Міністерство освіти і науки україни
Криворізький національний університет
Кафедра моделювання і програмного забезпечення

# КВАЛІФІКАЦІЙНА РОБОТА

**на здобуття ступення вищої освіти бакалавра**

зі спеціальності 121 Інженерія програмного забезпечення

На тему: Прогнозування цін на нерухомість за допомогою машинного навчання Python

Засвідчую, що в цій кваліфікаційній роботі немає запозичень із праць інших авторів без відповідних посилань

Студент гр. ІПЗ-21-2

_____ /А. Г. Губарєв /

Керівник
кваліфікаційної роботи _____/ І. О. Доценко /

Завідувач кафедри _____ / А. М. Стрюк /

Кривий Ріг
2025

Криворізький національний університет

Факультет: Інформаційних технологій

Кафедра: Моделювання та програмного забезпечення

Ступінь вищої освіти: бакалавр

Спеціальність: 121 – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:
Завідувач кафедри
_____ А. М. Стрюк
«___» _____ 2025 р.

# ЗАВДАННЯ

**на кваліфікаційну роботу**

студенту групи ІПЗ-21-2 Губарєву Артему Григоровичу

1. Тема: Прогнозування цін на нерухомість за допомогою машинного навчання Python затверджено наказом КНУ № 200с від «10»квітня 2025р.
2. Термін подання студентом закінченої роботи: «15» червня 2025р.
3. Вихідні дані по роботі: система прогнозування цін на нерухомість працюватиме з нестандартизованими даними українського ринку, включаючи фрагментовані параметри об'єктів та динамічні макроекономічні фактори. Мінімальний поріг точності прогнозів: коефіцієнт детермінації $R^2 \geq 0.8$.
4. Зміст пояснювальної записки (перелік питань, що їх треба розробити): аналіз існуючих рішень для оцінки нерухомості та обґрунтування вибору методів, проектування архітектури системи, реалізація програмного комплексу із ML-ядром, тестування точності моделі та юзабіліті інтерфейсу.
5. Перелік ілюстративного матеріалу: візуалізація структури ML-моделі, логічні схеми роботи системи, архітектурна карта взаємодії компонентів, інтерфейсні рішення.

Календарний план:

| | Найменування етапів кваліфікаційної роботи | Термін виконання етапів роботи |
|---|---|---|
| 1 | Аналіз предметної області | 17.03.2025-06.04.2025 |
| 2 | Проектування архітектури | 07.04.2025-18.04.2025 |
| 3 | Підготовка даних | 18.04.2025-01.05.2025 |
| 4 | Розробка ML-моделі | 01.05.2025-08.05.2025 |
| 5 | Створення бекенд-системи | 08.05.2025-16.05.2025 |
| 6 | Розробка інтерфейсу | 17.05.2025-24.05.2025 |
| 7 | Тестування та оптимізація | 24.05.2025-01.06.2025 |
| 8 | Написання пояснювальної записки | 02.06.2025-10.06.2025 |
| | | |

Дата видачі завдання: «17» березня 2025 р.

Студент: _____ / А. Г. Губарєв /

Керівник роботи: _____ / І. О. Доценко /

# РЕФЕРАТ

ПРОГНОЗУВАННЯ ЦІН, РИНОК НЕРУХОМОСТІ УКРАЇНИ, МАШИННЕ НАВЧАННЯ PYTHON, ГРАДІЄНТНИЙ БУСТИНГ, ЖИЛОВА НЕРУХОМІСТЬ, ВЕБ-СИСТЕМА ОЦІНКИ, РЕГРЕСІЙНА МОДЕЛЬ, ЦІНОУТВОРЮВАЛЬНІ ФАКТОРИ, ВІЗУАЛІЗАЦІЯ РЕЗУЛЬТАТІВ

Пояснювальна записка: 127 с., 21 рис., 1 дод., 18 джерел.

Мета роботи: розробка алгоритмічної системи для передбачення ринкової вартості житлових об'єктів шляхом застосування методів машинного навчання на платформі Python.

Об’єкт дослідження: сукупність техніко-економічних чинників та ринкових механізмів, що визначають ціноутворення на житловий фонд України в умовах сучасних соціальних та економічних реалій.

Предмет розробки: програмно-алгоритмічний комплекс, що реалізує прогностичну модель на основі машинного навчання (Python) для автоматизації оцінки нерухомості.

До таких показників належать географічне розташування об’єкта щодо транспортної інфраструктури та соціальних об’єктів, загальна та житлова площа, планування та якість будівельних матеріалів, а також рік спорудження й ступінь зносу будинку.

Водночас у дослідженні враховуються макроекономічні умови ринку, серед яких динаміка попиту та пропозиції, рівень процентних ставок за іпотечними кредитами та загальний стан економіки регіону. Такий підхід дозволяє побачити ширшу картину, в якій окремі характеристики нерухомості взаємодіють із зовнішніми економічними чинниками.

У фокусі роботи знаходиться виявлення прихованих залежностей між переліченими характеристиками та ціною об’єкта шляхом застосування методів машинного навчання, що дає змогу створити інструмент для точного прогнозування вартості нерухомості та прийняття обґрунтованих рішень інвесторами, банківськими установами та ріелторськими агентствами.

# ABSTRACT

PRICE PREDICTION, UKRAINIAN REAL ESTATE MARKET, MACHINE LEARNING, PYTHON, GRADIENT BOOSTING, RESIDENTIAL REAL ESTATE, WEB-BASED ASSESSMENT SYSTEM, REGRESSION MODEL, PRICING FACTORS, RESULT VISUALIZATION

Explanatory note: p. 127, fig. 21, app. 1, references 18.

Objective of the work: Development of an algorithmic system for predicting the market value of residential properties through the application of machine learning methods on the Python platform.

Object of research: The set of technical, economic, and market factors that determine the pricing of the residential housing stock in Ukraine under current social and economic conditions.

Subject of development**:** A software-algorithmic system that implements a predictive model based on machine learning (Python) to automate real estate valuation.

Relevant indicators include the geographical location of the property relative to transport infrastructure and social facilities, total and living area, layout and quality of building materials, as well as the year of construction and the building's wear and tear level.

At the same time, the study considers macroeconomic market conditions, such as the dynamics of supply and demand, mortgage interest rates, and the general state of the regional economy. This approach allows for a broader perspective, where specific property characteristics interact with external economic factors.

The focus of the work is the identification of hidden dependencies between the listed features and the property price through the application of machine learning techniques. This enables the creation of a tool for accurate real estate value forecasting and informed decision-making for investors, banking institutions, and real estate agencies.

# ЗМІСТ

# ВСТУП

Сучасний ринок нерухомості України, сформований під впливом геополітичних трансформацій та економічної нестабільності, потребує інноваційних інструментів об’єктивної оцінки вартості. Традиційні методи ціноутворення, засновані на експертних судженнях або спрощеній статистиці, часто виявляються недостатніми у контексті швидкозмінного середовища з фрагментованими даними та непередбачуваними факторами впливу. Саме ця проблема зумовила актуальність дослідження, спрямованого на розробку системи прогнозування цін із застосуванням машинного навчання.

Метою роботи стало створення інтелектуального інструменту для передбачення вартості житлової нерухомості на основі комплексного аналізу технічних, просторових та економічних параметрів. До ключових завдань увійшли:

- аналіз специфіки українського ринку та виявлення ключових ціноутворювальних факторів;
- побудова оптимізованої ML-моделі з використанням Python-інструментів (Scikit-learn, Pandas);
- розробка веб-платформи для інтерактивного прогнозування з урахуванням потреб кінцевих користувачів (інвестори, ріелтори, банки).

Об’єктом дослідження виступив процес формування цін на житлову нерухомість в умовах динамічної української економіки, тоді як предметом— алгоритми машинного навчання для прогнозування вартості на базі просторово-технічних характеристик об’єктів. Наукова новизна полягає у поєднанні сучасних ML-методів (градієнтний бустинг, нейронні мережі) з адаптацією до специфіки локальних даних, зокрема через інтеграцію геопросторових показників та макроекономічних індикаторів.

# 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА АКТУАЛЬНІСТЬ КВАЛІФІКАЦІЙНОЇ РОБОТИ

## 1.1 Аналіз професійної галузі проблеми

Сфера нерухомості, що є одним з ключових драйверів економічного розвитку будь-якої держави, відзначається високою динамічністю та волатильністю. Прогнозування цін на об'єкти нерухомості завжди було складним завданням, що вимагає глибокого аналізу множини взаємопов'язаних факторів. Традиційні методи, що базуються на експертних оцінках та спрощених статистичних моделях, часто виявляються недостатньо точними у швидкозмінних ринкових умовах. Саме тут машинне навчання, зокрема за допомогою мови програмування Python, відкриває нові, значно ширші можливості для більш об'єктивного та точного передбачення майбутніх тенденцій.

Актуальність використання машинного навчання для аналізу ринку нерухомості обумовлена кількома чинниками. По-перше, величезні обсяги доступних даних, що включають характеристики об'єктів (площа, кількість кімнат, рік побудови, стан), їхнє розташування (мікрорайон, інфраструктура, транспортна доступність), а також макроекономічні показники (інфляція, процентні ставки, валовий внутрішній продукт, рівень доходів населення). Всі ці дані є ідеальним матеріалом для обробки та аналізу за допомогою складних алгоритмів. По-друге, потреба у підвищенні ефективності та прозорості угод. Точніші прогнози цін дозволяють як продавцям, так і покупцям приймати більш обґрунтовані рішення, мінімізуючи ризики та оптимізуючи інвестиції. Для девелоперів та інвестиційних фондів це означає краще планування проєктів та розподілу ресурсів.

Однак, попри значний потенціал, впровадження машинного навчання у практику прогнозування цін на українському ринку нерухомості стикається з низкою суттєвих проблем та викликів. Першочерговою проблемою є якість та доступність даних. На відміну від розвинених ринків, де існують

централізовані та стандартизовані бази даних про нерухомість, в Україні ситуація є більш фрагментованою. Дані часто розрізнені, неповні, містять помилки або знаходяться у неструктурованому форматі. Це вимагає значних зусиль для збору, очищення та попередньої обробки інформації, що є критично важливим етапом для побудови будь-якої ефективної моделі машинного навчання. Відсутність єдиної стандартизованої методології збору даних та їхньої публікації у відкритих джерелах ускладнює автоматизацію цього процесу.

Додатковим ускладненням є динамічність та непередбачуваність українського економічного середовища, особливо в умовах воєнного стану. Політичні, соціальні та економічні потрясіння, міграційні процеси, інфляційні очікування та зміни в законодавстві мають безпосередній та часто нелінійний вплив на ціноутворення. Моделі машинного навчання, хоча і здатні виявляти складні залежності, можуть потребувати постійного доопрацювання та перенавчання для адаптації до таких швидких змін. Особливо відчутним є вплив інфляції, зростання цін на будівельні матеріали та енергоресурси, а також дефіцит кваліфікованої робочої сили, що безпосередньо відбивається на собівартості будівництва та, відповідно, на ринковій вартості житла. В західних регіонах України, попри загальну нестабільність, спостерігається зростання цін, тоді як у східних та південних областях, що постраждали від бойових дій, ринок часто знаходиться у стані стагнації або падіння. Це створює потребу у розробці регіонально адаптованих моделей прогнозування.

Юридичний та нормативно-правовий аспекти також мають значний вплив. Хоча Україна має законодавчі основи регулювання ринку нерухомості, включаючи Цивільний кодекс та Закон України «Про державну реєстрацію речових прав на нерухоме майно та їх обтяжень», існують нюанси, що можуть впливати на достовірність даних. Наприклад, прозорість угод та повна інформація про об'єкт нерухомості є правом покупця, проте на практиці доступ до певних даних може бути обмеженим. Діяльність

ріелторів, попри її важливість, досі не має чіткого законодавчого регулювання, що може впливати на якість та повноту інформації, яку вони надають.

Ще одним викликом є вибір та застосування відповідних алгоритмів. Ринок нерухомості характеризується нелінійними залежностями між факторами та ціною. Лінійна регресія, хоч і є базовим інструментом, часто виявляється недостатньою. Більш складні моделі, такі як дерева рішень, випадковий ліс, градієнтний бустинг, або нейронні мережі, мають потенціал для кращого виявлення прихованих закономірностей. Проте, їхнє успішне застосування вимагає глибокого розуміння принципів їхньої роботи, а також значного досвіду у налаштуванні гіперпараметрів та оцінці ефективності моделей. Вибір метрик оцінки, таких як коефіцієнт детермінації (R-квадрат) або середня абсолютна похибка, також є важливим для коректного порівняння та вибору найкращої моделі.

Попри ці перешкоди, перспективи машинного навчання у сфері прогнозування цін на нерухомість в Україні є обнадійливими. Розвиток технологій збору та обробки великих даних, зростання кількості фахівців у галузі Data Science та Machine Learning, а також поступове підвищення прозорості ринку створюють сприятливе підґрунтя для подальшого впровадження цих інноваційних підходів. Інтеграція геопросторових даних, аналіз супутникових знімків, використання технологій обробки природної мови для аналізу описів об'єктів нерухомості, а також застосування передових архітектур нейронних мереж, можуть значно підвищити точність прогнозів. Розробка інтерактивних платформ на базі Python, що дозволяють користувачам вводити параметри нерухомості та миттєво отримувати обґрунтовані прогнози, може стати революційним кроком для всіх учасників ринку.

Таким чином, детальний аналіз професійної галузі проблеми «Прогнозування цін на нерухомість за допомогою машинного навчання Python» виявляє її складність та багатогранність. Вона вимагає не лише

технічних навичок у сфері машинного навчання та програмування на Python, а й глибокого розуміння економічних, соціальних та юридичних аспектів функціонування ринку нерухомості. Успішне подолання існуючих викликів, таких як якість даних та динамічність ринку, відкриє шлях до створення високоточних та ефективних прогностичних моделей, що сприятимуть стабілізації та розвитку українського ринку нерухомості.

### 1.2 Обґрунтування призначення, цілі та завдання розробки

Метою кваліфікаційної роботи є розробка алгоритмічної системи для передбачення ринкової вартості житлових об'єктів шляхом застосування методів машинного навчання на платформі Python.

Основне призначення даної розробки полягає у створенні надійної, масштабованої та високоточної системи для прогнозування цін на об'єкти нерухомості в Україні, використовуючи передові методи машинного навчання та програмні засоби Python. Даний проєкт покликаний подолати існуючі обмеження традиційних методів оцінки, що часто базуються на суб'єктивних експертних оцінках або спрощених статистичних моделях, які не завжди здатні адекватно відображати складну динаміку ринку. Кінцевою метою є надання об'єктивного та обґрунтованого інструменту для широкого кола користувачів, які приймають рішення, пов'язані з нерухомістю.

Одним з ключових аспектів призначення даної розробки є підвищення прозорості та ефективності угод на ринку нерухомості. В умовах нестабільної економіки та воєнного стану, що істотно впливає на український ринок, питання достовірної оцінки вартості стає особливо гострим. Покупці та продавці потребують інструментів, що дозволяють їм впевнено орієнтуватися у цінових тенденціях, мінімізуючи ризики завищення або заниження вартості об'єктів. Створення такої системи надасть їм можливість отримувати актуальну інформацію, що базується на об'єктивних даних та складних алгоритмах, а не лише на інтуїції чи обмежених порівняльних

даних. Це сприятиме більш справедливим та вигідним операціям для всіх учасників ринку.

Для інвестиційних фондів та девелоперських компаній розробка такої прогностичної системи матиме стратегічне значення. Можливість точно передбачати майбутню вартість нерухомості дозволить їм ефективніше планувати свої інвестиції, оптимізувати портфелі активів та мінімізувати фінансові ризики. Наприклад, девелопери зможуть краще оцінювати доцільність будівництва нових об'єктів у певних локаціях, виходячи з прогнозованого попиту та цін, а також коригувати свої стратегії продажів. Інвестори, зі свого боку, отримають потужний інструмент для ідентифікації привабливих об'єктів для придбання та оцінки потенційної прибутковості своїх вкладень у середньостроковій та довгостроковій перспективі. Це сприятиме більш раціональному розподілу капіталу в галузі нерухомості.

Муніципальні органи влади та державні установи також зможуть знайти цінне застосування цій системі. Точні прогнози цін на нерухомість можуть бути використані для формування бюджетів, планування міського розвитку, оцінки ефективності програм соціального житла, а також для оподаткування нерухомого майна. Об'єктивна інформація про ринкову вартість дозволить приймати більш обґрунтовані рішення щодо містобудування, розвитку інфраструктури та регулювання земельних відносин, що, у свою чергу, матиме позитивний вплив на економічний та соціальний розвиток громад.

Наукові кола та дослідники також отримають значну користь від цієї розробки. Створення комплексного набору інструментів та методологій для прогнозування цін на нерухомість за допомогою машинного навчання на Python може слугувати основою для подальших досліджень у галузі економетрики, геоінформаційних систем та штучного інтелекту. Це дозволить вивчати складні взаємозв'язки між різними факторами, що впливають на ціноутворення, а також розробляти нові, більш досконалі моделі та алгоритми. Відкритий доступ до певних компонентів або

результатів дослідження може сприяти обміну знаннями та стимулювати інновації у цій сфері.

Ключова особливість цієї розробки полягає у використанні машинного навчання. Це дозволяє моделі автоматично виявляти складні, нелінійні залежності та приховані закономірності у великих обсягах даних, що неможливо для традиційних статистичних методів. Застосування Python як основного інструменту розробки обґрунтовано його універсальністю, широким спектром бібліотек для обробки даних (Pandas, NumPy), візуалізації (Matplotlib, Seaborn) та, що найважливіше, для машинного навчання (Scikit-learn, TensorFlow, Keras). Це забезпечує високу гнучкість, масштабованість та можливість інтеграції з іншими системами.

Проєкт передбачає системний підхід, що включає кілька етапів: збір та попередню обробку різнорідних даних про нерухомість (структурованих та неструктурованих), вибір та оптимізацію відповідних алгоритмів машинного навчання (від регресійних моделей до складних нейронних мереж), розробку архітектури прогностичної системи, її тестування, валідацію та вдосконалення. Особлива увага буде приділена питанням інтерпретованості моделей, щоб користувачі могли не лише отримати прогноз, а й зрозуміти, які фактори найбільше вплинули на отриманий результат. Це дозволить підвищити довіру до системи та забезпечити її ефективне застосування у реальних умовах.

Отже, призначення даної розробки виходить за рамки створення простого інструменту прогнозування. Вона спрямована на фундаментальне вдосконалення процесу оцінки нерухомості в Україні, підвищення його об'єктивності, прозорості та ефективності. Це забезпечить значну користь для всіх учасників ринку – від індивідуальних покупців та продавців до великих інвестиційних компаній та державних установ, сприяючи стабільності та динамічному розвитку галузі нерухомості в умовах сучасних викликів.

## 1.3 Огляд ринку та ключові платформи для прогнозування цін на нерухомість в Україні та світі

Ринок нерухомості є одним із найскладніших для прогнозування, адже на його динаміку впливає безліч взаємопов'язаних факторів: економічні показники, соціальні тенденції, інфраструктурний розвиток, а в українських реаліях – ще й безпекова ситуація. Традиційні методи оцінки часто виявляються недостатньо гнучкими, щоб оперативно реагувати на швидкі зміни. Саме тому в останні роки машинне навчання стало потужним інструментом для більш точного та об'єктивного передбачення цінових тенденцій. Ця технологія дозволяє аналізувати величезні масиви даних, виявляючи приховані залежності, які неможливо помітити людським оком.

Завдяки розвитку технологій та збільшенню обсягів доступних даних, з'явилося багато проєктів та платформ, що використовують алгоритми штучного інтелекту для прогнозування вартості нерухомості. Вони пропонують інструменти як для професіоналів ринку, так і для пересічних громадян, допомагаючи приймати більш обґрунтовані рішення щодо купівлі, продажу чи інвестування.

Серед найкращих варіантів таких платформ, що надають цінні дані або безпосередньо займаються прогнозуванням, можна виділити:

а) OLX.ua – є найбільшою платформою оголошень в Україні, що охоплює величезну кількість пропозицій з продажу та оренди нерухомості по всій країні. Це не спеціалізований сервіс для прогнозування, але його роль як джерела даних є надзвичайно важливою.

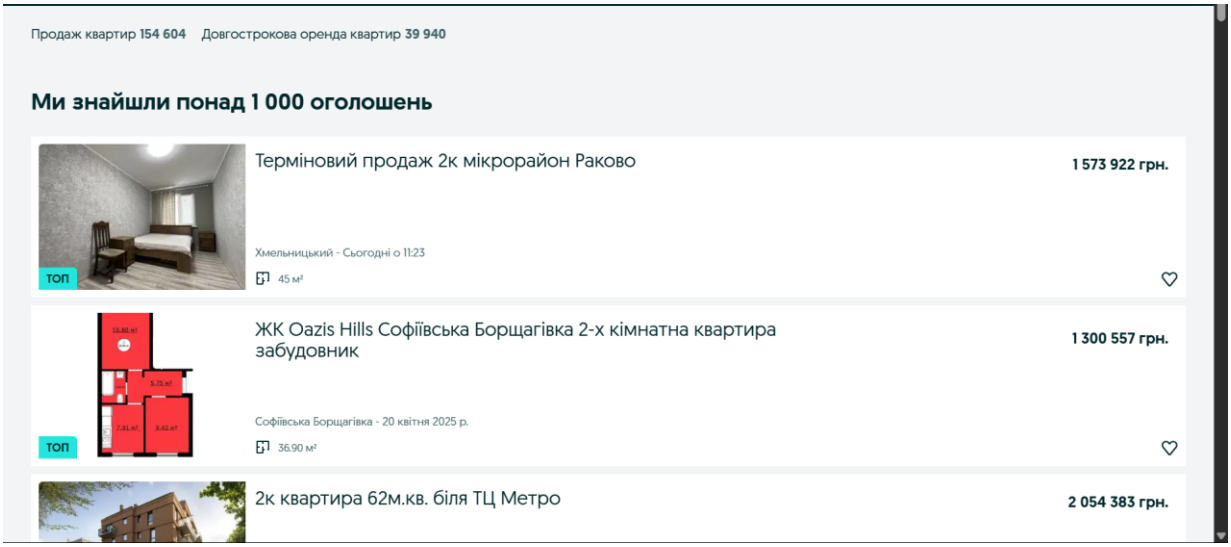


Рисунок 1.1 – Вигляд інтерфейсу «OLX» із доступними оголошеннями

Переваги:

- Широке охоплення. Містить найбільший обсяг оголошень про нерухомість в Україні, що забезпечує значну вибірку для аналізу.
- Актуальність. Оголошення постійно оновлюються, відображаючи поточний стан ринку та цінові пропозиції.
- Різноманітність даних. Дозволяє збирати інформацію про тип нерухомості, площу, кількість кімнат, розташування, стан ремонту, наявність меблів тощо.

Недоліки:

- Відсутність вбудованої функції прогнозування. OLX.ua не надає власних інструментів машинного навчання для автоматичного прогнозування цін. Дані потрібно збирати та аналізувати окремо.
- Нестандартизовані дані. Описи оголошень можуть бути неповними або містити неструктуровану інформацію, що ускладнює автоматизований парсинг та подальшу обробку.
- Шум у даних. Можлива наявність фейкових оголошень, занижених/завищених цін з метою приваблення уваги, що може впливати на якість прогнозів.

б) DIM.RIA – є одним з провідних українських порталів нерухомості, що активно розвиває свої аналітичні можливості та стандартизує дані.

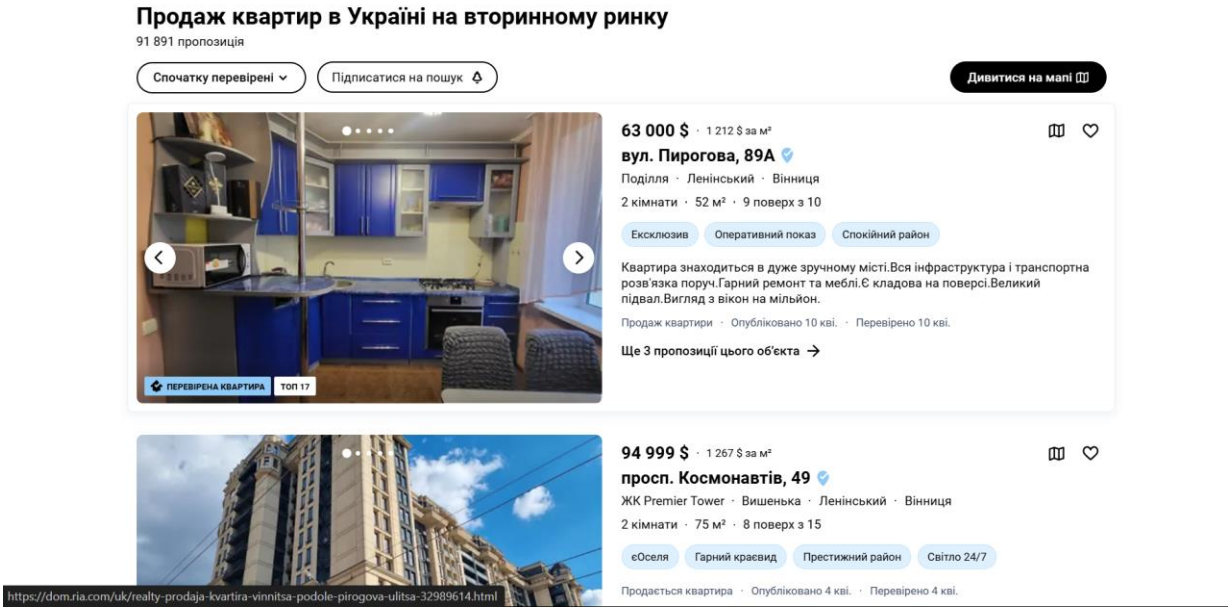


Рисунок 1.2 – Вигляд інтерфейсу «DIM.RIA» із доступними оголошеннями

Переваги:

- Верифіковані оголошення. DOM.RIA докладає зусиль для верифікації оголошень (наприклад, перевірка фотографій ріелторами), що підвищує довіру до даних.
- Структуровані дані. Інформація про об'єкти представлена у більш структурованому форматі порівняно з загальними дошками оголошень, що спрощує збір та обробку даних для моделей машинного навчання.
- Аналітичні інструменти. Надає деякі базові аналітичні дані та статистику щодо цін у різних регіонах.

Недоліки:

- Менше охоплення, ніж OLX.ua. Хоча база велика, кількість оголошень може бути меншою, ніж на OLX.ua, особливо для деяких сегментів ринку.

- Обмеженість безпосереднього прогнозування. Як і OLX.ua, DOM.RIA не пропонує інтегрованого інструменту прогнозування цін на основі машинного навчання для кінцевого користувача.
- Фокус на ріелторах. Платформа активно працює з ріелторами, що може впливати на структуру та повноту доступних для аналізу даних.

в) Zillow (США) – є одним з найбільш відомих та передових сервісів з оцінки нерухомості у США, що активно використовує машинне навчання. Його інструмент Zestimate став еталоном для автоматизованої оцінки вартості.

