МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Острозька академія»
Навчально-науковий інститут інформаційних технологій та бізнесу Кафедра інформаційних технологій та аналітики даних

КВАЛІФІКАЦІЙНА РОБОТА
на здобуття освітнього ступеня бакалавра

на тему: **«Розробка модуля карти з інтерактивним порівнянням оголошень для платформи "Пошук житла в Острозі"»**

Виконала: студентка 4 курсу, групи КН-42
першого (бакалаврського) рівня вищої освіти
спеціальності 122 Комп'ютерні науки
освітньо-професійної програми «Комп'ютерні науки»
Мельник Дарина Анатоліївна

Керівник: доктор філософії з прикладної математики, старший викладач кафедри інформаційних технологій та аналітики даних, фахівець-практик
Красюк Богдан Віталійович

Рецензент: *науковий ступінь, вчене звання, посада, ПІБ рецензента*

РОБОТА ДОПУЩЕНА ДО ЗАХИСТУ

Завідувач кафедри інформаційних технологій та аналітики даних _____ (проф., д.е.н. Кривицька О.Р.)

Протокол № ____ від « ___ » _____ 2026 р.

Острог, 2026

АНОТАЦІЯ
кваліфікаційної роботи
на здобуття освітнього ступеня бакалавра

Тема: Розробка модуля карти з інтерактивним порівнянням оголошень для платформи "Пошук житла в Острозі"

Автор: Мельник Дарина Анатоліївна

Науковий керівник: *доктор філософії з прикладної математики, викладач, фахівець-практик, Красюк Богдан Віталійович*

Захищена «……»…… 20__ року.

Пояснювальна записка до кваліфікаційної роботи: *69 с., 31 рис., 1 табл., 4 додатки, 38 джерел.*

Ключові слова: *вебплатформа, фронтенд, React.js, Vite, інтерактивна карта, Leaflet, SignalR, UI/UX дизайн, порівняння оголошень оренди, пошук житла для студентів в Острозі.*

Короткий зміст праці:
Кваліфікаційна робота присвячена розробці клієнтської частини платформи «UniStay» для пошуку житла в Острозі. Виконано проєктування інтерфейсу на базі React 19 та TypeScript із використанням середовища Vite для максимальної швидкодії. Основний акцент зроблено на створенні інтерактивного модуля карти (Leaflet), який дозволяє візуалізувати відстані до навчальних корпусів та інфраструктурних об'єктів у реальному часі.
У роботі реалізовано унікальний інструмент синхронного порівняння оголошень з автоматичним підсвічуванням вигідних характеристик. Технічна частина включає чат-сервіс на базі SignalR для миттєвої комунікації, систему захищених маршрутів (JWT) та інтеграцію з хмарним сховищем Cloudinary для оптимізації медіаданих. Результатом є сучасний вебзастосунок, що автоматизує процес вибору житла та покращує користувацький досвід за рахунок інноваційних інструментів аналітики.

ANNOTATION
of qualification paper
for bachelor's degree

Theme: *Development of a map module with interactive listing comparison for the "Housing Search in Ostroh" platform*

Author: *Daryna Melnyk*

Scientific supervisor: *Doctor of Philosophy in Applied Mathematics, Lecturer, Practitioner Specialist, Bohdan Vitaliyovych Krasiuk*

Defended on «……»…… 20__.

Explanatory note to the qualification work: *69 pages, 31 figures, 1 table, 4 appendices, 38 sources*

Keywords: *web platform, frontend, React.js, Vite, interactive map, Leaflet, SignalR, UI/UX design, rental listing comparison, student housing search in Ostroh.*

Brief summary of the work:

This qualification work is dedicated to the development of the client-side of the "UniStay" platform for finding housing in Ostroh. The interface was designed using React 19 and TypeScript, leveraging the Vite build tool for maximum performance. A central focus was placed on creating an interactive map module (Leaflet), which enables real-time visualization of distances to university buildings and key infrastructure. The project implements a unique tool for synchronous listing comparison with automatic highlighting of advantageous features. The technical implementation includes a SignalR-based chat service for instant communication, a protected route system (JWT), and integration with Cloudinary cloud storage for media data optimization. The result is a modern web application that automates the housing selection process and enhances user experience through innovative analytical tools.

