeserializeObject<List<UserObj>>(json);

                List<User> users = new List<User>();

                foreach (var item in readList)

                {

                    User user = new User();

                    user.UserId = item.UserId;

                    user.Name = item.Name;

                    user.Email = item.Email;

                    user.Password = item.Password;

                    user.BoughtListings = readAdFromCSV(Path.Combine(databaseDirectory, $"user\_{user.UserId}\_BoughtListings.csv"));

                    user.AddedListings = readAdFromCSV(Path.Combine(databaseDirectory, $"user\_{user.UserId}\_AddedListings.csv"));

                    users.Add(user);

                }

                return users;

            }

            catch (Exception ex)

            {

                ExceptionManager.HandleException(ex, "Не вдалося зчитати дані з файлу.", "Помилка зчитування даних");

                return null;

            }

        }

        // метод для видалення зайвих даних перед записом

        public static void DeleteFilesExcept(string directoryPath, string fileToKeep)

        {

            try

            {

                var files = Directory.GetFiles(directoryPath);

                var filesToDelete = files.Where(file => file != fileToKeep);

                foreach (var fileToDelete in filesToDelete)

                {

                    File.Delete(fileToDelete);

                }

            }

            catch (Exception ex)

            {

                ExceptionManager.HandleException(ex, "Не вдалося видалити дані.", "Помилка видалення даних");

            }

        }

    }

}

Б19 – GlobalData

using System.Collections.Generic;

using System.ComponentModel;

namespace code.Classes

{

    // клас, що відповідає за збереження активних даних в момент виконання програми

    public static class GlobalData

    {

        public static BindingList<Advertisement> AvailableListings { get; set; }

        public static List<User> Users { get; set; }

    }

}

Б20 – LoginManager

using System;

using System.Security.Cryptography;

using System.Text;

using System.Threading.Tasks;

using MailKit.Net.Smtp;

using MimeKit;

namespace code.Classes

{

    // клас, що відповідає за перевірку даних, входу до системи та

    // кодування й декодування паролю

    public static class LoginManager

    {

        public static bool IsLoggedIn

        {

            get; set;

        }

        public static User CurrentUser

        {

            get; set;

        }

        // метод для гешування паролю

        public static string hashPassword(string password)

        {

            using (SHA256 sha256 = SHA256.Create())

            {

                byte[] hashedBytes = sha256.ComputeHash(Encoding.UTF8.GetBytes(password));

                StringBuilder hashedPassword = new StringBuilder();

                for (int i = 0; i < hashedBytes.Length; i++)

                {

                    hashedPassword.Append(hashedBytes[i].ToString("x2"));

                }

                return hashedPassword.ToString();

            }

        }

        // метод для перевірки совпадіння гешу з введенним паролем

        public static bool verifyPassword(string password, string hashedPassword)

        {

            string hashedInput = hashPassword(password);

            return string.Equals(hashedInput, hashedPassword, StringComparison.OrdinalIgnoreCase);

        }

        // метод для генерації коду веріфікаціїї з 6 букв та цифр

        public static string generateVerificationCode()

        {

            const string characters = "ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789";

            Random random = new Random();

            char[] code = new char[6];

            for (int i = 0; i < code.Length; i++)

            {

                code[i] = characters[random.Next(characters.Length)];

            }

            return new string(code);

        }

        // метод для відправлення сгенерованого коду верифікації на пошту

        public static async Task sendVerificationCodeToEmail(string toEmail, string verificationCode)

        {

            string mailFrom = "realestateapp@ukr.net";

            string password = "XI56uR64bNjtfvsg";

            var message = new MimeMessage();

            message.From.Add(new MailboxAddress("Real Estate App Bot", mailFrom));

            message.To.Add(new MailboxAddress("Recipient", toEmail));

            message.Subject = "Верифікація вашої поштової скриньки";

            var bodyBuilder = new BodyBuilder();

            bodyBuilder.TextBody = $"Ваш код верифікації: {verificationCode}";

            message.Body = bodyBuilder.ToMessageBody();

            using (var client = new SmtpClient())

            {

                await client.ConnectAsync("smtp.ukr.net", 465, true);

                await client.AuthenticateAsync(mailFrom, password);

                await client.SendAsync(message);

                await client.DisconnectAsync(true);

            }

        }

        // функція, що генерує випадковий пароль

        public static string generateNewPassword()

        {

            int length = 12;

            const string chars = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789!@#$%^&\*()-\_+=<>?";

            StringBuilder newPassword = new StringBuilder();

            Random random = new Random();

            for (int i = 0; i < length; i++)

            {

                int index = random.Next(chars.Length);

                newPassword.Append(chars[index]);

            }

            return newPassword.ToString();

        }

        public static async Task sendNewPasswordToEmail(string toEmail, string newPassword)

        {

            string mailFrom = "realestateapp@ukr.net";

            string password = "XI56uR64bNjtfvsg";

            var message = new MimeMessage();

            message.From.Add(new MailboxAddress("Real Estate App Bot", mailFrom));

            message.To.Add(new MailboxAddress("Recipient", toEmail));

            message.Subject = "Новий пароль від вашого аккаунту";

            var bodyBuilder = new BodyBuilder();

            bodyBuilder.TextBody = $"Ваш новий пароль: {newPassword}\nЗа потреби можете його змінити у особистому кабінеті.";

            message.Body = bodyBuilder.ToMessageBody();

            using (var client = new SmtpClient())

            {

                await client.ConnectAsync("smtp.ukr.net", 465, true);

                await client.AuthenticateAsync(mailFrom, password);

                await client.SendAsync(message);

                await client.DisconnectAsync(true);

            }

        }

    }

}

Б21 – User

using System;

using System.ComponentModel;

namespace code.Classes

{

    // клас користувача

    public class User

    {

        public string UserId

        {

            get; set;

        }

        public string Name

        {

            get; set;

        }

        public string Email

        {

            get; set;

        }

        public string Password

        {

            get; set;

        }

        public BindingList<Advertisement> BoughtListings

        {

            get; set;

        }

        public BindingList<Advertisement> AddedListings

        {

            get; set;

        }

        public User()

        {

            UserId = Guid.NewGuid().ToString();

            BoughtListings = new BindingList<Advertisement>();

            AddedListings = new BindingList<Advertisement>();

        }

        public User(string name, string email, string password)

        {

            UserId = Guid.NewGuid().ToString();

            Name = name;

            Email = email;

            Password = password;

            BoughtListings = new BindingList<Advertisement>();

            AddedListings = new BindingList<Advertisement>();

        }

    }

}

Додаток В Слайди презентації

Рисунок В.1 – Слайд 1

Рисунок В.2 – Слайд 2

….

Увага! В презентації обов`язково нумеруєте слайди!