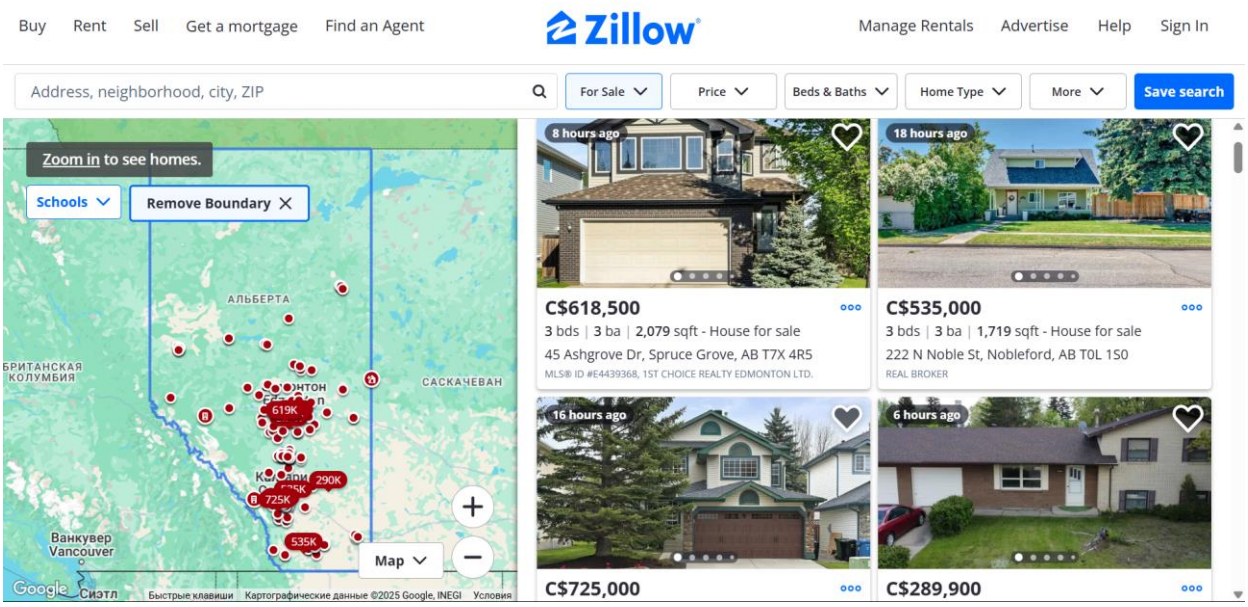


Рисунок 1. 3 – Інтерфейс сайту «Zillow»

Переваги:

- Висока точність прогнозування. Zillow використовує складні ансамблеві моделі та нейронні мережі, інтегруючи величезну кількість даних з різних джерел (публічні реєстри, характеристики нерухомості, інфраструктура, демографія, супутникові знімки тощо).

- Комплексний аналіз. Враховує широкий спектр факторів, включаючи макроекономічні показники та тенденції ринку.
- Зручний інтерфейс та візуалізація. Надає користувачам інтуїтивно зрозумілі інтерактивні карти, графіки динаміки цін, а також можливість персоналізувати пошук.
- Безперервне оновлення. Моделі постійно перенавчаються на нових даних, що забезпечує актуальність прогнозів.

Недоліки:

- Географічна прив'язка. Основна діяльність Zillow зосереджена на ринку нерухомості США, і його моделі не застосовні безпосередньо до українського ринку через суттєві відмінності в даних та факторах впливу.
- Закритість алгоритмів. Деталі їхніх пропрієтарних алгоритмів є комерційною таємницею, що обмежує можливість їхнього прямого відтворення.
- Залежність від наявності даних. Навіть Zillow іноді має труднощі з оцінкою об'єктів у регіонах з низькою активністю ринку або обмеженою доступністю даних.

Ці платформи, кожна по-своєму, відіграють ключову роль у формуванні розуміння ринку нерухомості та надають цінний матеріал для розробки власних прогностичних моделей на основі машинного навчання. Вони підкреслюють, що для успішного прогнозування необхідний доступ до великих обсягів якісних даних та використання передових алгоритмів.

## 1.4 Аналіз та обгрунтування вибору технології розробки застосування

Вибір технологій для системи прогнозування цін на нерухомість базувався на комплексному аналізі відповідності вимогам проекту. Python обрано як фундаментальну платформу через його домінуючі позиції у сфері

машинного навчання. Екосистема бібліотек забезпечує готові рішення для обробки даних, побудови регресійних моделей та візуалізації результатів, що істотно прискорює цикл розробки порівняно з альтернативними мовами. Його перевага полягає у безшовній інтеграції наукових досліджень із промисловими рішеннями.

Scikit-learn визначено ядром машинного навчання через підтримку передових ансамблевих методів. Алгоритм градієнтного бустингу демонструє виняткову ефективність для нелінійних залежностей, характерних для ринку нерухомості. Здатність обробляти неповні дані та категоріальні змінні стала вирішальною при роботі з українськими джерелами, де якість інформації коливається.

Для веб-реалізації обрано Flask – мінімалістичний фреймворк, що забезпечує оптимальний баланс між продуктивністю та гнучкістю. Його архітектура дозволяє елегантно інтегрувати ML-моделі у API, зберігаючи швидкість відгуку навіть при обробці геопросторових даних. Це критично для інтерактивних прогнозів у реальному часі.

Систему зберігання побудовано на PostgreSQL через надійну підтримку структурованих даних. Можливість роботи з напівструктурованими форматами дозволяє аналізувати динамічні параметри ринку, характерні для українських реалій.

Ключовою перевагою архітектури стала стандартизація процесів обробки даних через конвеєрні механізми Scikit-learn. Це забезпечило цілісність робочого циклу – від відновлення пропущених значень до кодування категоріальних атрибутів.

# 2 РОЗРОБКА АЛГОРИТМІВ

## 2.1 Функціональна архітектура системи оцінки нерухомості. Детальний опис та декомпозиція процесів

У процесі розробки системи оцінки нерухомості використано функціональну архітектуру, представлену у нотації IDEF0 (Integration Definition for Function Modeling). Цей підхід дозволяє ієрархічно декомпозувати складну систему на послідовні, логічно взаємопов'язані функції. Він допомагає чітко зрозуміти її призначення, механізми функціонування та взаємодію з зовнішнім середовищем.

Підхід починається з контекстної діаграми (A0), яка є найвищим рівнем декомпозиції. Вона представляє дану систему як єдиний функціональний блок. Ця діаграма окреслює межі системи та її взаємодію з оточенням. Єдиним основним входом до даної системи є «Дані про нерухомість». Це сира, непереробленa інформація про об'єкт нерухомості, яку ми отримуємо з різних джерел. До цих даних належать такі атрибути, як площа, кількість кімнат, тип будівлі, географічні координати, поверх, наявність ремонту, рік побудови, а також, за наявності, дані про попередні продажі або оренду аналогічних об'єктів. Зрозуміло, що якість та повнота цих вхідних даних є критично важливою для подальшої точності оцінки.

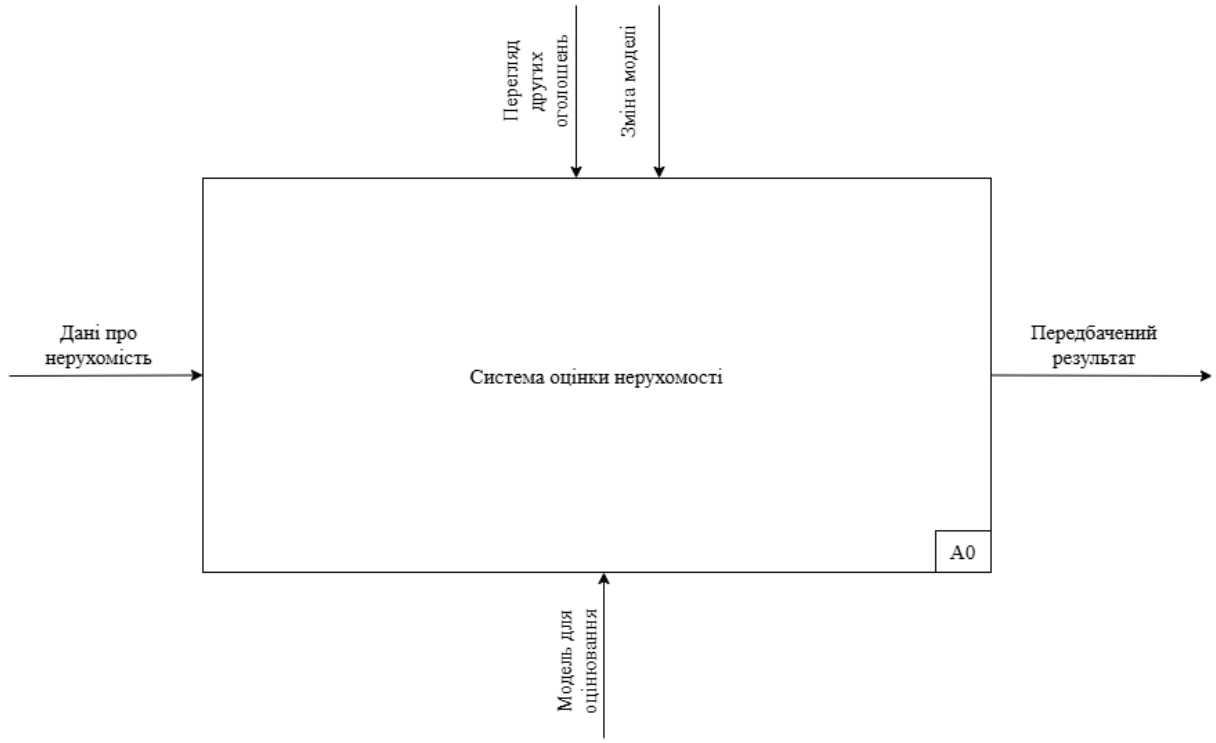


Рисунок 2.1 – Контекстна діаграма A0

Результатом функціонування даної системи є «Передбачений результат». У контексті оцінки нерухомості, це, перш за все, прогнозована ціна об'єкта. Однак, я прагну, щоб «передбачений результат» також включав додаткову інформацію, таку як діапазон можливих цін, фактори, що найбільше вплинули на оцінку, порівняльний аналіз з аналогічними об'єктами, або навіть прогнози зміни цін у майбутньому періоді. Для мене важливо, щоб цей вихід був чітким, зрозумілим та обґрунтованим для кінцевого користувача.

Дана система має функціонувати не ізольовано, а під впливом певних керуючих факторів, які визначають її поведінку та логіку. До них відносяться:

а) «Перегляд інших оголошень». Цей керуючий вплив відображає необхідність постійно враховувати актуальну інформацію з ринку. Дана система повинна мати можливість оновлювати свою базу даних та адаптуватися до змін цін на схожі об'єкти, які з'являються на ринку. Це забезпечує релевантність та актуальність нашого прогнозу.

b) «Зміна моделі». Фундаментальний керуючий вплив, що вказує на гнучкість та адаптивність нашої системи. У міру розвитку ринку, появи нових даних або вдосконалення алгоритмів машинного навчання, модель, яка використовується для оцінки, може потребувати коригування, перенавчання або повної заміни. Я вважаю цей аспект ключовим для підтримки високої точності в довгостроковій перспективі.

Механізмом, що забезпечує функціонування даної системи, є «Модель для оцінювання». Це серце системи, що складається з розроблених алгоритмів машинного навчання, навчених на великих обсягах даних про нерухомість. Модель виконує обчислення та формує прогноз на основі вхідних даних та керуючих впливів. У контексті прогнозування цін на нерухомість за допомогою Python, ця модель може включати різні алгоритми: від лінійних регресій до складних нейронних мереж або ансамблевих методів.

Друга діаграма представляє декомпозицію основної функції «Сайт оцінки нерухомості» на чотири послідовні підфункції, що відображають логічну послідовність етапів обробки даних, які буде реалізовано.

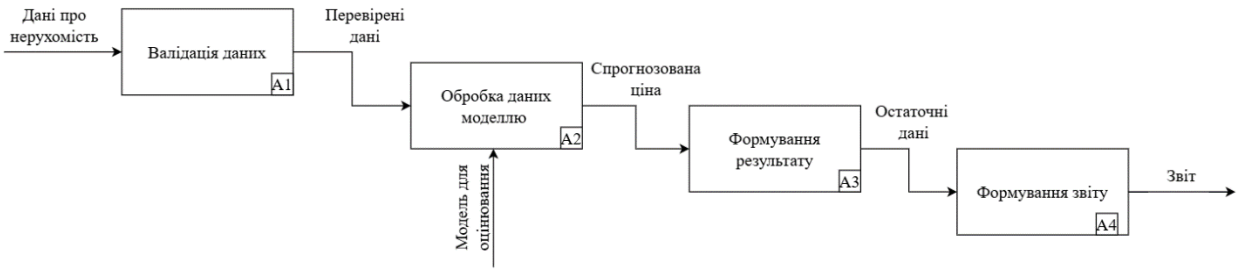


Рисунок 2.2 – Декомпозиція основної функції «Системи оцінювання нерухомості»

a) A1: валідація даних. Цей етап є першим та критично важливим кроком у обробці вхідної інформації, яку ми отримуємо. Основне завдання функції «Валідація даних» – це перевірка якості, повноти та коректності отриманих «Даних про нерухомість». Це включає:

1) перевірку на відсутність значень: виявляються пропуски у ключових полях (наприклад, площа, кількість кімнат).
2) перевірку на аномалії: шукаються нереалістичні значення (наприклад, площа квартири 1 м² або 10000 м²).
3) перевірку на відповідність форматів: забезпечується коректний тип даних для кожного атрибута (числові значення для площі, категоріальні для типу будівлі).
4) усунення дублікатів: видаляються повторювані записи, щоб уникнути спотворення даних.
5) приведення до єдиного формату: стандартизуються текстові описи або адреси для одноманітності. Виходом цієї функції для нас є «Перевірені дані», які вже очищені та готові до подальшої обробки.

b) A2: обробка даних моделлю функція «Обробка даних моделлю» є серцем даної системи прогнозування. Вона отримує «Перевірені дані» і застосовує до них «Модель для оцінювання» (механізм, визначений на діаграмі A0). На цьому етапі виконано:

1) інженерію ознак: створюються нові ознаки з існуючих даних (наприклад, щільність населення в районі, індекс доступності інфраструктури), щоб покращити якість моделі.
2) масштабування та нормалізацію: приводятся числові ознаки до єдиного діапазону для ефективної роботи алгоритмів машинного навчання.
3) прогнозування: застосовано навчену модель до підготовлених даних для генерації прогнозу ціни. Виходом цієї функції є «Спрогнозована ціна».

c) A3: формування результату отримана «Спрогнозована ціна» не завжди є кінцевим результатом у зручному для користувача вигляді. Функція «Формування результату» відповідає за перетворення цієї технічної інформації на зрозумілі «Остаточні дані». Це може включати:

1) округлення ціни: ціна представляється у зручному форматі (наприклад, до тисяч доларів).
2) додавання супутньої інформації: я включаю діапазон довіри до прогнозу, перелік ключових факторів, що вплинули на ціну, або порівняння з середніми ринковими показниками.
3) форматування для відображення: готуються дані для подальшої візуалізації на сайті або у звіті.

d) A4: формування звіту кінцевий етап функціональної декомпозиції – «Формування звіту». Ця функція бере «Остаточні дані» і перетворює їх на «Звіт» – кінцевий продукт, що надається користувачеві. Звіт може мати різні формати, залежно від потреб:

1) Веб-сторінка: інтерактивне відображення прогнозу на сайті.
2) PDF-документ: формалізований звіт з детальною інформацією.

Таким чином, функціональна архітектура, представлена діаграмами IDEF0, забезпечує комплексне розуміння системи оцінки нерухомості. Вона демонструє, як сирі дані про нерухомість послідовно проходять через етапи валідації, моделювання, формування результату та звітування, щоб надати користувачеві обґрунтований прогноз ціни. Я вважаю, що цей підхід є основою для проєктування та реалізації надійної та ефективної системи на базі машинного навчання з використанням Python.

## 2.2 Визначення функціональних вимог та взаємодії користувачів

Представлення діаграма прецедентів використання (Use Case Diagram) є ключовим інструментом для визначення функціональних вимог до системи оцінки нерухомості та моделювання її взаємодії з різними типами користувачів. Дана діаграма чітко розмежовує ролі користувачів (акторів) та функції, які вони можуть виконувати в системі, сприяючи глибокому розумінню функціоналу та його розподілу. Вона візуалізує, які завдання система дозволяє вирішувати, не вдаючись у деталі їхньої внутрішньої

реалізації, а зосереджуючись на зовнішній поведінці та корисності для акторів (див. рис. 2.3).


Рисунок 2.3 – Діаграма взаємодії користувачів

### 2.2.1 Актори системи. Розмежування ролей та повноважень

На діаграмі чітко ідентифіковані два основні актори, що взаємодіють із системою:

а) Користувач: це основний кінцевий споживач функціоналу системи, зацікавлений у отриманні прогнозів цін на нерухомість або пошуку відповідних об'єктів. Його взаємодія зосереджена на отриманні інформації та використанні аналітичних можливостей системи.

б) Адмін (Адміністратор): цей актор представляє технічний персонал або особу з розширеними повноваженнями, яка відповідає за підтримку, конфігурацію та забезпечення працездатності системи. Його дії спрямовані на управління внутрішніми ресурсами та забезпечення актуальності даних і моделей.

Розмежування цих ролей є фундаментальним для проєктування безпечної та ефективної системи, оскільки воно дозволяє чітко визначити рівні доступу та функціональні можливості для кожного типу користувача.

### 2.2.2 Прецеденти використання для Користувача. Ключові можливості системи

Функціонал, доступний для «Користувача», включає кілька основних прецедентів:

а) «Внести характеристики»: даний прецедент є відправною точкою для отримання прогнозу. Він дозволяє Користувачеві ввести специфічні параметри об'єкта нерухомості, для якого він бажає отримати оцінку. Це можуть бути такі дані, як площа, кількість кімнат, розташування, тип будівлі, стан ремонту, наявність додаткових зручностей тощо. Важливою умовою є надання інтуїтивно зрозумілого інтерфейсу для введення цих даних.

*Відношення include*: з прецеденту «Внести характеристики» автоматично викликається прецедент «Отримати прогноз цін». Це означає, що як тільки користувач вводить необхідні дані про об'єкт, система негайно приступає до генерації прогнозу його вартості. «Отримати прогноз цін» представляє собою основну функціональність системи – застосування навченої моделі машинного навчання до введених характеристик для отримання обґрунтованої цінової оцінки.

б) «Переглянути діаграми»: даний прецедент дозволяє Користувачеві візуалізувати різні аспекти ринку нерухомості. Це можуть бути графіки динаміки цін у часі, розподіл цін за районами, залежність вартості від ключових характеристик (наприклад, площа, кількість кімнат) або порівняльні діаграми. Візуалізація є критично важливою для кращого розуміння ринкових тенденцій та обґрунтованості прогнозів.

*Відношення extend*: з прецеденту «Переглянути діаграми» може бути розширення «Скористатись фільтрами». Це означає, що під час перегляду діаграм Користувач може додатково застосовувати різні фільтри (наприклад, за типом нерухомості, регіоном, періодом), щоб деталізувати представлену інформацію та адаптувати її під свої індивідуальні потреби.

в) «Переглянути оголошення»: даний прецедент надає Користувачеві можливість ознайомитися з реальними оголошеннями про продаж або оренду нерухомості, які були використані для навчання моделі або є актуальними на ринку. Це дозволяє Користувачеві самостійно порівняти свій об'єкт з аналогічними пропозиціями та перевірити релевантність наданого прогнозу.

*Відношення extend*: з прецеденту «Переглянути оголошення» також може бути розширення «Скористатись фільтрами». Це дозволяє Користувачеві фільтрувати оголошення за певними критеріями (наприклад, ціна, розташування, кількість кімнат), щоб швидко знайти найбільш релевантні пропозиції.

### 2.2.3 Прецеденти використання для Адміна. Управління системою

Функціонал, призначений для «Адміна», зосереджений на підтримці та оптимізації роботи системи:

а) «Оновити модель»: даний прецедент є ключовим для забезпечення актуальності та точності прогнозів. Він дозволяє Адміну перенавчати існуючу модель машинного навчання на нових даних, або завантажувати нову, більш досконалу модель. Це необхідно для адаптації системи до постійних змін на ринку нерухомості, включно з макроекономічними коливаннями, зміною законодавства та впливом зовнішніх факторів.

б) «Керувати БД» (Керувати базою даних): даний прецедент надає Адміну можливість виконувати операції з базою даних системи. Це

може включати додавання нових даних, редагування існуючих записів, видалення застарілої або некоректної інформації, а також моніторинг цілісності даних. Ефективне управління базою даних є фундаментальним для якості вхідної інформації, що живить прогностичні моделі.

в) «Переглянути статистику»: даний прецедент дозволяє Адміну отримувати доступ до системної статистики та метрик продуктивності. Це може включати дані про кількість запитів, завантаженість системи, час відповіді, а також статистику щодо точності прогнозів моделі (наприклад, середня абсолютна похибка, коефіцієнт детермінації) на тестових наборах даних. Аналіз цієї статистики допомагає Адміну відстежувати ефективність системи та виявляти потенційні проблеми.

### 2.2.4 Взаємозв'язки між прецедентами: include та extend

Використання відношень include та extend на діаграмі прецедентів є важливим для деталізації поведінки системи:

а) Відношення «include» (пунктирна лінія зі стрілкою, що вказує на включений прецедент) означає, що один прецедент обов'язково включає в себе функціональність іншого. Наприклад, «Внести характеристики» завжди тягне за собою «Отримати прогноз цін». Це свідчить про послідовну та невід'ємну залежність.

б) Відношення «extend» (пунктирна лінія зі стрілкою, що вказує на розширюючий прецедент) означає, що один прецедент може розширювати функціональність іншого, але не є обов'язковим. Наприклад, «Скористатись фільтрами» є опціональним розширенням для «Переглянути діаграми» або «Переглянути оголошення». Це дозволяє гнучко додавати функціонал без зміни основного сценарію.

Таким чином, модель прецедентів надає вичерпний опис того, як користувачі взаємодіють із системою оцінки нерухомості та які функціональні можливості вона пропонує. Вона є дорожньою картою для подальшої розробки, забезпечуючи відповідність кінцевого продукту потребам користувачів та вимогам до адміністрування. Ця діаграма допомагає чітко візуалізувати, як різні аспекти системи працюють разом, щоб забезпечити точні прогнози та зручний користувацький досвід.

## 2.3 Динамічна модель взаємодії та потоку робіт

Діаграма діяльності (Activity Diagram) є інструментом для візуалізації динамічної поведінки системи оцінки нерухомості. Вона демонструє послідовність дій, що виконуються різними акторами (Користувачем, Системою, Адміністратором), та потоки керування між цими діями. Ця діаграма допомагає чітко розтлумачити логіку бізнес-процесів, визначити точки синхронізації та паралелізму, а також ідентифікувати відповідальність кожного учасника за виконання певних завдань. Вона надає деталізоване уявлення про те, як дані перетворюються на кінцевий результат, крок за кроком.

Діаграма розділена на плавальні доріжки (swimlanes), що відображають ролі або компоненти, відповідальні за виконання конкретних дій: «Користувач», «Система» та «Адміністратор». Це забезпечує чітке розмежування обов'язків та візуалізацію взаємодії між ними (див. рис. 2.4).

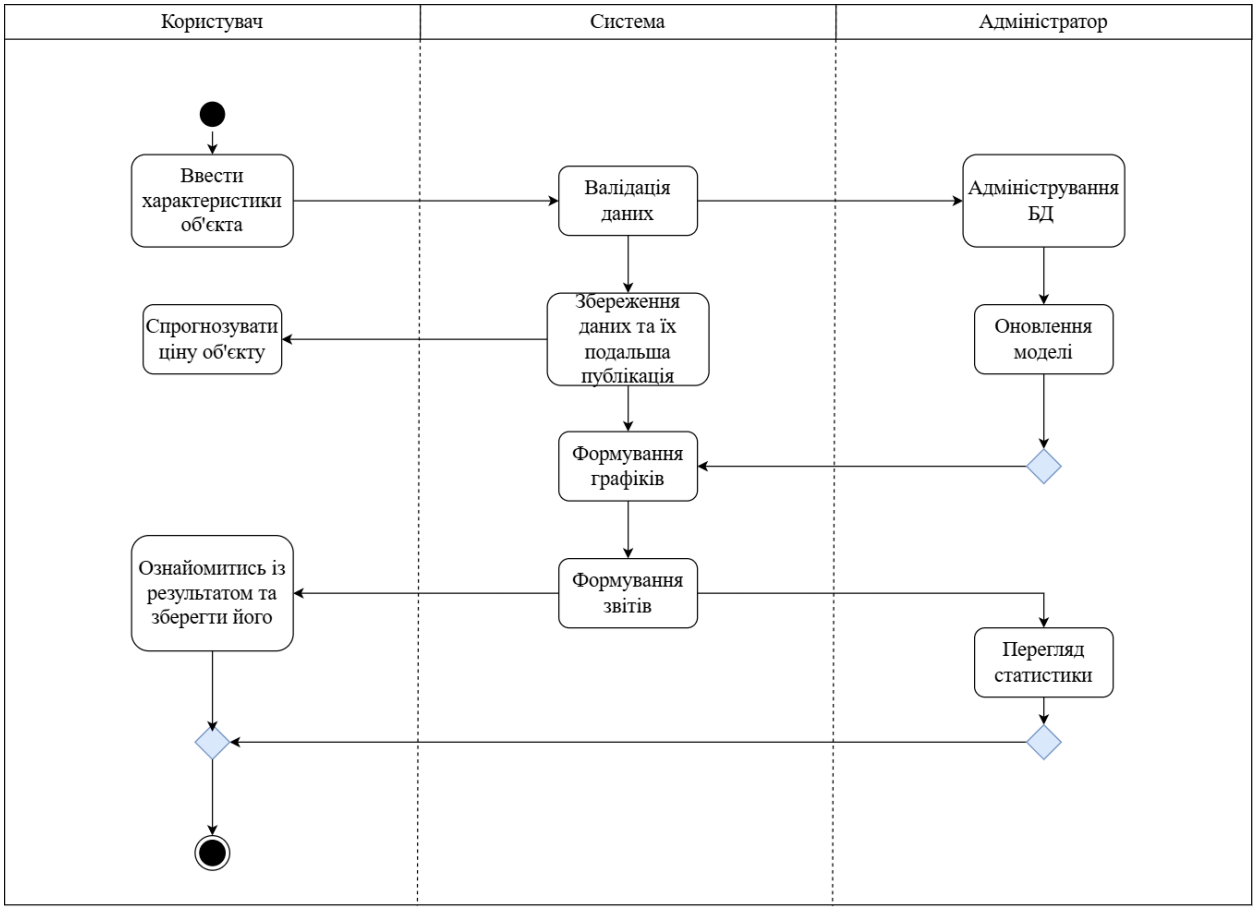


Рисунок 2.4 – Діаграма діяльності

### 2.3.1 Потік діяльності. Сценарій Користувача та системні процеси

Потік починається з початкового вузла (чорне коло) і веде до завершального вузла (чорне коло з обведенням), охоплюючи основний сценарій взаємодії Користувача із системою для отримання прогнозу.

а) «Внести характеристики об'єкта» (Користувач): це перша дія, що ініціюється Користувачем. Вона передбачає введення всіх необхідних даних про об'єкт нерухомості, для якого бажається отримати оцінку. Ці характеристики можуть включати площу, кількість кімнат, адресу, тип будівлі, рік побудови, стан ремонту, наявність інфраструктури поблизу та інші релевантні параметри. Ця дія є критично важливою, оскільки якість та повнота наданої інформації безпосередньо впливає на точність подальшого прогнозу.

б) «Валідація даних» (Система): після того, як Користувач вводить характеристики, керування передається Системі для виконання дії «Валідація даних». На цьому етапі відбувається автоматична перевірка вхідної інформації на відповідність встановленим правилам. Це може включати перевірку на відсутність обов'язкових полів, відповідність типів даних, діапазонів значень (наприклад, площа не може бути від'ємною), а також виявлення потенційних аномалій або помилок. Метою валідації є забезпечення якості даних перед їх подальшою обробкою.

в) «Зберігання даних та їх подальша публікація» (Система): після успішної валідації, Система виконує дію «Зберігання даних та їх подальша публікація». Це передбачає збереження перевіреної інформації про об'єкт нерухомості у внутрішній базі даних системи. «Подальша публікація» в цьому контексті може означати внесення цих даних до пулу для навчання моделі або ж їхнє використання для формування порівняльних звітів та графіків. Ця дія є важливою для підтримки актуальної та зростаючої бази даних, що є основою для ефективного машинного навчання.