ЗМІСТ

ВСТУП

У сучасному цифровому світі перша взаємодія користувача з будь-яким сервісом відбувається через його візуальний інтерфейс, від якості якого залежить успішність продукту. Для ринку оренди нерухомості, зокрема у студентських містечках з високим рівнем освітньої міграції, швидкий та зручний пошук житла є критично важливим. Платформи з пошуку нерухомості стали основним інструментом у цьому процесі, проте вимоги сучасних користувачів до вебдодатків постійно зростають: вони очікують не просто списку оголошень, а інтуїтивно зрозумілого, швидкого та візуально інформативного середовища.

Аналіз існуючих традиційних вебсервісів з пошуку житла виявляє суттєві недоліки в їхній клієнтській частині. Більшість платформ пропонують користувачам шаблонні інтерфейси зі стандартним відображенням інформації у вигляді текстових списків та статичних зображень. Для цільової аудиторії студентів, для яких ключовим фактором вибору є розташування житла відносно навчального закладу та міської інфраструктури, такий підхід є незручним. Користувачам доводиться перемикатися між дошкою оголошень та сторонніми картографічними сервісами, самостійно вираховувати відстані та намагатися утримати в голові просторову картину. Це створює високе когнітивне навантаження. Крім того, відсутність зручного інтерфейсу для обговорення знайдених варіантів у реальному часі змушує студентів розпорошувати комунікацію по сторонніх месенджерах.

Ефективне розв'язання цієї проблеми полягає у створенні спеціалізованого, високопродуктивного вебдодатка, який надає принципово новий рівень взаємодії з даними завдяки повністю кастомному дизайну та продуманому UX. У цьому контексті актуальною є розробка інноваційного картографічного модуля з можливістю інтерактивного порівняння об'єктів нерухомості, доповненого мікроанімаціями для кращого візуального сприйняття просторових даних. Такий підхід у поєднанні з інтегрованим клієнтським модулем чатів дозволить орендарям приймати рішення на основі наочної інформації в межах однієї платформи.

Метою даної кваліфікаційної роботи є проєктування та розробка клієнтської частини (frontend) вебплатформи «Пошук житла в Острозі» (UniStay) з унікальним користувацьким інтерфейсом та інтерактивним картографічним модулем. Рішення базується на використанні сучасного фреймворку React, збирача Vite та екосистеми спеціалізованих бібліотек, що забезпечує створення чуйного (responsive), масштабованого та візуально привабливого вебдодатка.

Для досягнення поставленої мети необхідно виконати такі **основні завдання**:

- спроєктувати модульну архітектуру клієнтського вебдодатка на базі технологічного стека React, забезпечивши оптимізацію процесів збірки за допомогою Vite;
- розробити повністю кастомний користувацький інтерфейс (UI) з нуля, імплементувавши сучасні UX-патерни та систему візуальних акцентів;
- спроєктувати та інтегрувати картографічний модуль на базі бібліотеки Leaflet, реалізувавши алгоритми візуалізації відстаней та одночасного порівняння оголошень;
- застосувати технології CSS-анімацій (keyframes, transitions) для створення інтерактивних візуальних ефектів (пульсуючі маркери, анімовані вектори відстаней, режими фокусування), що знижують когнітивне навантаження користувача;
- розробити клієнтську логіку для системи групового підбору житла, інтегрувавши технологію SignalR для забезпечення безшовного обміну повідомленнями у реальному часі;
- реалізувати асинхронну взаємодію з серверним REST API за допомогою Axios для відображення актуального контенту.

Об’єкт дослідження: процеси взаємодії користувачів із вебсистемами пошуку нерухомості та методи візуалізації просторових і динамічних даних у вебсередовищі.

