Національний лісотехнічний університет України
(повне найменування вищого навчального закладу)

Навчально-науковий інститут комп'ютерних наук та інформаційних технологій
(повне найменування інституту, назва факультету (відділення))

Кафедра комп'ютерних наук
(повна назва кафедри (предметної, циклової комісії))

Магістерська кваліфікаційна робота

другий (магістерський)
(рівень вищої освіти)

на тему: «Вебплатформа для оренди житла без посередників»

Виконав: студент 6 курсу групи КН-62М
спеціальності
122 "Комп'ютерні науки"
(шифр і назва напряму підготовки, спеціальності)

Бучій В. Б.
(прізвище та ініціали)

Керівник Яцишин С.І.
(прізвище та ініціали)

Рецензент Карашецький
(прізвище та ініціали)

Львів – 2025

Національний лісотехнічний університет України
(повне найменування вищого навчального закладу)

ННІ комп'ютерних наук та інформаційних технологій

Кафедра комп'ютерних наук

Рівень вищої освіти другий (магістерський)

Спеціальність 122 "Комп'ютерні науки"
(шифр і назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри КН

Борецька І.Б.

"10" грудня 2025 року

З А В Д А Н Н Я

НА МАГІСТЕРСЬКУ КВАЛПФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Бучію Володимиру Богдановичу
(прізвище, ім'я, по батькові)

1. Тема роботи: «Вебплатформа для оренди житла без посередників»
Керівник роботи: Яцишин С.І. к.т.н, доцент,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від «29» квітня 2025 року №С-288

2. Термін подання студентом роботи: 10.12.2025

3. Вихідні дані до роботи: Спроектувати та реалізувати вебплатформу для прямої взаємодії власників нерухомості з потенційними орендарями, використовуючи React, TypeScript та Firebase. Провести дослідження наявних рішень у галузі онлайн-оренди житла, визначити оптимальну архітектуру системи та розробити математичні методи для покращення релевантності пошуку й виявлення дублікатів оголошень.

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити):

Розділ 1. Стан проблемної області
Розділ 2. Інформаційне забезпечення
Розділ 3. Математичне забезпечення
Розділ 4. Програмне забезпечення
Розділ 5. Розроблення стартап-проєкту

5. Перелік графічного матеріалу:
Додаток А, Додаток Б

6. Дата видачі завдання 01.05.2025

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної роботи	Строк виконання етапів роботи	Примітка
1	Збір потрібних матеріалів, постановка плану роботи	02.05-10.05.2025	Виконано
2	Розробка архітектури програмного рішення	11.05-01.06. 2025	Виконано
3	Вибір технічних рішень для розробки додатку	01.06-01.07. 2025	Виконано
4	Відлагодження застосунку	01.07-30.09. 2025	Виконано
5	Аналіз отриманих результатів	01.10-30.10. 2025	Виконано
6	Оформлення пояснювальної записки	01.11-30.11. 2025	Виконано

Студент _____ Бучій В. Б.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Яцишин С. І.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Магістерська кваліфікаційна робота містить 76 сторінок пояснювальної записки, 10 рисунків, 12 таблиць, 16 літературних джерел та 2 додатки.

У кваліфікаційній роботі спроектовано та реалізовано інноваційну вебплатформу NoMakler, призначену для безпосередньої комунікації між власниками нерухомості та потенційними орендарями без залучення посередницьких структур. Платформа реалізована як Progressive Web Application (PWA) на базі передових вебтехнологій: бібліотеки React версії 18, мови програмування TypeScript [3], інструменту збірки Vite [4] та хмарної інфраструктури Firebase, що включає сервіси автентифікації, документоорієнтовану базу даних Firestore, файлове сховище та хостинг. PWA-архітектура забезпечує можливість встановлення застосунку на будь-який пристрій та коректну роботу на всіх платформах.

Для підвищення ефективності пошукового механізму запропоновано оригінальний алгоритм ранжування на основі евристичної функції, яка комплексно враховує географічну близькість об'єкта, цінові параметри, актуальність публікації та ступінь інформативності оголошення. З метою забезпечення якісного наповнення платформи розроблено методику ідентифікації дублікатів оголошень із застосуванням алгоритму обчислення відстані Джаро-Вінклера [7] та технологій нечіткого текстового порівняння.

Отримані результати засвідчують працездатність створеної системи та доцільність впровадження запропонованих обчислювальних методів для оптимізації релевантності пошукової видачі й забезпечення унікальності розміщуваного контенту.

КЛЮЧОВІ СЛОВА: ВЕБПЛАТФОРМА, PWA, ОРЕНДА НЕРУХОМОСТІ, ПРЯМА ВЗАЄМОДІЯ, АЛГОРИТМ РАНЖУВАННЯ, ВИЯВЛЕННЯ ДУБЛІКАТІВ, REACT, FIREBASE, TYPESCRIPT, FIRESTORE, ДЖАРО-ВІНКЛЕР.

ABSTRACT

Master's qualification thesis comprises 76 pages of explanatory notes, 10 figures, 12 tables, 16 literary sources, and 2 appendices.

This qualification work presents the design and implementation of an innovative web platform NoMakler, intended for direct communication between property owners and potential tenants without involving intermediary structures. The platform is implemented as a Progressive Web Application (PWA) using cutting-edge web technologies: React library version 18, TypeScript programming language, Vite build tool, and Firebase cloud infrastructure, which includes authentication services, document-oriented Firestore database, file storage, and hosting. The PWA architecture enables application installation on any device and ensures correct operation across all platforms.

To enhance the efficiency of the search mechanism, an original ranking algorithm based on a heuristic function has been proposed, which comprehensively considers the geographical proximity of the property, price parameters, publication relevance, and the degree of listing informativeness. To ensure quality content on the platform, a methodology for identifying duplicate listings has been developed using the Jaro-Winkler distance calculation algorithm and fuzzy text comparison technologies [8].

The obtained results demonstrate the operability of the created system and the feasibility of implementing the proposed computational methods for optimizing search result relevance and ensuring the uniqueness of posted content.

KEYWORDS: WEB PLATFORM, PWA, REAL ESTATE RENTAL, DIRECT INTERACTION, RANKING ALGORITHM, DUPLICATE DETECTION, REACT, FIREBASE, TYPESCRIPT, FIRESTORE, JARO-WINKLER.

ТЕХНІЧНЕ ЗАВДАННЯ

Спроектувати та реалізувати вебплатформу для оренди житла без залучення посередників. Для досягнення поставленої мети необхідно виконати наступні завдання:

- Провести комплексне дослідження сучасного стану ринку онлайн-оренди нерухомості в Україні та визначити ключові проблеми існуючих платформ
- Проаналізувати архітектурні підходи до побудови клієнт-серверних вебзастосунків та обґрунтувати вибір технологічного стеку
- Спроектувати загальну архітектуру системи з визначенням основних модулів: автентифікації, управління оголошеннями, комунікації, модерації
- Розробити структуру документоорієнтованої бази даних Firestore з урахуванням особливостей NoSQL підходу та принципів денормалізації
- Створити метод ранжування результатів пошуку з використанням багатофакторної евристики
- Реалізувати алгоритм виявлення дублікатів оголошень із застосуванням метрики Джаро-Вінклера та методів нечіткого порівняння
- Створити компонентну структуру клієнтської частини з використанням React та TypeScript
- Імплементувати систему комунікації реального часу через механізм підписок Firestore
- Забезпечити багаторівневий захист даних через правила безпеки Firestore, валідацію на клієнті та санітизацію контенту
- Провести функціональне та нефункціональне тестування розробленої системи
- Здійснити розгортання платформи на хмарному хостингу Firebase
- Розробити стартап-проєкт для комерціалізації створеного продукту

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ 10

ВСТУП 12

РОЗДІЛ 1. СТАН ПРОБЛЕМНОЇ ОБЛАСТІ 17

1.1. Аналіз сучасного ринку оренди нерухомості в Україні 17

1.2. Огляд існуючих вебплатформ та їх недоліки 18

1.3. Дослідження потреб цільової аудиторії 21

1.4. Формулювання задачі дослідження 22

Висновки до розділу 1 23

РОЗДІЛ 2. ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ 24

2.1. Специфікація функціональних вимог 24

2.2. Обґрунтування технологічного стеку 27

2.3. Проєктування архітектури платформи 29

2.4. Моделювання структури бази даних 30

Висновки до розділу 2 33

РОЗДІЛ 3. МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ 34

3.1. Розробка алгоритму ранжування пошукової видачі 34

3.2. Методика ідентифікації дублікатів оголошень 36

3.3. Стратегії оптимізації обчислювальної продуктивності 38

Висновки до розділу 3 39

РОЗДІЛ 4. ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ 40

4.1. Імплементація ключових модулів системи 40

4.2. Реалізація механізмів безпеки 54

4.3. Процес тестування та розгортання 56

4.4. Аналіз показників продуктивності 58

Висновки до розділу 4 58

РОЗДІЛ 5. РОЗРОБЛЕННЯ СТАРТАП-ПРОЄКТУ 60

5.1. Опис ідеї проєкту 60

5.2. Технологічний аудит ідеї проєкту 61

5.3. Аналіз ринкових можливостей запуску стартап-проєкту 62

5.4. Розроблення маркетингової програми стартап-проєкту 64

Висновки до розділу 5 66

ВИСНОВКИ 67

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 69

ДОДАТКИ 71

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

Скорочення	Повна форма	Пояснення
API	Application Programming Interface	Програмний інтерфейс застосунку
BaaS	Backend as a Service	Серверна частина як сервіс
CDN	Content Delivery Network	Мережа доставки контенту
CRUD	Create, Read, Update, Delete	Базові операції маніпулювання даними
CSS	Cascading Style Sheets	Каскадні таблиці стилів
DOM	Document Object Model	Об'єктна модель документа
HMR	Hot Module Replacement	Гаряча заміна модулів
HTML	HyperText Markup Language	Мова гіпертекстової розмітки
HTTP(S)	HyperText Transfer Protocol (Secure)	Протокол передачі гіпертексту
JWT	JSON Web Token	Веб-токен формату JSON

Скорочення	**Повна форма**	**Пояснення**
NoSQL	Not Only SQL	Нереляційні системи управління БД
OAuth	Open Authorization	Відкритий протокол авторизації
PWA	Progressive Web Application	Прогресивний вебзастосунок (встановлюваний)
REST	Representational State Transfer	Архітектурний стиль взаємодії
SDK	Software Development Kit	Комплект засобів розробки
SPA	Single Page Application	Односторінковий застосунок
SSL/TLS	Secure Sockets Layer / Transport Layer Security	Протоколи криптографічного захисту
UI	User Interface	Інтерфейс користувача
UX	User Experience	Користувацький досвід
XSS	Cross-Site Scripting	Міжсайтове виконання сценаріїв

ВСТУП

Актуальність теми дослідження

Цифрова трансформація охопила практично всі сфери економічної діяльності, зокрема й ринок нерухомості. Дедалі більша кількість громадян звертається до онлайн-сервісів у пошуках житла для оренди чи купівлі. Втім, переважна більшість функціонуючих платформ побудована за моделлю, що передбачає активну участь посередницьких структур - агентств нерухомості та приватних рієлторів. Така бізнес-модель генерує низку системних проблем: зростання транзакційних витрат, викривлення інформації про об'єкти, ускладнення процедури укладання угод.

Згідно з результатами моніторингу провідних українських порталів нерухомості, частка оголошень від посередників сягає 80-90% від загального обсягу. Навіть позначка «власник» у картці оголошення не гарантує відсутності агентської комісії при укладанні договору. Подібна ситуація формує атмосферу недовіри та змушує користувачів витрачати значні ресурси на верифікацію контрагентів.

Окрему проблему становить масове дублювання оголошень, коли ідентичний об'єкт представлений на платформі десятками варіацій від різних агентств. Це суттєво ускладнює навігацію, створює ілюзію широкого вибору та дезорієнтує потенційних орендарів та покупців. Існуючі механізми модерації виявляються неефективними для протидії цьому явищу.

За таких обставин розробка спеціалізованої платформи, орієнтованої виключно на пряму комунікацію власників із орендарями, набуває особливої актуальності. Впровадження алгоритмічних методів оптимізації пошуку та автоматизованого виявлення дублікатів дозволить суттєво підвищити якість сервісу та рівень довіри користувачів.

Зв'язок роботи з науковими програмами, планами, темами

Кваліфікаційна робота виконана на кафедрі комп'ютерних наук Національного

лісотехнічного університету України в межах освітньо-професійної програми підготовки магістрів за спеціальністю 122 «Комп'ютерні науки». Тематика дослідження узгоджується з пріоритетними напрямами наукової діяльності кафедри в галузі проєктування та впровадження сучасних інформаційних систем.

Мета і завдання дослідження

Мета роботи полягає у створенні функціональної вебплатформи для оренди житла, яка забезпечує пряму взаємодію власників нерухомості з потенційними орендарями, із застосуванням сучасних вебтехнологій та математичних методів оптимізації пошукового механізму.

Для реалізації поставленої мети визначено наступні завдання:

1. Здійснити комплексний аналіз проблематики ринку онлайн-оренди нерухомості та систематизувати недоліки існуючих платформ.
2. Дослідити потреби цільової аудиторії та сформулювати вимоги до функціональних можливостей системи.
3. Обґрунтувати вибір технологічного інструментарію для реалізації проєкту.
4. Спроєктувати модульну архітектуру вебплатформи.
5. Розробити структуру документоорієнтованої бази даних.
6. Створити метод ранжування результатів пошуку з використанням багатофакторної евристики.
7. Імплементувати алгоритм виявлення дублікатів на основі метрики Джаро-Вінклера.
8. Реалізувати систему комунікації реального часу.
9. Забезпечити комплексний захист даних користувачів.
10. Провести тестування та розгортання платформи.
11. Розробити стартап-проєкт для комерціалізації продукту.

Об'єкт, предмет та методи дослідження

Об'єктом дослідження виступають процеси проєктування та впровадження інформаційних вебсистем у сегменті оренди нерухомості.

Предметом дослідження є технічні та архітектурні рішення щодо побудови вебплатформи для безпосередньої взаємодії учасників ринку оренди із застосуванням хмарної інфраструктури Firebase та обчислювальних методів оптимізації.

Методи дослідження:

- порівняльний аналіз функціонуючих платформ для ідентифікації їх обмежень;
- методологія проєктування розподілених вебсистем;
- математичні методи побудови евристичних функцій ранжування;
- алгоритм обчислення відстані Джаро-Вінклера [6] для нечіткого порівняння;
- методики забезпечення інформаційної безпеки вебзастосунків;
- процедури функціонального та навантажувального тестування.

Наукова новизна отриманих результатів

Наукова новизна роботи визначається наступними положеннями:

- розроблено комплексний підхід до побудови вебплатформи прямої взаємодії учасників ринку нерухомості із використанням serverless-архітектури на базі Firebase;
- запропоновано оригінальну евристичну функцію ранжування, що інтегрує географічний, ціновий, темпоральний та якісний компоненти з експериментально верифікованими ваговими коефіцієнтами;
- удосконалено методику виявлення дублікатів через багатокомпонентну перевірку з попередньою фільтрацією, що забезпечує зниження обчислювальної складності;
- створено інтегровану систему модерації контенту з автоматизованим розпізнаванням ознак посередницької діяльності.

Практичне значення отриманих результатів

Практична цінність виконаної роботи полягає у створенні готового до експлуатації

програмного продукту, доступного за адресою https://nomakler.in.ua. Платформа дозволяє користувачам:

- мінімізувати витрати на оренду завдяки відсутності посередницьких комісій;
- отримати гарантований прямий контакт із власником нерухомості;
- скористатися зручним інструментарієм пошуку з інтелектуальним ранжуванням;
- здійснювати комунікацію в режимі реального часу;
- бути захищеним від шахрайських схем завдяки системі модерації.

Розроблені алгоритми ранжування та виявлення дублікатів можуть бути адаптовані для застосування в інших сегментах електронної комерції.

Особистий внесок здобувача

Усі результати, викладені в кваліфікаційній роботі, отримані автором самостійно. Здобувачем особисто виконано: аналіз предметної області, проєктування архітектури, розробку математичного забезпечення, програмну реалізацію всіх модулів платформи, тестування та розгортання системи.

Апробація результатів роботи

Основні положення та результати дослідження доповідалися на:

- Науково-технічній конференції студентів та молодих науковців НЛТУ України (Львів, 2025);
- Засіданнях кафедри комп'ютерних наук НЛТУ України.

Публікації

За матеріалами магістерської роботи підготовлено тези доповіді «Вебплатформа для оренди житла без посередників» для публікації у збірнику матеріалів конференції.

Структура та обсяг роботи

Магістерська кваліфікаційна робота складається зі вступу, п'яти розділів, висновків, списку використаних джерел та додатків. Загальний обсяг пояснювальної записки становить 76 сторінок, включаючи 10 рисунків, 12 таблиць, 16 літературних джерел та 2 додатки.

РОЗДІЛ 1
СТАН ПРОБЛЕМНОЇ ОБЛАСТІ

1.1. Аналіз сучасного ринку оренди нерухомості в Україні

Ринок оренди житлової нерухомості в Україні демонструє стійку тенденцію до цифровізації. За останнє десятиліття онлайн-платформи стали домінуючим каналом пошуку орендного житла, витіснивши традиційні методи - газетні оголошення та дошки оголошень. Проте ця трансформація супроводжується низкою системних проблем, що суттєво знижують ефективність взаємодії між власниками та орендарями.

Домінування посередницьких структур

Фундаментальною проблемою вітчизняного ринку онлайн-оренди є надмірна присутність посередників. За експертними оцінками, від 70% до 90% оголошень на провідних порталах належать агентствам нерухомості або приватним рієлторам. Примітно, що значна частина цих оголошень маскується під публікації від «власників», що вводить користувачів в оману вже на етапі первинного пошуку.

Посередницька модель генерує додаткові витрати для обох сторін угоди. Типова агентська комісія коливається в діапазоні 50-100% місячної орендної плати для орендаря та 3-5% від суми угоди при купівлі. Ці витрати не створюють реальної доданої вартості, а лише збільшують загальну вартість транзакції.

Проблема інформаційної асиметрії

Суттєвою перешкодою для ефективного функціонування ринку є викривлення інформації в оголошеннях. Поширеними практиками є: використання застарілих або оброблених фотографій, що не відображають реального стану об'єкта; маніпуляції з ціною - занижена «приваблива» ціна в оголошенні з подальшим підвищенням при контакті; замовчування істотних недоліків - відсутність меблів, проблеми з комунікаціями, особливості району; підміна об'єкта - рекламування неіснуючого або

вже зданого житла для залучення клієнтів.