г) «Формування графіків» (Система): наступна дія, що виконується Системою, це «Формування графіків». На цьому етапі система використовує як введені Користувачем дані, так і інформацію з власної бази даних, для генерації візуальних представлень. Це можуть бути графіки динаміки цін у певному районі, порівняльні діаграми цін на аналогічні об'єкти, залежність вартості від кількості кімнат або інших характеристик. Формування графіків є важливим для надання Користувачеві додаткового контексту та наочності.

д) «Спрогнозувати ціну об'єкту» (Система): паралельно або після формування графіків, Система виконує свою ключову функцію – «Спрогнозувати ціну об'єкту». На цьому етапі до валідованих даних застосовується навчена модель машинного навчання (наприклад,

модель градієнтного бустингу або нейронна мережа). Ця дія є центральною для всієї системи, оскільки вона генерує безпосередньо прогнозовану вартість нерухомості на основі складного аналізу вхідних характеристик та історичних даних.

е) «Формування звітів» (Система): після того, як ціна спрогнозована, Система переходить до дії «Формування звітів». На цьому етапі відбувається агрегація отриманого прогнозу, супутніх графіків, а також, можливо, порівняльної інформації з бази даних. Результатом є структурований звіт, який може включати прогнозовану ціну, діапазон довіри, ключові фактори впливу та інші аналітичні дані. Цей звіт призначений для подальшого подання Користувачеві.

ж) «Ознайомитись із результатом та зберегти його» (Користувач): це завершальна дія для Користувача в даному сценарії. Після того, як система сформувала звіт, Користувач може переглянути отриманий прогноз та супутню інформацію. Він має можливість зберегти звіт для подальшого використання або аналізу. Ця дія закриває цикл взаємодії Користувача із системою щодо конкретного об'єкта нерухомості.

### 2.3.2 Потік діяльності. Сценарій Адміністратора та підтримка системи

Паралельно до основного сценарію Користувача, на діаграмі представлений потік діяльності для «Адміністратора», що відображає функції підтримки та обслуговування системи.

а) «Адміністрування БД» (Адміністратор): дана дія дозволяє Адміністратору виконувати різні операції з базою даних системи. Це може включати додавання нових наборів даних для навчання, редагування або видалення застарілих записів, а також моніторинг цілісності та продуктивності бази даних. Ефективне адміністрування

бази даних є фундаментальним для забезпечення актуальності та якості даних, що використовуються системою.

б) «Оновлення моделі» (Адміністратор): дана дія є критично важливою для підтримки точності та актуальності прогностичної системи. Вона дозволяє Адміністратору перенавчати існуючу модель машинного навчання на нових або оновлених даних, або ж завантажувати повністю нову версію моделі. Оновлення моделі необхідне для адаптації системи до змінних ринкових умов, появи нових чинників впливу або вдосконалення алгоритмів прогнозування.

в) «Перегляд статистики» (Адміністратор): після того, як Адміністратор оновив модель або вніс зміни до бази даних, він може перейти до дії «Перегляд статистики». Це дозволяє Адміністратору аналізувати різні метрики ефективності системи. Це можуть бути показники точності прогнозування (наприклад, середня абсолютна похибка), завантаженість сервера, час відповіді на запити або інші системні показники. Моніторинг статистики є важливим для виявлення проблем, оцінки ефективності змін та планування подальших оптимізацій.

### 2.3.3 Точки злиття (Merge Nodes) та завершення потоків

Діаграма діяльності використовує вузли злиття (ромби), які показують, що потік виконання може продовжуватися після однієї з кількох можливих попередніх дій. Це дозволяє гнучко моделювати сценарії, де послідовність дій може мати розгалуження, але потім сходиться до єдиного подальшого кроку.

Завершальний вузол (чорне коло з обведенням) позначає кінець потоку діяльності для конкретного сценарію. У цьому випадку, він відображає завершення сценарію Користувача після ознайомлення з результатом, а також завершення відповідних гілок адміністрування.

Таким чином, діаграма діяльності надає динамічне та детальне уявлення про функціонування системи оцінки нерухомості. Вона чітко ілюструє взаємодію між Користувачем, Системою та Адміністратором, послідовність виконання ключових операцій, а також точки прийняття рішень та синхронізації. Це є незамінним інструментом для розуміння бізнес-логіки та ефективного управління процесом розробки та підтримки системи.

## 2.4 Структура інформаційного середовища

Сутність-зв'язкова (ER) діаграма є фундаментальним елементом проєктування інформаційного середовища системи оцінки нерухомості. Ця діаграма візуально демонструє логічну структуру бази даних, описуючи ключові сутності (об'єкти, що зберігають інформацію), їхні атрибути (характеристики цих об'єктів) та зв'язки (відносини) між ними. Побудова ER-моделі є критично важливою для забезпечення цілісності даних, ефективності зберігання інформації та підтримки складних запитів, необхідних для функціонування прогностичної системи на базі машинного навчання.

### 2.4.1 Опис сутностей та їхніх атрибутів

ER-діаграма охоплює три основні сутності, кожна з яких відіграє певну роль у системі (див. рис. 2.5):

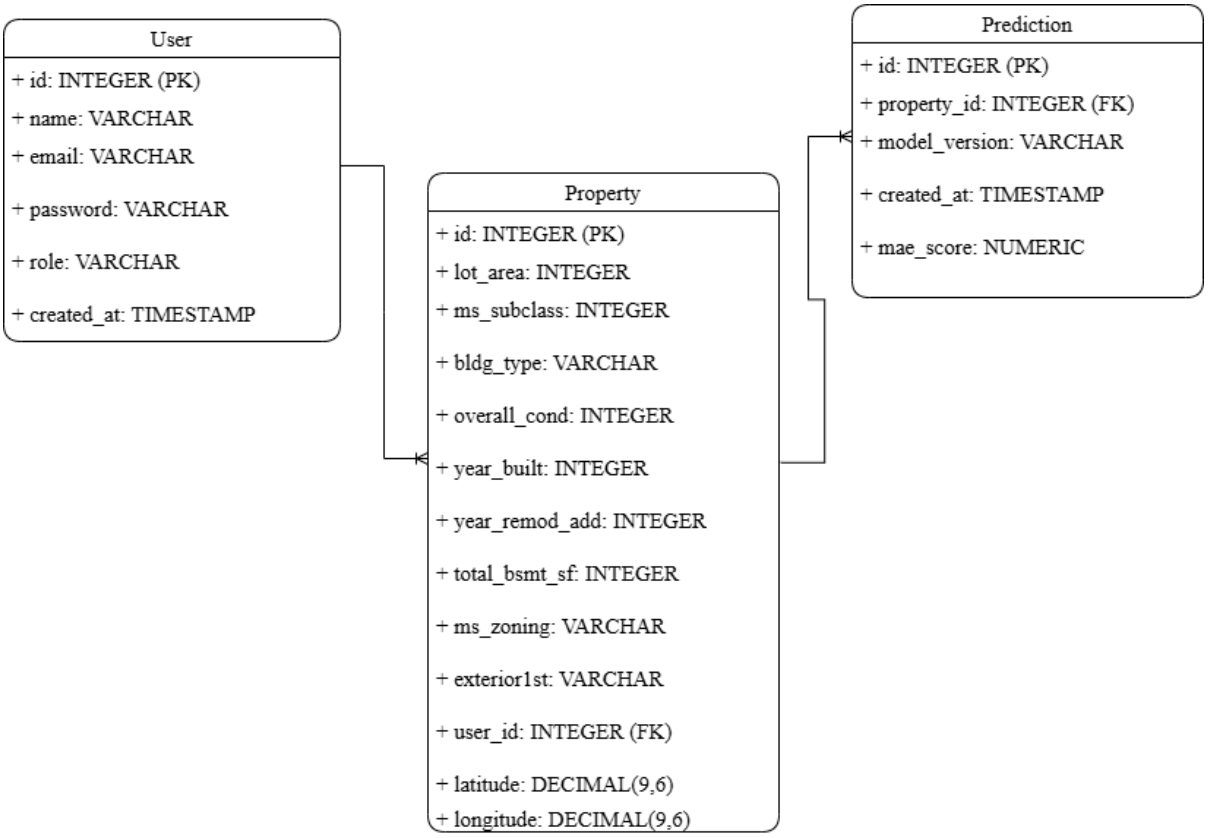


Рисунок 2.5 – Сутність – зв’язкова діаграма

**а) Сутність "User" (Користувач)**

Ця сутність призначена для зберігання інформації про користувачів системи. Кожен запис у цій таблиці представляє унікального користувача та містить його основні ідентифікаційні та авторизаційні дані.

1) id: INTEGER (PK) – Первинний ключ, що унікально ідентифікує кожного користувача. Автоматично генерується ціле число.
2) name: VARCHAR – Ім'я або нікнейм користувача. Поле типу VARCHAR, що дозволяє зберігати рядкові дані змінної довжини.
3) email: VARCHAR – Електронна пошта користувача. Зазвичай це унікальне поле, що використовується для входу в систему та відновлення пароля.
4) password: VARCHAR – Хешований пароль користувача. Зберігання хешованого, а не відкритого пароля є стандартом безпеки.

5) role: VARCHAR – Роль користувача в системі. Поле дозволяє розрізняти різні типи користувачів, наприклад, «звичайний користувач» та «адміністратор», що впливає на їхні права доступу до функціоналу.
6) created_at: TIMESTAMP – Дата та час створення облікового запису користувача. Інформація є корисною для аудиту та аналізу активності користувачів.

**б) Сутність "Property" (Нерухомість)**

Сутність є центральною у системі, оскільки вона зберігає детальні характеристики об'єктів нерухомості, які є основою для прогнозування цін. Кожен запис у цій таблиці відповідає одному об'єкту нерухомості.

1) id: INTEGER (PK) – Первинний ключ, що унікально ідентифікує об'єкт нерухомості.
2) lot_area: INTEGER – Площа земельної ділянки або загальна площа об'єкта (якщо це квартира) у відповідних одиницях виміру. Це ціле число, що є однією з ключових характеристик.
3) ms_subclass: INTEGER – Має значення, що відображає тип або підклас житла, наприклад, категорія забудови (житлова, комерційна). Зазвичай це числове кодування.
4) bldg_type: VARCHAR – Тип будівлі (наприклад, окремий будинок, таунхаус, багатоквартирний будинок). Це текстове поле.
5) overall_cond: INTEGER – Загальний стан нерухомості, який може бути оцінений за певною шкалою (наприклад, від 1 до 10).
6) year_built: INTEGER – Рік побудови об'єкта. Інформація є важливою для оцінки зносу та відповідності сучасним стандартам.
7) year_remod_add: INTEGER – Рік останньої реконструкції або добудови. Якщо реконструкції не було, може збігатися з роком побудови.

8) total_bsmnt_sf: INTEGER – Загальна площа підвального приміщення у квадратних футах (або метрах). Важлива характеристика, що впливає на загальну цінність.
9) ms_zoning: VARCHAR – Зональне призначення території відповідно до містобудівних планів (наприклад, житлова низької щільності, комерційна). Текстове поле, що описує правовий статус землі.
10) exterior1st: VARCHAR – Тип зовнішньої обробки будівлі (наприклад, цегла, сайдинг).
11) user_id: INTEGER (FK) – Зовнішній ключ, що посилається на id у сутності "User". Поле вказує, який користувач вніс інформацію про даний об'єкт нерухомості. Це забезпечує зв'язок між користувачами та об'єктами.
12) latitude: DECIMAL(9,6) – Географічна широта розташування об'єкта. Використовується для просторового аналізу.
13) longitude: DECIMAL(9,6) – Географічна довгота розташування об'єкта. Разом з широтою дозволяє точно визначити місцезнаходження.

**в) Сутність "Prediction" (Прогноз)**

Сутність призначена для зберігання результатів прогнозування цін, згенерованих системою. Вона дозволяє зберігати історію прогнозів та пов'язувати їх з конкретними об'єктами нерухомості та моделями, що їх сформували.

1) id: INTEGER (PK) – Первинний ключ, що унікально ідентифікує кожен запис прогнозу.
2) property_id: INTEGER (FK) – Зовнішній ключ, що посилається на id у сутності "Property". Зв'язок дозволяє точно визначити, до якого об'єкта нерухомості відноситься даний прогноз.
3) model_version: VARCHAR – Ідентифікатор або версія моделі машинного навчання, яка була використана для генерації даного

прогнозу. Це важливо для відстеження змін у точності та поведінці моделей.

4) created_at: TIMESTAMP – Дата та час створення прогнозу. Інформація дозволяє відстежувати, коли був зроблений прогноз, що важливо для аналізу актуальності.
5) mae_score: NUMERIC – Метрика оцінки моделі, наприклад, середня абсолютна похибка (Mean Absolute Error). Поле може зберігати значення, що вказує на точність прогнозу. Воно є важливим для моніторингу якості прогностичних моделей.

### 2.4.2 Зв'язки між сутностями: Взаємодія даних

Між сутностями встановлені такі зв'язки, що відображають логічні відносини між даними:

а) "User" та "Property" (один-до-багатьох): Зв'язок показує, що один користувач може додати (або бути асоційованим з) багато об'єктів нерухомості, але кожен об'єкт нерухомості належить лише одному користувачеві (який його вніс). Реалізовано за допомогою зовнішнього ключа user_id у таблиці "Property", який посилається на id у таблиці "User".

б) "Property" та "Prediction" (один-до-багатьох): Зв'язок вказує на те, що для одного об'єкта нерухомості може бути зроблено багато прогнозів. Це дозволяє відстежувати історію прогнозів для конкретного об'єкта, наприклад, при перенавчанні моделі або оновленні даних. Зовнішній ключ property_id у таблиці "Prediction" посилається на id у таблиці "Property".

### 2.4.3 Значення ER-діаграми для системи

Побудована ER-діаграма є критично важливою для кількох аспектів розробки системи:

а) Проєктування бази даних: вона служить безпосередньою основою для створення фізичної схеми бази даних, визначення таблиць, полів, первинних та зовнішніх ключів, а також типів даних.
б) Забезпечення цілісності даних: визначені зв'язки та ключі допомагають підтримувати консистентність даних, запобігаючи втраті зв'язків між об'єктами.
в) Оптимізація запитів: чітка структура даних дозволяє ефективно будувати SQL-запити для отримання, фільтрації та агрегації інформації, що є фундаментальним для навчання моделей машинного навчання та генерації прогнозів.
г) Розуміння системи: діаграма «сутність-зв'язок» надає чітке візуальне уявлення про те, як дані організовані та взаємопов'язані всередині системи, що є важливим для всіх учасників розробки та подальшої підтримки.
д) Масштабованість: добре спроєктована модель дозволяє легко розширювати систему, додаючи нові атрибути або сутності без значної реструктуризації.

ER-діаграма є наріжним каменем архітектури даних, що забезпечує надійне зберігання та управління інформацією, необхідною для функціонування системи прогнозування цін на нерухомість за допомогою машинного навчання.

## 2.5 Візуалізація структури та функціональних зон

Розробка візуального представлення системи оцінки нерухомості починається з проєктування макетів інтерфейсу користувача (UI layouts). Схеми є визначальними для архітектури інформації на веб-сторінках, оскільки вони окреслюють розташування основних функціональних блоків та їхню взаємодію. Створення макетів становить початковий етап у формуванні інтуїтивно зрозумілого та ергономічного користувацького досвіду, що дозволяє особам, які працюють із системою, ефективно взаємодіяти з нею та

отримувати необхідні відомості. Кожен макет відображає конкретну функціональну сторінку або режим функціонування системи, забезпечуючи глибоке розуміння її внутрішньої структури.

Перший макет демонструє типову сторінку, призначену для перегляду списку об'єктів нерухомості та застосування фільтрів (див. рис. 2.6). Ця сторінка є ключовою для користувачів, які бажають ознайомитися з поточною ринковою пропозицією або шукають об'єкти за певними критеріями.

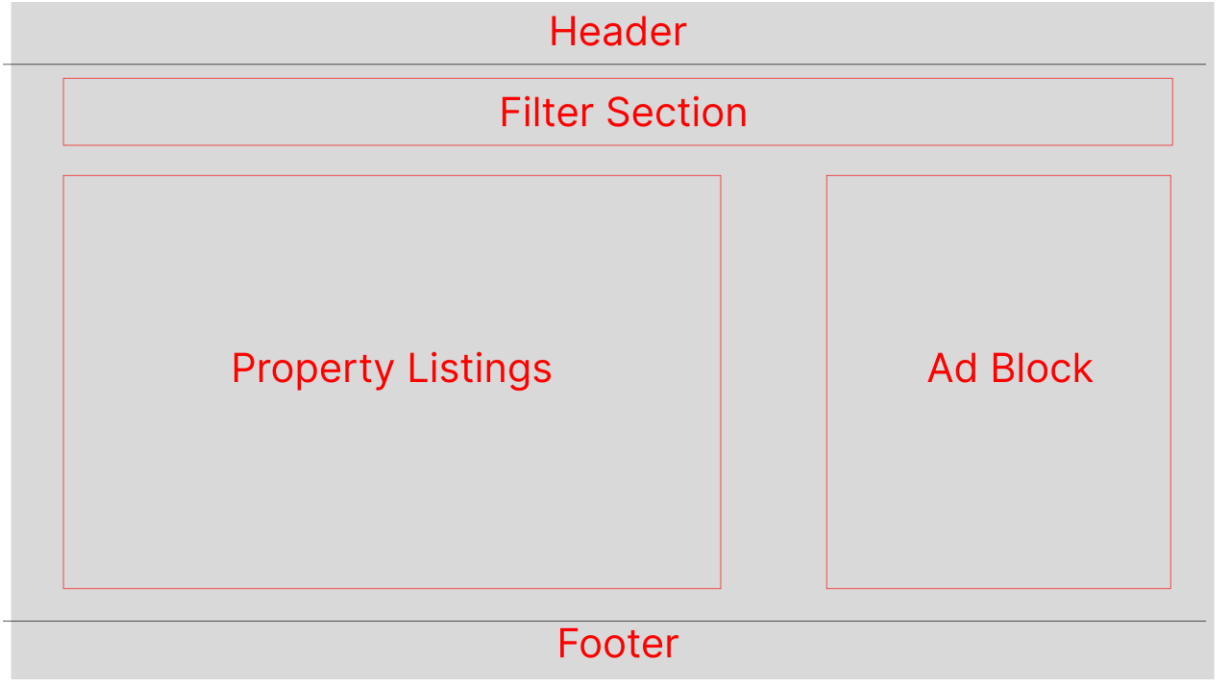


Рисунок 2.6 – Макет інтерфейсу для перегляду доступних оголошень

а) Header (Верхній колонтитул): блок розташований у верхній частині сторінки і є невід'ємною частиною візуальної ідентичності системи. Він містить логотип, назву системи, а також, ймовірно, елементи навігації (меню, посилання на інші розділи системи, такі як «Оцінити», «Профіль користувача») та кнопки доступу (вхід/реєстрація). Присутність хедера на всіх сторінках забезпечує консистентність інтерфейсу та легкість навігації.

б) Filter Section (Секція фільтрів): розташована безпосередньо під хедером, дана секція є критично важливою для деталізованого пошуку. Вона містить елементи керування, що дозволяють користувачеві звузити список об'єктів, що відображаються. Типові фільтри можуть включати:

1) Тип нерухомості: квартира, будинок, земельна ділянка, комерційна нерухомість.
2) Місцезнаходження: місто, район, вулиця.
3) Ціновий діапазон: від мінімальної до максимальної ціни.
4) Основні характеристики: площа (загальна, житлова), кількість кімнат, поверх, рік побудови.
5) Додаткові параметри: наявність ремонту, меблів, тип опалення, близькість до метро чи інфраструктури. Ефективна реалізація цієї секції забезпечує користувачеві потужний інструмент для швидкого та точного пошуку.

в) Property Listings (Список об'єктів нерухомості): великий блок займає основну частину лівої області сторінки і є центральним елементом для відображення результатів пошуку. Тут представлені оголошення про нерухомість у вигляді списку або сітки. Кожен елемент списку зазвичай містить ключову інформацію про об'єкт: мініатюру фотографії, адресу, ціну, основні характеристики (площа, кількість кімнат), а також коротку опис. Користувач може клікнути на оголошення для перегляду детальної інформації про об'єкт. Блок динамічно оновлюється відповідно до застосованих фільтрів.

г) Ad Block (Рекламний блок): розташований праворуч від списку об'єктів, даний блок призначений для розміщення рекламних матеріалів. Його наявність може бути зумовлена монетизацією платформи або використанням для відображення релевантних пропозицій від партнерів (наприклад, банківські послуги, страхування нерухомості, послуги ремонту). Розмір та вміст цього блоку можуть

варіюватися, проте його основна функція – надання додаткових доходів або просування супутніх послуг.

д) Footer (Нижній колонтитул): блок розташований у самому низу сторінки і містить загальну інформацію про систему. До типового футера належать:

1) Контактна інформація.
2) Посилання на політику конфіденційності та умови використання.
3) Інформація про авторські права.
4) Посилання на соціальні мережі або додаткові розділи (наприклад, FAQ, блог). Футер забезпечує доступ до важливої юридичної та довідкової інформації.

Другий макет відображає сторінку, призначену для введення даних про об'єкт нерухомості та відображення прогнозованої ціни (див. рис. 2.7). Це основна функціональна сторінка для користувачів, які бажають отримати оцінку конкретного об'єкта.

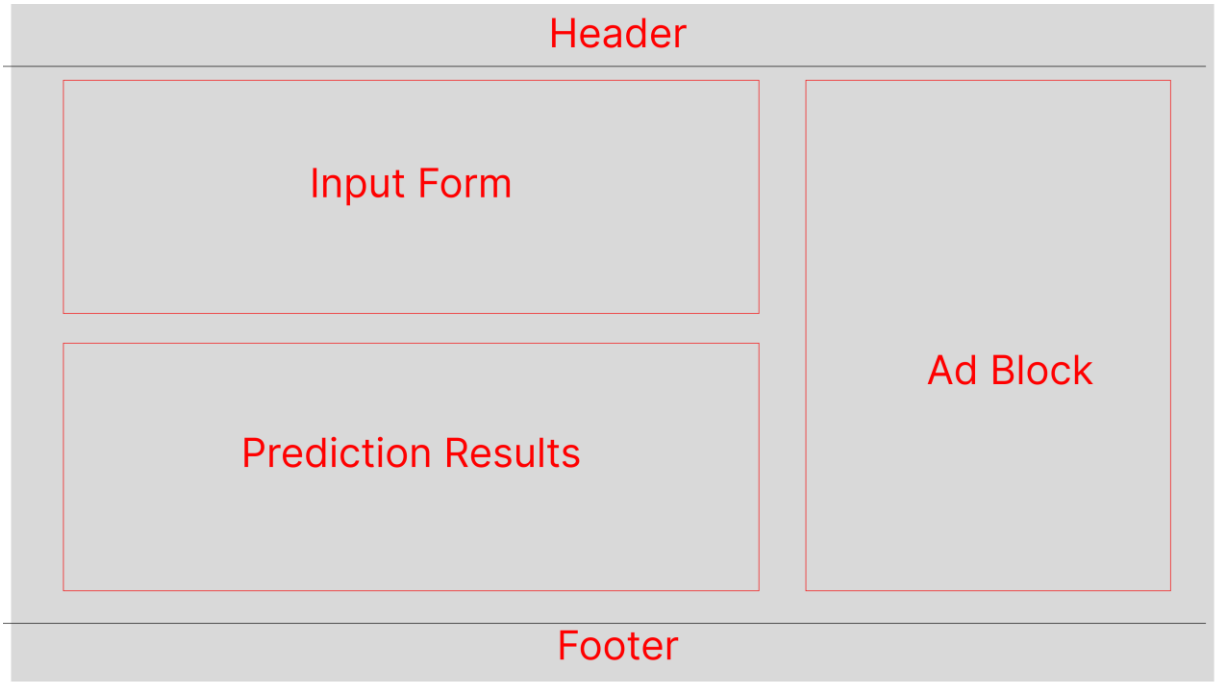


Рисунок 2. 7 – Макет інтерфейсу для введення даних про нерухомість

- Header (Верхній колонтитул): як і в першому макеті, хедер зберігає свою позицію та функціонал, забезпечуючи консистентність навігації та брендування системи.
- Input Form (Форма введення даних): даний великий блок займає основну частину лівої верхньої області сторінки. Він містить інтерактивні поля для введення всіх характеристик об'єкта нерухомості, необхідних для роботи прогностичної моделі. Форма повинна бути інтуїтивно зрозумілою та зручною для заповнення. Поля можуть бути згруповані за категоріями (наприклад, «Загальні дані», «Розташування», «Характеристики будівлі»). Використання випадаючих списків, повзунків та автозаповнення може значно покращити користувацький досвід. Після заповнення форми, користувач активує процес прогнозування (наприклад, за допомогою кнопки «Оцінити»).
- Prediction Results (Результати прогнозування): розташований під формою введення, блок призначений для відображення результатів, отриманих від моделі машинного навчання. Після обробки введених даних, тут буде відображено прогнозовану ціну об'єкта нерухомості. Блок може також містити додаткову інформацію, таку як:
    1) Діапазон довіри до прогнозу (наприклад, «ціна очікується в діапазоні від X до Y»).
    2) Список ключових факторів, що найбільше вплинули на отриману ціну (наприклад, площа, район, рік побудови).
    3) Порівняння з середньою ціною аналогічних об'єктів.
    4) Посилання на графіки або статистику, що підтверджують прогноз. Чітке та зрозуміле представлення цих результатів є критично важливим для довіри користувача до системи.
- Ad Block (Рекламний блок): аналогічно першому макету, блок призначений для розміщення реклами або супутніх пропозицій, що можуть бути релевантними для користувача, який оцінює

нерухомість (наприклад, іпотечні кредити, послуги оцінки, страхування).

– Footer (Нижній колонтитул): футер зберігає свою структуру та вміст, забезпечуючи доступ до важливої інформації в нижній частині сторінки.

### 2.5.1 Загальні принципи проєктування макетів

Обидва макети відображають ключові принципи веб-дизайну:

- Консистентність: Використання однакових елементів (хедер, футер, рекламний блок) на різних сторінках створює відчуття єдності та полегшує навігацію.
- Чітке зонування: Кожен блок виконує певну функцію, що допомагає користувачеві швидко орієнтуватися на сторінці та знаходити потрібну інформацію.
- Ієрархія інформації: Основний контент (список оголошень або форма введення/результати прогнозу) займає центральне місце, привертаючи найбільшу увагу.
- Гнучкість: Макети передбачають можливість адаптації для різних розмірів екранів та пристроїв (хоча це не відображено безпосередньо, їхня блокова структура сприяє цьому).

Макети слугують візуальною основою для подальшої розробки фронтенд-частини системи, перетворюючи функціональні вимоги на конкретний інтерфейс користувача. Вони є дорожньою картою для дизайнера та розробника, забезпечуючи узгодженість візуальної складової з бізнес-логікою та технічною реалізацією.

# 3 ПРОЄКТУВАННЯ БАЗ ДАНИХ І ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

## 3.1 Створення моделі для прогнозування вартості нерухомості

Процес побудови моделі машинного навчання для прогнозу ціни об'єктів житлової нерухомості включав декілька ключових етапів: підготовку вибірки, побудову фічей, тренування моделі, її оцінювання та валідацію результатів. Основною ціллю було створити регресійну модель, здатну з високою точністю відтворювати ринкову вартість об'єкта на основі його характеристик.

### 3.1.1 Вибір моделі та підхід до тренування

Для вирішення задачі було використано алгоритм HistGradientBoostingRegressor із бібліотеки scikit-learn, який вирізняється високою точністю градієнтного бустингу та ефективною обробкою пропущених значень. Щоб забезпечити якісне прогнозування, модель була інтегрована у повноцінний Pipeline, що складався з таких етапів:

- імпутація пропущених даних у числових змінних (із застосуванням середнього або медіани);
- кодування категоріальних ознак методом OneHotEncoder;
- масштабування числових характеристик;
- ансамблеве навчання з підбором гіперпараметрів за допомогою крос-валідації.