Предмет дослідження: розробка та технології розробки клієнтської частини (Frontend), інструменти створення інтерактивних картографічних модулів (Leaflet),

методи імплементації кастомного UI/UX дизайну та технології комунікації в реальному часі (SignalR) на базі екосистеми React.

Таким чином, розробка клієнтської частини проєкту «UniStay» з інтегрованим картографічним модулем та унікальним дизайном дозволить застосувати передові підходи фронтенд-інженерії на практиці. Це розв'яже реальну соціальну проблему студентів міста Острог, надавши їм швидкий, естетично привабливий та централізований інструмент для візуального аналізу нерухомості та комунікації.

РОЗДІЛ 1

АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ТЕОРЕТИЧНІ ЗАСАДИ ПРОЄКТУВАННЯ КЛІЄНТСЬКОЇ ЧАСТИНИ ПЛАТФОРМИ «ПОШУК ЖИТЛА В ОСТРОЗІ»

1.1. Аналіз проблематики пошуку житла та вимоги до користувацького інтерфейсу

Процес пошуку орендованого житла є одним із найскладніших етапів для студентів та молодих спеціалістів під час переїзду до нового міста. Специфіка студентських містечок, яким є Острог, полягає у яскраво вираженій сезонності попиту та чітко окреслених географічних пріоритетах цільової аудиторії. Для студента ключовим критерієм вибору житла (поряд із вартістю) є його просторове розташування — насамперед близькість до навчальних корпусів Національного університету «Острозька академія», а також до ключових об'єктів міської інфраструктури (супермаркетів, зупинок транспорту).

Аналіз існуючих підходів до пошуку житла виявляє суттєвий розрив між потребами користувачів та функціоналом традиційних інформаційних платформ. З погляду користувацького досвіду (UX), поточний процес супроводжується низкою проблем:

- високе когнітивне навантаження під час просторової орієнтації. Більшість вебсервісів надають інформацію про розташування об'єкта у вигляді текстової адреси або статичного фрагмента карти в самому кінці оголошення. Користувач змушений самостійно копіювати адресу, відкривати сторонні картографічні сервіси (наприклад, Google Maps) та вручну будувати маршрути до університету, щоб зрозуміти реальні відстані.
- Неможливість наочного порівняння. При виборі з кількох варіантів орендар тримає відкритими десятки вкладок у браузері. Традиційні інтерфейси не дозволяють одночасно вивести два чи більше об'єктів на єдину інтерактивну площину, щоб візуально порівняти їхню віддаленість від критично важливих точок.

- Фрагментованість комунікації. Знаходження житла часто передбачає спільну оренду для оптимізації витрат. На існуючих платформах відсутня можливість обговорити оголошення з потенційними сусідами безпосередньо в контексті самого об'єкта нерухомості. Користувачі змушені пересилати посилання у сторонні месенджери (Telegram, Viber), створюючи розрізнені діалоги, що ускладнює відстеження інформації та уповільнює процес прийняття рішень.
- Перевантаженість інтерфейсів та застарілий візуальний стиль. Багато платформ не відповідають сучасним стандартам UI/UX дизайну, містять зайвий візуальний шум (рекламні банери, нерелевантні фільтри) та не використовують сучасні підходи до структурування контенту, що відволікає користувача від головної мети — швидкого аналізу варіантів.

Для ефективного розв'язання зазначених проблем було ухвалено рішення розробити спеціалізовану вебплатформу «UniStay». Виходячи з проведеного аналізу, було сформовано ключові вимоги до клієнтської частини (Frontend) та користувацького інтерфейсу системи:

- інтерактивна просторова візуалізація: інтерфейс повинен містити повноцінний картографічний модуль, який дозволяє не лише переглядати розташування квартир, але й візуалізувати вектори та точні відстані від об'єкта оренди до ключових локацій міста.
- Синхронне порівняння даних: необхідна реалізація механізму (як у вигляді структурованих списків, так і безпосередньо на карті), який дозволяє обрати декілька оголошень та наочно порівняти їхні інфраструктурні переваги в єдиному вікні без необхідності перемикання між сторінками.
- Централізація комунікації: система повинна мати інтегрований чат-модуль, реалізований у вигляді безшовного компонента інтерфейсу. Це дозволить орендарям, їхнім майбутнім співмешканцям та орендодавцю вести перемовини в реальному часі безпосередньо на платформі.