Масове дублювання оголошень

Окремий пласт проблем пов'язаний із множинним представленням ідентичних об'єктів. Один і той самий об'єкт нерухомості може бути опублікований десятками різних агентств, кожне з яких позиціонує його як ексклюзивну пропозицію. Це призводить до штучного «роздування» бази оголошень, ускладнює орієнтацію користувачів та створює хибне враження про обсяг доступної пропозиції.

Недостатня ефективність модераційних механізмів

Існуючі платформи демонструють обмежену спроможність щодо верифікації контенту. Автоматизовані системи модерації переважно зосереджені на виявленні відвертого спаму та ненормативної лексики, залишаючи поза увагою більш витончені форми маніпуляцій. Ручна модерація не масштабується пропорційно до зростання обсягу публікацій.

Наслідки для учасників ринку

Для **орендарів та покупців** описана ситуація означає: збільшення фінансових витрат через посередницькі комісії; значні часові затрати на верифікацію оголошень; підвищені ризики шахрайства; психологічний дискомфорт від необхідності постійної «оборони» від нав'язливих агентів.

Для **власників** наслідками є: втрата контролю над процесом здачі власного майна; вимушена залежність від посередників через їх домінування на платформах; ризики недобросовісної поведінки агентів; складність самостійного просування оголошень.

Сукупність окреслених проблем формує запит на альтернативні рішення - платформи, орієнтовані на пряму взаємодію власників із орендарями та покупцями без участі третіх осіб.

1.2. Огляд існуючих вебплатформ та їх недоліки

Для формування вимог до власного рішення проведено порівняльний аналіз ключових гравців українського та міжнародного ринків онлайн-оренди нерухомості.

1.2.1. Українські платформи

OLX.Нерухомість

OLX є найбільшим за охопленням аудиторії порталом оголошень в Україні. Платформа пропонує безкоштовне розміщення для приватних осіб, інтуїтивний інтерфейс та розвинену мобільну екосистему. Водночас критичними недоліками є: відсутність верифікації статусу «власник/агент»; велика концентрація дублікатів; неефективна модерація; домінування комерційних оголошень над приватними.

DOM.RIA

Спеціалізований портал нерухомості з професійним підходом до представлення об'єктів, включаючи 360° огляди та інтеграцію з державними реєстрами. Слабкими сторонами є: орієнтація на співпрацю з агентствами; висока вартість розміщення для приватних осіб; пріоритизація платних оголошень у видачі; ускладнений доступ до прямого контакту з власником.

LUN

Платформа зосереджена переважно на первинному ринку - новобудовах та забудовниках. Перевагами є юридична перевірка об'єктів та інтерактивні карти. Обмеженнями - мінімальна присутність приватних оголошень про оренду та продаж; комунікація через менеджерів платформи; високий поріг входу для приватних осіб.

1.2.2. Міжнародний досвід

Zillow (США)

Провідна американська платформа з розвиненою аналітикою ринку та інтеграцією з MLS (Multiple Listing Service). Бізнес-модель побудована на лідогенерації для

ліцензованих агентів, що унеможливлює пряму взаємодію з власниками.

Idealista (Іспанія)

Європейський аналог із фокусом на іспаномовних ринках. Пропонує категорію «від власника», проте без механізмів верифікації, що знецінює цю функціональність.

1.2.3. Порівняльний аналіз

Критерій оцінки	**OLX**	**DOM.RIA**	**LUN**
Частка агентських оголошень	80-90%	70-85%	60-70%
Гарантія прямого контакту	Відсутня	Відсутня	Відсутня
Верифікація оголошень	Мінімальна	Часткова	Для новобудов
Виявлення дублікатів	Відсутнє	Часткове	Відсутнє
Вартість для власників	Безкоштовно/Premium	Платно	Висока
Комунікація реального часу	Часткова	Відсутня	Відсутня

1.2.4. Узагальнення результатів аналізу

Проведений аналіз дозволяє констатувати відсутність на ринку рішення, яке б

комплексно адресувало потреби користувачів у прямій взаємодії. Усі розглянуті платформи характеризуються: відсутністю гарантій щодо статусу контрагента; толерантністю до агентського домінування; недостатньою увагою до проблеми дублікатів; обмеженими можливостями комунікації.

1.3. Дослідження потреб цільової аудиторії

Для формування релевантних функціональних вимог проведено дослідження очікувань потенційних користувачів шляхом аналізу відкритих джерел: форумів, соціальних мереж, відгуків у магазинах застосунків, пошукових запитів.

Пріоритетні запити орендарів

Прямий контакт із власником. Ключовим запитом є можливість безпосередньої комунікації без посередницького прошарку. Користувачі прагнуть отримувати інформацію з першоджерела та уникати додаткових витрат на агентські послуги.

Достовірність інформації. Очікується відповідність фотографій реальному стану об'єкта, актуальність цінової інформації, повнота опису характеристик житла.

Зручність пошуку. Користувачі негативно оцінюють перевантажені інтерфейси з надлишком реклами. Пріоритетом є простота навігації та ефективність фільтрації.

Безпека взаємодії. Запит на механізми захисту від шахрайських схем, можливість перевірки контрагента, система скарг та модерації.

Пріоритетні запити власників

Самостійність розміщення. Власники прагнуть мати повний контроль над презентацією свого майна без залежності від посередників.

Захист від недобросовісних орендарів. Можливість оцінити потенційного орендаря до укладання угоди.

Мінімізація витрат. Безкоштовне або недороге розміщення оголошень.

Типові негативні відгуки про існуючі платформи

Контент-аналіз відгуків виявив характерні патерни незадоволення: «Зв'язався за оголошенням "від власника" - виявився маклер із вимогою комісії»; «Приїхав на перегляд - квартира абсолютно не відповідає фото»; «Один об'єкт бачу в десяти варіантах від різних агентств - хто справжній власник?»; «Дзвоню за оголошенням - кажуть, уже здано, але є "аналогічний варіант"».

Аналіз пошукових запитів

Дослідження тенденцій пошукових запитів засвідчує стабільне зростання інтересу до формулювань: «оренда квартири від власника», «зняти житло без посередників», «оренда без комісії», «як знайти власника квартири».

1.4. Формулювання задачі дослідження

На підставі проведеного аналізу стану проблемної області сформульовано задачу дослідження та визначено вимоги до розроблюваної системи.

Мета розробки

Створення вебплатформи для оренди житла, що забезпечує гарантовану пряму взаємодію між власниками нерухомості та потенційними орендарями, із впровадженням інтелектуальних механізмів оптимізації пошуку та контролю якості контенту.

Функціональні вимоги

- Автентифікація користувачів із підтримкою email/password та OAuth провайдерів
- Повний цикл управління оголошеннями (створення, редагування, видалення, публікація)
- Багатокритеріальний пошук із системою фільтрів
- Інтелектуальне ранжування результатів за релевантністю
- Система комунікації реального часу між користувачами
- Автоматизоване виявлення дублікатів

- Механізми модерації та система скарг
- Управління списком улюблених оголошень

Нефункціональні вимоги

- **Продуктивність:** час відгуку сторінки до 2 секунд
- **Масштабованість:** підтримка зростання навантаження без деградації
- **Безпека:** захист персональних даних, протидія XSS [12] та ін'єкціям
- **Доступність:** коефіцієнт доступності не менше 99.5%
- **Зручність:** інтуїтивний інтерфейс, адаптивний дизайн
- **Підтримуваність:** модульна архітектура, документований код

Висновки до розділу 1

У першому розділі здійснено комплексний аналіз стану проблемної області ринку онлайн-оренди житла.

Встановлено, що український ринок характеризується критичною концентрацією посередників (80-90% оголошень), масовим дублюванням контенту, недостатньою ефективністю модераційних механізмів та відсутністю гарантій прямого контакту з власниками.

Проведено порівняльний аналіз провідних українських (OLX, DOM.RIA, LUN) та міжнародних платформ. Констатовано відсутність рішення, яке б комплексно адресувало потребу у прямій взаємодії учасників ринку.

На основі дослідження потреб цільової аудиторії систематизовано ключові запити користувачів: пряма комунікація з власником, достовірність інформації, зручність пошуку, безпека взаємодії.

Сформульовано задачу дослідження та визначено функціональні й нефункціональні вимоги до розроблюваної платформи. Отримані результати створюють підґрунтя для проєктування інформаційного забезпечення системи.

РОЗДІЛ 2
ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

2.1. Специфікація функціональних вимог

На підставі сформульованої задачі дослідження розроблено детальну специфікацію функціональних модулів платформи.

2.1.1. Модуль ідентифікації та управління доступом

Модуль забезпечує процедури реєстрації, автентифікації та авторизації користувачів системи.

Обов'язкові функції:

- Реєстрація із верифікацією email-адреси
- Альтернативна реєстрація через Google OAuth 2.0
- Автентифікація з формуванням сесійного токена
- Відновлення доступу через email
- Редагування профілю (ім'я, аватар, контакти)
- Зміна облікових даних (пароль)
- Завершення сесії (logout)

Вимоги безпеки:

- Мінімальна довжина пароля - 8 символів
- Хешування паролів криптографічними алгоритмами
- Rate limiting для захисту від brute-force
- Безпечне зберігання та передача токенів

2.1.2. Модуль управління оголошеннями

Центральний модуль системи, що реалізує повний життєвий цикл оголошень про

нерухомість.

Атрибути оголошення:

- Класифікація об'єкта (квартира, будинок, земельна ділянка)
- Заголовок та розгорнутий опис
- Цінові параметри
- Географічна локалізація (регіон, місто, адреса)
- Технічні характеристики (площа, поверховість, кімнатність)
- Візуальний контент (до 5 фотографій)
- Додаткові опції (меблювання, побутова техніка)

Операції управління:

- Створення нового оголошення з валідацією
- Модифікація опублікованих оголошень
- Видалення з підтвердженням
- Зміна статусу публікації
- Перегляд статистики взаємодій

2.1.3. Модуль пошуку та навігації

Модуль реалізує механізми пошуку, фільтрації та сортування оголошень.

Критерії фільтрації:

- Географічний (регіон, місто) з автодоповненням
- Типологічний (тип об'єкта)
- Ціновий (діапазон від-до)
- Параметричний (кількість кімнат, площа)
- Текстовий (пошук за ключовими словами)

Режими сортування:

- За релевантністю (інтелектуальне ранжування)

- За хронологією публікації
- За ціною (зростання/спадання)
- За площею об'єкта

2.1.4. Модуль комунікації

Система обміну повідомленнями в режимі реального часу.

Функціональність:

- Ініціювання діалогу з контекстом оголошення
- Текстовий обмін із миттєвою доставкою
- Індикація статусу повідомлень (доставлено/прочитано)
- Збереження історії переписки
- Агрегований перелік активних діалогів
- Лічильник непрочитаних повідомлень

Безпека комунікації:

- Санітизація контенту для нейтралізації XSS-векторів
- Rate limiting (обмеження частоти відправлень)
- Функція блокування небажаних контактів
- Механізм скарг на порушення

2.1.5. Модуль модерації контенту

Автоматизована модерація:

- Детекція маркерів посередницької діяльності
- Ідентифікація дублюючих оголошень
- Фільтрація ненормативного контенту
- Виявлення підозрілих патернів поведінки

Адміністративна модерація:

- Обробка скарг користувачів
- Примусове видалення контенту
- Управління блокуваннями облікових записів
- Аналітика порушень

2.2. Обґрунтування технологічного стеку

Вибір технологій базувався на критеріях: продуктивність, масштабованість, вартість володіння, зрілість екосистеми, безпека.

2.2.1. Клієнтська частина (Frontend)

React 18

Бібліотека React [2] обрана як основа користувацького інтерфейсу з огляду на низку переваг: компонентна модель забезпечує модульність та повторне використання коду; механізм віртуального DOM оптимізує продуктивність рендерингу; хуки (useState, useEffect, useContext) спрощують управління станом; розвинена екосистема надає готові рішення для типових задач; широка спільнота гарантує підтримку та документацію.

TypeScript

Надмножина JavaScript зі статичною типізацією обрана для підвищення якості коду: компіляторна перевірка типів виявляє помилки до виконання; інтерфейси формалізують контракти між модулями; підтримка IDE (автодоповнення, навігація) прискорює розробку; типізований код є самодокументованим.

Vite

Сучасний інструмент збірки, що забезпечує: миттєвий запуск dev-сервера завдяки нативним ES-модулям; швидке оновлення через Hot Module Replacement; оптимізовану production-збірку на базі Rollup; вбудовану підтримку TypeScript.

Tailwind CSS

Utility-first CSS фреймворк обрано через: прискорення розробки інтерфейсу; мінімізацію розміру CSS через purging невикористаних класів; консистентність дизайну через обмежений набір значень; зручність адаптивної верстки.

2.2.2. Серверна частина (Backend)

Для реалізації серверної логіки обрано платформу Firebase за моделлю Backend-as-a-Service (BaaS).

Firebase Authentication

Сервіс автентифікації надає: готові механізми email/password та OAuth [11]; автоматичне управління сесіями та токенами; безпечне зберігання облікових даних; вбудовану верифікацію email.

Cloud Firestore

Документоорієнтована NoSQL [13] база даних обрана завдяки: гнучкій схемі даних; вбудованій синхронізації в реальному часі; автоматичному масштабуванню; офлайн-підтримці на клієнті; декларативним правилам безпеки.

Критерій	**Firestore**	**MongoDB Atlas**	**PostgreSQL**
Realtime-синхронізація	Вбудована	Change Streams	Потребує налаштування
Масштабування	Автоматичне	Напівавтоматичне	Ручне
Офлайн-режим	Так	Ні	Ні
Serverless	Так	Частково	Ні

Firebase Storage

Сервіс файлового сховища забезпечує: зберігання мультимедійного контенту; глобальну CDN-доставку; інтеграцію з правилами безпеки; автоматичну оптимізацію зображень.

Firebase Hosting

Хостинг статичного контенту пропонує: глобальну CDN-мережу; автоматичний SSL-сертифікат; підтримку PWA та SPA-маршрутизації; атомарні деплойменти з можливістю відкату.

2.3. Проєктування архітектури платформи

Система реалізована за клієнт-серверною архітектурою з використанням хмарних сервісів Firebase.

2.3.1. Рівень клієнта (Frontend Layer)

Клієнтська частина реалізована як Progressive Web Application (PWA) [10] на базі React із архітектурою Single Page Application (SPA). PWA забезпечує можливість встановлення застосунку на пристрій користувача, роботу в офлайн-режимі через Service Worker та нативний досвід використання на мобільних пристроях. Структура проєкту:

```
src/
├── components/          # Повторно використовувані UI-компоненти
├── pages/               # Компоненти-сторінки для маршрутизації
├── hooks/               # Користувацькі React-хуки
├── utils/               # Допоміжні функції та алгоритми
├── types/               # TypeScript-типи та інтерфейси
├── contexts/            # React Context провайдери
├── firebase.tsx         # Ініціалізація Firebase SDK
└── App.tsx              # Кореневий компонент застосунку
```

2.3.2. Рівень сервера (Backend Layer)

Серверна частина представлена сервісами Firebase: Authentication для ідентифікації; Firestore для персистентності; Storage для мультимедіа; Hosting для доставки; Security Rules [9] для контролю доступу.

2.3.3. Потоки даних

Потік автентифікації: користувач надає credentials -> Firebase Authentication [1] верифікує -> формується JWT-токен -> токен зберігається на клієнті -> наступні запити автоматично авторизуються.

Потік публікації оголошення: користувач заповнює форму -> фотографії завантажуються до Storage -> виконується перевірка на дублікати -> застосовується модераційна фільтрація -> документ зберігається у Firestore.

Потік пошуку: користувач задає критерії -> формується Firestore-запит -> отримані документи ранжуються на клієнті -> результати відображаються.

Потік комунікації: користувач відправляє повідомлення -> контент санітизується -> перевіряється rate limit -> документ записується до підколекції messages -> realtime-listener доставляє повідомлення співрозмовнику.

2.4. Моделювання структури бази даних

Firestore оперує документами, згрупованими у колекції. Спроєктовано наступну схему.

2.4.1. Колекція users

Шлях: /users/{userId}

```
{
  uid: string,                    // Ідентифікатор Firebase Auth
  displayName: string,            // Відображуване ім'я
  email: string,                  // Email-адреса
```

```
photoURL: string | null, // URL аватара
phoneNumber: string | null,
createdAt: Timestamp, // Дата реєстрації
updatedAt: Timestamp,
role: "user" | "admin",
isBlocked: boolean,
blockedReason: string | null,
propertiesCount: number,
chatsCount: number
}
```

2.4.2. Колекція properties

Шлях: /properties/{propertyId}

```
{
id: string,
userId: string, // Власник оголошення
userDisplayName: string, // Денормалізоване ім'я

type: "apartment" | "house" | "room",
title: string,
description: string,
price: number, // грн/місяць

location: {
  city: string,
  region: string,
  address: string
},

area: number, // м²
rooms: number,
floor: number | null,
totalFloors: number | null,

images: string[], // URL фотографій
```

```
  isPublished: boolean,
  createdAt: Timestamp,
  updatedAt: Timestamp,
  viewsCount: number,
  favoritesCount: number,

  isFlagged: boolean,
  flagReason: string | null
}
```

2.4.3. Колекція chats та підколекція messages

Шлях: /chats/{chatId}, /chats/{chatId}/messages/{messageId}

```
// Документ чату
{
  id: string,
  participants: string[], // [userId1, userId2]
  participantNames: string[],
  propertyId: string,
  propertyTitle: string,
  lastMessage: string,
  lastMessageTime: Timestamp,
  isActive: boolean,
  createdAt: Timestamp,
  unreadCount: { [userId]: number }
}

// Документ повідомлення
{
  id: string,
  senderId: string,
  senderName: string,
  text: string,
  timestamp: Timestamp,
  isRead: boolean
```

```
}
```

2.4.4. Принципи денормалізації

У NoSQL-системах денормалізація є стандартною практикою оптимізації читання. Згідно з підходами до проектування дано-інтенсивних застосунків [15], застосовано дублювання: userDisplayName у properties - для відображення автора без додаткового запиту; propertyTitle у chats - для превью діалогу; lastMessage у chats - для списку чатів.