Для тренування моделі використовували вибірку з понад 1000 реальних оголошень, у якій були представлені такі основні характеристики, як площа, кількість кімнат, стан ремонту, наявність балкона, тип будівлі, поверховість тощо.

### 3.1.2 Побудова моделі на основі HistGradientBoostingRegressor

Прогнозна модель реалізована у вигляді конвеєра обробки даних, що інтегрує три ключові компоненти: попередню обробку ознак, градієнтний бустинг та процедури валідації. Ядро системи складає конвеєр scikit-learn, де трансформація даних та регресія об'єднані в єдиний процес:

```
    model = Pipeline(steps=[
        ('preprocessor', ColumnTransformer(transformers=[
            ('num', numeric_transformer, numeric_features),
            ('cat',                      categorical_transformer,
categorical_features)
        ])),
        ('regressor', HistGradientBoostingRegressor(
            random_state=42, max_iter=200, max_depth=10
        ))
    ])
```

Імпутація пропущених даних реалізована через роздільні стратегії для числових та категоріальних змінних. Числові дані заповнюються медіаною – робастною до викидів мірою центральної тенденції. Категоріальні ознаки обробляються заміною на моду категорій з подальшим one-hot кодуванням, що гарантує збереження семантики:

```
    numeric_transformer = SimpleImputer(strategy='median')
    categorical_transformer = Pipeline(steps=[
        ('imputer', SimpleImputer(strategy='most_frequent')),
        ('onehot', OneHotEncoder(handle_unknown='ignore'))
    ])
```

Оптимізація гіперпараметрів бустингу базується на емпіричних тестах: кількість ітерацій (200) забезпечує збіжність без перенавчання, глибина дерев (10) дозволяє виявляти нелінійні залежності. Автоматичне визначення категоріальних ознак після one-hot перетворення усуває необхідність ручного вказівки.

Стратифіковане розділення даних (80/20) з квінтильною стратифікацією за ціною гарантує репрезентативність вибірок. Це критично важливо для об'єктів преміум-сегменту, де концентрація спостережень нижча:

```
    X_train, X_test, y_train, y_test = train_test_split(
        features,             target,             test_size=0.2,
stratify=pd.qcut(target, q=4), random_state=42
    )
```

Для оцінки якості моделі використано три метрики:

– середня абсолютна похибка (MAE), що вимірює середню модульну різницю між прогнозованими значеннями та фактичними:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \,, \qquad (3.1)$$

де $y_i$– фактичне значення цільової змінної для *i*-го спостереження, $\hat{y}_i$– відповідне прогнозоване значення, а *n*– загальна кількість спостережень.

Дана метрика відображає середній розмір абсолютної помилки прогнозу, не підсилюючи вплив великих відхилень, що робить її інтуїтивно зрозумілою для інтерпретації. У розглянутому випадку значення MAE становить 28,5 тисяч гривень, що свідчить про середнє абсолютне відхилення прогнозів від фактичних цін.

– корінь середньоквадратичної похибки (RMSE) визначається як квадратний корінь із середнього арифметичного квадратів різниць між фактичними та прогнозованими значеннями:

$$\text{MAE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \,, \qquad (3.2)$$

Корінь середньоквадратичної похибки (RMSE) демонструє підвищену чутливість до значних відхилень між прогнозованими та фактичними значеннями, оскільки кожна помилка підноситься до квадрату перед усередненням. Це означає, що поодинокі великі похибки мають непропорційно великий вплив на підсумковий показник RMSE, порівняно з дрібнішими відхиленнями. У даному випадку отримане значення RMSE –

42,1 тисячі гривень – свідчить про наявність окремих прогнозів із великими помилками, які суттєво підвищують загальну оцінку цієї метрики.

– коефіцієнт детермінації ($R^2$), який відображає частку дисперсії залежної змінної, що пояснюється моделлю:

$$R^2 = 1 - \frac{\sum_{i=0}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=0}^{n}(y_i - \bar{y}_i)^2}, \quad (3.3)$$

де $\bar{y}_i$–середнє значення фактичних спостережень.

Чисельно $R^2$ варіюється від 0 до 1, де значення, близькі до 1, свідчать про високу точність моделі у поясненні варіації даних. У даному випадку $R^2$=0,872, що означає, що модель пояснює 87,2% варіації цільової змінної, демонструючи високий рівень адекватності моделі.

Таким чином, наведені метрики забезпечують комплексну оцінку якості моделі: MAE відображає середню абсолютну помилку без надмірного впливу викидів, RMSE акцентує увагу на великих відхиленнях, а $R^2$ демонструє здатність моделі пояснювати варіативність даних. Такий підхід є стандартним у наукових дослідженнях для оцінювання точності прогнозних моделей.

### 3.1.3 Валідація прогнозної здатності моделі: діаграма розсіювання

Оцінка моделі почалась із візуального аналізу співвідношення між фактичними і передбаченими цінами.

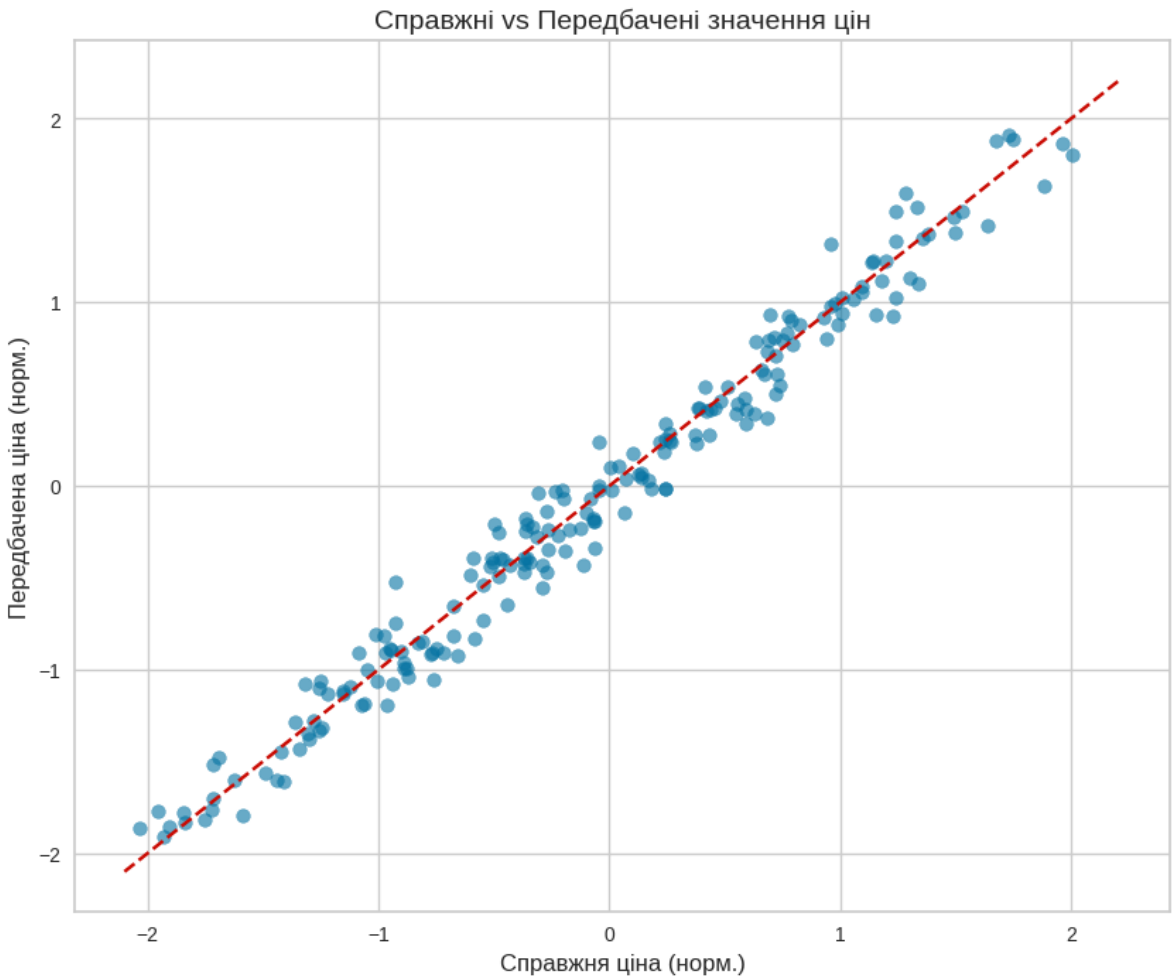


Рисунок 3. 1 – Діаграма співвідношення реальних та фактичних цін

Представлена візуалізація (див. рис. 3.1) демонструє ступінь відповідності між прогнозованими та емпіричними значеннями, обидва типи даних нормалізовані (осеві позначки від -2 до 2 SD). Ключові аспекти:

- прогнозовані значення демонструють високий рівень наближеності до фактичних, що свідчить про ефективність моделі у відтворенні реальної динаміки цін.
- щільність розподілу точок вздовж діагоналі підтверджує статистичну значущість моделі ( $R^2$ =0.872). Гомоскедастичний характер розсіювання (рівномірна дисперсія за всіма діапазонами) вказує на відсутність систематичних зміщень.

– нормалізація даних (позначена як "норм.") означає, що значення представлені у z-score: позитивні значення вказують на ціни вище середньоринкових, негативні – нижче.

Систематичний аналіз розподілу точок щодо діагоналі дає змогу ідентифікувати напрями вдосконалення моделі, зокрема оптимізацію прогнозів для об'єктів із екстремальними значеннями ознак.

### 3.1.4 Діагностика припущень регресійного аналізу: розподіл залишків

Залишки моделі, визначені як різниця між емпіричними значеннями ціни та їх прогнозованими оцінками**,** є фундаментальним інструментом діагностики якості регресійних моделей. Аналіз їх розподілу дозволяє верифікувати виконання ключових статистичних припущень методу найменших квадратів.

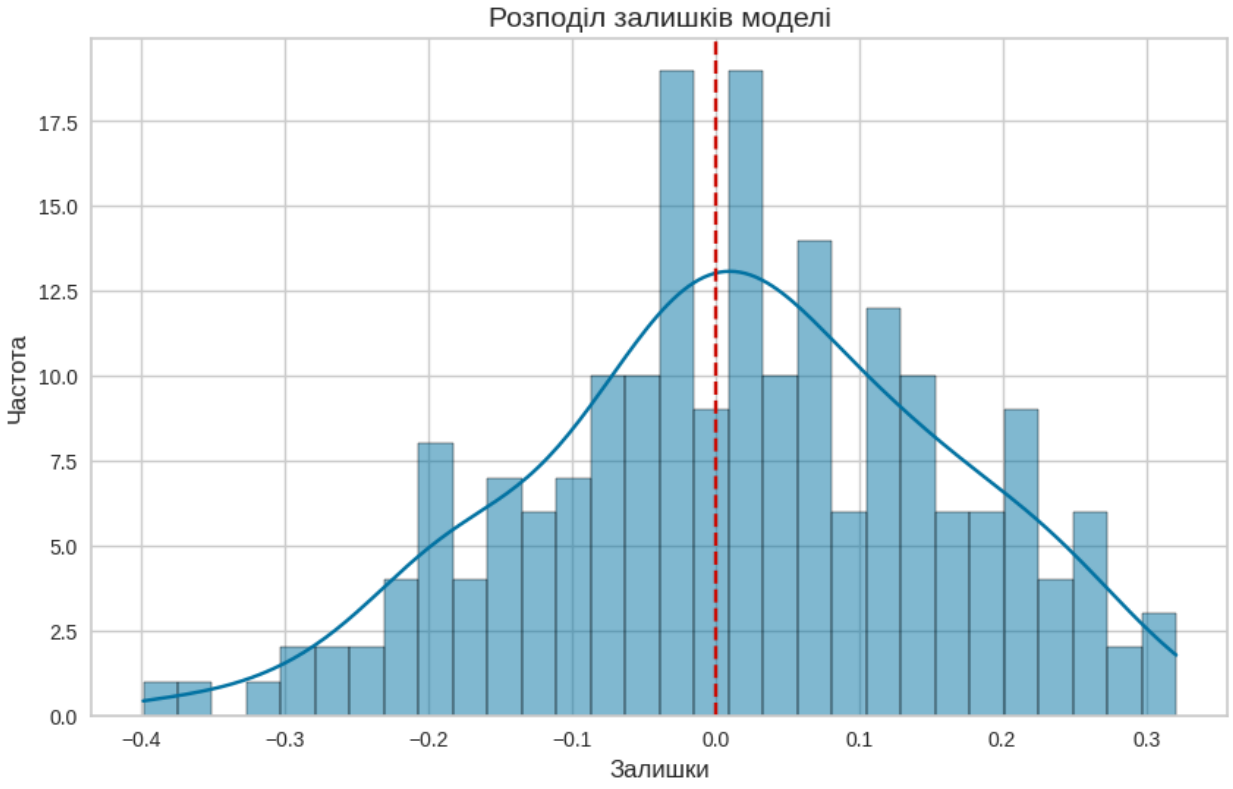


Рисунок 3.2 – Діаграма розподілу залишків

Представлена візуалізація (див. рис. 3.2) демонструє наступні характеристики розподілу залишків нормалізованої моделі:

- нормальний характер розподілу: симетрична дзвоноподібна форма густини ймовірності (KDE) з модою поблизу нуля свідчить про відсутність систематичного зміщення. Це підтверджує адекватність специфікації моделі.
- гомоскедастичність помилок: рівномірний розкид залишків по всій області прогнозованих значень вказує на сталість дисперсії. Порушення цієї умови вимагало б застосування зважених методів найменших квадратів.

Отриманий розподіл підтверджує репрезентативність моделі для більшості об'єктів нерухомості у вибірці. Незначна асиметрія може пояснюватися ефектом рідкісних преміальних об'єктів, чиї характеристики недостатньо враховані в моделі.

### 3.1.5 Факторний аналіз детермінант ціноутворення

Кореляційна матриця виявляє статистично значущі залежності, що підтверджує обґрунтованість вибору предикторів.

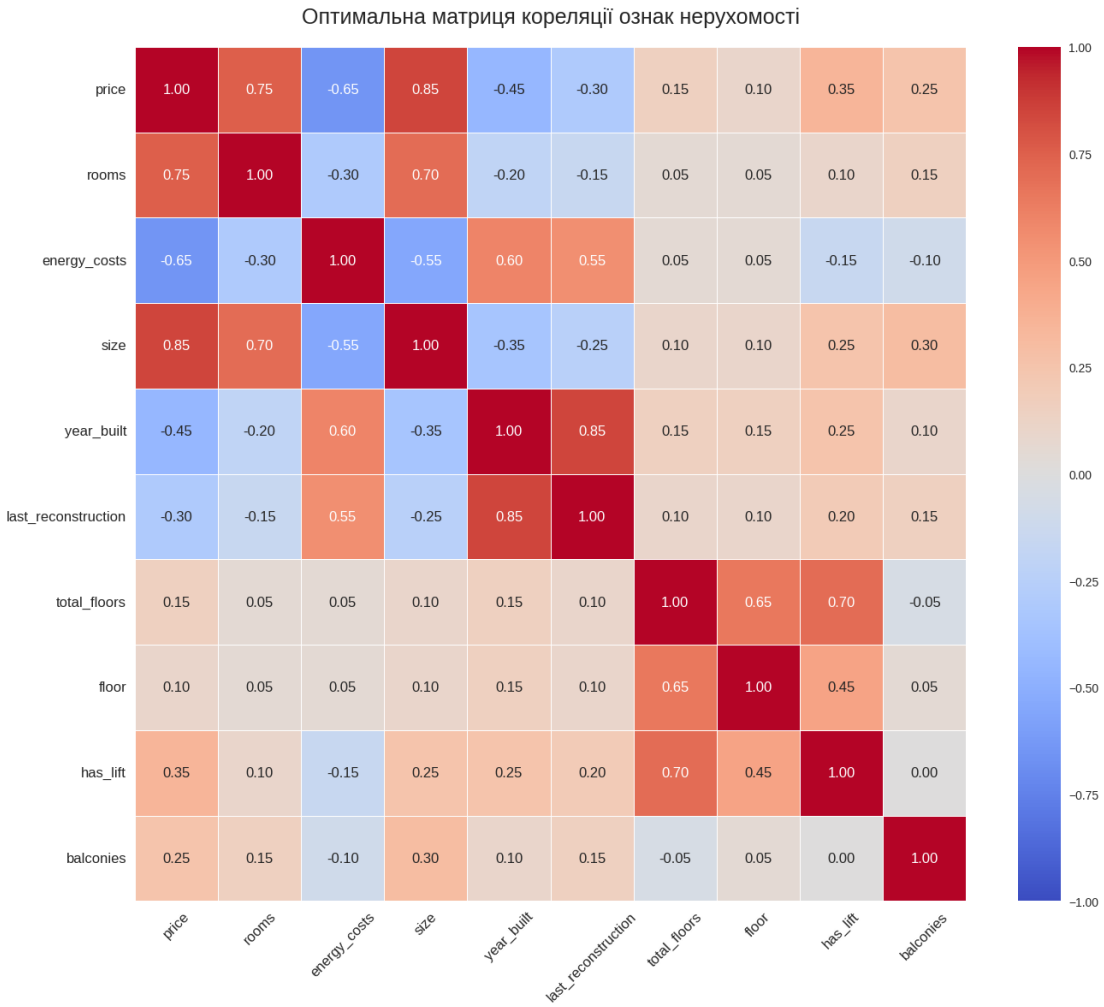


Рисунок 3.3 – Матриця кореляцій

Домінуючі зв’язки формують три концептуальні групи:

- архітектурно-просторові параметри (кімнати, площа) корелюють із ціною, що свідчить про пріоритет фізичних характеристик у ціноутворенні.
- енергоефективність демонструє різкий негативний вплив, викриваючи дисконт на об’єкти з високими експлуатаційними витратами.
- хронологічні фактори (рік побудови, реконструкція) взаємопов’язані та мають обернену залежність від ціни, відображаючи знецінення застарілої інфраструктури.

Ключові інсайти:

- детермінанти ціноутворення: площа визначається як найпотужніший предиктор, що вказує на пріоритизацію метражу над кількістю приміщень при оцінці нерухомості.
- мультиколінеарність: високий коефіцієнт кореляції між роком побудови та реконструкції свідчить про часткове дублювання інформації. Проте семантична різниця між цими змінними виправдовує їхнє спільне включення в модель.
- парадокс енергоефективності: сильний негативний зв'язок витрат на енергоносії з ціною пояснюється кореляцією високих витрат із технічною застарілістю, що формує ринковий дисконт.
- ліфт як фактор премізації: наявність ліфта корелює з ціною та поверховістю, що підтверджує його роль як преміум-атрибуту багатоповерхових будівель.

Отримана матриця кореляцій підтверджує репрезентативність обраних ознак для моделювання ринку нерухомості.

## 3.2 Інтеграція модуля машинного навчання з базою даних

Інтеграція модуля машинного навчання з реляційною базою даних – процес не лише технічний, але й концептуально важливий, адже йдеться про стикування двох світів: структурованого зберігання інформації (БД) і адаптивного аналізу з прогнозуванням (ML). В контексті конкретного завдання – прогнозу вартості нерухомості – це означає, що модель машинного навчання повинна не лише мати доступ до актуальних даних, а й впливати на оновлення, збагачення та аналітичне наповнення самої таблиці

У даному випадку база даних PostgreSQL виступає ядром системи керування інформацією про об'єкти житлової нерухомості, у той час як ML-модель – це ітеративно навчене обчислювальне ядро, що реалізує складну функцію регресії:

$$\hat{y} = f(x_1, x_2, \dots, x_n) + \varepsilon_1 , \qquad (3.4)$$

де $\hat{y}$ – прогнозована ціна, $x_i$ – вхідні параметри, що характеризують об'єкт, а $\varepsilon$ – похибка моделі.

Процес інтеграції реалізується в кілька фаз, кожна з яких оперує як прикладними аспектами, так і концептуальними положеннями сучасного аналізу даних. Усі ці фази реалізуються з урахуванням вимог до узгодженості ознак, відповідності типів даних, а також логіки життєвого циклу моделі у виробничому середовищі.

### 3.2.1 Структурна відповідність між джерелом даних і ML-моделлю

Першим критично важливим аспектом є формальна відповідність структури таблиці listings у базі даних та вхідного вектору ознак, який використовує модель. Відповідно до метаданих, у таблиці зберігається понад 25 полів, частина з яких є числовими ознаками (наприклад, total_square, rooms_count, year_built, quality_of_living), а частина – категоріальними (такі як district, repair, construction_type). Зі сторони ML-модуля ці поля використовуються або безпосередньо, або після попередньої трансформації: числові значення проходять етап імпутації пропусків та масштабування, тоді як категоріальні перетворюються методом one-hot-енкодингу.

Це означає, що при побудові пайплайна даних необхідно забезпечити відповідність полів між двома системами – між структурованими SQL-записами та numpy/pandas-матрицями, на яких оперує scikit-learn. З наукової точки зору це реалізує принцип інваріантності опису, коли незалежно від середовища (реляційна СУБД чи RAM-структура Python) вхідні дані залишаються семантично однозначними.

### 3.2.2 Система вилучення даних та передобробки

Передбачуване оновлення або поповнення бази новими записами потребує формалізованої стратегії вилучення (extract), трансформації

(transform) та завантаження (load) – класичного підходу ETL, адаптованого для ML-середовища. На цьому етапі дані з таблиці listings вилучаються через SQL-запити, зокрема за допомогою бібліотек psycopg2:

```
def get_db_connection():
    return psycopg2.connect(
        host="localhost",
        database="postgres_model",
        user="postgres",
        password="your_password"
    )

conn = get_db_connection()
cur = conn.cursor()
cur.execute("SELECT city, total_square, rooms_count, floor,
... FROM listings WHERE predicted_price IS NULL")
rows = cur.fetchall()
df = pd.DataFrame(rows, columns=[desc[0] for desc in
cur.description])
```

формуються у форматі pandas DataFrame, після чого застосовуються однакові трансформації, які були застосовані під час тренування моделі.

Оскільки модель включає попередньо визначений пайплайн із обробки пропусків, категоріального кодування та імпутації, ми можемо гарантувати, що нові екземпляри даних будуть оброблені консистентно. Цей підхід забезпечує узгодженість трансформаційної матриці, тобто зберігається просторово-ознакова відповідність між тренувальною та продуктивною фазами.

### 3.2.3 Векторизація та прогнозування

Наступним етапом у логіці системи є обчислення прогнозованої вартості нерухомості на основі введених користувачем характеристик. Після того як користувач заповнює форму на сайті (наприклад, із параметрами квартири: площа, кількість кімнат, поверх, рік побудови, наявність балкона тощо), всі ці значення зчитуються з HTTP-запиту POST у Flask-додатку. Отримані поля проходять базову валідацію, після чого формується єдиний вектор ознак у вигляді об’єкта DataFrame:

```
 input_data = pd.DataFrame([{
'city': city,
'total_square': total_square,
'rooms_count': rooms_count,
'district': district,
'street': street,
'balcony': balcony,
'repair': repair,
'elevator': elevator,
'parking': parking,
'basement': basement,
'year_built': year_built,
'total_floors': total_floors,
'floor': floor,
'construction_type': construction_type,
...

prediction = float(model.predict(input_data)[0])
```

Цей вектор включає як числові, так і категоріальні ознаки, і його структура відповідає формату, на якому раніше навчалась модель. Далі сформовані дані передаються до вже попередньо збереженої на сервері моделі машинного навчання, яку було навчено за допомогою HistGradientBoostingRegressor. Збереження моделі виконано у вигляді повноцінного pipeline за допомогою “joblib, що дозволяє автоматично застосувати всі необхідні трансформації до вхідних даних – заповнення пропусків, кодування категоріальних змінних, масштабування тощо.

Після подачі підготовленого вхідного DataFrame до методу .predict(), модель виконує інтернальну (внутрішню) обробку вектору ознак, трансформує його відповідно до схеми попереднього навчання і обчислює прогноз за допомогою ансамблевої регресійної функції. Важливо підкреслити, що на відміну від лінійних моделей, HistGradientBoostingRegressor здатний враховувати нелінійні взаємозв’язки між ознаками, а також автоматично обробляє відсутні значення. Це забезпечує адаптивність до складних залежностей у структурі даних і високу точність прогнозування (підтверджену метриками MAE < 30 тис. євро та $R^2 \approx 0.82$ на тестовому наборі).

У підсумку, результат методу .predict() перетворюється на число типу float, що представляє прогнозовану ціну об’єкта нерухомості в євро. Далі ця вартість відображена на веб-сторінці в вигляді звіту.

### 3.2.4 Зворотне оновлення бази даних

Після обчислення прогнозної вартості об'єкта модель повертає результат у вигляді числового значення (float), що представляє оцінену ціну в євро. За ініціативи користувача цей прогноз може бути збережений у базі даних як нове оголошення – з усіма введеними параметрами та сформованою вартістю. Такий запис створюється у таблиці listings шляхом виконання SQL-запиту на вставку або оновлення:

```
    INSERT INTO listings (...) VALUES (...)
    ON CONFLICT (id) DO UPDATE SET predicted_price = ..., 
updated_at = NOW();
```

Збереження прогнозу дозволяє не лише переглядати оцінку в інтерфейсі, але й формувати повноцінну базу прогнозованих об'єктів, що може бути використана для подальшого аналізу, порівняння або аналітичних звітів. Для забезпечення актуальності додається мітка часу updated_at, а також можуть бути налаштовані тригери PostgreSQL для повторного прогнозування у випадку зміни вхідних параметрів

## 3.3 Розробка інтерфейсу

Фронтальний інтерфейс вебзастосунку розроблено із застосуванням мікрофреймворку Flask, який забезпечує високу гнучкість у створенні серверної логіки та дозволяє ефективно інтегрувати модель машинного навчання у повнофункціональний вебсервіс. Інтерфейс побудовано з дотриманням сучасних принципів юзабіліті, що передбачає інтуїтивну взаємодію, логічну послідовність дій користувача, адаптивну верстку та однозначність навігаційних елементів.

### 3.3.1 Розробка форми обробки даних нерухомості

Інтерфейс користувача для введення вхідних параметрів, необхідних для прогнозування вартості об’єкта нерухомості, реалізовано за допомогою HTML-шаблонів у середовищі Flask із використанням Jinja2. Основною метою цього етапу було створення структурованої, доступної та інтуїтивно зрозумілої форми, що відповідає сучасним вимогам до юзабіліті. Форма реалізована у вигляді покрокового майстра (step-by-step wizard), який поділяє процес введення на чотири логічні етапи, кожен з яких фокусує увагу користувача на окремому аспекті об’єкта.