- Інтуїтивно зрозумілий кастомний дизайн: необхідно розробити унікальний візуальний стиль системи з використанням мікроанімацій та правильним розставленням акцентів (наприклад, динамічна зміна станів маркерів, фокусування на обраних елементах). Це дозволить логічно керувати увагою користувача.

Таким чином, розробка клієнтської частини має бути сфокусована на створенні єдиної, інтуїтивно зрозумілої платформи, яка закриватиме всі потреби студента: від просторового аналізу інфраструктури на карті до швидкої домовленості в чаті з орендодавцем.

1.2. Огляд аналогів та існуючих вебсервісів оренди нерухомості.

Розробка сучасного спеціалізованого вебдодатка вимагає детального аналізу існуючих лідерів ринку нерухомості. В Україні найпопулярнішими платформами для пошуку житла є універсальний маркетплейс OLX та спеціалізовані сервіси LUN (Flatfy) і DOM.RIA. Незважаючи на їхню технічну потужність та великі бази даних, з погляду клієнтської архітектури та користувацького досвіду (UX), вони мають суттєві обмеження під час вирішення специфічних завдань студентської аудиторії.

1.2.1. Платформа OLX

OLX є найбільшою дошкою оголошень в Україні. Платформа має звичний для більшості інтерфейс, проте її архітектура розрахована на максимально широкий спектр товарів, що негативно впливає на специфічний функціонал пошуку нерухомості.

- Переваги: звичний патерн взаємодії (list-view), базова система фільтрації, адаптивність під різні пристрої.
- Недоліки UI/UX та картографії: на платформі OLX карта є другорядним елементом. Вона реалізована у вигляді статичного блоку в правому боці сторінки оголошення. Користувач не може шукати житло безпосередньо на інтерактивній карті, не бачить точних відстаней до інфраструктурних об'єктів і позбавлений можливості одночасного просторового порівняння двох різних помешкань.