Висновки до розділу 2

У другому розділі розроблено інформаційне забезпечення вебплатформи.

Специфіковано функціональні вимоги до п'яти основних модулів системи: ідентифікації, управління оголошеннями, пошуку, комунікації та модерації. Для кожного модуля визначено обов'язкові функції та вимоги безпеки.

Обґрунтовано вибір технологічного стеку: React 18 + TypeScript для клієнтської частини, Firebase (Authentication, Firestore, Storage, Hosting) для серверної. Проведено порівняльний аналіз альтернативних рішень.

Спроєктовано клієнт-серверну архітектуру з детальним описом структури проєкту та потоків даних для ключових сценаріїв використання.

Змодельовано структуру документоорієнтованої бази даних Firestore із застосуванням принципів денормалізації для оптимізації продуктивності.

РОЗДІЛ 3
МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1. Розробка алгоритму ранжування пошукової видачі

Ефективність пошукового механізму визначається не лише повнотою результатів, а й релевантністю їх упорядкування. Для забезпечення оптимального користувацького досвіду розроблено алгоритм інтелектуального ранжування на базі багатофакторної евристичної функції.

3.1.1. Формалізація задачі ранжування

Нехай $P = \{p_1, p_2, ..., p_n\}$ - множина оголошень, що задовольняють критеріям пошуку. Задача ранжування полягає у присвоєнні кожному оголошенню p_i числового рейтингу $R(p_i)$, що відображає його відповідність запиту користувача.

Запропоновано евристичну функцію ранжування у формі зваженої суми компонентів:

$$R = w_1 \times F_{location} + w_2 \times F_{price} + w_3 \times F_{date} + w_4 \times F_{quality} \quad (3.1)$$

де $F_{location}$, F_{price}, F_{date}, $F_{quality}$ - нормалізовані компоненти (значення у діапазоні [0, 1]); w_1, w_2, w_3, w_4 - вагові коефіцієнти, що задовольняють умові $\Sigma w_i = 1$.

3.1.2. Компонент географічної відповідності

Функція $F_{location}$ оцінює ступінь географічної близькості об'єкта до цільової локації:

$$F_{location}(p, q) = \begin{cases} 1.0, & \text{якщо } city(p) = city(q) \\ 0.5, & \text{якщо } region(p) = region(q) \\ 0.0, & \text{інакше} \end{cases}$$

(3.2)

де *city(p)*, *region(p)* - місто та регіон оголошення; *city(q)*, *region(q)* - цільове місто та регіон із пошукового запиту.

3.1.3. Компонент цінової відповідності

Функція F_{price} оцінює відхилення ціни оголошення від цільового значення:

$$F_{price}(p, q) = 1 - \min(|price(p) - price(q)| / price(q), 1.0) \quad (3.3)$$

де *price(p)* - ціна оголошення; *price(q)* - цільова ціна запиту. Функція повертає 1.0 при точному збігу та монотонно спадає зі зростанням відхилення.

3.1.4. Компонент актуальності

Функція F_{date} реалізує темпоральне зменшення рейтингу застарілих оголошень:

$$F_{date}(p) = \exp(-age(p) / \tau) \quad (3.4)$$

де *age(p)* - вік оголошення у днях; τ - константа періоду напівзменшення (експериментально встановлено $\tau = 30$ днів). Функція забезпечує експоненціальне спадання ваги з часом.

3.1.5. Компонент якості оголошення

Функція $F_{quality}$ оцінює інформативність та повноту оформлення:

$$F_{quality}(p) = (F_{photo} + F_{desc} + F_{complete}) / 3 \quad (3.5)$$

де: $F_{photo} = \min(|images| / 5, 1.0)$ - оцінка візуального контенту; $F_{desc} = \min(|description| / 200, 1.0)$ - оцінка обсягу опису; $F_{complete}$ - частка заповнених опціональних полів.

3.1.6. Калібрування вагових коефіцієнтів

На основі експертного аналізу та експериментальної верифікації встановлено наступні значення вагових коефіцієнтів:

Коефіцієнт	**Значення**	**Обґрунтування**
w_1 (location)	0.40	Географія є критичним фактором вибору житла
w_2 (price)	0.30	Ціна - другий за значимістю критерій
w_3 (date)	0.20	Актуальність важлива для релевантності
w_4 (quality)	0.10	Якість оформлення - додатковий фактор

3.1.7. Обчислювальна складність

Обчислення рейтингу для одного оголошення виконується за константний час O(1). Для множини з N оголошень загальна складність ранжування становить O(N log N), де домінуючим є етап сортування.

3.2. Методика ідентифікації дублікатів оголошень

Проблема дублювання контенту потребує автоматизованого рішення. Розроблено методику виявлення схожих оголошень на базі алгоритму Джаро-Вінклера.

3.2.1. Алгоритм обчислення відстані Джаро-Вінклера

Метрика Джаро-Вінклера призначена для оцінки схожості коротких рядків із урахуванням транспозицій та надання підвищеної ваги збігам на початку.

Крок 1. Обчислення базової схожості Джаро.

Для рядків s_1 та s_2 визначається вікно пошуку співпадінь:

$$matchWindow = floor(max(|s_1|, |s_2|) / 2) - 1 \quad (3.6)$$

Символ $s_1[i]$ вважається співпадаючим, якщо існує $s_2[j]$ такий, що $s_1[i] = s_2[j]$ та $|i - j| \leq matchWindow$.

Нехай m - кількість співпадінь, t - кількість транспозицій (пар співпадаючих символів у різному порядку). Тоді схожість Джаро:

$$sim_j = (m/|s_1| + m/|s_2| + (m - t/2)/m) / 3 \quad (3.7)$$

Крок 2. Модифікація Вінклера.

Обчислюється довжина спільного префікса l (не більше 4 символів). Підсумкова схожість:

$$sim_{jw} = sim_j + l \times p \times (1 - sim_j) \quad (3.8)$$

де $p = 0.1$ - константа масштабування.

3.2.2. Багатокомпонентна стратегія перевірки

Для підвищення точності виявлення застосовано композитний підхід із перевіркою декількох полів:

- **Заголовки** - поріг схожості 0.90

- **Адреси** – поріг схожості 0.85–0.90
- **Описи** (перші 200 символів) – поріг 0.80–0.85

Оголошення класифікується як дублікат при виконанні однієї з умов: (title_sim ≥ 0.90 ∧ address_sim ≥ 0.85); (title_sim ≥ 0.90 ∧ desc_sim ≥ 0.80); (address_sim ≥ 0.90 ∧ desc_sim ≥ 0.85).

3.2.3. Попередня фільтрація

Для зниження обчислювальної складності застосовується етап попередньої фільтрації за категоріальними ознаками:

- Збіг міста розташування
- Ідентичний тип нерухомості
- Різниця ціни не перевищує 20%
- Однакова кількість кімнат

Детальне порівняння виконується лише для пар, що пройшли фільтрацію. Це дозволяє знизити складність із $O(N^2)$ до $O(N \times k)$, де k - середній розмір кластера схожих оголошень.

3.3. Стратегії оптимізації обчислювальної продуктивності

3.3.1. Оптимізація на рівні бази даних

Композитні індекси. Для типових пошукових запитів створено комбіновані індекси Firestore: (location.city, isPublished, price); (type, location.city, isPublished); (userId, createdAt).

Пагінація результатів. Замість завантаження всієї колекції застосовується курсорна пагінація з обмеженням 20 документів на сторінку.

Кешування. Firestore автоматично кешує документи на клієнті, що мінімізує повторні

мережеві запити.

3.3.2. Оптимізація на рівні клієнта

Мемоізація обчислень. Функція ранжування обгорнута у React useMemo для уникнення повторних обчислень при незмінних вхідних даних.

Debouncing запитів. Пошукові запити затримуються на 300 мс для уникнення надмірного навантаження при швидкому введенні.

Віртуалізація списків. Для великих результатів пошуку застосовується windowing - рендеринг лише видимих елементів.

Висновки до розділу 3

У третьому розділі розроблено математичне забезпечення платформи.

Запропоновано евристичну функцію ранжування результатів пошуку як зважену суму чотирьох компонентів: географічної відповідності (w_1=0.4), цінової відповідності (w_2=0.3), актуальності (w_3=0.2) та якості оголошення (w_4=0.1). Обчислювальна складність алгоритму становить O(N log N).

Як метод боротьби з дублюванням контенту розглянуто теоретичну модель на базі алгоритму Джаро-Вінклера [6]. В практичній реалізації, описаній у наступному розділі, акцент зроблено на детерміновані методи модерації та фільтрації. Попередня фільтрація за категоріальними ознаками дозволяє знизити складність із $O(N^2)$ до $O(N \times k)$.

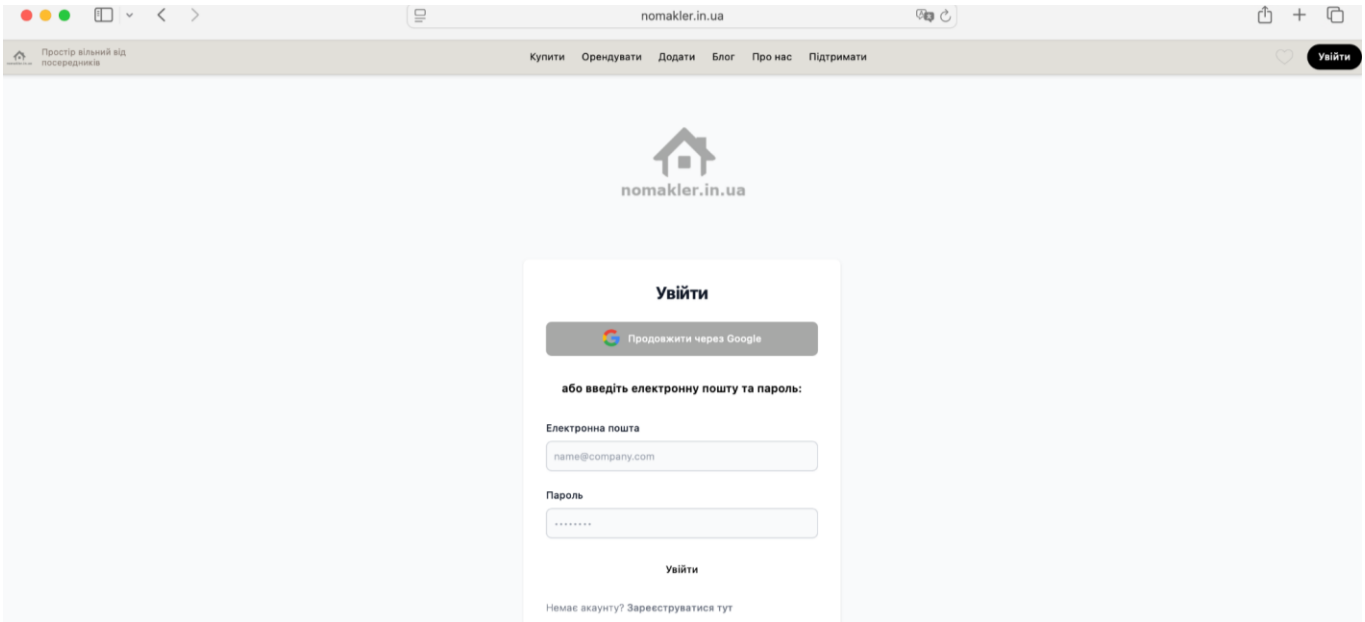
Визначено стратегії оптимізації продуктивності: композитні індекси, пагінація, кешування, мемоізація, debouncing та віртуалізація списків.

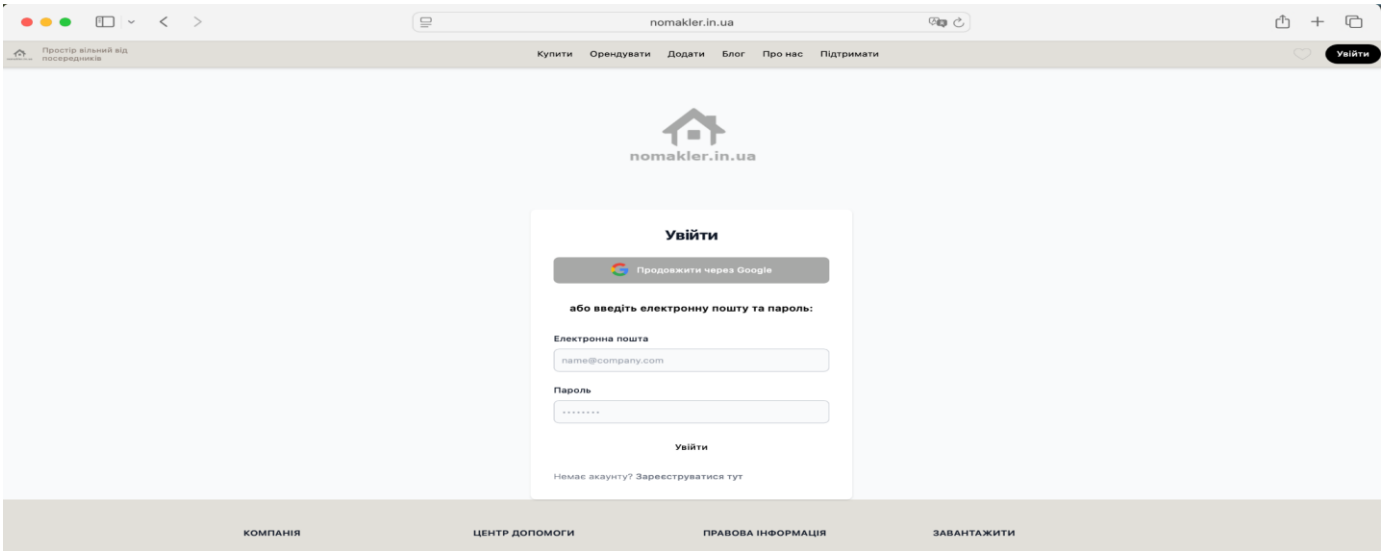
РОЗДІЛ 4
ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

4.1. Імплементація ключових модулів системи

Реалізація модулів системи виконана з дотриманням принципів чистої архітектури та якості коду [14]. Код структуровано за принципами модульності, повторного використання та single responsibility, що забезпечує підтримуваність та розширюваність системи.

4.1.1. Модуль автентифікації





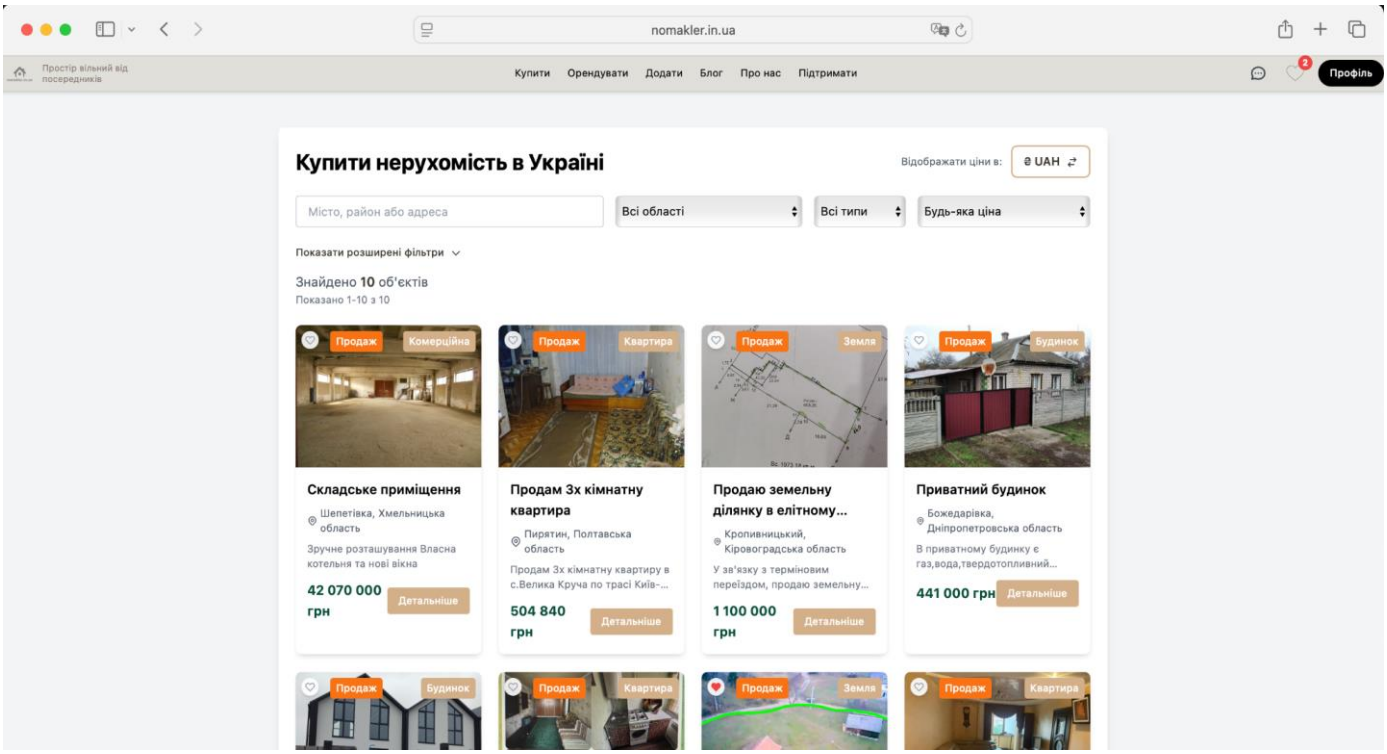
Для забезпечення безпечного доступу користувачів до системи розроблено модуль автентифікації, який базується на використанні сервісів Firebase Authentication. Програмна реалізація інкапсульована у спеціалізованому React-хуку useUserProfile, який відповідає за повний життєвий цикл сесії користувача:

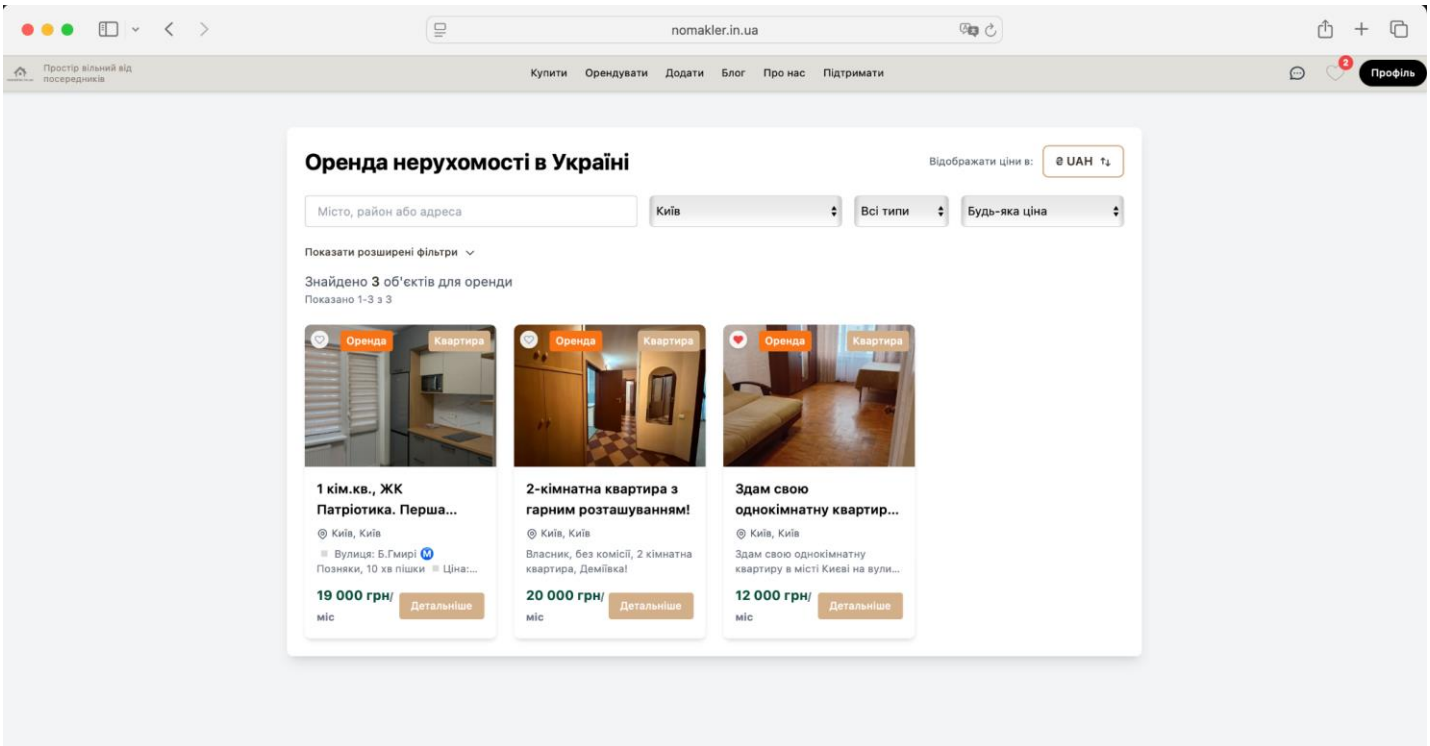
```tsx
export function useUserProfile() {
  const [profile, setProfile] = useState<UserProfile | null>(null);
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState<string | null>(null);

  useEffect(() => {
    const unsubscribe = auth.onAuthStateChanged(async (user) => {
      if (user) {
        try {
          setLoading(true);
          const userDoc = await getDoc(doc(firestore, 'users', user.uid));

          if (userDoc.exists()) {
            setProfile(userDoc.data() as UserProfile);
          } else {
            // Create profile if it doesn't exist
            const newProfile: UserProfile = {
              uid: user.uid,
              displayName: user.displayName || '',
              email: user.email || '',
              phoneNumber: user.phoneNumber || '',
              photoURL: user.photoURL || '',
              role: 'owner',
              createdAt: new Date(),
              updatedAt: new Date(),
            };

            await setDoc(doc(firestore, 'users', user.uid), newProfile);
            setProfile(newProfile);
          }
        } catch (err) {
          console.error('Error fetching user profile:', err);
          setError('Failed to load user profile');
        } finally {
          setLoading(false);
        }
      } else {
        setProfile(null);
        setLoading(false);
      }
    });
```

```
    return unsubscribe;
  }, []);

  const updateProfile = async (updates: Partial<UserProfile>) => {
    if (!profile) return false;

    try {
      const userRef = doc(firestore, 'users', profile.uid);
      await updateDoc(userRef, {
        ...updates,
        updatedAt: new Date(),
      });

      setProfile({ ...profile, ...updates, updatedAt: new Date() });
      return true;
    } catch (err) {
      console.error('Error updating profile:', err);
      setError('Failed to update profile');
      return false;
    }
  };

  return { profile, loading, error, updateProfile };
}

```

Логіка роботи модуля передбачає прослуховування змін стану авторизації в реальному часі. При вході користувача система автоматично отримує його унікальний ідентифікатор (UID) та ініціює запит до колекції users у базі даних Firestore для завантаження розширеного профілю (ім'я, роль, аватар, дата реєстрації). Отримані дані об'єднуються в єдиний об'єкт стану, який стає доступним глобально для всього застосунку через React Context. Це дозволяє уникнути надлишкових запитів до серверу при переході між сторінками та миттєво адаптувати інтерфейс залежно від прав доступу поточного користувача. Хук підписується на зміни стану автентифікації та автоматично завантажує розширений профіль із Firestore.

4.1.2. Модуль управління оголошеннями та пошуку





Центральним елементом системи є модуль роботи з оголошеннями, реалізований через хук **useProperties**:

```
export function useProperties() {
  const [properties, setProperties] = useState<Property[]>([]);
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState<string | null>(null);

  const fetchProperties = useCallback(async () => {
    try {
      setLoading(true);
      setError(null);


      const propertiesRef = collection(firestore, 'properties');
      const q = query(propertiesRef, orderBy('createdAt', 'desc'));
      const querySnapshot = await getDocs(q);

      const rawProperties = querySnapshot.docs.map((doc) => ({
        id: doc.id,
        ...doc.data(),
      })) as Property[];

      const validProperties = await filterValidProperties(rawProperties);

      console.log(
        `📊 Loaded ${validProperties.length} properties (filtered from ${rawProperties.length})`,
      );
      setProperties(validProperties);
    } catch (err) {
      console.error('Error fetching properties:', err);
      setError('Failed to load properties');
    } finally {
      setLoading(false);
    }
  }, []);

  useEffect(() => {
    fetchProperties();
  }, [fetchProperties]);

  return { properties, loading, error, refetch: fetchProperties };
}
```

Він забезпечує завантаження та первинну обробку даних про нерухомість.

Алгоритм отримання даних спроєктовано з акцентом на продуктивність та безпеку. Замість навантаження каналу постійним прослуховуванням бази даних, реалізовано стратегію оптимізованих запитів. При ініціалізації система звертається до колекції properties у Firestore, отримуючи список оголошень, відсортованих за хронологією.

Ключовою особливістю реалізації є багаторівнева фільтрація отриманих даних, винесена в окрему асинхронну процедуру **filterValidProperties**:

```
// Helper function to filter properties
async function filterValidProperties(
  properties: Property[],
): Promise<Property[]> {
  // 1. Filter out deleted properties first
  const activeProperties = properties.filter((p) => !p.deleted);

  // 2. Get unique user IDs
  const uniqueUserIds = Array.from(
    new Set(
      activeProperties.map((p) => p.userId).filter((id): id is string => !!id),
    ),
  );

  // 3. Batch check blacklist status
  const blacklistStatus = await checkUsersBlacklistStatus(uniqueUserIds);

  // 4. Final filter
  return activeProperties.filter((property) => {
    if (property.userId && blacklistStatus.get(property.userId)) {
      console.log(
        '🚫 Filtering out property from blacklisted user:',
        property.id,
      );
      return false;
    }
    return true;
  });
}
```

Вона виконує:

1. Миттєве відсіювання логічно видалених записів (soft delete).

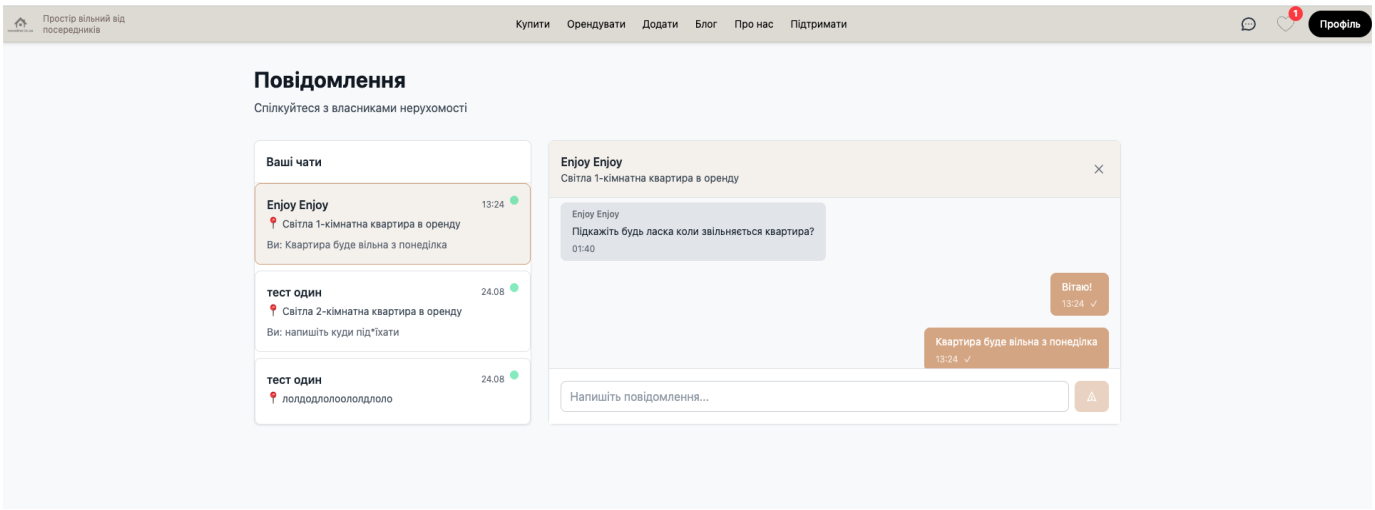
2. Агрегацію унікальних ідентифікаторів авторів оголошень.

3. Пакетну перевірку авторів на наявність у чорному списку (blacklist) через ModerationService. Це вирішує проблему "N+1 запитів", замінюючи сотні окремих перевірок на один паралельний процес.

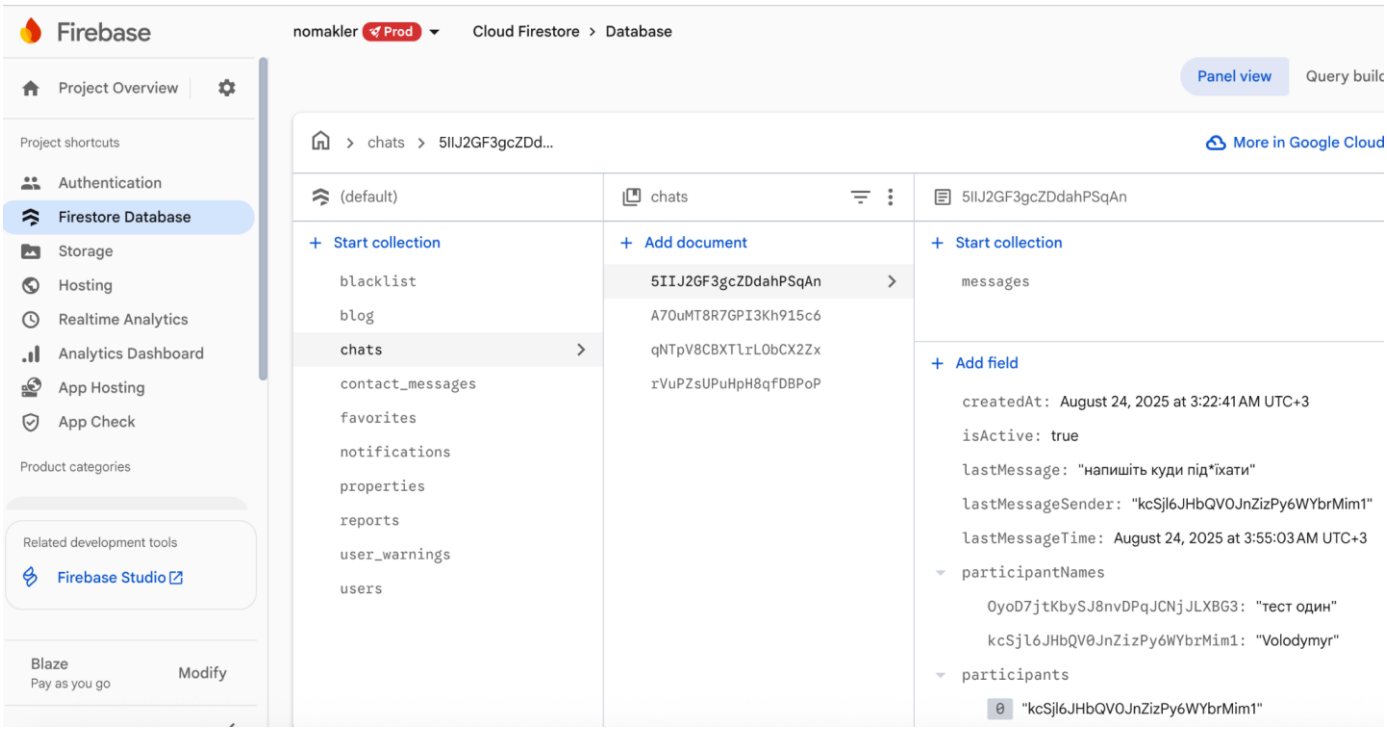
4. Фінальне формування валідного списку оголошень для відображення користувачу.

Такий підхід забезпечує швидке завантаження інтерфейсу та гарантує, що користувач не побачить небажаний контент.

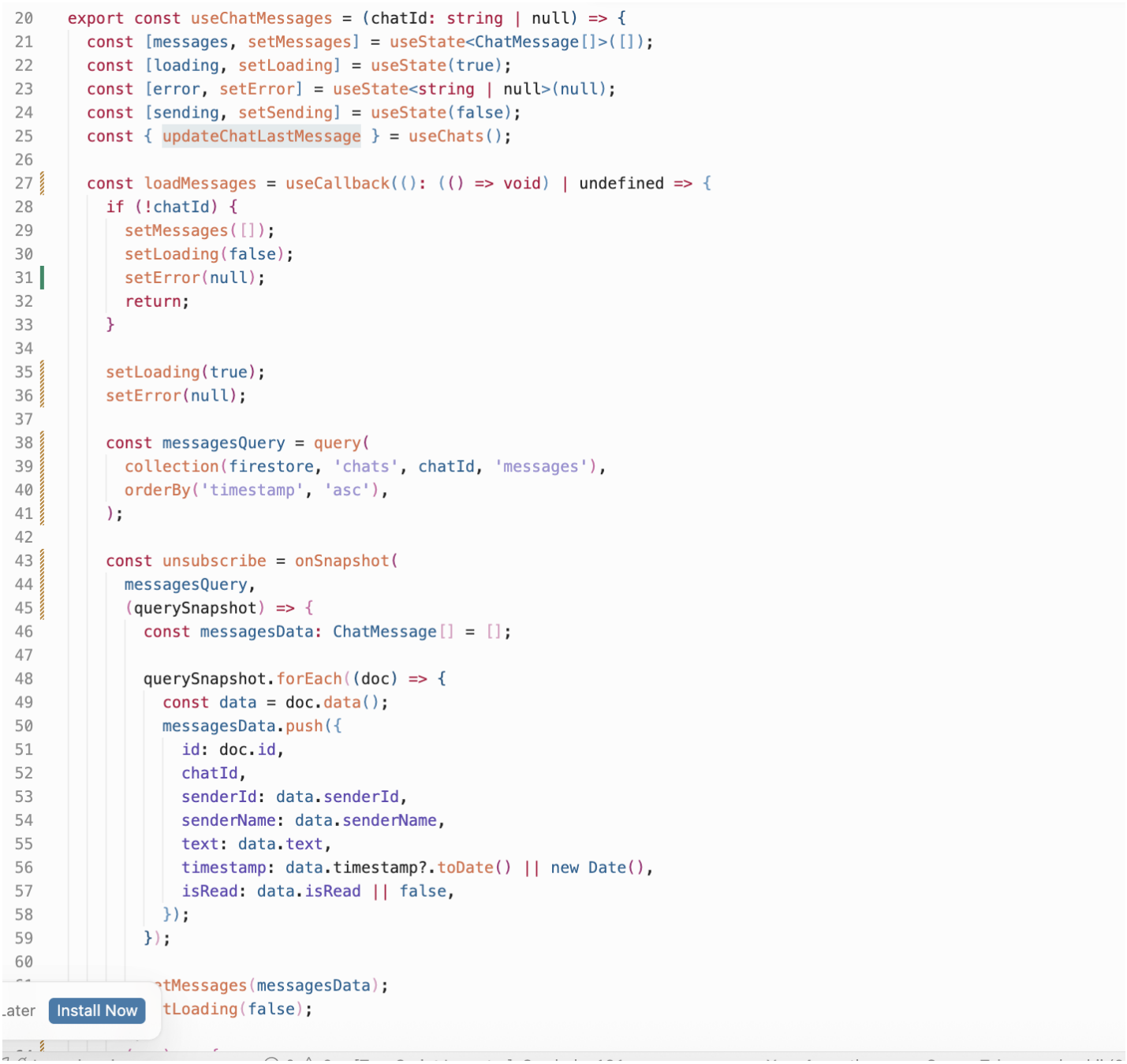
4.1.3. Модуль комунікації в реальному часі



Система миттєвих повідомлень спроєктована на основі підколекцій Firestore, де кожен чат є окремим документом у колекції chats, що містить вкладену колекцію messages:



Така структура забезпечує масштабованість та швидкий доступ до історії переписки, оскільки запити до повідомлень конкретного чату обмежені лише цією підколекцією, а не всією базою даних. Програмна реалізація інкапсульована у React-хуку **useChatMessages**, який управляє життєвим циклом діалогу:



```
export const useChatMessages = (chatId: string | null) => {
  const [messages, setMessages] = useState<ChatMessage[]>([]);
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState<string | null>(null);
  const [sending, setSending] = useState(false);
  const { updateChatLastMessage } = useChats();

  const loadMessages = useCallback((): (() => void) | undefined => {
    if (!chatId) {
      setMessages([]);
      setLoading(false);
      setError(null);
      return;
    }

    setLoading(true);
    setError(null);

    const messagesQuery = query(
      collection(firestore, 'chats', chatId, 'messages'),
      orderBy('timestamp', 'asc'),
    );

    const unsubscribe = onSnapshot(
      messagesQuery,
      (querySnapshot) => {
        const messagesData: ChatMessage[] = [];

        querySnapshot.forEach((doc) => {
          const data = doc.data();
          messagesData.push({
            id: doc.id,
            chatId,
            senderId: data.senderId,
            senderName: data.senderName,
            text: data.text,
            timestamp: data.timestamp?.toDate() || new Date(),
            isRead: data.isRead || false,
          });
        });

        tMessages(messagesData);
        tLoading(false);
later  Install Now
```

Модуль забезпечує:

1. **Ініціалізацію підписки на нові повідомлення в реальному часі.** При відкритті конкретного чату система встановлює **onSnapshot** підписку на колекцію messages цього чату, відсортовану за хронологією (orderBy('timestamp', 'asc')). Будь-які нові повідомлення автоматично синхронізуються з інтерфейсом без перезавантаження сторінки, що забезпечує миттєву доставку та відображення.