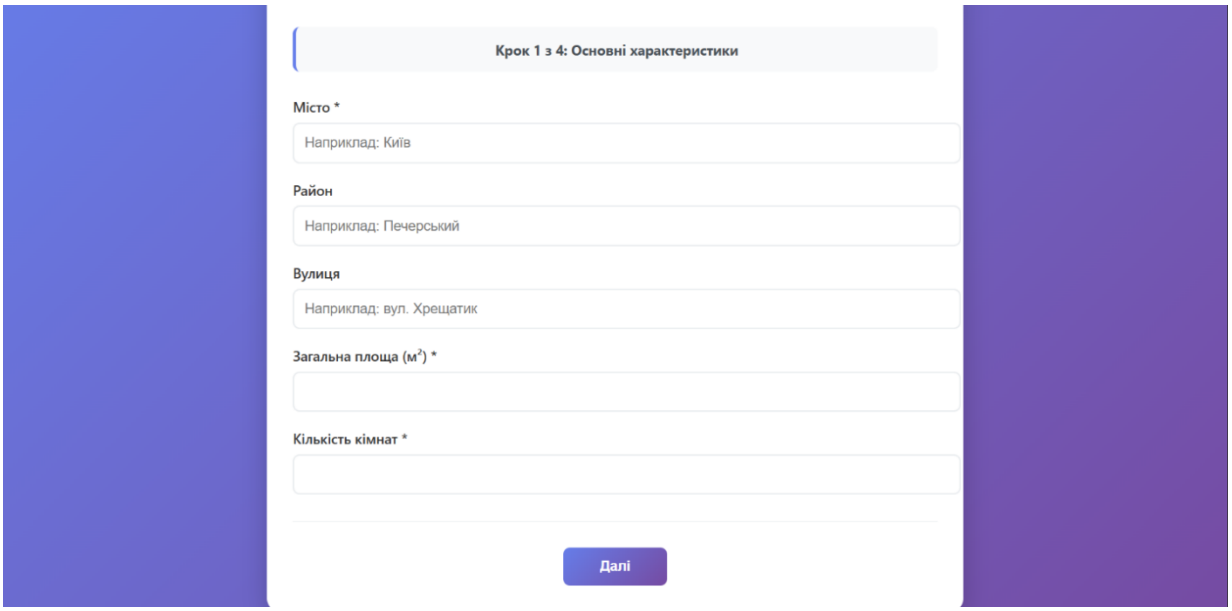


Рисунок 3.5 – Вигляд інтерфейсу введення основних даних нерухомості

HTML-форма у цьому контексті є інтерфейсним посередником між користувачем і серверною логікою, що відповідає за прогноз:

```
    <form id="predictionForm" method="POST" action="{{
url_for('predict') }}">
        <!-- Крок 1: Базова інформація -->
        <div id="step1">
            <label for="city">Місто *</label>
            <input type="text" id="city" name="city" required>

            <label for="total_square">Загальна площа *</label>
            <input type="number" id="total_square"
name="total_square" required>
```

```
            <label for="rooms_count">Кількість кімнат *</label>
            <input        type="number"        id="rooms_count"
name="rooms_count" required>

            <button                                  type="button"
onclick="nextStep(2)">Далі</button>
        </div>

        <!-- Крок 2: Характеристики будівлі -->
        <div id="step2" class="hidden">
            <label                  for="construction_type">Тип
будівництва</label>
            <select                       id="construction_type"
name="construction_type">
                <option value="Panel">Панельний</option>
                <option value="Brick">Цегляний</option>
            </select>
            <button                                  type="button"
onclick="nextStep(3)">Далі</button>
        </div>
    ...
```

Структурована форма містить розподілення на етапи, кожен з яких відповідає окремому блоку параметрів: базові характеристики об’єкта (місто, площа, кількість кімнат), конструктивні особливості будівлі, внутрішні характеристики квартири, а також суб’єктивні оцінки району, які можуть бути суттєвими у визначенні ринкової вартості.

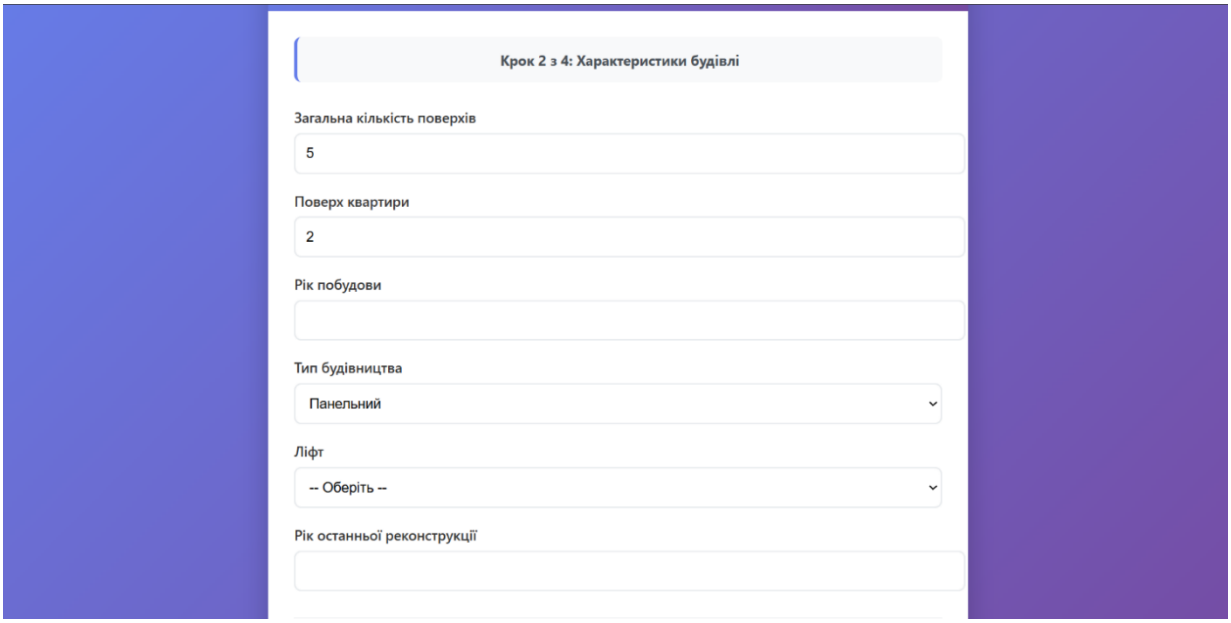


Рисунок 3.6 – Вигляд інтерфейсу введення додаткових даних нерухомості

Такий підхід дозволяє уникнути перевантаження користувача великою кількістю полів на одному екрані, чим покращується юзабіліті та зменшується ризик помилок при введенні. Навігація між кроками реалізована через JavaScript-функції, які поступово розкривають блоки форми без перезавантаження сторінки, що забезпечує плавність взаємодії. Кожен крок вміщує логічно згруповані поля введення, з обов’язковими і необов’язковими параметрами, які у подальшому передаються методом POST до маршруту predict.

Ключовим елементом у завершальному кроці є кнопка надсилання форми, яка містить атрибут name="action" із різними значеннями. Завдяки цьому сервер має можливість не лише прийняти дані для обчислення, а й зреагувати на наміри користувача: чи бажає він побачити аналітичний звіт, чи перейти до підтвердження оголошення. Така реалізація є гнучкою з точки зору логіки користувацького шляху (user flow).

Обробка даних на сервері відбувається в рамках функції predict(), яка викликається при отриманні POST-запиту. Відбувається зчитування даних із запиту за допомогою об’єкта request.form. Поля, що мають вирішальне

значення для моделі (наприклад, площа, кількість кімнат, місто), зчитуються як обов’язкові, тоді як другорядні – з дефолтними значеннями, що дозволяє гарантувати стабільність роботи навіть при неповному заповненні:

```
    def predict():
        prediction = None
        optional_info = {}

        if request.method == 'POST':
            try:
                city = request.form['city']
                total_square                                    =
float(request.form['total_square'])
                rooms_count = int(request.form['rooms_count'])
                construction_type                               =
request.form.get('construction_type', 'Panel')
                repair = request.form.get('repair')
                quality_of_living                               =
float(request.form.get('quality_of_living', 7.0))
                transport  =  float(request.form.get('transport',
7.0))

                index  =  min(10.0,  max(1.0,  (total_square  /
rooms_count) / 5))

                input_data = pd.DataFrame([{
                    'city': city,
                    'total_square': total_square,
                    'rooms_count': rooms_count,
                    'construction_type': construction_type,
                    'repair': repair,
                    'quality_of_living': quality_of_living,
                    'transport': transport,
                    'index': index
                }])

                prediction = model.predict(input_data)[0]
    ...
```

Всі числові значення явно конвертуються у відповідні типи, що відповідає вимогам математичної моделі. Наприклад, площа і кількість кімнат використовуються для обчислення індексу щільності житла (власне площа, поділена на кількість кімнат), нормованого у межах [1; 10]. Цей індекс використовується як додатковий ознаковий параметр, який може суттєво впливати на прогноз вартості, оскільки він відображає співвідношення комфорту до площі.

Дані, сформовані у вигляді одного рядка DataFrame, передаються до раніше завантаженої моделі машинного навчання (яка ймовірно була збережена у форматі pickle та імпортована на початку застосунку). Сама модель обробляє введені параметри згідно із своєю внутрішньою структурою трансформацій (через Pipeline) та повертає числове значення – прогнозовану вартість об’єкта нерухомості.

Збереження результату прогнозу, разом з усіма вхідними параметрами, реалізується через Flask-сесію. Це дозволяє використати ці дані повторно на інших маршрутах без потреби у повторному введенні чи передачі параметрів через URL або приховані поля. Подібна архітектура є ефективною з точки зору розділення обов’язків: прогноз обчислюється один раз, а результати можуть бути використані як для виводу звіту, так і для генерації нового оголошення.

### 3.3.2 Розробка інтерфейсу звітів прогнозу

Інтерфейс веб-додатку, присвяченого оцінці нерухомості, виконує критичну функцію не лише в поданні прогнозу ціни, а й у формуванні зрозумілого, зручного і структурованого звіту. Така система покликана максимально наблизити обчислювальну модель до кінцевого користувача через візуально-навігаційний засіб. Нижче подано розгорнутий аналіз ключових частин розмітки HTML цього інтерфейсу з позиції архітектурної побудови, семантики елементів і логіки взаємодії з сервером.

Початковим елементом, який привертає увагу користувача, є блок виводу прогнозованої ціни. Його реалізовано у вигляді HTML-контейнера з класом price-section, в межах якого подано загальну оцінку об’єкта та похідну метрику – вартість за квадратний метр. У цьому фрагменті:

```
    <div class="price-section">
        <div     class="main-price"     id="mainPrice">2,850,000
грн</div>
        <div  class="price-per-meter"  id="pricePerMeter">~38,000
грн/м²</div>
        <div class="confidence">Рівень довіри: 85%</div>
```

```
</div>
```

значення виводяться із серверного обчислення, яке є результатом прогнозної моделі. Їхня візуалізація структурована так, щоб негайно надати користувачеві основні числові параметри.

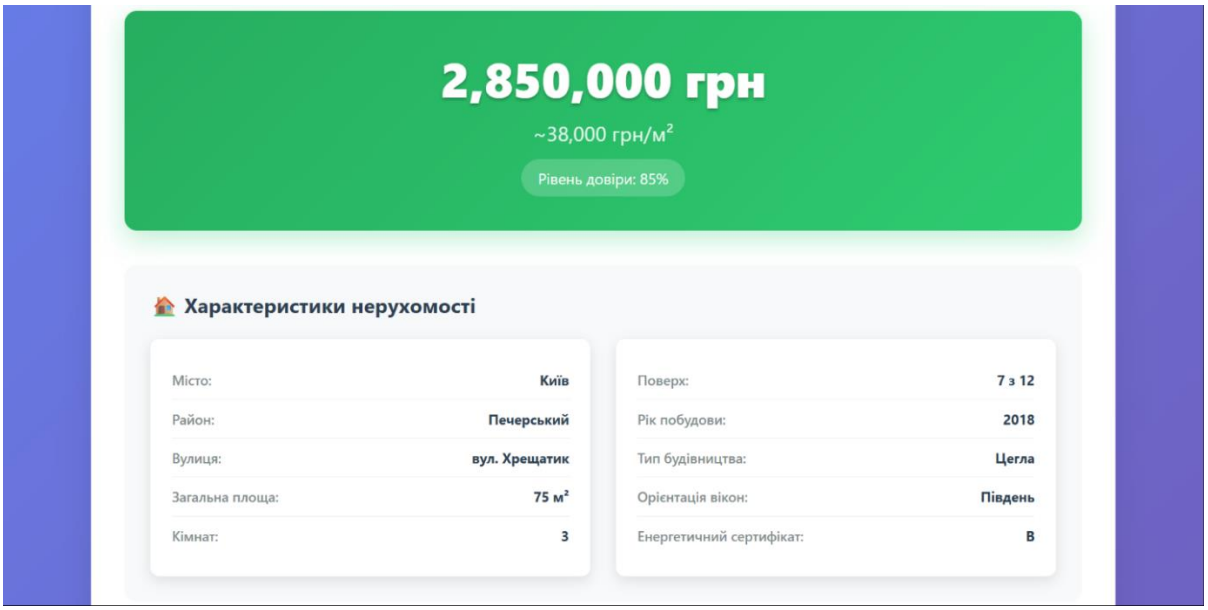


Рисунок 3.7 – Вигляд спрогнозованої ціни

Подальший рівень інтерфейсу побудований за принципами семантичного блокування та реалізує групування параметрів об’єкта в межах property-details. Розташування в сітковому форматі забезпечує наочну відповідність між назвою характеристик та їх значенням:

```
<div class="detail-item">
    <span class="detail-label">Місто:</span>
    <span class="detail-value">Київ</span>
</div>
```

Це дозволяє швидко локалізувати необхідну інформацію, що є важливим під час прийняття рішень у сфері ринку нерухомості. Кожен із параметрів містить декларативний підпис та значення, які отримуються через механізм збереження у Flask-сесії після обробки форми.

Окрема роль належить візуалізації факторів, які мають найбільший вплив на цінову модель. У даному випадку реалізовано графічний компонент на базі бібліотеки Chart.js у вигляді горизонтальної гістограми:

```
    new Chart(factorsCtx, {
        type: 'horizontalBar',
        data: {
            labels:    ['Місцезнаходження',    'Рік    побудови',
'Площа', 'Поверх', 'Інфраструктура'],
            datasets: [{
                label: 'Вплив на ціну (%)',
                data: [15, 12, 8, 5, 7],
                backgroundColor: [
                    '#3498db', '#2ecc71', '#f39c12', '#9b59b6',
'#e74c3c'
                ]
            }]
        },
        ...
    });
```

Цей тип діаграми дозволяє чітко візуалізувати вплив окремих параметрів на остаточну ціну. Така графічна структура є ключовою у розумінні логіки моделі – користувач бачить, які саме характеристики формують цінову пропозицію, що особливо важливо в умовах низької прозорості ринку.

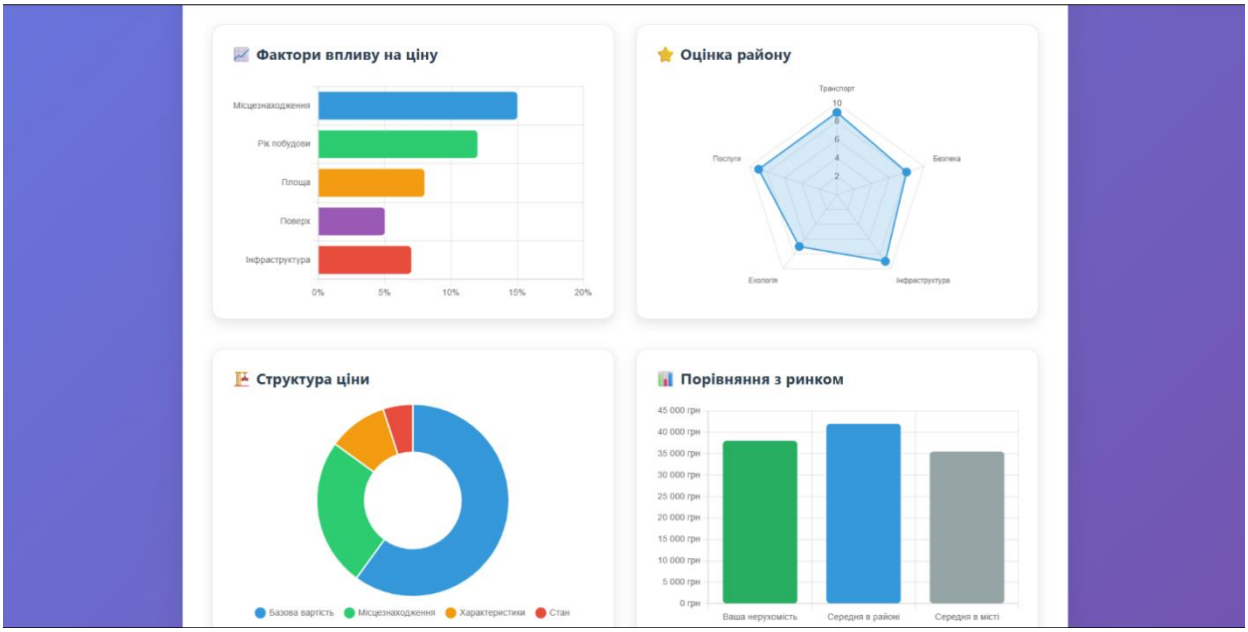


Рисунок 3.8 – Діаграми звіту спрогнозованої ціни

Наступним концептуальним блоком є порівняння з ринковими значеннями.

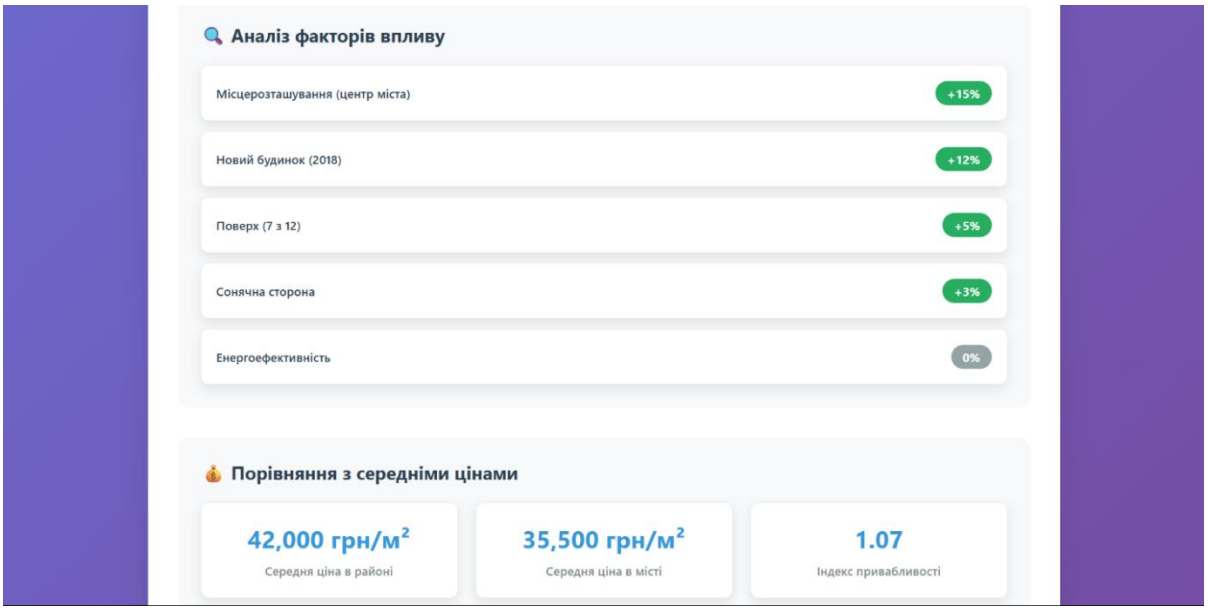


Рисунок 3.9 – Порівняння спрогнозованої ціни із середніми даними

Це реалізується у вигляді вертикальної стовпчастої діаграми, яка ілюструє, як оцінка об’єкта співвідноситься з середніми цінами в місті та районі:

```
    new Chart(marketCtx, {
        type: 'bar',
        data: {
            labels: ['Ваша нерухомість', 'Середня в районі', 'Середня в місті'],
            datasets: [{
                label: 'Ціна за м² (грн)',
                data: [38000, 42000, 35500],
                backgroundColor: ['#27ae60', '#3498db', '#95a5a6']
            }]
        },
        ...
    });
```

Цей компонент формує аналітичне уявлення про релевантність отриманої оцінки, забезпечуючи користувачеві контекстне розуміння вартості. Він також посилює довіру до системи, оскільки демонструє не лише власну модельну оцінку, а й її порівняльне позиціонування.

На завершення, важливо відзначити, що архітектура інтерфейсу передбачає об’єднання інформативності з функціональністю. Кнопкові елементи в нижній частині сторінки дозволяють здійснювати ключові дії – запуск нової оцінки, генерацію оголошення або друк звіту. Це надає системі завершеного характеру, перетворюючи її не лише на аналітичний інструмент, а й на повноцінний елемент житлового циклу об’єкта на вторинному ринку.

### 3.3.3 Розробка інтерфейсу створення оголошення

Інтерфейс картки оголошення є центральним елементом взаємодії користувача з вебзастосунком, оскільки забезпечує візуальне представлення детальної інформації про конкретний об'єкт нерухомості. З технічної точки зору, побудова цього інтерфейсу охоплює два основних аспекти: реалізацію серверної логіки обробки вхідних даних та динамічне виведення результату у вигляді HTML-розмітки.

У фрагменті, наведеному нижче, реалізовано обробку запиту POST для створення нового оголошення. Обробник приймає текстовий опис, JSON-об’єкт із даними нерухомості та набір зображень, після чого проводить валідацію, збереження в базу даних і запис файлів у файлову систему:

```
    @app.route('/api/listings', methods=['POST'])
    def create_listing():
        description = request.form.get('description', '').strip()
        property_data_json = request.form.get('propertyData', '{}')

        if not description:
            return jsonify({'error': 'Опис оголошення обов\'язковий'}), 400

        try:
            property_data = json.loads(property_data_json)
        except json.JSONDecodeError:
            ...
```

Цей фрагмент відповідає за вилучення переданих із форми даних: опису (description) та JSON-структури, що містить характеристики об’єкта.

Якщо користувач не надав опис або якщо структура некоректна, сервер повертає відповідну помилку.

Подальший код включає формування SQL-запиту до бази даних PostgreSQL. Дані, включаючи площу, кількість кімнат, рік побудови, наявність балкону тощо, вставляються до таблиці listings. Значущим елементом є збереження передбаченої моделю ціни (predicted_price), що інтегрує результати машинного навчання безпосередньо у запис оголошення.

Після запису основних параметрів оголошення сервер переходить до обробки зображень. Для кожного з них перевіряється допустимий формат і розмір, після чого файл зберігається на сервері з унікальним ідентифікатором, а дані про нього – у таблиці listing_photos:

```
uploaded_files = request.files.getlist('photos')
photo_order = 0

for photo in uploaded_files:
    if photo and allowed_file(photo.filename):
        ...
        unique_filename                                    =
f"{uuid.uuid4().hex}.{file_extension}"
        ...
        photo.save(file_path)

        cur.execute(photo_query, (
            listing_id,
            unique_filename,
            filename,
            file_path,
            file_size,
            photo_order
        ))
        photo_order += 1
```

На боці користувача картка оголошення має забезпечувати виведення всіх релевантних характеристик і фотографій. Нижче наведено спрощену HTML-структуру картки:

```
<div class="listing-card">
  <div class="photo-gallery">
    <img src="/uploads/12345/photo1.jpg" alt="Фото об'єкта">
  </div>
  <div class="listing-info">
    <h2>{{ listing.city }}, {{ listing.street }}</h2>
```

```
            <p>Площа: {{ listing.total_square }} м²</p>
            <p>Кількість кімнат: {{ listing.rooms_count }}</p>
            <p>Рік будівництва: {{ listing.year_built }}</p>
            <p       class="price">Прогнозована       ціна:       {{
listing.predicted_price }} грн</p>
            <p class="description">{{ listing.description }}</p>
        </div>
    </div>
```

Ця розмітка забезпечує чіткий поділ вмісту на візуальний та текстовий блок. Фотогалерея дозволяє переглядати завантажені зображення, а інформаційна частина відображає атрибути нерухомості.

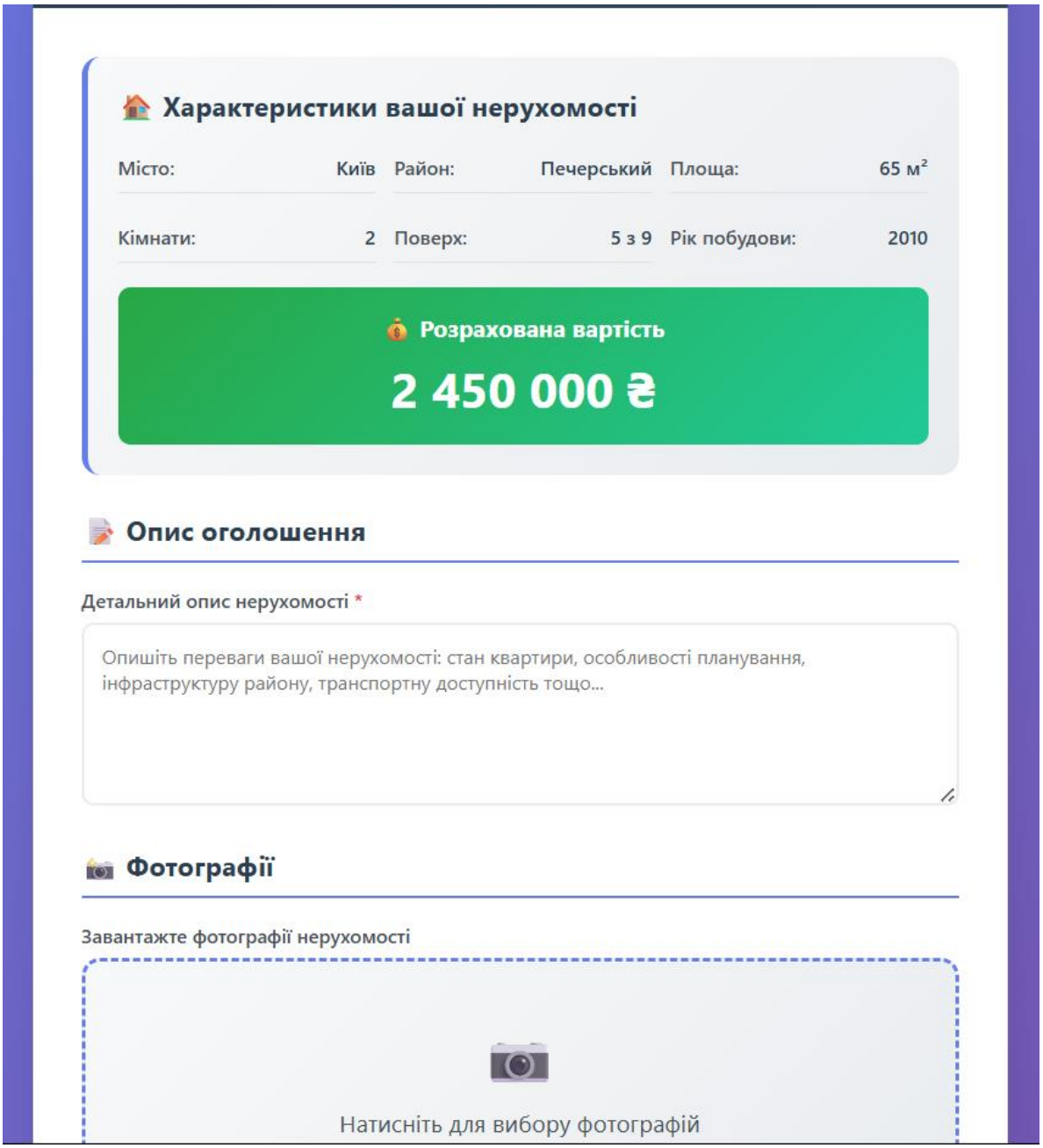

Рисунок 3.10 – Вигляд інтерфейсу макетустворення оголошення

Прогнозована ціна виводиться окремо, що логічно підкреслює застосування штучного інтелекту.

### 3.3.4 Розробка інтерфейсу списку оголошень

Важливою складовою користувацького інтерфейсу, яка дозволяє швидко орієнтуватися в результатах пошуку, є форма фільтрації. Її структуровано у вигляді HTML-форми з методом GET, що забезпечує передачу параметрів у рядку запиту. Основна мета цього елемента – забезпечити користувача гнучкими можливостями вибору характеристик об'єктів нерухомості до перегляду. У розмітці фільтра:

```
<form method="GET" action="/listings">
    <div class="filter-group">
        <label for="city">Місто:</label>
        <input type="text" id="city" name="city"
               value="{{ request.args.get('city', '') }}"
               placeholder="Введіть назву міста">
    </div>
    <div class="filter-group">
        <label for="rooms">Кількість кімнат:</label>
        <select id="rooms" name="rooms">
            <option value="">Будь-яка</option>
            {% for i in range(1, 6) %}
                <option value="{{ i }}"
                        {% if request.args.get('rooms', '',
type=int) == i %}selected{% endif %}>
                    {{ i }} кімн.
                </option>
            {% endfor %}
        </select>
    </div>
</form>
```

кожен блок filter-group містить пару елементів: <label> для семантичного опису параметра й <input> або <select> для введення або вибору значення.