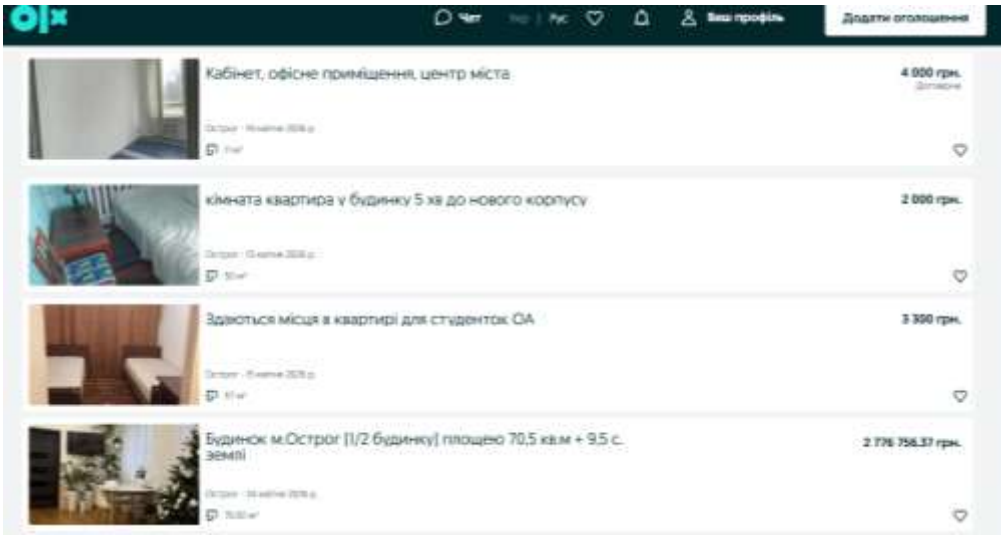


Рис. 1.1. Вигляд сторінки пошуку житла в м. Острог на платформі OLX

Джерело: https://www.olx.ua/uk/nedvizhimost/

Під час аналізу інтерфейсу було здійснено пошук житла в Острозі. Як видно з рис. 1.1, пошукові результати платформи є вкрай нерелевантними для студентської аудиторії. Серед десятків запропонованих системою варіантів лише одиничні оголошення (наприклад, оренда кімнати чи місця для студенток) фактично відповідають запиту. Більшість списку заповнена інформаційним шумом: пропозиціями щодо продажу будинків, земельних ділянок або оренди комерційних офісних приміщень. Для користувача необхідність вручну відфільтровувати 2-3 підходящі варіанти серед десятків нерелевантних суттєво погіршує досвід користування (User Experience). Додатково, відсутність спеціалізованої карти міста та неможливість створити груповий чат прямо на сторінці оголошення роблять цей сервіс неефективним для спільного пошуку житла студентами.

1.2.2. Платформи Flatfy (від LUN)

Сервіси екосистеми LUN (зокрема Flatfy як платформа для оренди) орієнтовані на більш сучасний підхід до пошуку нерухомості з активним використанням інтерфейсу карти.

- Переваги: сучасний мінімалістичний дизайн, інтерактивний пошук по карті.

- Недоліки UI/UX та картографії: хоча карта тут є центральним елементом, інтерфейс працює виключно як «вітрина». Користувач може клікнути на один маркер, але не може виділити кілька об'єктів для синхронного порівняння. Відсутній функціонал візуалізації відстаней (векторів) до кастомних точок (наприклад, до конкретного навчального корпусу). Відповідно, користувачу доводиться запам'ятовувати розташування одного варіанта, закривати його і відкривати інший, що є порушенням принципів зручного UX при складному виборі.

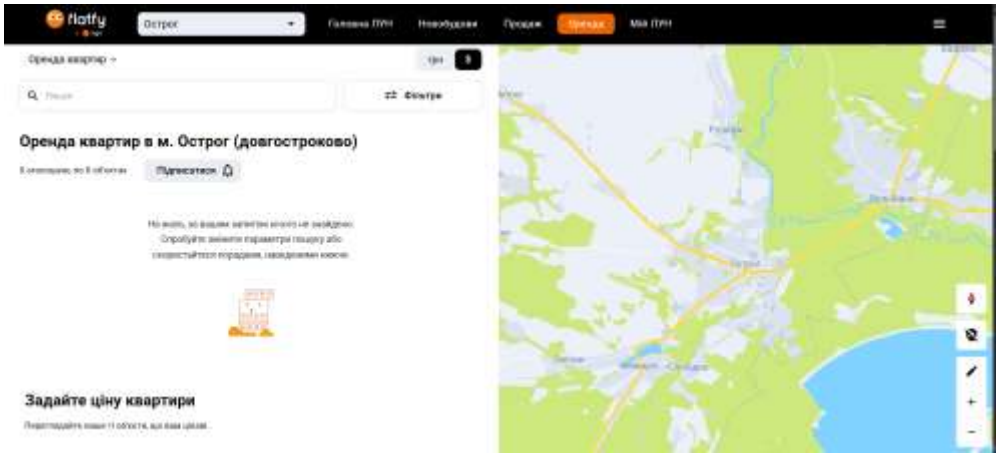


Рис. 1.2. Вигляд сторінки пошуку житла в м. Острог на платформі Flatfy

Джерело: https://flatfy.ua/uk/

Аналогічно до попереднього сервісу, інтерфейс Flatfy в умовах невеликого студентського містечка виявляється абсолютно неінформативним. Як видно з рис. 1.2, пошуковий запит щодо довгострокової оренди квартир в Острозі повертає порожній результат («0 оголошень по 0 об'єктах»). Відповідно, головний елемент платформи — інтерактивна карта — залишається порожнім і не виконує жодної корисної функції для користувача. Це яскраво демонструє, що великі платформи ігнорують специфіку малих міст із високим рівнем освітньої міграції.