2. Механізм відправки повідомлень із захистом від спаму. Перед записом у базу даних система виконує перевірку частоти відправки через **checkMessageRateLimit()**, яка обмежує інтервал між повідомленнями до 1 секунди на користувача. Це запобігає флудингу та зловживанням системою:

```
53
54  // Rate limiting for message sending (client-side only, server should also enforce)
55  const messageCooldowns = new Map<string, number>();
56  const MESSAGE_COOLDOWN_MS = 1000; // 1 second between messages
57
58  export const checkMessageRateLimit = (): boolean => {
59    const currentUser = auth.currentUser;
60    if (!currentUser) return false;
61
62    const now = Date.now();
63    const lastMessageTime = messageCooldowns.get(currentUser.uid) || 0;
64
65    if (now - lastMessageTime < MESSAGE_COOLDOWN_MS) {
66      return false; // Too fast
67    }
68
69    messageCooldowns.set(currentUser.uid, now);
70    return true;
71  };
```

3. Санітизацію текстових даних для запобігання XSS-атакам. Функція **sanitizeMessage()** очищає вхідний текст від потенційно небезпечного контенту: видаляє HTML-теги (особливо <script>), блокує javascript: протоколи, обрізає довжину до 1000 символів. Це гарантує, що користувацький контент не може виконати довільний код у браузері іншого користувача.

```
7   // Sanitize message text to prevent XSS and malicious content
8   export const sanitizeMessage = (text: string): string => {
9     if (!text || typeof text !== 'string') return '';
10
11    return text
12      .trim()
13      .replace(/<script\b[^<]*(?:(?!<\/script>)<[^<]*)*<\/script>/gi, '') // Remove script tags
14      .replace(/<[^>]*>/g, '') // Remove HTML tags
15      .replace(/javascript:/gi, '') // Remove javascript: protocols
16      .slice(0, 1000); // Limit message length
17  };
18
```

4. Автоматичне оновлення статусу прочитання повідомлень. Система відстежує, які

повідомлення переглянув отримувач, та оновлює поле isRead у документі повідомлення. Це дозволяє відображати лічильник непрочитаних повідомлень та індикатори статусу доставки.

Додатково модуль оновлює метадані чату (останнє повідомлення, час останньої активності) в документі батьківського чату, що забезпечує швидкий доступ до прев'ю діалогу без завантаження всієї історії.

4.1.4. Модуль ручної модерації

Для забезпечення якості контенту та боротьби з порушеннями правил платформи реалізовано систему ручної модерації. Модерація здійснюється адміністратором через адміністраторську панель на основі скарг користувачів. Система модерації інтегрована в клас ModerationService, який забезпечує повний цикл обробки скарг від моменту подання до прийняття рішення та виконання відповідних дій.

Архітектура системи модерації:

Система модерації складається з таких компонентів:

Модуль подання скарг - дозволяє користувачам повідомляти про порушення: підозра на посередницьку діяльність, фейкові оголошення, неприйнятний контент, дублікати або спам.

Адміністраторська панель - інтерфейс для перегляду, аналізу та обробки скарг адміністратором

Система пріоритизації - автоматичне призначення пріоритету скаргам на основі типу порушення

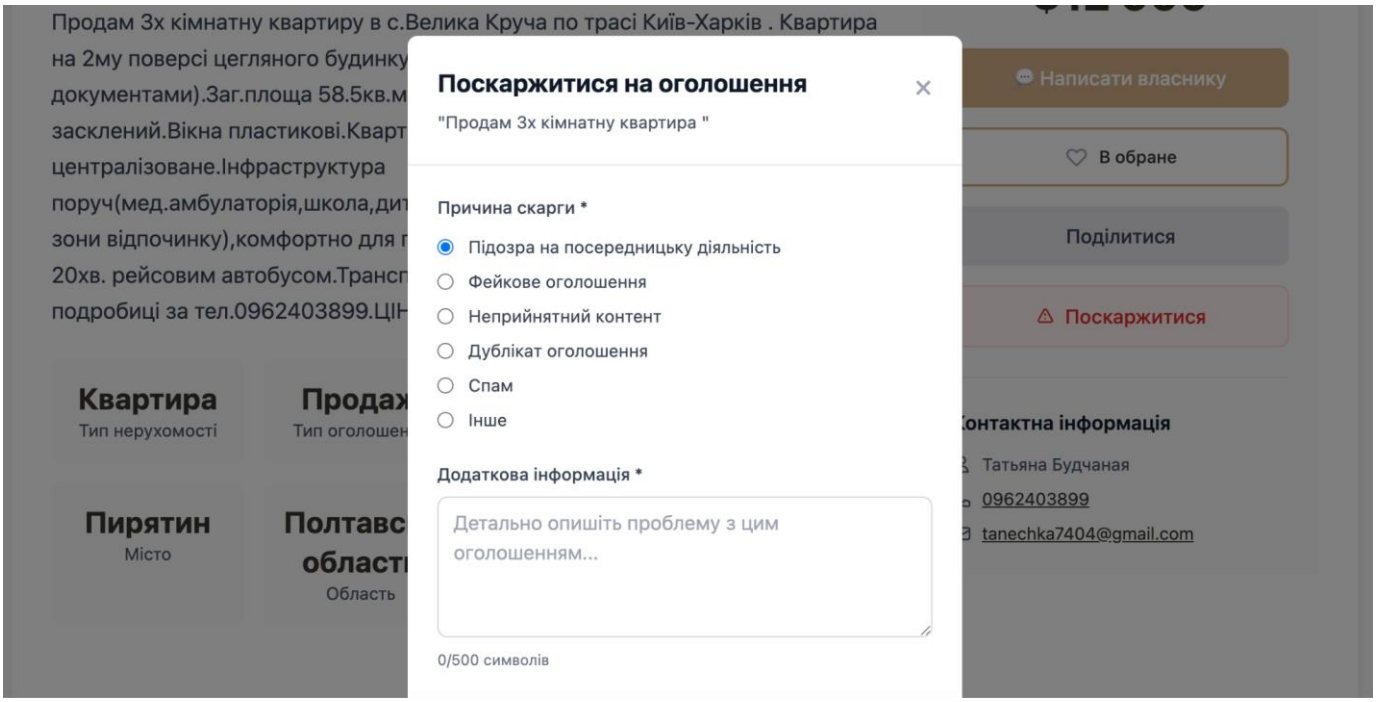
Модуль виконання дій - автоматичне виконання дій після прийняття рішення адміністратором

Система сповіщень - інформування користувачів про результати розгляду скарг

Процес модерації:

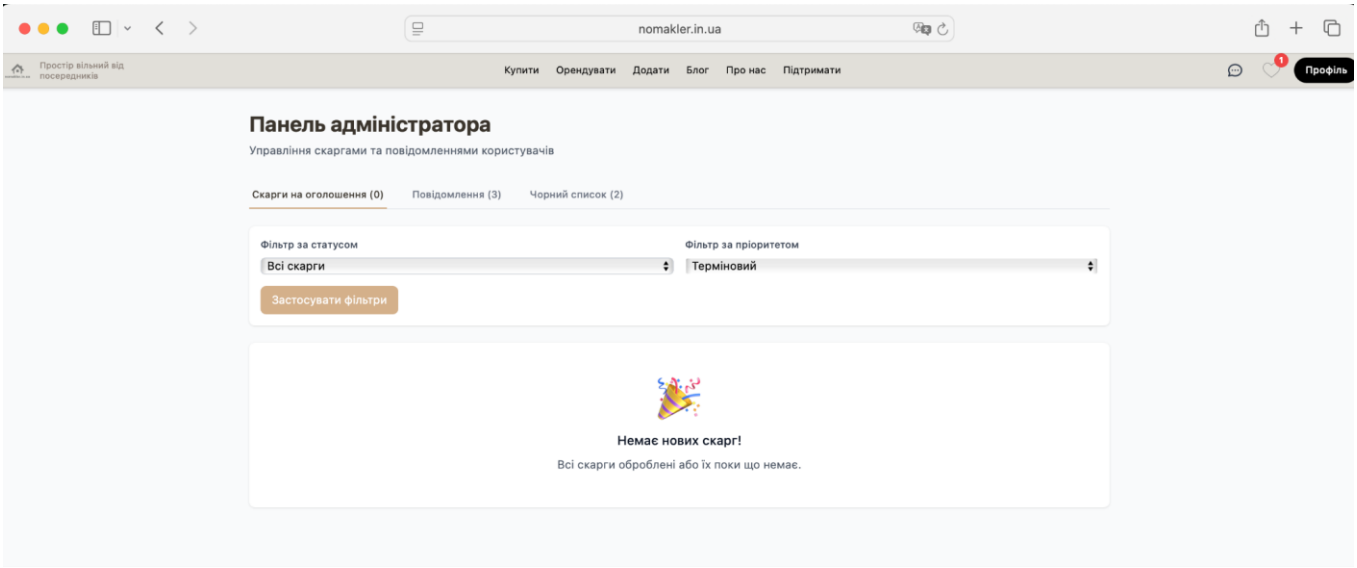
Процес ручної модерації включає такі етапи:

Подання скарги - користувач подає скаргу через інтерфейс, вказуючи причину (посередницька діяльність, фейкове оголошення, неприйнятний контент тощо) та опис проблеми:

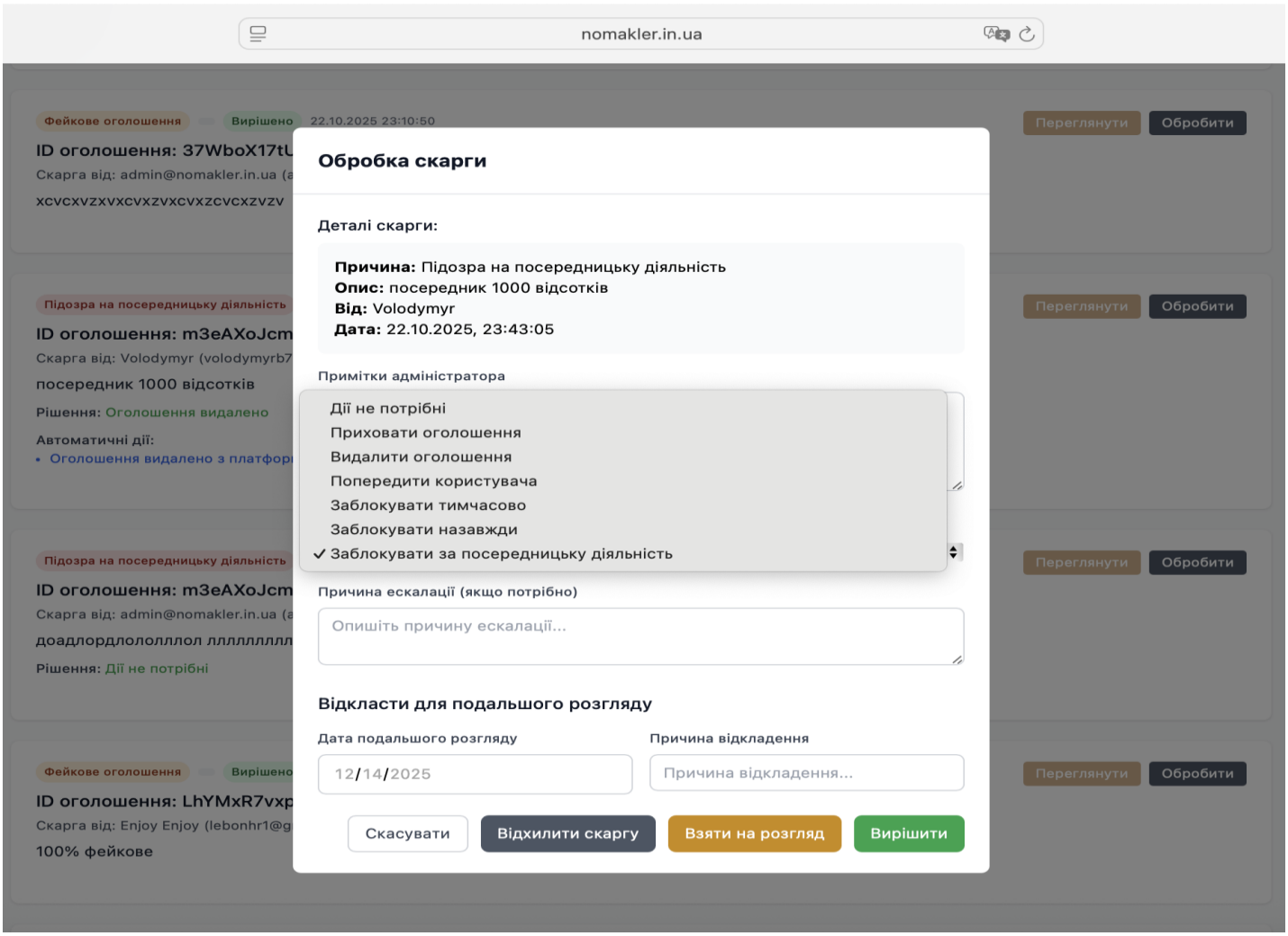


Автоматична пріоритизація - система призначає пріоритет скаржі: терміновий для неприйнятного контенту, високий для посередницької діяльності, фейкових оголошень та спаму, низький для дублікатів та інших причин

Розгляд адміністратором - адміністратор переглядає скарги в панелі, яка відображає їх з кольоровим кодуванням за типом та пріоритетом:



Аналіз та прийняття рішення - адміністратор вручну аналізує кожну скаргу, переглядає відповідне оголошення та приймає рішення: відхилити скаргу або підтвердити та вирішити проблему



Виконання дій - після прийняття рішення система автоматично виконує відповідні дії: приховування або видалення оголошення, попередження користувача, тимчасове або постійне блокування акаунту

Сповіщення користувачів - система інформує як автора скарги, так і власника оголошення про результати модерації

Типи рішень та дій:

Система підтримує такі типи рішень, які може прийняти адміністратор:

property_hidden - приховання оголошення від публічного перегляду

property_deleted - повне видалення оголошення з платформи

user_warned - відправка попередження користувачу

user_suspended - тимчасове блокування користувача

user_banned - постійне блокування користувача

realtor_banned - спеціальне блокування за посередницьку діяльність з автоматичним видаленням всіх оголошень користувача

no_action_needed - дії не потрібні, скарга відхилена

Реалізація ключових функцій:

Подача скарги користувачем реалізована через метод **submitReport**, який створює новий запис у колекції reports зі статусом pending та автоматично призначеним пріоритетом. Метод перевіряє авторизацію користувача, визначає пріоритет на основі причини скарги та зберігає дані у Firestore.

```
  static async submitReport(
    report: Omit<Report, 'id' | 'timestamp' | 'status' | 'priority'>,
  ): Promise<boolean> {
    try {
      const user = auth.currentUser;
      if (!user) {
        return false;
      }

      if (!report.propertyId?.trim() || !report.description?.trim()) {
        return false;
      }

      const priority = this.determineReportPriority(report.reason);

      const reportData = {
        ...report,
        timestamp: serverTimestamp(),
        status: 'pending' as const,
        priority,
        reporterNotified: false,
        propertyOwnerNotified: false,
      };

      await addDoc(collection(firestore, 'reports'), reportData);
      return true;
    } catch (error) {
      console.error('Error submitting report:', error);
      return false;
    }
  }
```

Оновлення статусу скарги адміністратором виконується через метод **updateReportStatus**, який оновлює статус скарги, додає коментарі адміністратора, призначає рішення та викликає автоматичне виконання відповідних дій. Метод також ініціює сповіщення власника оголошення та автора скарги про результати модерації.

Метод executeAutomaticActions виконує дії на основі прийнятого рішення. Наприклад, при рішенні property_deleted оголошення повністю видаляється з бази даних, а при realtor_banned користувач блокується та всі його оголошення видаляються. Кожна виконана дія фіксується в полі autoActions скарги для подальшого відстеження.

Система також забезпечує повну історію модерації: кожна скарга містить інформацію про час подання, адміністратора, який її розглянув, час розгляду, коментарі та виконані автоматичні дії. Це дозволяє відстежувати ефективність модерації та забезпечує

прозорість процесу.

Таким чином, система ручної модерації забезпечує гнучкий контроль якості контенту на платформі, дозволяючи адміністратору приймати обґрунтовані рішення на основі контексту кожної конкретної ситуації, при цьому автоматизуючи технічну частину виконання прийнятих рішень.

4.2. Реалізація механізмів безпеки

4.2.1. Firestore Security Rules

Правила безпеки Firestore забезпечують контроль доступу на рівні бази даних, виконуючи роль першого ешелону захисту. Правила перевіряються на сервері Firebase перед виконанням будь-якої операції з даними, що забезпечує навіть у випадку компрометації клієнтського коду неможливість несанкціонованого доступу до даних.