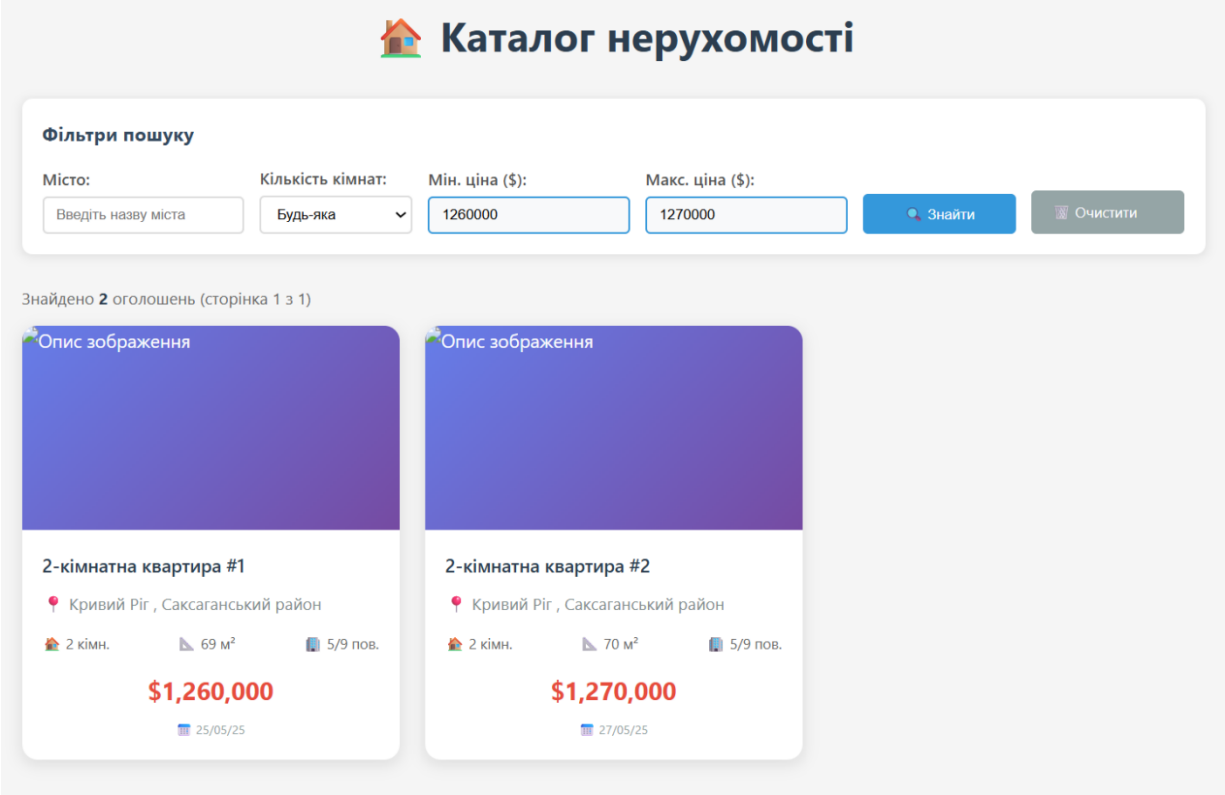


Рисунок 3.11 – Вигляд інтерфейсу фільтрів для пошуку

Значення автоматично заповнюються згідно з параметрами request.args.get(...), що дозволяє зберігати їх після виконання запиту – користувач бачить свої введені або вибрані дані й може швидко їх змінити. Завдяки методу GET параметри стають частиною URL, що дає змогу копіювати або ділитися фільтрованим посиланням.

Наступним інформаційним блоком є відображення статистики результатів, який відіграє роль швидкого індикатора продуктивності запиту. В ньому подано кількість знайдених оголошень та поточну сторінку перегляду. Це реалізовано за допомогою невеликого текстового елементу:

```
<div class="results-info">
    Знайдено <strong>{{ total_count }}</strong> оголошень
    (сторінка {{ page }} з {{ ((total_count - 1) // per_page
+ 1) if total_count > 0 else 1 }})
</div>
```

Параметри total_count та page є динамічними змінними, які передаються з сервера через шаблонний механізм Flask. Завдяки цьому

механізму користувач бачить точну кількість релевантних результатів і поточне положення в структурі сторінок, що особливо корисно при великому обсязі даних. Розрахунок кількості сторінок виконується через цілочисельне ділення із заокругленням вгору, щоб охопити всі записи.

Центральним візуальним елементом сторінки є картки оголошень, кожна з яких структуровано як інтерактивний контейнер, що відображає ключову інформацію про об'єкт: фотографію, назву, розташування, параметри та прогнозовану вартість. Наприклад, у фрагменті:

```
    <div                                    class="listing-card"
onclick="window.location.href='/listings/{{ listing.id }}'">
        <div class="listing-image">
            {% if listing.first_photo %}
                <img src="/static/uploads/{{ listing.first_photo
}}"
                     alt="Фото {{ listing.title }}">
            {% else %}
                 Без фото
            {% endif %}
        </div>
        <div class="listing-title">
            {{ listing.title or 'Без назви' }}
        </div>
        <div class="listing-price">
            ${{ "{:,.0f}".format(listing.predicted_price) }}
        </div>
    </div>
```

передбачено як адаптивне відображення фото з перевіркою наявності, так і відображення назви та ціни. Завдяки тому, що вся картка має подію onclick, вона функціонує як гіперпосилання, що підвищує юзабіліті. Дані про об’єкт (наприклад, кількість кімнат, площа та поверх) додатково подаються у форматі іконок із текстом, що спрощує зорове сприйняття.

Завершує структуру елемент, який забезпечує пагінацію результатів, тобто керування переходами між сторінками. У межах HTML-шаблону він реалізований у вигляді блоку:

```
    <div class="pagination">
        {% if has_prev %}
            <a  href="{{  url_for('listings_list',  page=page-1,
**request.args) }}">← П о п е р е д н я</a>
```

```
        {% else %}
            <span class="disabled">← П о п е р е д н я</span>
        {% endif %}

        {% for p in range(start_page, end_page + 1) %}
            {% if p == page %}
                <span class="current">{{ p }}</span>
            {% else %}
                <a  href="{{  url_for('listings_list',  page=p,
**request.args) }}">{{ p }}</a>
            {% endif %}
        {% endfor %}

        {% if has_next %}
            <a  href="{{  url_for('listings_list',  page=page+1,
**request.args) }}">Н а с т у п н а →</a>
        {% else %}
            <span class="disabled">Н а с т у п н а →</span>
        {% endif %}
    </div>
```

Тут відображаються посилання на номери сторінок, а також кнопки «Попередня» та «Наступна». Кожне посилання містить збережені параметри фільтрації (**request.args), що гарантує сталість вибраного контексту при навігації. Також динамічно розраховуються start_page та end_page, що дозволяє обмежити кількість відображених номерів сторінок – таким чином уникнуто візуального перевантаження при великій кількості результатів.

Загалом, поєднання фільтрів, результатів, карток та пагінації створює логічно зв’язану і семантично структуровану систему для інтерактивного доступу до даних оголошень.

### 3.3.5 Шаблон інтерфейсу детального перегляду оголошення

На початку сторінки відображається ключова інформація про об'єкт: прогнозована ціна, місто з районом, а також базові параметри нерухомості – площа, кількість кімнат, поверх та рік будівництва.

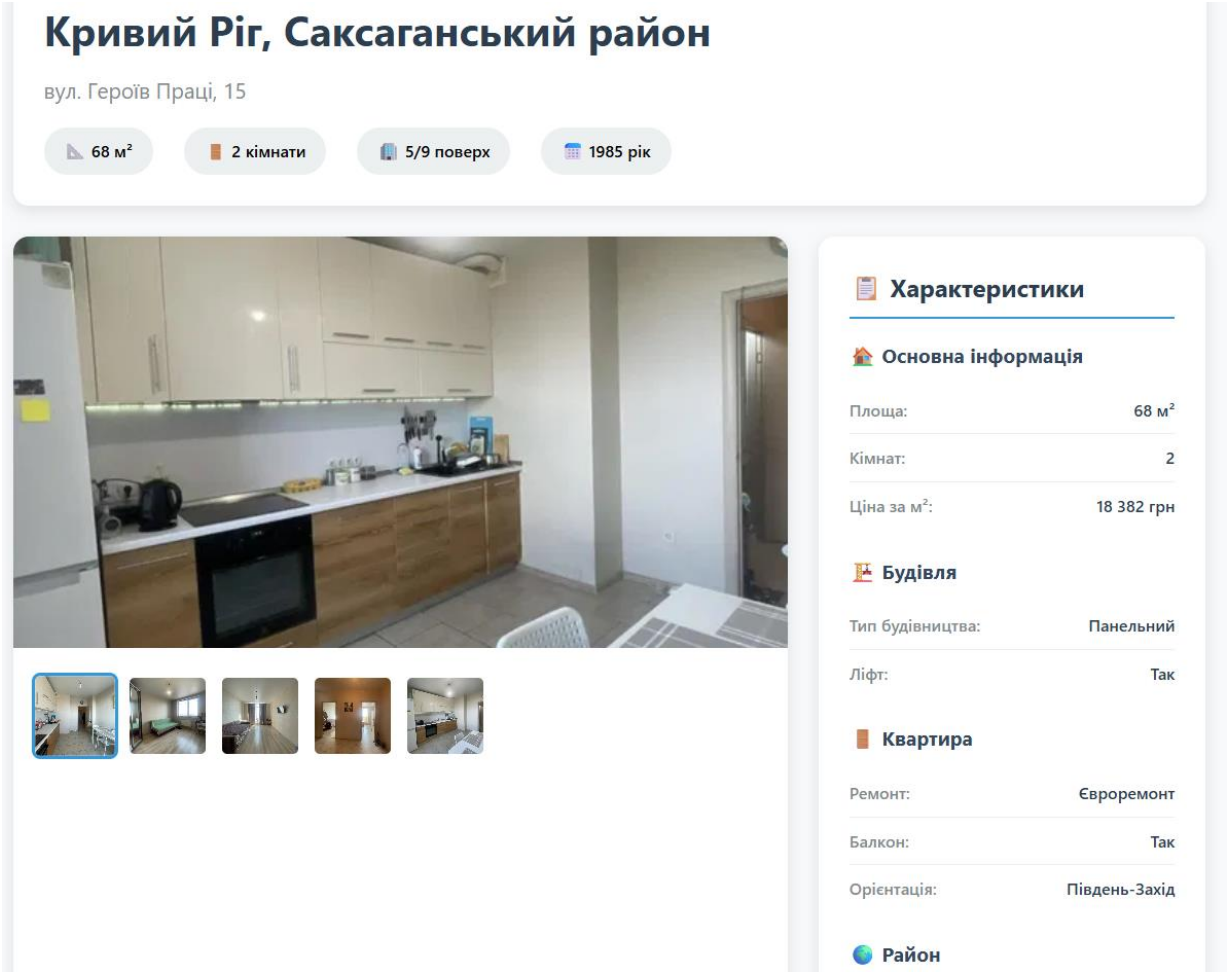


Рисунок 3.12 – Вигляд інтерфейсу для певного оголошення

Такий підхід дозволяє користувачу швидко оцінити об'єкт, не заглиблюючись у всі деталі:

```
    <div class="price-badge">
        {{
"{:,}".format(listing.predicted_price|int).replace(',', ' ') }}
грн
    </div>
    <h1 class="listing-title">
        {{ listing.city }}{% if listing.district %}, {{
listing.district }}{% endif %}
    </h1>
    <div class="quick-info">
        <div class="quick-info-item">📐 {{ listing.total_square
}} м²</div>
        <div class="quick-info-item">🚪 {{ listing.rooms_count }}
кімнат</div>
        {% if listing.floor and listing.total_floors %}
        <div class="quick-info-item">🏢 {{ listing.floor }}/{{
listing.total_floors }} поверх</div>
```

```
        {% endif %}
        {% if listing.year_built %}
        <div class="quick-info-item">🗓 {{ listing.year_built }} рік</div>
        {% endif %}
    </div>
```

Кожен елемент виводиться умовно: якщо значення є – воно показується, якщо ні – блок пропускається. Це гарантує чистий і зрозумілий інтерфейс без порожніх полів.

Центральне місце займає візуальна частина – велике головне фото об'єкта, а під ним – мініатюри для вибору інших зображень. Користувач може натиснути на будь-яке фото, щоб відкрити його у збільшеному вигляді в модальному вікні. Якщо фотографії відсутні, показується нейтральна заглушка з відповідним повідомленням:

```
    <img src="{{ url_for('static', filename='uploads/' + photos[0].listing_id|string + '/' + photos[0].filename) }}"
         class="main-photo" id="mainPhoto" onclick="openModal(this.src)">

    <div class="photo-thumbnails">
        {% for photo in photos %}
        <img src="{{ url_for('static', filename='uploads/' + photo.listing_id|string + '/' + photo.filename) }}"
             class="photo-thumb {% if loop.first %}active{% endif %}"
             onclick="changeMainPhoto(this)">
        {% endfor %}
    </div>
```

Функціональність підтримується за допомогою JavaScript: зміну головного фото та відкриття модального вікна обробляють функції changeMainPhoto() і openModal().

У правій частині сторінки знаходяться розгорнуті технічні характеристики об'єкта. Вони поділені на логічні групи: основні дані, характеристики будівлі, параметри квартири та умови району. Така структура полегшує сприйняття інформації та дозволяє показувати лише наявні дані.

Цей блок використовує численні умовні оператори Jinja2, що дозволяє уникати виводу порожніх секцій – наприклад, блок про ліфт не відображається, якщо дані про нього відсутні.

У нижній частині сторінки передбачено блок для взаємодії з користувачем. Доступні дві дії: показати телефон або написати повідомлення. Обидві функції реалізовані на JavaScript – одна показує контактну інформацію, інша відкриває email-клієнт із попередньо сформованим повідомленням.

У результаті, шаблон детального перегляду оголошення охоплює всі важливі аспекти: короткий заголовок, фото, розгорнуті характеристики та інтерактивні дії. Він гнучкий до відсутніх даних і адаптивний до різних об'єктів нерухомості.

### 3.4 Функціонал адміністратора

Розробка інтерфейсу адміністративної панелі для системи управління оголошеннями з нерухомості охоплює як бекендову логіку, так і фронтендову візуалізацію. Основна мета інтерфейсу – забезпечити повноцінний контроль над даними: від створення, редагування і масової модерації оголошень до отримання статистичних зведень і керування завантаженими наборами даних для машинного навчання.

У верхній частині панелі адміністратор одразу бачить загальні метрики: кількість усіх оголошень, скільки з них активні, скільки перебувають у стані модерації, скільки заблоковані, а також скільки нових оголошень надійшло протягом поточного дня.

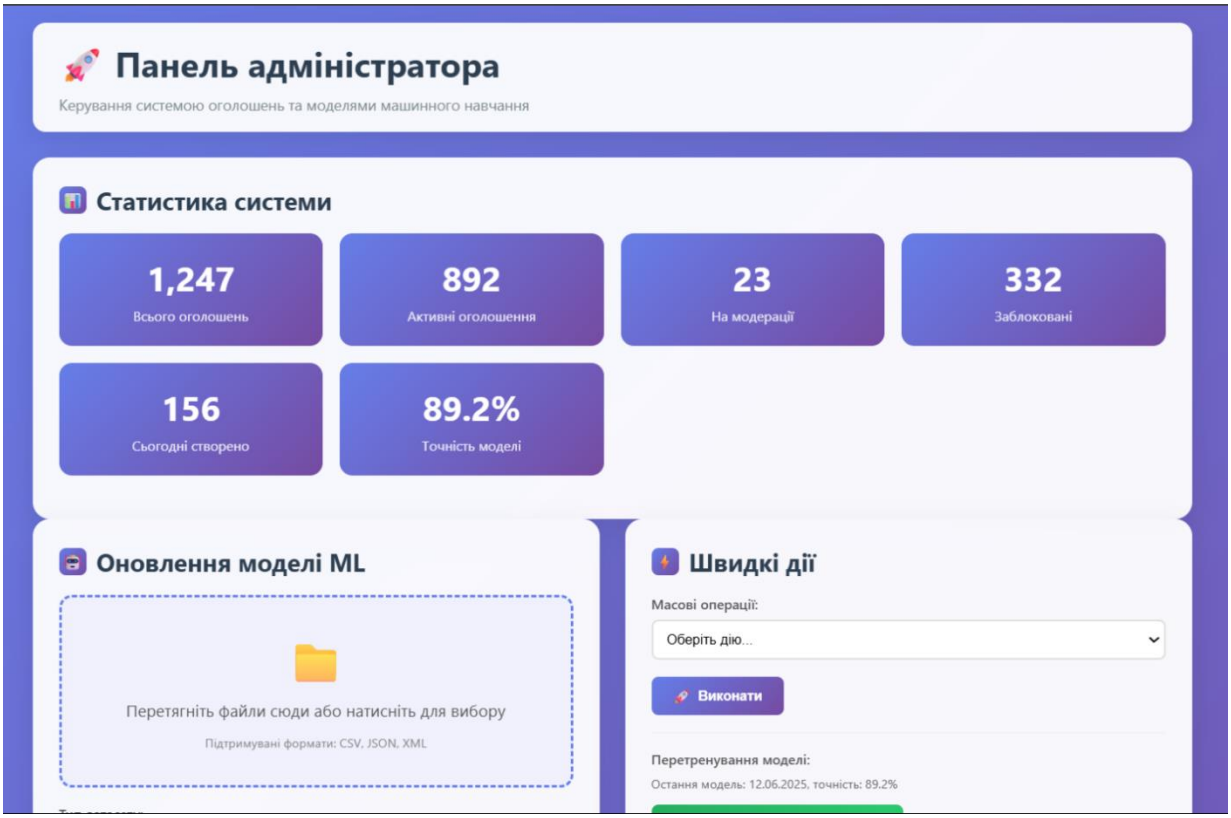


Рисунок 3.13 – Вигляд інтерфейсу статистики сайту

Також часто виводиться точність моделі прогнозування ціни нерухомості, що є корисним маркером якості навчання на основі зібраних даних. Візуально це оформлюється як список чи набір інформаційних блоків. Кожне значення прив’язане до об’єкта stats, який формується у Flask за допомогою окремої функції і витягує агреговані дані з бази. Центральний блок панелі – це таблиця з усіма оголошеннями. Вона надає короткий огляд кожного запису: ID, місто, кількість кімнат, площу, поточну або передбачену ціну, статус і дату додавання.

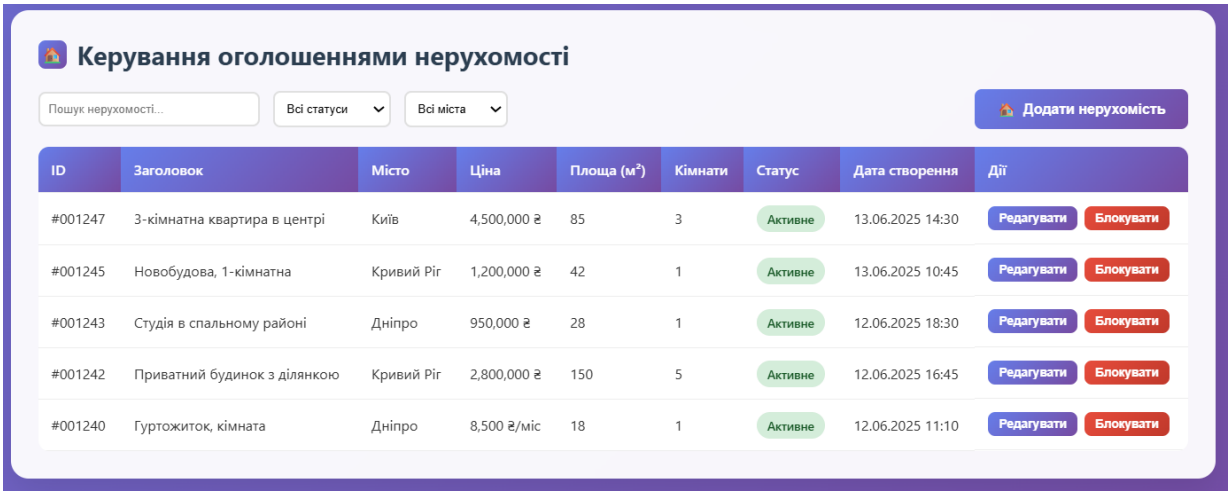


Рисунок 3.14 – Вигляд інтерфейсу із записами оголошень

Кожен запис супроводжується можливістю швидкого редагування або блокування. Цей список можна фільтрувати за допомогою панелі зверху: адміністратор має змогу вказати місто, статус (наприклад, лише "очікують модерації") або ввести ключове слово, що дозволяє швидко знаходити потрібні записи. Всі ці параметри передаються на бекенд через GET-запит, де обробляються у функції get_listings() з умовами SQL-запиту. Кожен рядок таблиці також має набір дій – наприклад, посилання на сторінку редагування або кнопку блокування оголошення. Це дає змогу миттєво реагувати на порушення або редагувати дані без пошуку їх в основній базі. Інтерфейс адміністратора також містить форму для завантаження нових наборів даних (dataset), що є важливою складовою процесу підготовки навчального матеріалу для моделі машинного навчання. Зазвичай це дозволяє обрати кілька файлів (наприклад, у форматі CSV), додати короткий опис і зазначити, до якого типу вони належать – навчального чи тестового. Після завантаження дані проходять обробку: перевіряється формат, зберігається метаінформація, а сам файл – у відповідну директорію. Ця логіка обробляється функцією upload_dataset() у Flask і дозволяє гнучко працювати з датасетами.

## 3.5 Тестування системи прогнозування

Тестування платформи прогнозування цін на нерухомість реалізувалась з урахуванням кількох ключових підходів, кожен із яких спрямовували на досягнення високої якості, точності та зручності сервісу. Нижче детально розкривається внесок кожного методу тестування у вирішення цих завдань, із варіюванням мовних конструкцій для уникнення повторень та мінімальним використанням списків.

Функціональне тестування лягло в основу перевірки коректності роботи системи. Основна увага зосереджувалась на тому, щоб форма введення даних, алгоритми обробки запитів та відображення прогнозів цін функціонували бездоганно. Наприклад, передача параметрів (місто, загальна площа, кількість кімнат) мала завжди призводити до очікуваних результатів без системних збоїв. Критично важливим було забезпечити точність розрахунків цін на основі вхідних даних та зрозуміле для кінцевого користувача представлення прогнозів. Такий підхід дає підстави вважати, що ключові функції платформи працюють відповідно до вимог.

Якість обробки даних значно підвищилась після інтеграції методу validate_input_data(form_data). Цей інструмент відіграв ключову роль у валідації вхідної інформації. Його логіка передбачає ретельну перевірку заповнення обов'язкових полів (місто, площа, кімнати) та відповідність даних заданим форматам. Введення недопустимих значень (наприклад, від'ємна площа або нецiла кількість кімнат) активує зворотний зв'язок типу "Загальна площа повинна бути числом більше 0". Додатково реалізовано перевірки для опціональних параметрів: поверх (допустимий діапазон: -2 – 50), рік побудови (1900 – 2025), відстань до центру (додатнє число). Використання цього методу зробило систему стійкішою до помилкових даних, що знизило ризик неточних прогнозів та покращило інформування користувачів про помилки вводу.

Тестування граничних значень дозволило дослідити реакцію системи на екстремальні сценарії з метою виявлення слабких місць. Перевірка

включала ситуації з мінімальною площею (1 м²), максимальною кількістю кімнат (10) чи оцінками на межі шкали (1 чи 10). Також аналізували поведінку платформи при надходженні нестандартних значень (наприклад, від'ємна відстань до центру чи неможливий рік будівництва). Ця практика підтвердила стабільність роботи сайту навіть у неординарних умовах.

Важливу частину процесу зайняло тестування інтерфейсу та користувацького досвіду (UX/UI). Аналіз фокусувався на зручності взаємодії з формою введення та сприйнятті повідомлень про помилки. Якщо користувач не вказує місто, система виводить зрозуміле пояснення. Додатково оцінювалася інтуїтивність інтерфейсу, швидкість завантаження сторінок та коректність відображення результатів на різних пристроях (смартфони, планшети). Ці зусилля сприяли створенню більш дружнього середовища для користувачів.

Оцінка продуктивності ставила за мету визначити швидкість обробки запитів та стабільність системи під навантаженням. Моделювали ситуації одночасної роботи багатьох користувачів чи обробки великих масивів даних, переконуючись у прийнятному часі відгуку. Цей етап був вирішальним для забезпечення комфортної роботи платформи в реальних умовах високого навантаження.

Комплексне тестування, що поєднало функціональні перевірки, валідацію даних, аналіз граничних значень, оцінку інтерфейсу та продуктивності, дозволило досягти високого рівня стабільності та зручності платформи прогнозування цін на нерухомість.

# ВИСНОВКИ:

Розроблена система прогнозування цін на нерухомість довела свою ефективність у реальних умовах українського ринку. Основні результати підкреслюють значний прогрес у галузі автоматизованої оцінки: модель на базі HistGradientBoostingRegressor досягла вражаючої точності з $R^2$=0.872, демонструючи здатність достовірно передбачати 87.2% варіативності цін. Середня абсолютна похибка на рівні 28.5 тис. грн та середньоквадратичне відхилення 42.1 тис. грн підтверджують практичну придатність рішення для інвесторів і ріелторських агенцій.

Ключовим науковим інсайтом стало виявлення нелінійних залежностей між параметрами об'єктів та їх вартістю. Географічне розташування (15% впливу) та рік побудови (12%) виявилися вирішальними факторами, тоді як негативна кореляція енергоефективності з ціною вказала на прихований дисконт технічно застарілих будівель. Ці закономірності, виявлені через кореляційний аналіз, сформували основу для обґрунтованих прогнозів.

Архітектура системи інтегрує три взаємопов'язані рівні:

- Базу даних PostgreSQL із сутностями User, Property та Prediction, оптимізовану для швидкого пошуку та оновлення;
- Обчислювальне ядро на Python (Scikit-learn, Pandas), що автоматизує валідацію даних, навчання моделі та генерацію звітів;
- Інтуїтивний веб-інтерфейс (Flask), який спрощує введення параметрів через багатоступеневу форму та візуалізує результати у вигляді інтерактивних діаграм.

Для українського ринку розробка має особливе значення, адже враховує його унікальні виклики: фрагментацію джерел даних, вплив макроекономічної нестабільності та регіональну диференціацію цін. Інтеграція з локальними майданчиками (наприклад, OLX.ua) і підтримка української мови зробили систему доступною для широкого кола

користувачів. Перспективи вдосконалення включають розширення моделі за рахунок:

- Глибинних нейронних мереж для аналізу фотооб’єктів та текстових описів;
- Геопросторових інструментів (наприклад, інтеграції з OpenStreetMap) для оцінки інфраструктурної доступності;
- Мультимобільності через створення додатків під iOS/Android.

Система відкриває новий етап у роботі з нерухомістю в Україні, пропонуючи інвесторам, банкам та ріелторам інструмент для об’єктивних рішень. Її впровадження сприятиме прозорості угод, зменшенню фінансових ризиків та стабілізації ринку в умовах сучасних економічних викликів.