Крім того, навіть за гіпотетичної наявності оголошень, платформа жодним чином не локалізована під потреби студентів: на карті не виділено будівлі Острозької академії як головного орієнтиру, а комунікація зводиться до звичайного телефонного дзвінка, без можливості обговорення варіантів у вбудованому чаті.

1.2.3. Платформа DIM.RIA

DIM.RIA є ще одним лідером українського ринку у сфері нерухомості. На відміну від OLX, цей сервіс є вузькоспеціалізованим і пропонує більш просунутий інструментарій для пошуку житла, включаючи окремий режим перегляду оголошень на карті.

- Переваги: наявність спеціалізованих фільтрів для нерухомості, можливість перемикання між режимом списку та картою, функція підписки на нові оголошення.
- Недоліки UI/UX та картографії: незважаючи на наявність вбудованої карти, логіка взаємодії з її елементами (маркерами) є неочевидною. Під час тестування інтерфейсу виявлено проблеми з маршрутизацією (навігацією) користувача: клік по маркеру на карті не завжди інтуїтивно переводить до деталей оголошення. Крім того, як і в попередніх сервісах, тут повністю відсутній функціонал для інтерактивного порівняння відстаней між кількома об'єктами. Карта виконує роль лише статичного географічного довідника, а не аналітичного інструмента.

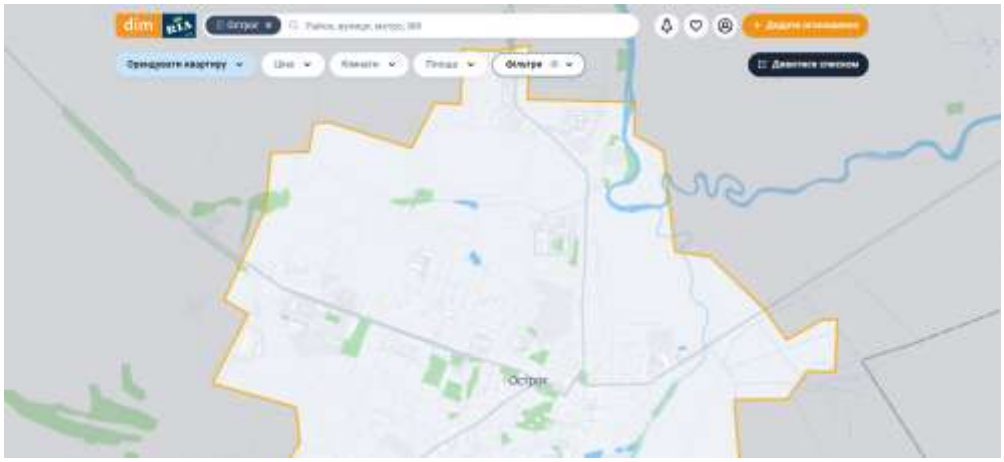


Рис. 1.3. Результати пошуку оренди квартири в м. Острог на карті на платформі DIM.RIA

Джерело: https://dom.ria.com/uk/

Рис. 1.4. Результати пошуку оренди кімнати в м. Острог на платформі DIM.RIA

Джерело: https://dom.ria.com/uk/

Результати локального пошуку по місту Острог (рис. 1.3 та рис. 1.4) вкотре підтверджують тенденцію ігнорування малих студентських міст універсальними сервісами. Платформа повертає нульовий результат як у списковому відображенні, так і на карті. Для студента це означає повну неможливість знайти житло або оцінити його просторове розташування відносно університету. Організація спільного проживання через цю платформу також неможлива через відсутність внутрішніх групових чатів.