Архітектура правил безпеки:

Правила безпеки організовані за принципом найменших привілеїв: за замовчуванням доступ заборонений, і лише явно визначені правила дозволяють операції. Кожне правило складається з трьох компонентів:

- **Шлях до документа** - визначає, до яких колекцій та документів застосовується правило
- **Тип операції** - read (читання) або write (створення, оновлення, видалення)
- **Умова доступу** - логічний вираз, який має бути істинним для дозволу операції

Основні принципи безпеки:

- **Автентифікація** - більшість операцій вимагають авторизованого користувача (перевірка через request.auth != null)
- **Власність даних** - користувачі можуть модифікувати лише свої власні дані (перевірка request.auth.uid == resource.data.userId)
- **Адміністративні права** - адміністратори мають розширені права через custom claims або перевірку email

- **Незворотність критичних операцій** - видалення важливих даних (користувачі, скарги, повідомлення) заборонене через правила

Приклади правил для основних колекцій:

Колекція users: Користувачі можуть читати профілі інших користувачів (для відображення контактної інформації в чатах), але змінювати лише свій власний профіль. Адміністратори мають право модифікувати будь-які профілі для потреб модерації.

Колекція properties: Оголошення доступні для читання всім користувачам. Створювати оголошення можуть лише авторизовані користувачі, причому поле userId має відповідати ID авторизованого користувача. Оновлювати та видаляти оголошення можуть лише їх власники або адміністратори.

Колекція chats та messages: Доступ до чатів та повідомлень обмежений лише учасниками конкретного чату. Перевірка здійснюється через перевірку наявності ID користувача в масиві participants. Це забезпечує конфіденційність листування та неможливість доступу сторонніх осіб до приватних повідомлень.

Колекція reports: Користувачі можуть створювати скарги, але змінювати їх статус можуть лише адміністратори. Це запобігає маніпуляціям зі скаргами та забезпечує цілісність процесу модерації.

Колекція blacklist: Доступ до чорного списку обмежений: користувачі можуть переглядати лише свої власні записи, а повний доступ та можливість додавання записів мають лише адміністратори.

Правила безпеки також включають захист від поширених вразливостей: перевірку валідності даних при створенні, заборону на зміну критичних полів (наприклад, userId не може бути змінено після створення документа), та обмеження на операції видалення для важливих колекцій.

Приклад правил:

```
service cloud.firestore {
  match /databases/{database}/documents {
    // Users collection - users can read/write their own profile, admins can moderate
    match /users/{userId} {
      allow read: if true; // Anyone can read user profiles for contact info (needed for chats)
      allow create: if request.auth != null; // Allow creation during auth
      allow update: if request.auth != null && (
        request.auth.uid == userId || // Owner can edit
        request.auth.token.admin == true || // Admins with custom claims
        request.auth.token.email in ['admin@nomakler.in.ua'] // Admin emails (development fallback)
      );
      allow delete: if false; // Never allow user deletion
    }
```

```
    // Properties collection - anyone can read, owners can write their own, admins can moderate
    match /properties/{propertyId} {
      allow read: if true; // Anyone can read properties
      allow create: if request.auth != null &&
        request.auth.uid == request.resource.data.userId; // Only authenticated users, must be owner
      allow update, delete: if request.auth != null && (
        request.auth.uid == resource.data.userId || // Owner can edit
        request.auth.token.admin == true || // Admins with custom claims
        request.auth.token.email in ['admin@nomakler.in.ua', 'vladyslav@nomakler.in.ua'] // Admin emails (development fallback)
      );
    }

    // Favorites collection - users can read/write their own favorites
    match /favorites/{favoriteId} {
      allow read: if request.auth != null &&
        request.auth.uid == resource.data.userId;
      allow create: if request.auth != null &&
        request.auth.uid == request.resource.data.userId;
      allow update, delete: if request.auth != null &&
        request.auth.uid == resource.data.userId;
    }
```

4.2.2. Санітизація контенту

Як згадувалося в розділі 4.1.3, система чату використовує санітизацію повідомлень для захисту від XSS-атак.

4.2.3. Rate Limiting

Механізм обмеження частоти запитів, який використовується в модулі комунікації (розділ 4.1.3), запобігає зловживанням та DoS-атакам.

4.3. Процес тестування та розгортання

4.3.1. Функціональне тестування

Перевірено коректність роботи всіх модулів системи:

Модуль	Тестові сценарії	Результат
Автентифікація	Реєстрація, вхід, OAuth	Пройдено
Оголошення	CRUD операції, валідація, завантаження фото	Пройдено
Пошук	Фільтрація, сортування, ранжування	Пройдено
Чат	Відправка, доставка, realtime-синхронізація	Пройдено
Модерація	Відображення скарги на адмін панелі, опрацювання скарги	Пройдено

4.3.2. Тестування безпеки

Верифіковано стійкість системи до типових векторів атак: XSS-ін'єкції успішно нейтралізуються санітизацією; спроби несанкціонованого доступу блокуються Security Rules; rate limiting ефективно протидіє DoS-атакам на рівні застосунку.

4.3.3. Розгортання

Платформа розгорнута на Firebase Hosting за адресою https://nomakler.in.ua. Процес деплою автоматизовано через Firebase CLI:

```
npm run build                    # Production-збірка
firebase deploy                  # Деплой на хостинг
```

Скріншоти основних сторінок та інтерфейсів платформи наведені в додатку В. Детальна демонстрація роботи системи з поясненням особливостей реалізації представлена у скріншотах розділу 4.1 та 4.2.

4.4. Аналіз показників продуктивності

Проведено аудит продуктивності за допомогою Google Lighthouse [16]:

Метрика	**Значення**	**Оцінка**
Performance Score	94/100	Відмінно
First Contentful Paint	0.7 с	Відмінно
Largest Contentful Paint	1.4 с	Добре
Speed Index	1.5 с	Добре
Total Blocking Time	0 с	Відмінно
Cumulative Layout Shift	0.06	Відмінно
Bundle Size (gzip)	860 KB	Задовільно

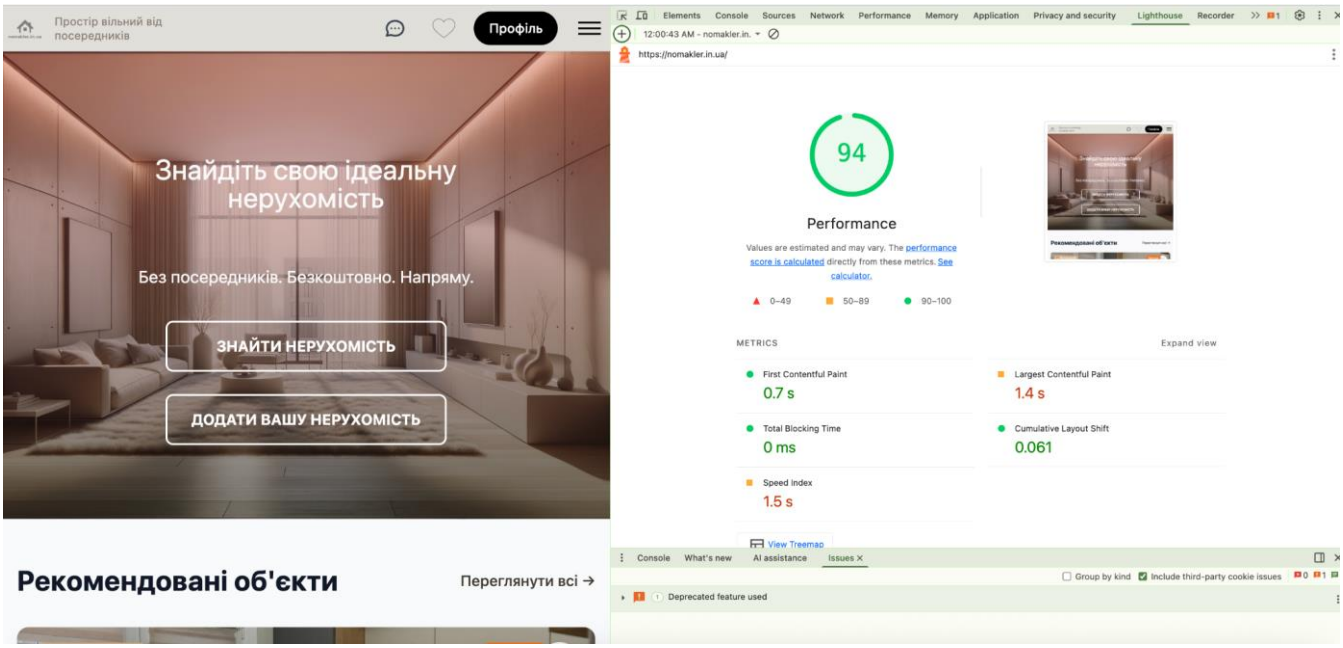
Висновки до розділу 4

У четвертому розділі описано програмну реалізацію платформи.

Імплементовано ключові модулі системи: автентифікації (useAuth), управління оголошеннями (useProperties), комунікації (useChat), ранжування (rankProperties). Код структуровано за принципами модульності та повторного використання.

Реалізовано багаторівневу систему безпеки: Firestore Security Rules для контролю доступу, санітизація контенту для захисту від XSS, rate limiting для протидії зловживанням.

Проведено функціональне тестування всіх модулів та тестування безпеки. Аудит продуктивності засвідчив високі показники: Performance Score 94/100, FCP 0.7 с, LCP 1.4 с (див рис. 1), що відповідає встановленим нефункціональним вимогам.



РОЗДІЛ 5
РОЗРОБЛЕННЯ СТАРТАП-ПРОЄКТУ

5.1. Опис ідеї проєкту

Стартап-проєкт NoMakler представляє вебплатформу для оренди та купівлі нерухомості без посередників, що забезпечує пряму комунікацію між власниками нерухомості та потенційними орендарями чи покупцями.

Зміст ідеї

Напрям застосування	**Вигоди для користувача**
Пошук орендного житла	Прямий контакт із власником без комісій посередників; достовірна інформація про об'єкти; інтелектуальний пошук із релевантним ранжуванням
Здача житла в оренду	Самостійне управління оголошеннями; безкоштовне розміщення; прямий доступ до орендарів без агентських витрат
Комунікація сторін	Вбудований чат реального часу; збереження історії переписки; прив'язка до конкретного оголошення

Аналіз потенційних техніко-економічних переваг

Характеристика	**NoMakler**	**OLX**	**DOM.RIA**

Характеристика	**NoMakler**	**OLX**	**DOM.RIA**
Гарантія прямого контакту	**Так**	Ні	Ні
Виявлення дублікатів (в процесі реалізації)	**Автоматичне**	Відсутнє	Часткове
Інтелектуальне ранжування	**Так**	Ні	Ні
Вартість для власників	**Безкоштовно**	Freemium	Freemium
Realtime-чат	**Так**	Так	Так

5.2. Технологічний аудит ідеї проєкту

Аналіз технологічної здійсненності проєкту:

Ідея проєкту	**Технології реалізації**	**Наявність**	**Доступність**
Вебплатформа з адаптивним дизайном	React, TypeScript, Tailwind CSS [5]	Наявна	Вільно поширювані технології
Serverless-архітектура	Firebase (BaaS)	Наявна	Freemium модель
Realtime-комунікація	Firestore subscriptions	Наявна	Включено у Firebase

Ідея проєкту	Технології реалізації	Наявність	Доступність
Алгоритми фільтрації та сортування	Власна розробка (TypeScript)	Розроблено	Власний код
Алгоритм ранжування	Власна розробка (TypeScript)	В процесі розробки	Власний код
Виявлення дублікатів	Jaro-Winkler (власна реалізація)	В процесі розробки	Власний код
Система модерації	Власна розробка (TypeScript)	Розроблено	Власний код
PWA-функціонал	Service Worker, Web App Manifest	Наявна	Реалізовано

Висновок: проєкт є технологічно реалізовуваним. Усі необхідні технології доступні, ключові алгоритми розроблено або знаходяться в процеці розробки, PWA-функціонал забезпечує кросплатформність.

5.3. Аналіз ринкових можливостей запуску стартап-проєкту

Характеристика потенційного ринку

Показник	Характеристика
Обсяг ринку	Ринок онлайн-операцій з нерухомістю в Україні - включно з орендою, купівлею житла, земельними ділянками та комерційною нерухомістю - є одним із найбільших і найперспективніших сегментів цифрової економіки. Він характеризується значною місткістю, стабільно високим попитом і зростаючим переходом користувачів до онлайн-сервісів, що створює сприятливі умови для розвитку інноваційних платформ.
Динаміка	До початку повномасштабної війни український ринок нерухомості характеризувався стабільним двозначним зростанням у більшості сегментів, зокрема у первинному житлі, оренді та земельних ділянках, з темпами розвитку близько 15-20 % на рік, що свідчило про його високий потенціал та привабливість для інвестицій. Після початку війни ринок зазнав суттєвого шоку та нерівномірного скорочення активності в окремих регіонах, проте у західних та центральних областях спостерігається поступове відновлення попиту, що свідчить про збереження потенціалу для цифрових платформ і онлайн-сервісів у сфері нерухомості.
Обмеження входу	Низькі технічні бар'єри; високі маркетингові витрати на залучення аудиторії
Специфіка	Мережевий ефект: цінність платформи зростає з кількістю користувачів

SWOT-аналіз

Сильні сторони	**Слабкі сторони**
Унікальна ціннісна пропозиція (без посередників); інноваційні алгоритми; низька собівартість завдяки serverless	Відсутність початкової бази користувачів; обмежені маркетингові ресурси; залежність від Firebase
Можливості	**Загрози**
Зростання попиту на P2P-сервіси; Розширення користувацької бази через всі сегменти нерухомості; Стратегічні партнерства для підвищення охоплення платформи.	Реакція конкурентів; зміни в законодавстві; економічна нестабільність; репутаційні ризики

5.4. Розроблення маркетингової програми стартап-проєкту

Ключові переваги концепції

Потреба	**Вигода**	**Ключові переваги**
Уникнення комісій	Економія 50-100% місячної оренди	Безкоштовний сервіс для всіх сторін
Достовірна інформація	Впевненість у контрагенті	Модерація + виявлення дублікатів
Зручний пошук	Економія часу	Інтелектуальне

Потреба	Вигода	Ключові переваги
		ранжування

Канали збуту

- **Основний канал - вебплатформа NoMakler**, яка одночасно працює як **прогресивний вебзастосунок (PWA)** і доступна за адресою https://nomakler.in.ua.
- **Мобільний доступ через PWA**, що дозволяє користувачам зручно користуватися сервісом на смартфонах без встановлення окремої програми.

Маркетингові комунікації

Канал	Ціль	Бюджет (місяць)
SEO-оптимізація	Органічний трафік	$200-500
Контекстна реклама (Google Ads)	Залучення власників нерухомості та орендарів і покупців	$500-1000
SMM (Facebook, Instagram)	Впізнаваність бренду	$300-500
Партнерства (блогери, тематичні групи)	Довіра аудиторії	$200-400

Модель монетизації

- **Базовий функціонал** - безкоштовно (формування бази користувачів)

- **Premium-розміщення** - платне підняття оголошень у видачі
- **Верифікація власників** - платна послуга підтвердження статусу
- **Аналітика для власників** - розширена статистика за підпискою

Висновки до розділу 5

У п'ятому розділі розроблено стартап-проєкт для комерціалізації платформи NoMakler.

Сформульовано ціннісну пропозицію проєкту: пряма взаємодія без посередників, інтелектуальний пошук, безкоштовний базовий функціонал. Проведено порівняння з конкурентами та визначено конкурентні переваги.

Технологічний аудит підтвердив реалізовуваність проєкту: всі необхідні технології доступні, ключові алгоритми розроблено.

Аналіз ринкових можливостей виявив значний потенціал зростання та основні ризики. SWOT-аналіз систематизував сильні та слабкі сторони проєкту.

Розроблено маркетингову програму з визначенням каналів збуту, комунікаційних інструментів та моделі монетизації на основі freemium-підходу.

ВИСНОВКИ

У магістерській кваліфікаційній роботі спроєктовано та реалізовано вебплатформу для оренди житла без посередників із застосуванням сучасних технологій та оригінальних алгоритмічних рішень.

Основні результати дослідження:

Проведено дослідження проблемної області та розроблено комплексне рішення, що включає теоретичне обґрунтування, математичне забезпечення та практичну реалізацію. Створено функціональну вебплатформу NoMakler, яка забезпечує пряму взаємодію між власниками нерухомості та потенційними орендарями, усуваючи необхідність у посередницьких структурах. Розроблено та впроваджено математичні методи оптимізації пошукового механізму та виявлення дублікатів, що підвищують якість сервісу та рівень довіри користувачів. Реалізована система модерації забезпечує контроль якості контенту та захист від недобросовісних учасників ринку. Платформа успішно розгорнута у виробничому середовищі та доступна для реальних користувачів за адресою https://nomakler.in.ua. Проведене тестування підтвердило коректність роботи всіх модулів системи, а аудит продуктивності засвідчив високі показники, що відповідають встановленим нефункціональним вимогам.