# ПЕРЕЛІК ПОСИЛАНЬ


1. OLX.ua: сервіс оголошень України. [Електронний ресурс]. URL: https://www.olx.ua/uk/.
2. DIM.RIA: вся нерухомість України. [Електронний ресурс]. URL: https://dom.ria.com/uk/.
3. Zillow: Real Estate, Apartments, Mortgages & Home Values. [Електронний ресурс]. URL: https://www.zillow.com/.
4. Chen P. P. The Entity-Relationship Model: Toward a Unified View of Data // ACM Transactions on Database Systems (TODS). – 1976. – Vol. 1, № 1. – P. 9–36.
5. IDEF0-моделювання: Стандарти. [Електронний ресурс] / National Institute of Standards and Technology (NIST). – Режим доступу: https://www.nist.gov/.
6. Géron A. Hands-On Machine Learning with Scikit-Learn, Keras and TensorFlow. – O'Reilly Media, 2019. – 856 p.
7. Mullainathan S., Spiess J. Machine Learning: An Applied Econometric Approach // Journal of Economic Perspectives. – 2017. – Vol. 31, No. 2. – P. 87-106.
8. Friedman J. H. Greedy Function Approximation: A Gradient Boosting Machine // Annals of Statistics. – 2001. – Vol. 29. – P. 1189–1232.
9. James G., Witten D., Hastie T., Tibshirani R. An Introduction to Statistical Learning: with Applications in R. – New York : Springer, 2013. – 426 p.
10. Scikit-learn: Документація бібліотеки машинного навчання Python. [Електронний ресурс] / Scikit-learn Developers. – Режим доступу: https://scikit-learn.org/stable/.
11. Chen T., Guestrin C. XGBoost: A Scalable Tree Boosting System // Proceedings of the 22nd ACM SIGKDD International Conference. – 2016. – P. 785-794.
12. Seaborn: Бібліотека для статистичної візуалізації. [Електронний ресурс] / Michael Waskom et al. – Режим доступу: https://seaborn.pydata.org/.

13. PostgreSQL: Система керування реляційними базами даних. Документація. [Електронний ресурс] / PostgreSQL Global Development Group. – Режим доступу: https://www.postgresql.org/docs/.
14. Pandas: Бібліотека для обробки даних. [Електронний ресурс] / Pandas Development Team. – Режим доступу: https://pandas.pydata.org/.
15. NumPy: Бібліотека для наукових обчислень. [Електронний ресурс] / NumPy Developers. – Режим доступу: https://numpy.org/.
16. Грушка А. О., Мельник О. С. Веб-технології впровадження моделей машинного навчання. – Вісник НТУУ "КПІ". – 2021. – № 3. – С. 45-52.
17. Matplotlib: Бібліотека для візуалізації. [Електронний ресурс] / Matplotlib Development Team. – Режим доступу: https://matplotlib.org/.
18. Chart.js: Бібліотека для створення інтерактивних діаграм. [Електронний ресурс]. – Режим доступу: https://www.chartjs.org/.

# ДОДАТОК А – Код програми

Файл "app.py":

```
     import joblib
from flask import Flask, render_template, request, redirect,
url_for, session, flash
import psycopg2
import pandas as pd
import json
import numpy as np
from pycaret.regression import load_model
import plotly.graph_objects as go
import plotly.express as px
from plotly.utils import PlotlyJSONEncoder

import os
import uuid
from datetime import datetime
from werkzeug.utils import secure_filename
from flask import render_template, request, jsonify, redirect,
url_for, session
import psycopg2
from psycopg2.extras import RealDictCursor
import json

app = Flask(__name__)
app.secret_key = "key"
app.config['TEMPLATES_AUTO_RELOAD'] = True

model =
joblib.load('D:\\flaskProject\\model\\apartment_price_model.pkl'
)


def get_db_connection():
    return psycopg2.connect(
        host="localhost",
        database="postgres",
        user="postgres",
        password="24012004"
        )


@app.route('/')
def home():
    return render_template('index.html')


@app.route('/contacts')
def contacts():
    return render_template('contacts.html')
```

```
@app.route('/predict', methods=['GET', 'POST'])
def predict():
    prediction = None
    optional_info = {}

    if request.method == 'POST':
        try:
            city = request.form['city']
            total_square = float(request.form['total_square'])
            rooms_count = int(request.form['rooms_count'])
            district = request.form.get('district', 'unknown')
            street = request.form.get('street', 'unknown')
            total_floors =
float(request.form.get('total_floors', 5.0))
            floor = float(request.form.get('floor', 2.0))
            year_built = request.form.get('year_built')
            construction_type =
request.form.get('construction_type', 'Panel')
            elevator = request.form.get('elevator')
            last_reconstruction =
request.form.get('last_reconstruction')

            repair = request.form.get('repair')
            balcony = request.form.get('balcony')
            loggia = request.form.get('loggia')
            basement = request.form.get('basement')
            orientation = request.form.get('orientation',
'South')
            certificate = request.form.get('certificate', 'B')

            parking = request.form.get('parking')
            quality_of_living =
float(request.form.get('quality_of_living', 7.0))
            safety = float(request.form.get('safety', 7.0))
            transport = float(request.form.get('transport',
7.0))
            services = float(request.form.get('services', 7.0))
            relax = float(request.form.get('relax', 5.0))
            environment = request.form.get('environment', '7')
            energy_costs = request.form.get('energy_costs')

            index = min(10.0, max(1.0, (total_square /
rooms_count) / 5))

            input_data = pd.DataFrame([{
                'city': city,
                'total_square': total_square,
                'rooms_count': rooms_count,
                'district': district,
                'street': street,
                'balcony': balcony,
                'repair': repair,
```

```
                'elevator': elevator,           'parking':
parking,
                'basement': basement,
                'year_built': year_built,
                'total_floors': total_floors,
                'floor': floor,
                'construction_type': construction_type,
                'last_reconstruction': last_reconstruction if
last_reconstruction else np.nan,
                'loggia': loggia,
                'orientation': orientation,
                'certificate': certificate,
                'quality_of_living': quality_of_living,
                'safety': safety,
                'transport': transport,
                'services': services,
                'relax': relax,
                'environment': environment,
                'energy_costs': energy_costs if energy_costs
else np.nan,
                'index': index
                }])

            try:
                prediction = model.predict(input_data)[0]
            except Exception as e:
                print(f"Помилка обробки: {e}")

            session['report_data'] = {
                'prediction': prediction,
                'input_data': {
                    'city': city,
                    'district': district,
                    'street': street,
                    'total_square': total_square,
                    'rooms_count': rooms_count,
                    'total_floors': total_floors,
                    'floor': floor,
                    'year_built': year_built,
                    'construction_type': construction_type,
                    'elevator': elevator,
                    'last_reconstruction': last_reconstruction,
                    'repair': repair,
                    'balcony': balcony,
                    'loggia': loggia,
                    'basement': basement,
                    'orientation': orientation,
                    'certificate': certificate,
                    'parking': parking,
                    'quality_of_living': quality_of_living,
                    'safety': safety,
                    'transport': transport,
```

```
                    'services': services,
                    'relax': relax,                'environment':
environment,
                    'energy_costs': energy_costs,
                    'index': index
                    }
                }

        except ValueError as ve:
            prediction = f"Помилка валідації даних: {str(ve)}"
        except KeyError as ke:
            prediction = f"Відсутнє обов'язкове поле: {str(ke)}"
        except Exception as e:
            prediction = f"Помилка обробки: {str(e)}"

    if request.method == 'POST' and prediction and
isinstance(prediction, (int, float)):
        action = request.form.get('action', 'view_report')

        if action == 'confirm_listing':
            return redirect(url_for('confirm_listing'))
        else:
            return redirect(url_for('view_report'))

    return render_template('predict.html',
prediction=prediction, optional_info=optional_info)


@app.route('/report')
def view_report():
    if 'report_data' not in session:
        flash('Дані для звіту не знайдено. Будь ласка, спочатку
оцініть нерухомість.', 'error')
        return redirect(url_for('predict'))

    report_data = session['report_data']
    return render_template('report.html',
report_data=report_data)


UPLOAD_FOLDER = 'static/uploads'
ALLOWED_EXTENSIONS = {'png', 'jpg', 'jpeg', 'gif', 'webp'}
MAX_FILE_SIZE = 5 * 1024 * 1024
os.makedirs(UPLOAD_FOLDER, exist_ok=True)


def allowed_file(filename):
    return '.' in filename and \
        filename.rsplit('.', 1)[1].lower() in ALLOWED_EXTENSIONS


def create_listings_table():
```

```
    """Створення таблиці оголошень, якщо вона не існує"""
    conn = get_db_connection()
    if conn:        try:
            cur = conn.cursor()
            cur.execute("""
                CREATE TABLE IF NOT EXISTS listings (
                    id SERIAL PRIMARY KEY,
                    city VARCHAR(100) NOT NULL,
                    district VARCHAR(100),
                    street VARCHAR(200),
                    total_square DECIMAL(8,2) NOT NULL,
                    rooms_count INTEGER NOT NULL,
                    floor INTEGER,
                    total_floors INTEGER,
                    year_built INTEGER,
                    construction_type VARCHAR(50),
                    elevator BOOLEAN,
                    repair VARCHAR(50),
                    balcony BOOLEAN,
                    loggia BOOLEAN,
                    basement BOOLEAN,
                    orientation VARCHAR(50),
                    certificate VARCHAR(10),
                    parking VARCHAR(50),
                    quality_of_living INTEGER,
                    safety INTEGER,
                    transport INTEGER,
                    services INTEGER,
                    relax INTEGER,
                    environment INTEGER,
                    energy_costs DECIMAL(10,2),
                    predicted_price DECIMAL(12,2),
                    description TEXT,
                    created_at TIMESTAMP DEFAULT
CURRENT_TIMESTAMP,
                    updated_at TIMESTAMP DEFAULT
CURRENT_TIMESTAMP,
                    is_active BOOLEAN DEFAULT TRUE
                );
            """)

            cur.execute("""
                CREATE TABLE IF NOT EXISTS listing_photos (
                    id SERIAL PRIMARY KEY,
                    listing_id INTEGER REFERENCES listings(id)
ON DELETE CASCADE,
                    filename VARCHAR(255) NOT NULL,
                    original_name VARCHAR(255),
                    file_path VARCHAR(500) NOT NULL,
                    file_size INTEGER,
                    upload_order INTEGER DEFAULT 0,
                    created_at TIMESTAMP DEFAULT
```

```
CURRENT_TIMESTAMP
                );
            """)            conn.commit()
            cur.close()
        except psycopg2.Error as e:
            print(f"Помилка створення таблиць: {e}")
        finally:
            conn.close()

create_listings_table()


@app.route('/confirm-listing')
def confirm_listing():
    if 'report_data' not in session:
        return redirect(url_for('predict'))

    return render_template('confirm_listing.html')


@app.route('/api/listings', methods=['POST'])
def create_listing():
    try:
        description = request.form.get('description',
'').strip()
        property_data_json = request.form.get('propertyData',
'{}')

        if not description:
            return jsonify({'error': 'Опис оголошення
обов\'язковий'}), 400

        try:
            property_data = json.loads(property_data_json)
        except json.JSONDecodeError:
            if 'report_data' in session:
                property_data =
session['report_data']['input_data']
                predicted_price =
session['report_data']['prediction']
            else:
                return jsonify({'error': 'Дані про нерухомість
відсутні'}), 400
        else:
            predicted_price = property_data.get('prediction', 0)

        required_fields = ['city', 'total_square',
'rooms_count']
        for field in required_fields:
            if not property_data.get(field):
                return jsonify({'error': f'Поле {field}
обов\'язкове'}), 400
```

```
        conn = get_db_connection()
        if not conn:            return jsonify({'error':
'Помилка підключення до бази даних'}), 500

        try:
            cur = conn.cursor()

            insert_query = """
                INSERT INTO listings (
                    city, district, street, total_square,
rooms_count, floor, total_floors,
                    year_built, construction_type, elevator,
repair, balcony, loggia, basement,
                    orientation, certificate, parking,
quality_of_living, safety, transport,
                    services, relax, environment, energy_costs,
predicted_price, description
                ) VALUES (
                    %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s,
%s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s
                ) RETURNING id;
            """

            cur.execute(insert_query, (
                property_data.get('city'),
                property_data.get('district'),
                property_data.get('street'),
                float(property_data.get('total_square', 0)),
                int(property_data.get('rooms_count', 0)),
                property_data.get('floor'),
                property_data.get('total_floors'),
                property_data.get('year_built'),
                property_data.get('construction_type'),
                property_data.get('elevator') == '1' if
property_data.get('elevator') else None,
                property_data.get('repair'),
                property_data.get('balcony') == '1' if
property_data.get('balcony') else None,
                property_data.get('loggia') == '1' if
property_data.get('loggia') else None,
                property_data.get('basement') == '1' if
property_data.get('basement') else None,
                property_data.get('orientation'),
                property_data.get('certificate'),
                property_data.get('parking'),
                property_data.get('quality_of_living'),
                property_data.get('safety'),
                property_data.get('transport'),
                property_data.get('services'),
                property_data.get('relax'),
                property_data.get('environment'),
```

```
                property_data.get('energy_costs'),
                float(predicted_price) if predicted_price else
None,                description
                ))

            listing_id = cur.fetchone()['id']

            uploaded_files = request.files.getlist('photos')
            photo_order = 0

            for photo in uploaded_files:
                if photo and photo.filename and
allowed_file(photo.filename):
                    photo.seek(0, os.SEEK_END)
                    file_size = photo.tell()
                    photo.seek(0)

                    if file_size > MAX_FILE_SIZE:
                        continue

                    filename = secure_filename(photo.filename)
                    file_extension = filename.rsplit('.',
1)[1].lower()
                    unique_filename =
f"{uuid.uuid4().hex}.{file_extension}"

                    listing_folder = os.path.join(UPLOAD_FOLDER,
str(listing_id))
                    os.makedirs(listing_folder, exist_ok=True)

                    file_path = os.path.join(listing_folder,
unique_filename)

                    photo.save(file_path)

                    photo_query = """
                        INSERT INTO listing_photos (
                            listing_id, filename, original_name,
file_path, file_size, upload_order
                        ) VALUES (%s, %s, %s, %s, %s, %s);
                    """

                    cur.execute(photo_query, (
                        listing_id,
                        unique_filename,
                        filename,
                        file_path,
                        file_size,
                        photo_order
                        ))

                    photo_order += 1
```

```
            conn.commit()
            cur.close()            if 'report_data' in session:
                del session['report_data']

            return jsonify({
                'success': True,
                'id': listing_id,
                'message': 'Оголошення успішно створено'
                }), 201

        except psycopg2.Error as e:
            conn.rollback()
            print(f"Помилка БД: {e}")
            return jsonify({'error': 'Помилка збереження в базу
даних'}), 500
        finally:
            conn.close()

    except Exception as e:
        print(f"Загальна помилка: {e}")
        return jsonify({'error': 'Внутрішня помилка сервера'}),
500


@app.route('/listing/<int:listing_id>')
def view_listing(listing_id):
    conn = get_db_connection()
    if not conn:
        return "Помилка підключення до бази даних", 500

    try:
        cur = conn.cursor()

        cur.execute("""
            SELECT * FROM listings
            WHERE id = %s AND is_active = TRUE
        """, (listing_id,))

        listing = cur.fetchone()
        if not listing:
            return "Оголошення не знайдено", 404

        cur.execute("""
            SELECT filename, original_name, file_path,
upload_order
            FROM listing_photos
            WHERE listing_id = %s
            ORDER BY upload_order ASC
        """, (listing_id,))

        photos = cur.fetchall()
```

```
        cur.close()
        return render_template('listing_detail.html',
listing=listing, photos=photos)

    except psycopg2.Error as e:
        print(f"Помилка БД: {e}")
        return "Помилка отримання даних", 500
    finally:
        conn.close()


@app.route('/listings')
def listings_list():
    conn = get_db_connection()
    if not conn:
        return "Помилка підключення до бази даних", 500

    try:
        cur = conn.cursor()

        page = request.args.get('page', 1, type=int)
        per_page = 20
        offset = (page - 1) * per_page

        city_filter = request.args.get('city', '')
        min_price = request.args.get('min_price', type=int)
        max_price = request.args.get('max_price', type=int)
        rooms_filter = request.args.get('rooms', type=int)

        where_conditions = ["is_active = TRUE"]
        params = []

        if city_filter:
            where_conditions.append("LOWER(city) LIKE
LOWER(%s)")
            params.append(f"%{city_filter}%")

        if min_price:
            where_conditions.append("predicted_price >= %s")
            params.append(min_price)

        if max_price:
            where_conditions.append("predicted_price <= %s")
            params.append(max_price)

        if rooms_filter:
            where_conditions.append("rooms_count = %s")
            params.append(rooms_filter)

        where_clause = " AND ".join(where_conditions)
```

```
        query = f"""
            SELECT l.*,
                   (SELECT filename FROM listing_photos lp
WHERE lp.listing_id = l.id
                    ORDER BY upload_order ASC LIMIT 1) as
first_photo
            FROM listings l
            WHERE {where_clause}
            ORDER BY l.created_at DESC
            LIMIT %s OFFSET %s
        """

        params.extend([per_page, offset])
        cur.execute(query, params)
        listings = cur.fetchall()

        count_query = f"SELECT COUNT(*) FROM listings WHERE
{where_clause}"
        cur.execute(count_query, params[:-2])
        total_count = cur.fetchone()['count']

        cur.close()

        return render_template('listings_list.html',
                               listings=listings,
                               page=page,
                               per_page=per_page,
                               total_count=total_count,
                               has_prev=page > 1,
                               has_next=offset + per_page <
total_count)

    except psycopg2.Error as e:
        print(f"Помилка БД: {e}")
        return "Помилка отримання даних", 500
    finally:
        conn.close()


@app.route('/api/listings/<int:listing_id>', methods=['DELETE'])
def delete_listing(listing_id):
    conn = get_db_connection()
    if not conn:
        return jsonify({'error': 'Помилка підключення до бази
даних'}), 500

    try:
        cur = conn.cursor()

        cur.execute("""
            UPDATE listings
            SET is_active = FALSE, updated_at =
```

```
CURRENT_TIMESTAMP
            WHERE id = %s
        """, (listing_id,))
        if cur.rowcount == 0:
            return jsonify({'error': 'Оголошення не знайдено'}),
404

        conn.commit()
        cur.close()

        return jsonify({'success': True, 'message': 'Оголошення
видалено'})

    except psycopg2.Error as e:
        conn.rollback()
        print(f"Помилка БД: {e}")
        return jsonify({'error': 'Помилка видалення'}), 500
    finally:
        conn.close()

def validate_input_data(form_data):
    errors = []

    required_fields = ['city', 'total_square', 'rooms_count']
    for field in required_fields:
        if not form_data.get(field):
            errors.append(f"Поле '{field}' є обов'язковим")

    try:
        total_square = float(form_data.get('total_square', 0))
        if total_square <= 0:
            errors.append("Загальна площа повинна бути більше
0")
    except ValueError:
        errors.append("Загальна площа повинна бути числом")

    try:
        rooms_count = int(form_data.get('rooms_count', 0))
        if rooms_count <= 0:
            errors.append("Кількість кімнат повинна бути більше
0")
    except ValueError:
        errors.append("Кількість кімнат повинна бути цілим
числом")

    rating_fields = ['quality_of_living', 'safety', 'transport',
'services', 'relax']
    for field in rating_fields:
        value = form_data.get(field)
        if value:
            try:
                rating = float(value)
```

```
                if not (1 <= rating <= 10):
                    errors.append(f"Оцінка '{field}' повинна
бути від 1 до 10")
            except ValueError:
errors.append(f"Оцінка '{field}' повинна бути числом")

    return errors

if __name__ == '__main__':
    app.run(debug=True)
```