Підсумовуючи результати аналізу існуючих рішень, можна констатувати, що загальнонаціональні платформи (OLX, Flatfy, DIM.RIA) не здатні повною мірою задовольнити специфічні потреби студентської аудиторії через обмеження користувацького інтерфейсу та архітектури. Хоча спеціалізовані сервіси мають базовий картографічний функціонал, жоден із проаналізованих ресурсів не пропонує інструментів для синхронного порівняння характеристик об'єктів. Користувач позбавлений можливості зіставити параметри кількох оголошень в єдиному вікні (як на рівні текстових характеристик, так і на рівні візуалізації відстаней до навчальних корпусів), що змушує його самостійно структурувати інформацію поза межами сервісу. Крім того, інтерфейси цих платформ розраховані виключно на індивідуальну комунікацію, що унеможливлює ефективний груповий підбір житла. Проєктована платформа «UniStay» заповнює ці прогалини, пропонуючи унікальний картографічний модуль та окремий функціонал детального порівняння оголошень, що забезпечує новий рівень користувацького досвіду (UX) та автоматизує процес прийняття рішень.

1.3. Теоретичні аспекти проєктування UI/UX для картографічних вебдодатків

Проєктування користувацьких інтерфейсів (UI) та користувацького досвіду (UX) для вебдодатків, де центральним елементом є інтерактивна карта, суттєво відрізняється від створення класичних текстових або спискових платформ. Карта є складним інформаційним середовищем, яке вимагає від користувача високої концентрації уваги та просторового мислення. Тому головним завданням UI/UX дизайну в таких проєктах є зниження когнітивного навантаження та інтуїтивне управління увагою користувача.

В основу розробки клієнтської частини платформи «UniStay» було покладено наступні теоретичні принципи UI/UX дизайну та психології сприйняття:

1. Закон Міллера та мінімізація когнітивного навантаження. Згідно з цим психологічним законом, короткочасна пам'ять людини здатна одночасно утримувати лише обмежену кількість елементів (в середньому 7±2). У контексті пошуку житла це означає, що користувач не може запам'ятати всі характеристики квартири (ціну, площу, умови оренди), щоб порівняти їх з іншим варіантом, просто перемикаючись між сторінками. Саме цей принцип теоретично обґрунтовує необхідність створення розробленої сторінки для синхронного (side-by-side) порівняння параметрів об'єктів. Це дозволяє розвантажити пам'ять користувача, переклавши задачу зіставлення даних на інтерфейс програми.
2. Візуальна ієрархія та просторове акцентування. Для ефективної роботи з картою необхідно чітко розмежовувати головну та другорядну інформацію (позбутися візуального шуму). На картографічній підкладці увагу студента повинні привертати лише інтерактивні елементи (маркери квартир) та ключові орієнтири (корпуси Національного університету «Острозька академія»). Використання продуманої кольорової палітри, тіней та різних розмірів іконок дозволяє створити правильну ієрархію: користувач спочатку бачить цільову локацію, потім — доступні варіанти навколо неї, і лише після

взаємодії з маркером отримує детальну інформацію у спливаючому вікні (Popup).