Перспективи подальших досліджень:

1. Інтеграція картографічних сервісів для візуалізації локацій
2. Впровадження рекомендаційної системи на основі машинного навчання
3. Розширення офлайн-функціоналу PWA для перегляду кешованого контенту та відкладеної синхронізації дій
4. Система верифікації користувачів через державні реєстри

5. Віртуальні тури 360° для огляду приміщень
6. Інтеграція платіжних сервісів для бронювання

Таким чином, усі поставлені завдання виконано, мета роботи досягнута. Створено функціональну, безпечну та масштабовану вебплатформу, що вирішує актуальну проблему українського ринку нерухомості - забезпечує пряму комунікацію між власниками та орендарями без посередників і додаткових комісій.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

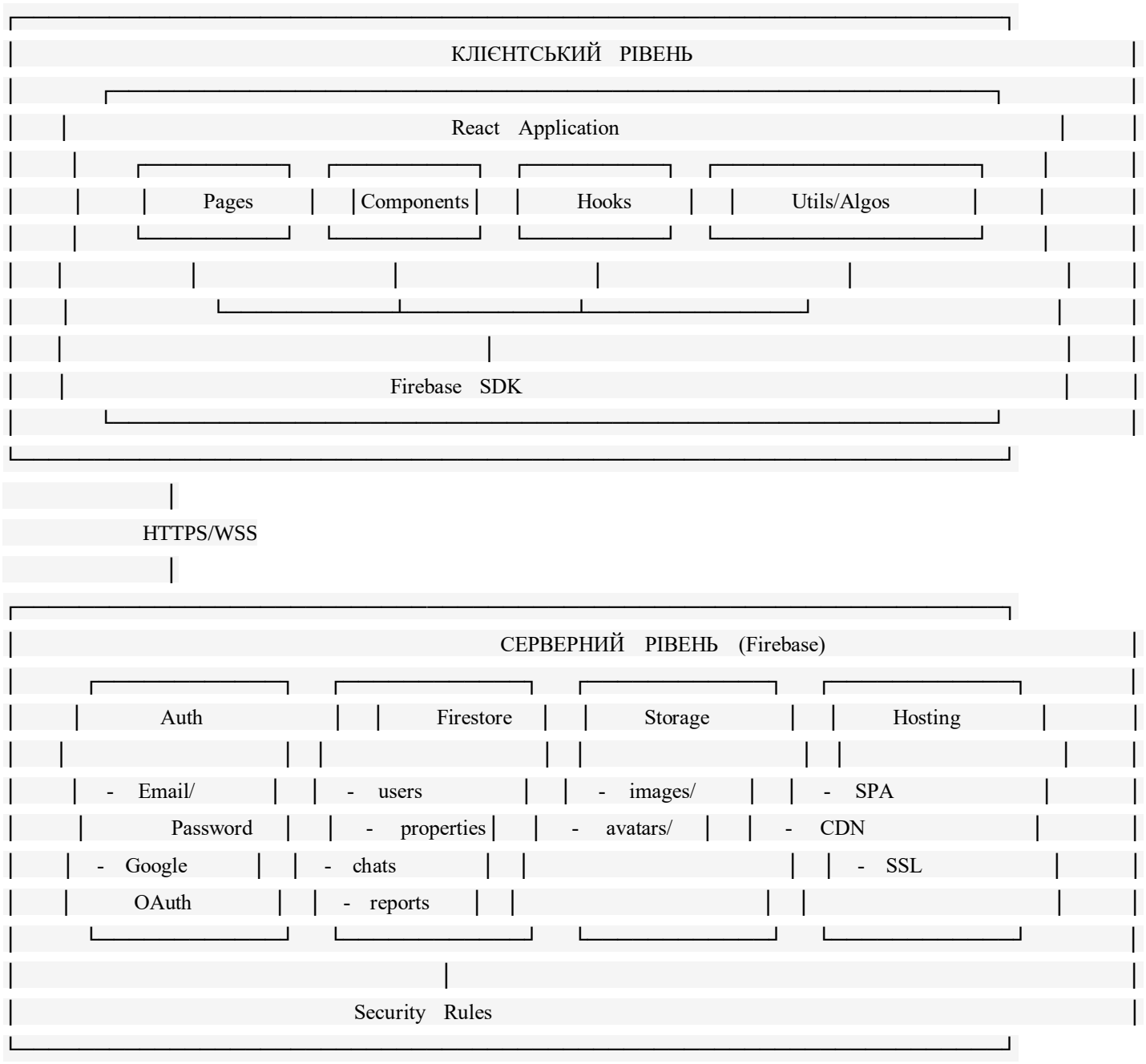
1. Firebase Documentation [Електронний ресурс] / Google Cloud Platform. – https://firebase.google.com/docs (дата звернення: 18.05.2025).
2. React Documentation [Електронний ресурс] / Meta Open Source. - https://react.dev (дата звернення: 27.06.2025).
3. TypeScript Documentation [Електронний ресурс] / Microsoft Corporation. - https://www.typescriptlang.org/docs (дата звернення: 20.08.2025).
4. Vite Documentation [Електронний ресурс] / Evan You. - https://vitejs.dev (дата звернення: 08.07.2025).
5. Tailwind CSS Documentation [Електронний ресурс] / Tailwind Labs. - https://tailwindcss.com/docs (дата звернення: 13.08.2025).
6. Winkler W. E. String Comparator Metrics and Enhanced Decision Rules in the Fellegi-Sunter Model of Record Linkage / W. E. Winkler // Proceedings of the Section on Survey Research Methods. - American Statistical Association, 1990. - P. 354-359.
7. Jaro M. A. Advances in Record-Linkage Methodology as Applied to Matching the 1985 Census of Tampa, Florida / M. A. Jaro // Journal of the American Statistical Association. - 1989. - Vol. 84, No. 406. - P. 414-420.
8. Fuse.js: Lightweight fuzzy-search library [Електронний ресурс]. - https://fusejs.io.
9. Cloud Firestore Security Rules [Електронний ресурс] / Firebase Documentation. - https://firebase.google.com/docs/firestore/security/rules-structure (дата звернення: 13.07.2025).
10. Progressive Web Apps [Електронний ресурс] / MDN Web Docs. - https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps (дата звернення: 07.05.2025).
11. OAuth 2.0 Authorization Framework [Електронний ресурс] / IETF RFC 6749. - https://tools.ietf.org/html/rfc6749 (дата звернення: 11.06.2025).

12. XSS Prevention Cheat Sheet [Електронний ресурс] / OWASP Foundation. - https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html (дата звернення: 22.09.2025).
13. Fowler M. NoSQL Databases: An Overview [Електронний ресурс] / M. Fowler. - https://martinfowler.com/nosql.html (дата звернення: 10.06.2025).
14. Martin R. C. Clean Code: A Handbook of Agile Software Craftsmanship / R. C. Martin. - Prentice Hall, 2008. - 464 p.
15. Kleppmann M. Designing Data-Intensive Applications / M. Kleppmann. - O'Reilly Media, 2017. - 616 p.
16. Lighthouse Performance Audits [Електронний ресурс] / Google Developers. - https://developer.chrome.com/docs/lighthouse (дата звернення: 30.11.2025).

ДОДАТКИ

Додаток А. Архітектура системи

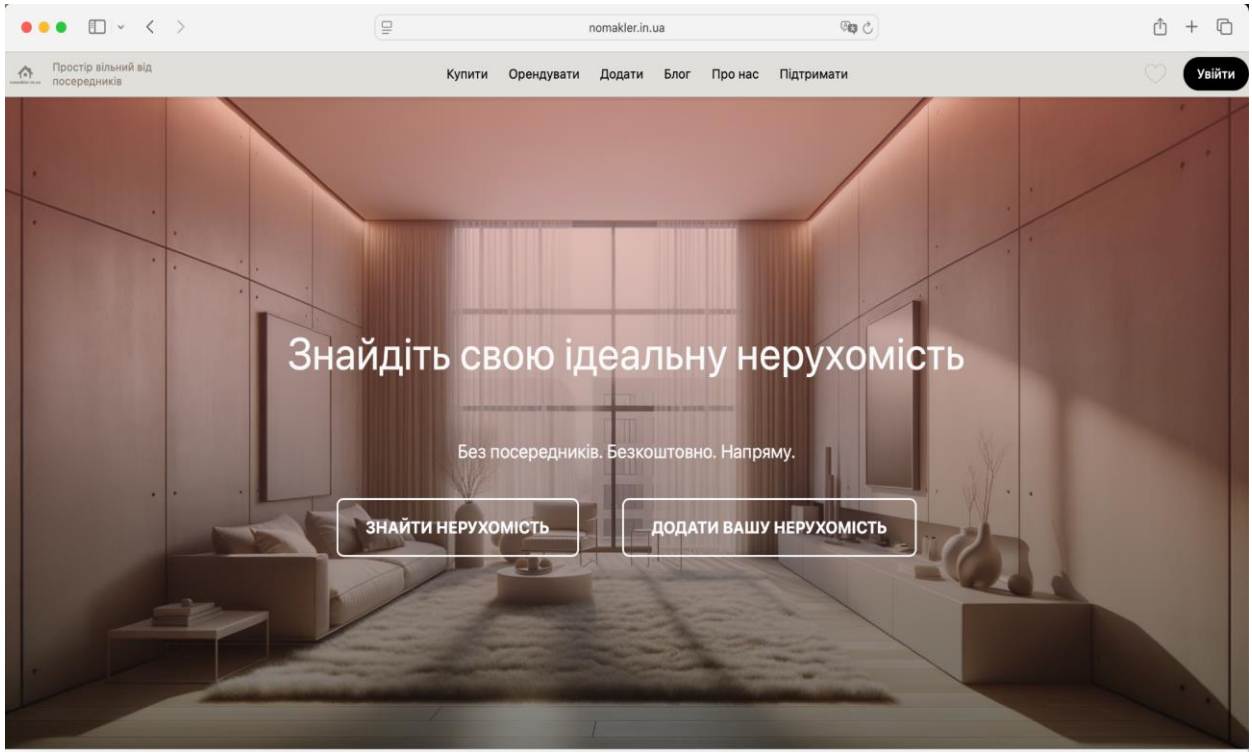
Схема архітектури вебплатформи NoMakler:

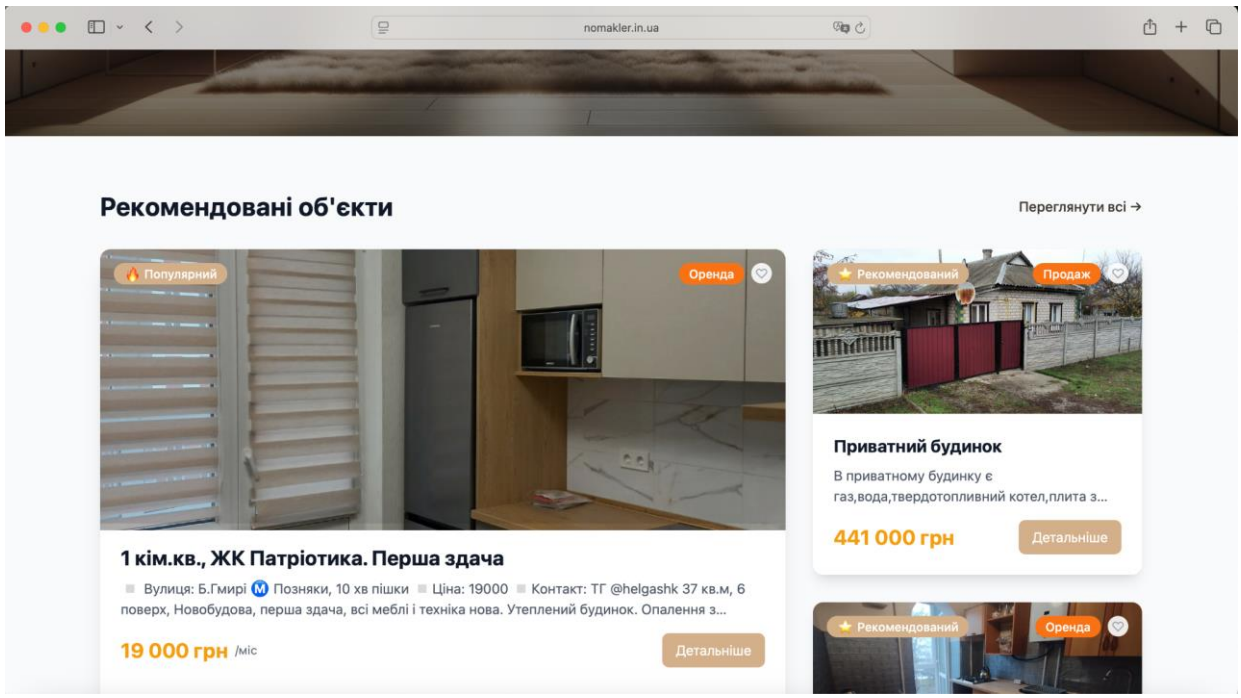


Додаток Б. Скріншоти інтерфейсу платформи

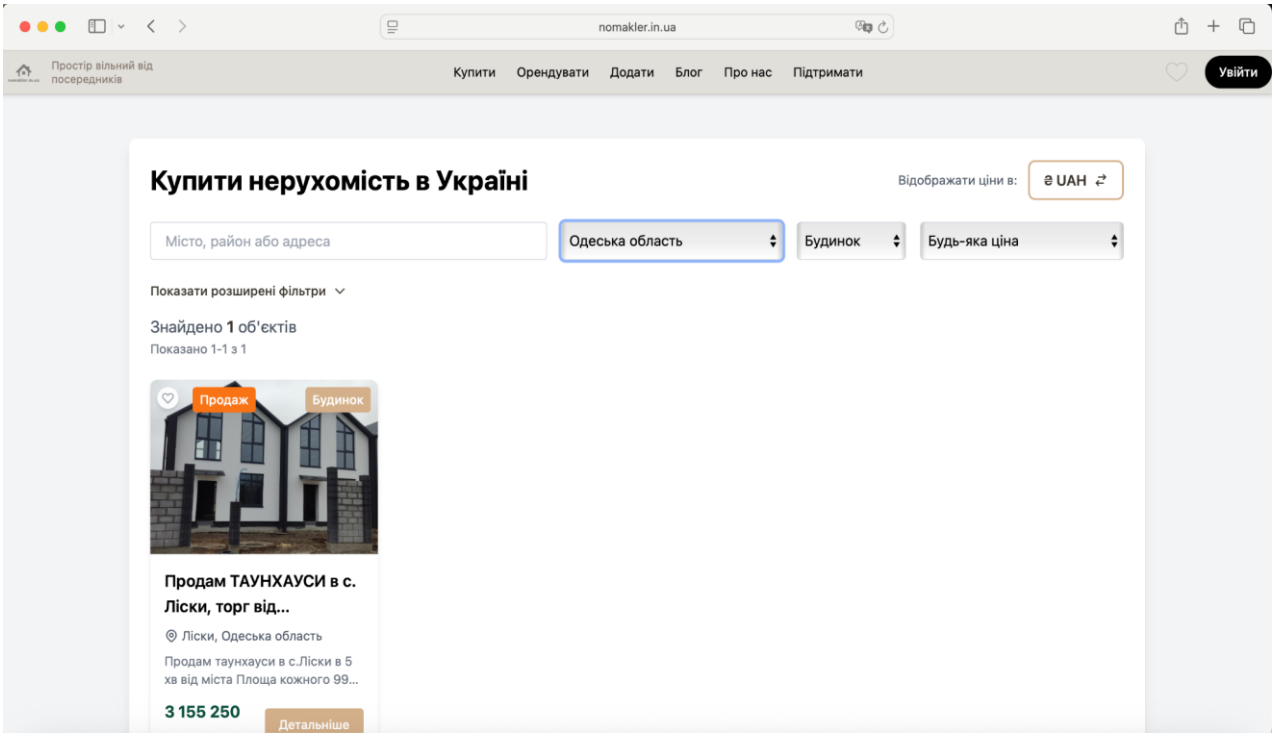
Основні екрани платформи:

- Головна сторінка з пошуковою формою та популярними оголошеннями

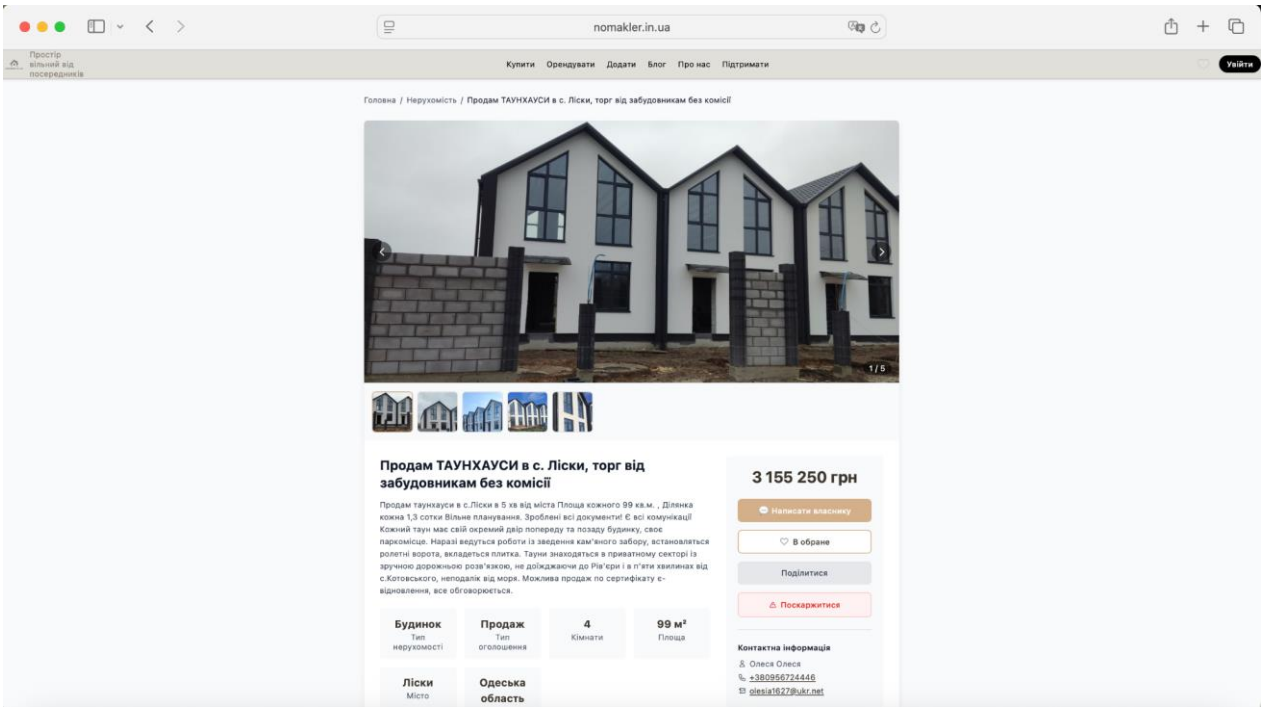




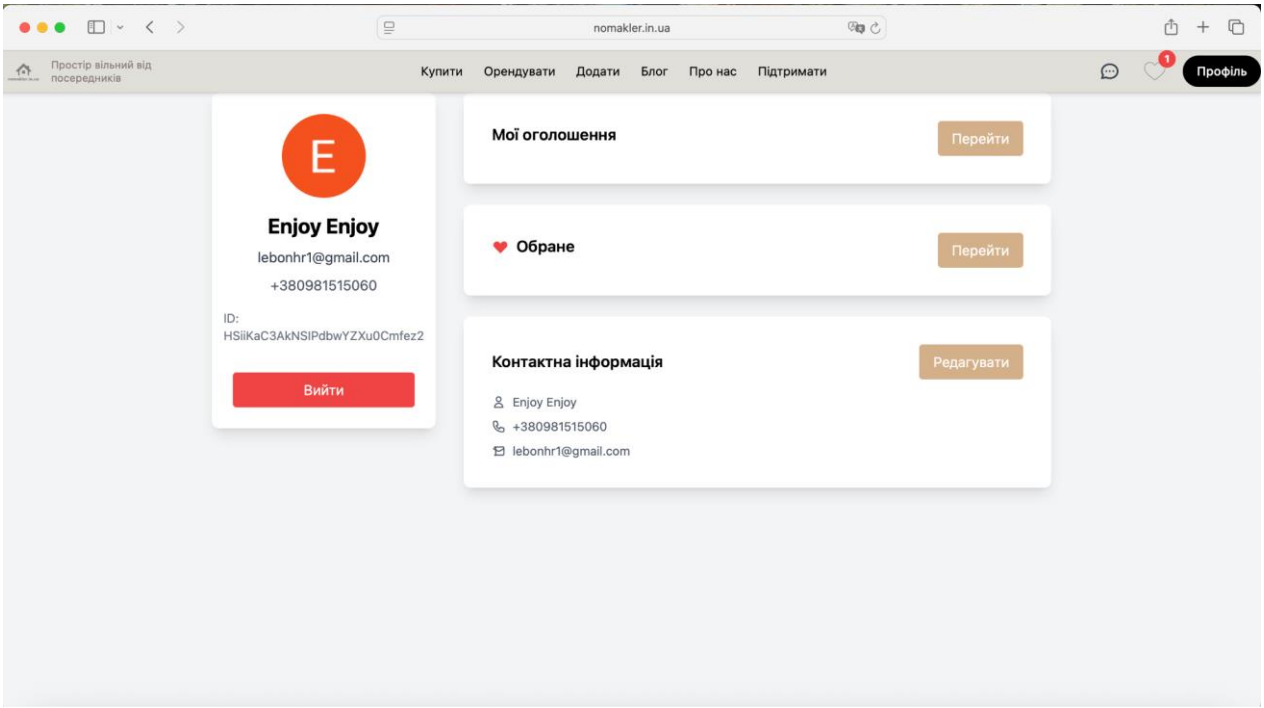
- Результати пошуку з фільтрами та сортуванням