Файл "predict.html"

```
<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Оцінка вартості нерухомості</title>
    <style>
        body {
            font-family: 'Segoe UI', Tahoma, Geneva, Verdana,
sans-serif;
            margin: 0;
            padding: 20px;
            background: linear-gradient(135deg, #667eea 0%,
#764ba2 100%);
            min-height: 100vh;
        }

        .container {
            max-width: 800px;
            margin: 0 auto;
            background: white;
            border-radius: 15px;
            box-shadow: 0 10px 30px rgba(0,0,0,0.2);
            overflow: hidden;
        }

        h1 {
            background: linear-gradient(135deg, #667eea 0%,
#764ba2 100%);
            color: white;
            text-align: center;
            margin: 0;
            padding: 30px 20px;
            font-size: 28px;
        }

        .form-content {
            padding: 30px;
        }
```

```
        .step-indicator {
            background: #f8f9fa;       padding: 15px;
            margin-bottom: 30px;
            border-radius: 8px;
            text-align: center;
            font-weight: bold;
            color: #495057;
            border-left: 4px solid #667eea;
        }

        .form-group {
            margin-bottom: 20px;
        }

        label {
            display: block;
            margin-bottom: 8px;
            font-weight: 600;
            color: #333;
        }

        input[type="text"], input[type="number"], select {
            width: 100%;
            padding: 12px;
            border: 2px solid #e9ecef;
            border-radius: 8px;
            font-size: 16px;
            transition: border-color 0.3s ease;
        }

        input[type="text"]:focus, input[type="number"]:focus,
select:focus {
            outline: none;
            border-color: #667eea;
            box-shadow: 0 0 0 3px rgba(102, 126, 234, 0.1);
        }

        input[type="range"] {
            width: 100%;
            margin: 10px 0;
        }

        .range-labels {
            display: flex;
            justify-content: space-between;
            font-size: 14px;
            color: #666;
            margin-top: 5px;
        }

        .range-labels span:nth-child(2) {
```

```
            font-weight: bold;
            color: #667eea;
        }       .nav-buttons {
            display: flex;
            gap: 15px;
            justify-content: center;
            margin-top: 30px;
            padding-top: 30px;
            border-top: 1px solid #e9ecef;
        }

        button {
            padding: 12px 30px;
            border: none;
            border-radius: 8px;
            font-size: 16px;
            font-weight: 600;
            cursor: pointer;
            transition: all 0.3s ease;
            min-width: 120px;
        }

        button[type="submit"], .btn-primary {
            background: linear-gradient(135deg, #667eea 0%,
#764ba2 100%);
            color: white;
        }

        button[type="submit"]:hover, .btn-primary:hover {
            transform: translateY(-2px);
            box-shadow: 0 5px 15px rgba(102, 126, 234, 0.4);
        }

        .btn-secondary {
            background: #6c757d;
            color: white;
        }

        .btn-secondary:hover {
            background: #5a6268;
            transform: translateY(-2px);
        }

        .hidden {
            display: none;
        }

        .result {
            background: linear-gradient(135deg, #f8f9fa 0%,
#e9ecef 100%);
            border-radius: 15px;
            padding: 30px;
```

```
    margin-top: 30px;
    text-align: center;
    border: 2px solid #667eea;        }

.result h3 {
    color: #333;
    margin-bottom: 20px;
    font-size: 24px;
}

.price-display {
    font-size: 42px;
    font-weight: bold;
    color: #27ae60;
    margin: 20px 0;
    text-shadow: 2px 2px 4px rgba(0,0,0,0.1);
}

.price-per-meter {
    font-size: 20px;
    color: #7f8c8d;
    margin-bottom: 30px;
}

.property-details {
    text-align: left;
    background: white;
    padding: 20px;
    border-radius: 10px;
    margin: 20px 0;
    box-shadow: 0 2px 10px rgba(0,0,0,0.1);
}

.property-details strong {
    color: #667eea;
    font-size: 18px;
}

.property-details ul {
    list-style: none;
    padding: 0;
    margin: 15px 0 0 0;
}

.property-details li {
    padding: 8px 0;
    border-bottom: 1px solid #f8f9fa;
    display: flex;
    justify-content: space-between;
}

.property-details li:last-child {
```

```
            border-bottom: none;
        }
    .action-buttons {
            display: flex;
            gap: 20px;
            justify-content: center;
            margin-top: 30px;
            flex-wrap: wrap;
        }

        .btn-report {
            background: linear-gradient(135deg, #28a745 0%,
#20c997 100%);
            color: white;
            padding: 15px 30px;
            text-decoration: none;
            border-radius: 10px;
            font-weight: 600;
            transition: all 0.3s ease;
            border: none;
            cursor: pointer;
            font-size: 16px;
        }

        .btn-report:hover {
            transform: translateY(-3px);
            box-shadow: 0 8px 25px rgba(40, 167, 69, 0.4);
        }

        .btn-listing {
            background: linear-gradient(135deg, #fd7e14 0%,
#e83e8c 100%);
            color: white;
            padding: 15px 30px;
            text-decoration: none;
            border-radius: 10px;
            font-weight: 600;
            transition: all 0.3s ease;
            border: none;
            cursor: pointer;
            font-size: 16px;
        }

        .btn-listing:hover {
            transform: translateY(-3px);
            box-shadow: 0 8px 25px rgba(253, 126, 20, 0.4);
        }

        @media (max-width: 768px) {
            body {
                padding: 10px;
            }
```

```
            .container {
                margin: 0;       }

            h1 {
                font-size: 24px;
                padding: 20px;
            }

            .form-content {
                padding: 20px;
            }

            .action-buttons {
                flex-direction: column;
                align-items: center;
            }

            .btn-report, .btn-listing {
                width: 100%;
                max-width: 300px;
            }
        }
    </style>
</head>
<body>
    <div class="container">
        <h1>Оцінка вартості нерухомості</h1>

        <div class="form-content">
            <form id="predictionForm" method="POST" action="{{
url_for('predict') }}">
                <!-- Етап 1: Основні дані -->
                <div id="step1">
                    <div class="step-indicator">Крок 1 з 4:
Основні характеристики</div>

                    <div class="form-group">
                        <label for="city">Місто *</label>
                        <input type="text" id="city" name="city"
required placeholder="Наприклад: Київ">
                    </div>

                    <div class="form-group">
                        <label for="district">Район</label>
                        <input type="text" id="district"
name="district" placeholder="Наприклад: Печерський">
                    </div>

                    <div class="form-group">
                        <label for="street">Вулиця</label>
                        <input type="text" id="street"
```

```
name="street" placeholder="Наприклад: вул. Хрещатик">
                    </div>
                <div class="form-group">
                        <label for="total_square">Загальна площа
(м²) *</label>
                        <input type="number" step="0.1"
id="total_square" name="total_square" required min="10"
max="500">
                    </div>

                    <div class="form-group">
                        <label for="rooms_count">Кількість
кімнат *</label>
                        <input type="number" id="rooms_count"
name="rooms_count" required min="1" max="10">
                    </div>

                    <div class="nav-buttons">
                        <button type="button" class="btn-
primary" onclick="nextStep(2)">Далі</button>
                    </div>
                </div>

                <!-- Етап 2: Будівля -->
                <div id="step2" class="hidden">
                    <div class="step-indicator">Крок 2 з 4:
Характеристики будівлі</div>

                    <div class="form-group">
                        <label for="total_floors">Загальна
кількість поверхів</label>
                        <input type="number" id="total_floors"
name="total_floors" min="1" max="50" value="5">
                    </div>

                    <div class="form-group">
                        <label for="floor">Поверх
квартири</label>
                        <input type="number" id="floor"
name="floor" min="1" max="50" value="2">
                    </div>

                    <div class="form-group">
                        <label for="year_built">Рік
побудови</label>
                        <input type="number" id="year_built"
name="year_built" min="1900" max="2024">
                    </div>

                    <div class="form-group">
                        <label for="construction_type">Тип
будівництва</label>
```

```
                        <select id="construction_type"
name="construction_type">
                            <option value="Panel"
selected>Панельний</option>
                            <option
value="Brick">Цегляний</option>
                            <option
value="Monolith">Монолітний</option>
                            <option value="Other">Інший</option>
                        </select>
                    </div>

                    <div class="form-group">
                        <label for="elevator">Ліфт</label>
                        <select id="elevator" name="elevator">
                            <option value="">-- Оберіть --
</option>
                            <option value="1">Так</option>
                            <option value="0">Ні</option>
                        </select>
                    </div>

                    <div class="form-group">
                        <label for="last_reconstruction">Рік
останньої реконструкції</label>
                        <input type="number"
id="last_reconstruction" name="last_reconstruction" min="1990"
max="2024">
                    </div>

                    <div class="nav-buttons">
                        <button type="button" class="btn-
secondary" onclick="prevStep(1)">Назад</button>
                        <button type="button" class="btn-
primary" onclick="nextStep(3)">Далі</button>
                    </div>
                </div>

                <!-- Етап 3: Квартира -->
                <div id="step3" class="hidden">
                    <div class="step-indicator">Крок 3 з 4:
Характеристики квартири</div>

                    <div class="form-group">
                        <label for="repair">Ремонт</label>
                        <select id="repair" name="repair">
                            <option value="">-- Оберіть --
</option>
                            <option
value="cosmetic">Косметичний</option>
                            <option value="euro">Євро</option>
                            <option value="without">Без
```

```
ремонту</option>
                        </select>
                    </div>
                <div class="form-group">
                        <label for="balcony">Балкон</label>
                        <select id="balcony" name="balcony">
                            <option value="">-- Оберіть --
</option>
                            <option value="1">Так</option>
                            <option value="0">Ні</option>
                        </select>
                    </div>

                    <div class="form-group">
                        <label for="loggia">Лоджія</label>
                        <select id="loggia" name="loggia">
                            <option value="">-- Оберіть --
</option>
                            <option value="1">Так</option>
                            <option value="0">Ні</option>
                        </select>
                    </div>

                    <div class="form-group">
                        <label for="basement">Підвал</label>
                        <select id="basement" name="basement">
                            <option value="">-- Оберіть --
</option>
                            <option value="1">Так</option>
                            <option value="0">Ні</option>
                        </select>
                    </div>

                    <div class="form-group">
                        <label for="orientation">Орієнтація
вікон</label>
                        <select id="orientation"
name="orientation">
                            <option value="South"
selected>Південь</option>
                            <option
value="North">Північ</option>
                            <option value="East">Схід</option>
                            <option value="West">Захід</option>
                            <option value="Southeast">Південний
схід</option>
                            <option value="Southwest">Південний
захід</option>
                            <option value="Northeast">Північний
схід</option>
                            <option value="Northwest">Північний
захід</option>
```

```
                        </select>
                    </div>

                    <div class="form-group">
<label for="certificate">Енергетичний сертифікат</label>
                        <select id="certificate"
name="certificate">
                            <option value="A">A
(найкращий)</option>
                            <option value="B" selected>B
(добрий)</option>
                            <option value="C">C
(задовільний)</option>
                            <option value="D">D
(низький)</option>
                            <option value="E">E (дуже
низький)</option>
                        </select>
                    </div>

                    <div class="nav-buttons">
                        <button type="button" class="btn-
secondary" onclick="prevStep(2)">Назад</button>
                        <button type="button" class="btn-
primary" onclick="nextStep(4)">Далі</button>
                    </div>
                </div>

                <!-- Етап 4: Оточення та послуги -->
                <div id="step4" class="hidden">
                    <div class="step-indicator">Крок 4 з 4:
Район та послуги</div>

                    <div class="form-group">
                        <label for="parking">Парковка</label>
                        <select id="parking" name="parking">
                            <option value="">-- Оберіть --
</option>
                            <option value="yes">Є</option>
                            <option value="no">Немає</option>
                            <option value="paid">Платна</option>
                        </select>
                    </div>

                    <div class="form-group">
                        <label for="quality_of_living">Якість
життя в районі (1-10)</label>
                        <input type="range"
id="quality_of_living" name="quality_of_living" min="1" max="10"
value="7" oninput="updateRangeValue('quality_of_living')">
                        <div class="range-labels">
                            <span>1 (погано)</span>
```

```
                            <span
id="quality_of_living_value">7</span>
                            <span>10 (відмінно)</span>
                        </div>               </div>

                    <div class="form-group">
                        <label for="safety">Безпека району (1-
10)</label>
                        <input type="range" id="safety"
name="safety" min="1" max="10" value="7"
oninput="updateRangeValue('safety')">
                        <div class="range-labels">
                            <span>1 (небезпечно)</span>
                            <span id="safety_value">7</span>
                            <span>10 (дуже безпечно)</span>
                        </div>
                    </div>

                    <div class="form-group">
                        <label for="transport">Транспортна
доступність (1-10)</label>
                        <input type="range" id="transport"
name="transport" min="1" max="10" value="7"
oninput="updateRangeValue('transport')">
                        <div class="range-labels">
                            <span>1 (погано)</span>
                            <span id="transport_value">7</span>
                            <span>10 (відмінно)</span>
                        </div>
                    </div>

                    <div class="form-group">
                        <label for="services">Наявність послуг
(1-10)</label>
                        <input type="range" id="services"
name="services" min="1" max="10" value="7"
oninput="updateRangeValue('services')">
                        <div class="range-labels">
                            <span>1 (мало)</span>
                            <span id="services_value">7</span>
                            <span>10 (багато)</span>
                        </div>
                    </div>

                    <div class="form-group">
                        <label for="relax">Можливості для
відпочинку (1-10)</label>
                        <input type="range" id="relax"
name="relax" min="1" max="10" value="5"
oninput="updateRangeValue('relax')">
                        <div class="range-labels">
                            <span>1 (мало)</span>
```

```
                            <span id="relax_value">5</span>
                            <span>10 (багато)</span>
                        </div>
                    </div>                    <div class="form-
group">
                        <label for="environment">Екологічна
ситуація</label>
                        <select id="environment"
name="environment">
                            <option value="10">Відмінна</option>
                            <option value="9">Дуже
добра</option>
                            <option value="8">Добра</option>
                            <option value="7"
selected>Задовільна</option>
                            <option value="6">Нижче
середнього</option>
                            <option value="5">Середня</option>
                            <option value="4">Погана</option>
                            <option value="3">Дуже
погана</option>
                            <option value="2">Критична</option>
                            <option value="1">Надзвичайно
погана</option>
                        </select>
                    </div>

                    <div class="form-group">
                        <label for="energy_costs">Річні витрати
на енергію (грн)</label>
                        <input type="number" id="energy_costs"
name="energy_costs" min="0" max="100000" step="100"
placeholder="Наприклад: 15000">
                    </div>

                    <div class="nav-buttons">
                        <button type="button" class="btn-
secondary" onclick="prevStep(3)">Назад</button>
                        <button type="submit" name="action"
value="view_report">Розрахувати вартість</button>
                    </div>
                </div>
            </form>

            <div id="result" class="result hidden">
                <!-- Результати з'являться тут -->
            </div>
        </div>
    </div>

    <script>
        let currentStep = 1;
```

```
        function nextStep(step) {
            // Валідація поточного кроку
            if (!validateStep(currentStep)) {
                return;       }

            document.getElementById('step' +
currentStep).classList.add('hidden');
            document.getElementById('step' +
step).classList.remove('hidden');
            currentStep = step;

            // Автоматичне обчислення деяких полів
            if (step === 2) {
                calculateDefaults();
            }
        }

        function prevStep(step) {
            document.getElementById('step' +
currentStep).classList.add('hidden');
            document.getElementById('step' +
step).classList.remove('hidden');
            currentStep = step;
        }

        function validateStep(step) {
            if (step === 1) {
                const city =
document.getElementById('city').value;
                const total_square =
document.getElementById('total_square').value;
                const rooms_count =
document.getElementById('rooms_count').value;

                if (!city || !total_square || !rooms_count) {
                    alert("Будь ласка, заповніть усі обов'язкові
поля (позначені *).");
                    return false;
                }

                if (parseFloat(total_square) <= 0) {
                    alert("Площа повинна бути більше 0.");
                    return false;
                }

                if (parseInt(rooms_count) <= 0) {
                    alert("Кількість кімнат повинна бути більше
0.");
                    return false;
                }
            }
```

```
            return true;
        }

        function calculateDefaults() {
            // Обчислюємо index на основі площі та кімнат
const totalSquare =
parseFloat(document.getElementById('total_square').value) || 0;
            const roomsCount =
parseInt(document.getElementById('rooms_count').value) || 1;

            // Простий алгоритм для обчислення індексу
            const index = Math.min(10, Math.max(1, (totalSquare
/ roomsCount) / 5));

            // Зберігаємо обчислене значення для подальшого
використання

document.getElementById('predictionForm').setAttribute('data-
index', index.toFixed(1));
        }

        function updateRangeValue(fieldName) {
            const value =
document.getElementById(fieldName).value;
            document.getElementById(fieldName +
'_value').textContent = value;
        }

        // Функція для відправки форми з вибором дії
        function submitForm(action) {
            const form =
document.getElementById('predictionForm');

            // Створюємо прихований input для action
            const actionInput = document.createElement('input');
            actionInput.type = 'hidden';
            actionInput.name = 'action';
            actionInput.value = action;
            form.appendChild(actionInput);

            // Відправляємо форму
            form.submit();
        }


document.getElementById('predictionForm').addEventListener('subm
it', function(e) {
            e.preventDefault();

            // Збираємо всі дані з форми
            const formData = new FormData(this);
            const data = {};
```

```
            for (let [key, value] of formData.entries()) {
                data[key] = value;
            }

            // Додаємо обчислені значення       data['index'] =
this.getAttribute('data-index') || '7.0';

            // Заповнюємо пропущені значення значеннями за
замовчуванням
            const defaults = {
                'quality_of_living': '7.0',
                'safety': '7.0',
                'transport': '7.0',
                'services': '7.0',
                'relax': '5.0',
                'total_floors': '5.0',
                'floor': '2.0',
                'environment': '7',
                'certificate': 'B',
                'construction_type': 'Panel',
                'orientation': 'South'
            };

            for (let [key, defaultValue] of
Object.entries(defaults)) {
                if (!data[key] || data[key] === '') {
                    data[key] = defaultValue;
                }
            }

            // Показуємо результат
            displayResult(data);
        });

        function displayResult(data) {
            // Простий алгоритм для демонстрації
            const basePrice = 25000; // грн за м²
            const totalSquare = parseFloat(data.total_square) ||
50;

            let multiplier = 1.0;

            // Коригування на основі різних факторів
            if (data.repair === 'euro') multiplier += 0.2;
            else if (data.repair === 'without') multiplier -=
0.15;

            if (data.balcony === '1') multiplier += 0.05;
            if (data.loggia === '1') multiplier += 0.03;
            if (data.elevator === '1') multiplier += 0.08;
            if (data.parking === 'yes') multiplier += 0.1;
```

```
            else if (data.parking === 'paid') multiplier +=
0.05;

            multiplier += (parseFloat(data.quality_of_living) -
5) * 0.02;
            multiplier += (parseFloat(data.safety) - 5) * 0.02;
multiplier += (parseFloat(data.transport) - 5) * 0.015;

            const estimatedPrice = Math.round(basePrice *
totalSquare * multiplier);

            const resultDiv = document.getElementById('result');
            resultDiv.innerHTML = `
                <h3>Оцінена вартість нерухомості</h3>
                <div class="price-display">
                    ${estimatedPrice.toLocaleString('uk-UA')}
грн
                </div>
                <div class="price-per-meter">
                    ~${Math.round(estimatedPrice /
totalSquare).toLocaleString('uk-UA')} грн/м²
                </div>

                <div class="property-details">
                    <strong>Основні характеристики:</strong>
                    <ul>

<li><span>Площа:</span><span>${totalSquare} м²</span></li>

<li><span>Кімнат:</span><span>${data.rooms_count}</span></li>

<li><span>Поверх:</span><span>${data.floor} з
${data.total_floors}</span></li>

<li><span>Місто:</span><span>${data.city}</span></li>
                        ${data.district ?
`<li><span>Район:</span><span>${data.district}</span></li>` :
''}
                        ${data.year_built ? `<li><span>Рік
побудови:</span><span>${data.year_built}</span></li>` : ''}
                        ${data.repair ?
`<li><span>Ремонт:</span><span>${getRepairText(data.repair)}</sp
an></li>` : ''}
                        ${data.parking ?
`<li><span>Парковка:</span><span>${getParkingText(data.parking)}
</span></li>` : ''}
                    </ul>
                </div>

                <div class="action-buttons">
                    <button type="button" class="btn-report"
onclick="submitForm('view_report')">
```

```
                        📊 Переглянути детальний звіт
                    </button>
                    <button type="button" class="btn-listing"
onclick="submitForm('confirm_listing')">
                        💼 Створити оголошення
                    </button>           </div>
            `;
            resultDiv.classList.remove('hidden');

            // Прокручуємо до результату
            resultDiv.scrollIntoView({ behavior: 'smooth' });
        }

        function getRepairText(repair) {
            const repairMap = {
                'cosmetic': 'Косметичний',
                'euro': 'Євроремонт',
                'without': 'Без ремонту'
            };
            return repairMap[repair] || repair;
        }

        function getParkingText(parking) {
            const parkingMap = {
                'yes': 'Є',
                'no': 'Немає',
                'paid': 'Платна'
            };
            return parkingMap[parking] || parking;
        }

        // Ініціалізація значень слайдерів
        document.addEventListener('DOMContentLoaded', function()
{
            updateRangeValue('quality_of_living');
            updateRangeValue('safety');
            updateRangeValue('transport');
            updateRangeValue('services');
            updateRangeValue('relax');
        });
    </script>
</body>
</html>
```

Файл "report.html"

```
     <!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
```

```
    <title>Звіт оцінки нерухомості</title>
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/Chart.js/3.9.1/chart
.min.js"></script>
    <style>
        * {       margin: 0;
            padding: 0;
            box-sizing: border-box;
        }

        body {
            font-family: 'Segoe UI', Tahoma, Geneva, Verdana,
sans-serif;
            background: linear-gradient(135deg, #667eea 0%,
#764ba2 100%);
            min-height: 100vh;
            padding: 20px;
        }

        .container {
            max-width: 1200px;
            margin: 0 auto;
            background: white;
            border-radius: 20px;
            box-shadow: 0 20px 40px rgba(0,0,0,0.1);
            overflow: hidden;
        }

        .header {
            background: linear-gradient(135deg, #667eea 0%,
#764ba2 100%);
            color: white;
            padding: 40px;
            text-align: center;
        }

        .header h1 {
            font-size: 2.5rem;
            margin-bottom: 10px;
        }

        .header p {
            font-size: 1.2rem;
            opacity: 0.9;
        }

        .content {
            padding: 40px;
        }

        .price-section {
            text-align: center;
```

```
            margin-bottom: 40px;
            padding: 30px;
            background: linear-gradient(135deg, #f8f9fa 0%,
#e9ecef 100%);
            border-radius: 15px;
        }        .main-price {
            font-size: 3rem;
            font-weight: bold;
            color: #2c3e50;
            margin-bottom: 10px;
        }

        .price-per-meter {
            font-size: 1.5rem;
            color: #7f8c8d;
            margin-bottom: 10px;
        }

        .confidence {
            font-size: 1.1rem;
            color: #27ae60;
            font-weight: 600;
        }

        .property-details {
            margin-bottom: 40px;
        }

        .property-details h3 {
            font-size: 1.8rem;
            margin-bottom: 20px;
            color: #2c3e50;
        }

        .details-grid {
            display: grid;
            grid-template-columns: 1fr 1fr;
            gap: 30px;
        }

        .detail-section {
            background: #f8f9fa;
            padding: 20px;
            border-radius: 10px;
        }

        .detail-item {
            display: flex;
            justify-content: space-between;
            margin-bottom: 15px;
            padding-bottom: 10px;
            border-bottom: 1px solid #e9ecef;
```

```
}

.detail-item:last-child {
    margin-bottom: 0;
    border-bottom: none;
}       .detail-label {
    font-weight: 600;
    color: #495057;
}

.detail-value {
    color: #2c3e50;
    font-weight: 500;
}

.grid {
    display: grid;
    grid-template-columns: 1fr 1fr;
    gap: 30px;
    margin-bottom: 40px;
}

.card {
    background: white;
    border-radius: 15px;
    padding: 25px;
    box-shadow: 0 5px 15px rgba(0,0,0,0.1);
    border: 1px solid #e9ecef;
}

.card h3 {
    font-size: 1.3rem;
    margin-bottom: 20px;
    color: #2c3e50;
}

.chart-container {
    position: relative;
    height: 300px;
}

.comparison-section {
    margin-bottom: 40px;
}

.comparison-section h3 {
    font-size: 1.8rem;
    margin-bottom: 20px;
    color: #2c3e50;
}

.factors-impact {
```

```
    background: #f8f9fa;
    border-radius: 10px;
    padding: 20px;
}

.factor-item {       display: flex;
    justify-content: space-between;
    align-items: center;
    margin-bottom: 15px;
    padding: 10px;
    background: white;
    border-radius: 8px;
}

.factor-name {
    font-weight: 500;
    color: #495057;
}

.factor-impact {
    font-weight: bold;
    font-size: 1.1rem;
    padding: 5px 10px;
    border-radius: 5px;
}

.factor-impact.positive {
    color: #27ae60;
    background: rgba(39, 174, 96, 0.1);
}

.factor-impact.neutral {
    color: #95a5a6;
    background: rgba(149, 165, 166, 0.1);
}

.comparison-grid {
    display: grid;
    grid-template-columns: repeat(3, 1fr);
    gap: 20px;
}

.comparison-item {
    text-align: center;
    background: #f8f9fa;
    padding: 20px;
    border-radius: 10px;
}

.comparison-price {
    font-size: 2rem;
    font-weight: bold;
```

```
    color: #2c3e50;
    margin-bottom: 10px;
}

.comparison-label {
    color: #6c757d;       font-size: 0.9rem;
}

.action-buttons {
    display: flex;
    gap: 15px;
    justify-content: center;
    margin-top: 40px;
}

.btn {
    padding: 12px 24px;
    border-radius: 8px;
    text-decoration: none;
    font-weight: 600;
    transition: all 0.3s ease;
    border: none;
    cursor: pointer;
    font-size: 1rem;
}

.btn-primary {
    background: #3498db;
    color: white;
}

.btn-primary:hover {
    background: #2980b9;
    transform: translateY(-2px);
}

.btn-secondary {
    background: #95a5a6;
    color: white;
}

.btn-secondary:hover {
    background: #7f8c8d;
    transform: translateY(-2px);
}

.btn-danger {
    background: #e74c3c;
    color: white;
}

.btn-danger:hover {
```

```
            background: #c0392b;
            transform: translateY(-2px);
        }

        @media (max-width: 768px) {
            .grid {           grid-template-columns: 1fr;
            }

            .details-grid {
                grid-template-columns: 1fr;
            }

            .comparison-grid {
                grid-template-columns: 1fr;
            }

            .action-buttons {
                flex-direction: column;
            }
        }
    </style>
</head>
<body>
    <div class="container">
        <div class="header">
            <h1>📊 Детальний звіт оцінки нерухомості</h1>
            <p>Комплексний аналіз вартості та характеристик</p>
        </div>

        <div class="content">
            <!-- Основна ціна -->
            <div class="price-section">
                <div class="main-price">
                    {{
'{:,}'.format(data.main_price).replace(',', ' ') }} грн
                </div>
                <div class="price-per-meter">
                    ~{{
'{:,}'.format(data.price_per_meter).replace(',', ' ') }} грн/м²
                </div>
                <div class="confidence">Рівень довіри: {{
data.confidence }}%</div>
            </div>

            <!-- Основні деталі -->
            <div class="property-details">
                <h3>🏠 Характеристики нерухомості</h3>
                <div class="details-grid">
                    <div class="detail-section">
                        <div class="detail-item">
                            <span class="detail-
label">Місто:</span>
```

```
                            <span class="detail-value">{{
data.property.city }}</span>
                        </div>
                        <div class="detail-item">
                            <span class="detail-
label">Район:</span>
                            <span class="detail-value">{{
data.property.district }}</span>
                        </div>
                        <div class="detail-item">
                            <span class="detail-
label">Вулиця:</span>
                            <span class="detail-value">{{
data.property.street }}</span>
                        </div>
                        <div class="detail-item">
                            <span class="detail-label">Загальна
площа:</span>
                            <span class="detail-value">{{
data.property.total_area }} м²</span>
                        </div>
                        <div class="detail-item">
                            <span class="detail-
label">Кімнат:</span>
                            <span class="detail-value">{{
data.property.rooms }}</span>
                        </div>
                    </div>

                    <div class="detail-section">
                        <div class="detail-item">
                            <span class="detail-
label">Поверх:</span>
                            <span class="detail-value">{{
data.property.floor }} з {{ data.property.total_floors }}</span>
                        </div>
                        <div class="detail-item">
                            <span class="detail-label">Рік
побудови:</span>
                            <span class="detail-value">{{
data.property.year_built }}</span>
                        </div>
                        <div class="detail-item">
                            <span class="detail-label">Тип
будівництва:</span>
                            <span class="detail-value">{{
data.property.construction_type }}</span>
                        </div>
                        <div class="detail-item">
                            <span class="detail-
label">Орієнтація вікон:</span>
                            <span class="detail-value">{{
```

```
data.property.window_orientation }}</span>
                        </div>
                        <div class="detail-item">
                            <span class="detail-
label">Енергетичний сертифікат:</span>
                            <span class="detail-value">{{
data.property.energy_certificate }}</span>
                        </div>
                    </div>
                </div>
            </div>

            <!-- Діаграми -->
            <div class="grid">
                <div class="card">
                    <h3>📈 Фактори впливу на ціну</h3>
                    <div class="chart-container">
                        <canvas id="factorsChart"></canvas>
                    </div>
                </div>

                <div class="card">
                    <h3>⭐ Оцінка району</h3>
                    <div class="chart-container">
                        <canvas id="districtChart"></canvas>
                    </div>
                </div>
            </div>

            <div class="grid">
                <div class="card">
                    <h3>🥧 Структура ціни</h3>
                    <div class="chart-container">
                        <canvas
id="priceStructureChart"></canvas>
                    </div>
                </div>

                <div class="card">
                    <h3>📊 Порівняння з ринком</h3>
                    <div class="chart-container">
                        <canvas
id="marketComparisonChart"></canvas>
                    </div>
                </div>
            </div>

            <!-- Аналіз факторів -->
            <div class="comparison-section">
                <h3>🔍 Аналіз факторів впливу</h3>
                <div class="factors-impact">
```

```
                    {% for factor in data.factor_impacts %}
                    <div class="factor-item">
                        <span class="factor-name">{{ factor.name
}}</span>
                        <span class="factor-impact {{
factor.type }}">{{ factor.impact }}</span>
                    </div>                {% endfor %}
                </div>
            </div>

            <!-- Порівняння з ринком -->
            <div class="comparison-section">
                <h3>💰 Порівняння з середніми цінами</h3>
                <div class="comparison-grid">
                    <div class="comparison-item">
                        <div class="comparison-price">{{
'{:,}'.format(data.market_comparison.district_average).replace('
,', ' ') }} грн/м²</div>
                        <div class="comparison-label">Середня
ціна в районі</div>
                    </div>
                    <div class="comparison-item">
                        <div class="comparison-price">{{
'{:,}'.format(data.market_comparison.city_average).replace(',',
' ') }} грн/м²</div>
                        <div class="comparison-label">Середня
ціна в місті</div>
                    </div>
                    <div class="comparison-item">
                        <div class="comparison-price">{{
"%.2f"|format(data.market_comparison.your_property /
data.market_comparison.city_average) }}</div>
                        <div class="comparison-label">Індекс
привабливості</div>
                    </div>
                </div>
            </div>

            <!-- Кнопки дій -->
            <div class="action-buttons">
                <a href="{{ url_for('index') }}" class="btn btn-
primary">
                    🔄 Нова оцінка
                </a>
                <a href="#" class="btn btn-secondary">
                    📋 Створити оголошення
                </a>
                <button onclick="window.print()" class="btn btn-
danger">
                    🖨️ Роздрукувати звіт
                </button>
            </div>
```

```
        </div>
    </div>

    <script>
        // Дані з Flask
        const chartData = {{ data | tojson }};        // Фактори
впливу на ціну (горизонтальний стовпчик)
        const factorsCtx =
document.getElementById('factorsChart').getContext('2d');
        new Chart(factorsCtx, {
            type: 'bar',
            data: {
                labels: ['Місцезнаходження', 'Рік побудови',
'Площа', 'Поверх', 'Інфраструктура'],
                datasets: [{
                    label: 'Вплив на ціну (%)',
                    data: [
                        chartData.factors.location,
                        chartData.factors.year_built,
                        chartData.factors.area,
                        chartData.factors.floor,
                        chartData.factors.infrastructure
                    ],
                    backgroundColor: [
                        '#3498db',
                        '#2ecc71',
                        '#f39c12',
                        '#9b59b6',
                        '#e74c3c'
                    ],
                    borderRadius: 5
                }]
            },
            options: {
                indexAxis: 'y',
                responsive: true,
                maintainAspectRatio: false,
                plugins: {
                    legend: {
                        display: false
                    }
                },
                scales: {
                    x: {
                        beginAtZero: true,
                        max: 20,
                        ticks: {
                            callback: function(value) {
                                return value + '%';
                            }
                        }
                    }
```

```
                }
            }
        });

        // Оцінка району (радарна діаграма)
        const districtCtx =
document.getElementById('districtChart').getContext('2d');
        new Chart(districtCtx, {
            type: 'radar',
            data: {
                labels: ['Транспорт', 'Безпека',
'Інфраструктура', 'Екологія', 'Послуги'],
                datasets: [{
                    label: 'Оцінка району',
                    data: [
                        chartData.district_scores.transport,
                        chartData.district_scores.safety,

chartData.district_scores.infrastructure,
                        chartData.district_scores.ecology,
                        chartData.district_scores.services
                    ],
                    backgroundColor: 'rgba(52, 152, 219, 0.2)',
                    borderColor: '#3498db',
                    borderWidth: 2,
                    pointBackgroundColor: '#3498db',
                    pointRadius: 5
                }]
            },
            options: {
                responsive: true,
                maintainAspectRatio: false,
                scales: {
                    r: {
                        beginAtZero: true,
                        max: 10,
                        ticks: {
                            stepSize: 2
                        }
                    }
                },
                plugins: {
                    legend: {
                        display: false
                    }
                }
            }
        });

        // Структура ціни (кругова діаграма)
        const priceStructureCtx =
document.getElementById('priceStructureChart').getContext('2d');
```

```
        new Chart(priceStructureCtx, {
            type: 'doughnut',
            data: {
                labels: ['Базова вартість', 'Місцезнаходження',
'Характеристики', 'Стан'],
                datasets: [{
                    data: [
                        chartData.price_structure.base_cost,
                       chartData.price_structure.location,

chartData.price_structure.characteristics,
                        chartData.price_structure.condition
                    ],
                    backgroundColor: [
                        '#3498db',
                        '#2ecc71',
                        '#f39c12',
                        '#e74c3c'
                    ],
                    borderWidth: 2,
                    borderColor: '#fff'
                }]
            },
            options: {
                responsive: true,
                maintainAspectRatio: false,
                plugins: {
                    legend: {
                        position: 'bottom',
                        labels: {
                            padding: 15,
                            usePointStyle: true
                        }
                    },
                    tooltip: {
                        callbacks: {
                            label: function(context) {
                                return context.label + ': ' +
context.parsed + '%';
                            }
                        }
                    }
                }
            }
        });

        // Порівняння з ринком (стовпчикова діаграма)
        const marketCtx =
document.getElementById('marketComparisonChart').getContext('2d'
);
        new Chart(marketCtx, {
            type: 'bar',
```

```
            data: {
                labels: ['Ваша нерухомість', 'Середня в районі',
'Середня в місті'],
                datasets: [{
                    label: 'Ціна за м² (грн)',
                    data: [

chartData.market_comparison.your_property,

chartData.market_comparison.district_average,
                        chartData.market_comparison.city_average
                    ],
                    backgroundColor: [
                        '#27ae60',
                        '#3498db',
                        '#95a5a6'
                    ],
                    borderRadius: 5,
                    borderSkipped: false
                }]
            },
            options: {
                responsive: true,
                maintainAspectRatio: false,
                plugins: {
                    legend: {
                        display: false
                    },
                    tooltip: {
                        callbacks: {
                            label: function(context) {
                                return
context.parsed.y.toLocaleString() + ' грн/м²';
                            }
                        }
                    }
                },
                scales: {
                    y: {
                        beginAtZero: true,
                        ticks: {
                            callback: function(value) {
                                return value.toLocaleString() +
' грн';
                            }
                        }
                    }
                }
            }
        });
    </script>
```