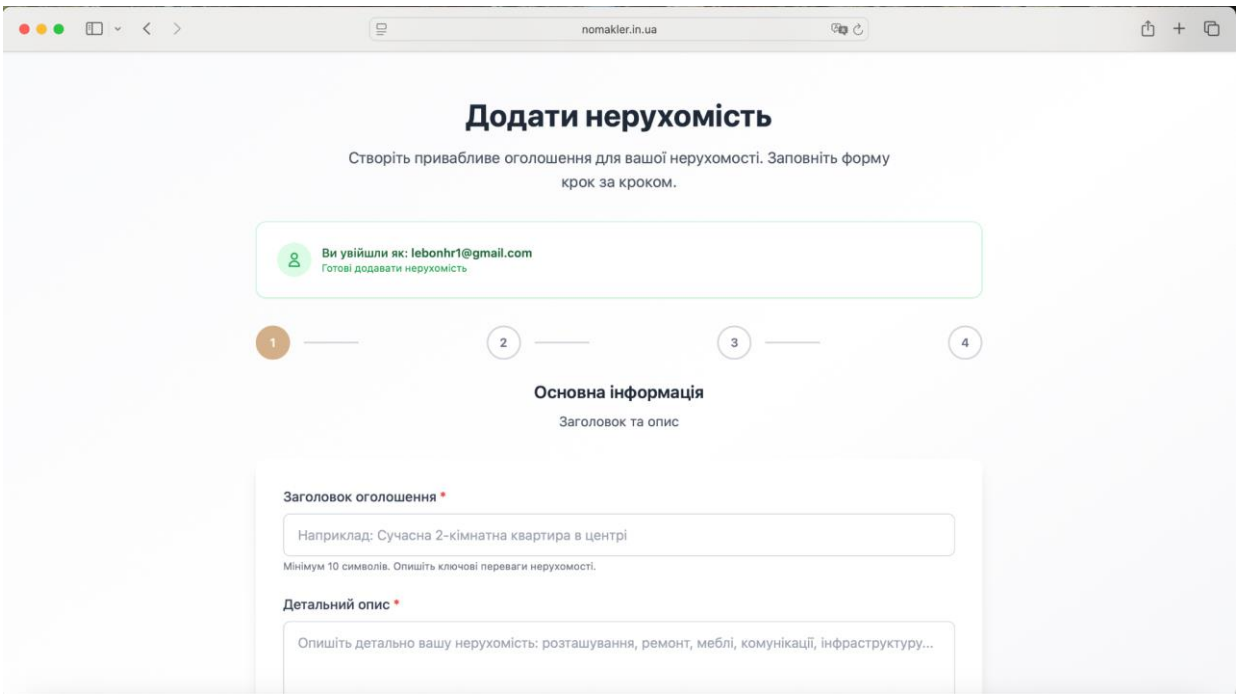
- Детальна сторінка оголошення з галереєю фото та контактами



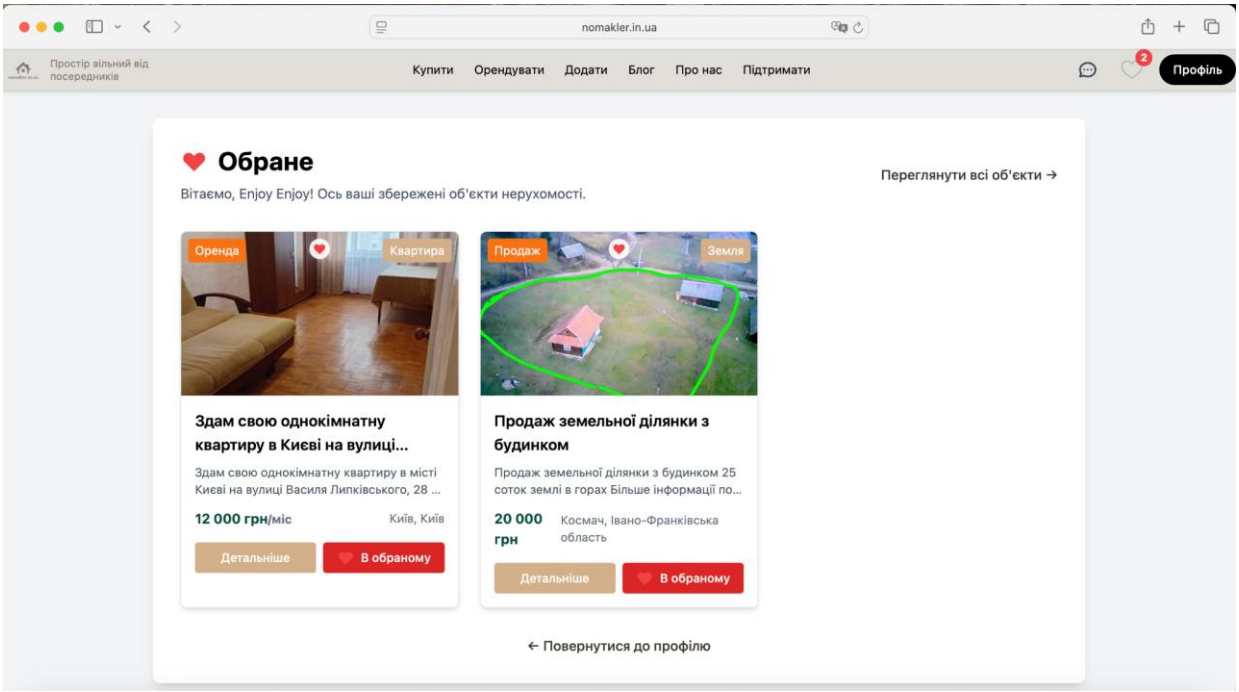
- Особистий кабінет із списком власних оголошень



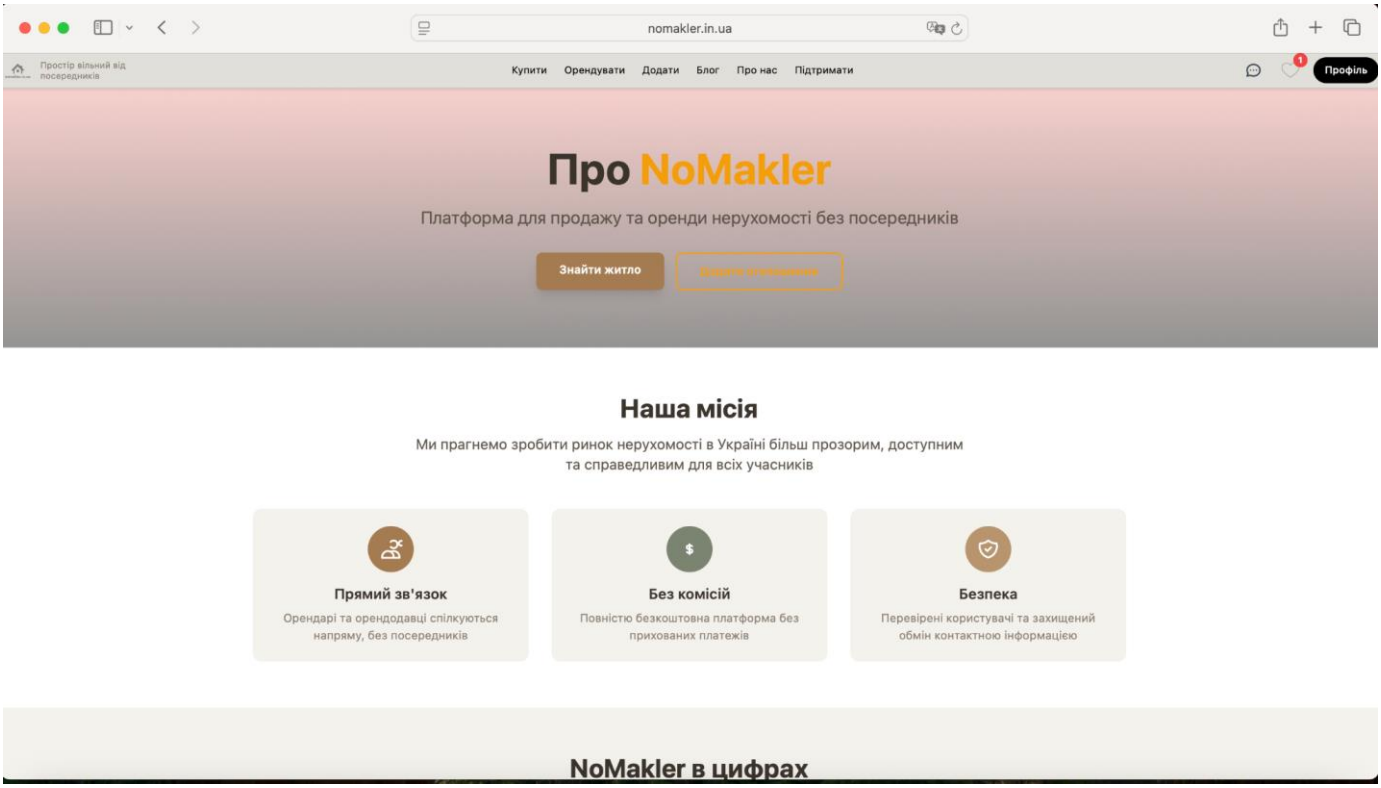
- Форма створення/редагування оголошення



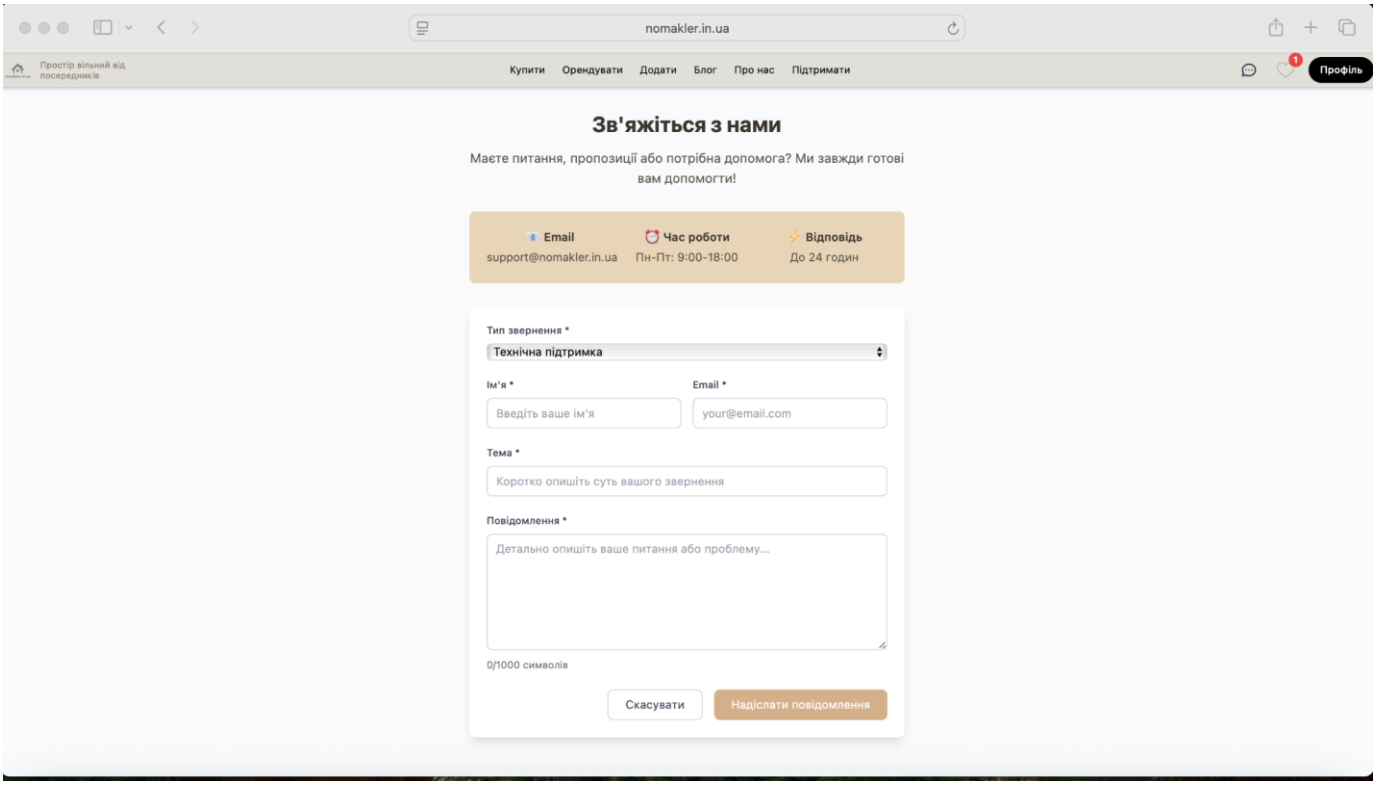
- Сторінка улюблених оголошень



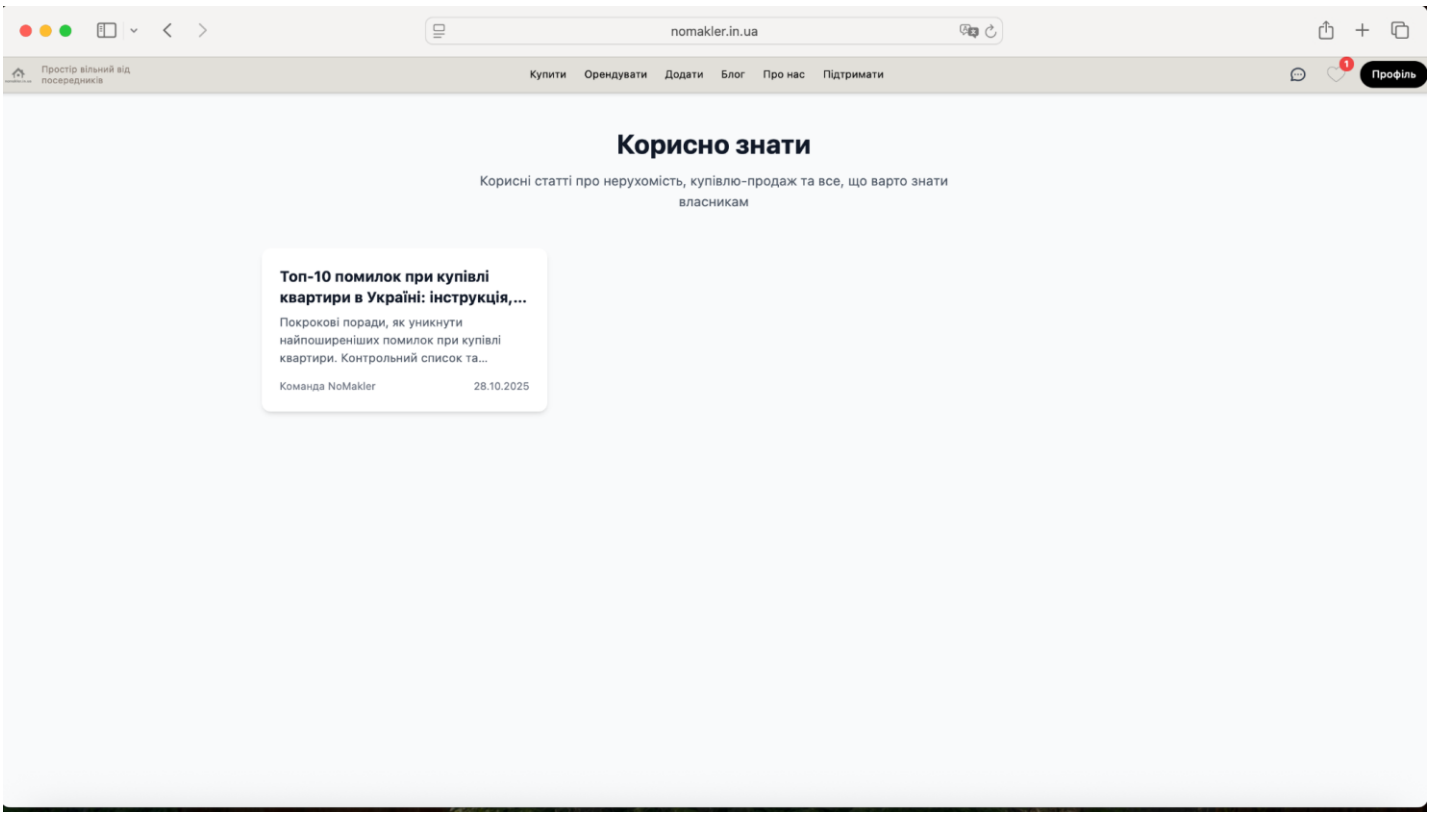
- Про нас



- Зворотній зв’язок



- Блог



Львів – 2025