3. Мікроанімації як інструмент зворотного зв'язку. Взаємодія з картографічним інтерфейсом має бути відчутною. Застосування мікроанімацій (наприклад, зміна кольору або розміру маркера при наведенні курсору, підсвічування обраного оголошення у списку поруч із картою) інформує користувача про те, що система прийняла його дію. Це суворо відповідає одному з головних евристичних принципів Якоба Нільсена — «Видимість стану системи» (Visibility of system status).
4. Закон Фітца та кластеризація даних. Цей закон визначає, що час, необхідний для досягнення цілі, залежить від відстані до неї та її розміру. Для зручності роботи з картою інтерактивні елементи (маркери) мають бути достатнього розміру для точного кліку мишею. У випадках, коли кілька об'єктів оренди знаходяться в одному будинку, інтерфейс повинен передбачати механізми кластеризації або рознесення маркерів, щоб уникнути їх накладання (overlapping) та унеможливлення вибору.
5. Збереження просторового контексту (технологія SPA). З погляду UX, при роботі з картографічними даними критично важливо уникати повних перезавантажень сторінки. Якби для перегляду деталей оголошення користувачу доводилося переходити на іншу сторінку браузера, він би щоразу втрачав орієнтацію в просторі. Оскільки платформа «UniStay» розроблена за принципом Single Page Application (на базі бібліотеки React), інтерфейс дозволяє миттєво відображати потрібну інформацію. Зокрема, при кліку на маркер детальна інформація про об'єкт оренди з'являється у спливаючому вікні (Popup) безпосередньо над картою. Це дозволяє залишити карту активною, зберігаючи візуальний контекст та обраний масштаб.

Використання цих принципів дозволяє перетворити складний процес аналізу просторових даних на зручний та інтуїтивно зрозумілий інструмент. Замість того,

щоб боротися з незручним інтерфейсом, користувач платформи «UniStay» може сфокусуватися безпосередньо на виборі найкращого варіанта житла.

1.4. Обґрунтування вибору технологічного стека для розробки Frontend-архітектури

При проєктуванні архітектури клієнтської частини платформи «Пошук житла в Острозі» критично важливим етапом був вибір базових технологій. Сучасна веб-розробка вимагає використання інструментів, які забезпечують високу продуктивність, масштабованість та зручність інтеграції спеціалізованих модулів (зокрема, картографічних та комунікаційних).

1.4.1. Вибір фронтенд-фреймворку та інструментів збірки

Сучасна екосистема веб-розробки пропонує три основні рішення для створення односторінкових застосунків (SPA): React.js, Angular та Vue.js. Після проведення порівняльного аналізу було обрано React.js.

- Angular був визнаний надлишковим для даного проєкту через жорстку структуру та складність інтеграції з легкими картографічними бібліотеками.
- Vue.js характеризується зручною системою реактивності, проте має меншу екосистему готових рішень для геопросторових даних у порівнянні з React.
- React.js забезпечує необхідний баланс між продуктивністю та гнучкістю. Використання віртуального DOM (Virtual DOM) гарантує високу швидкість при динамічному оновленні елементів на карті.

Додаткові переваги React.js для проєкту:

- Компонентна архітектура: дозволяє легко масштабувати проєкт та повторно використовувати UI-елементи (наприклад, картки оголошень).
- Ефективне управління станом: використання вбудованого Context API дозволило реалізувати механізм синхронного порівняння об'єктів без встановлення важких зовнішніх бібліотек (наприклад, Redux).

- Для ініціалізації та збірки проєкту замість застарілого Create React App було використано інструмент Vite. Він працює на базі нативних ES-модулів, що забезпечує миттєвий запуск локального сервера та надшвидке гаряче перезавантаження модулів (HMR), суттєво пришвидшуючи процес розробки.

1.4.2. Картографічна система та геодані

Оскільки центральним елементом платформи є інтерактивна карта міста Острог, критично важливо було обрати сервіс із підтримкою кастомних шарів, що не вимагає великих фінансових витрат. Розглядалися Google Maps API, Mapbox GL JS та Leaflet:

1. Google Maps API: має високу деталізацію, але використовує жорстку комерційну політику та є досить «важким» для SPA.
2. Mapbox GL JS: пропонує векторні тайли та WebGL, але має ліміти безкоштовного використання та складніший процес стилізації DOM-маркерів.
3. Leaflet (обрано): відкрита (Open Source) бібліотека, яка відрізняється легкістю (близько 39 КБ) та повною сумісністю з React через бібліотеку react-leaflet.

Вибір на користь Leaflet дозволив:

- Створити повністю безкоштовне рішення у поєднанні з тайлами OpenStreetMap (OSM), що актуально для студентського проєкту.
- Реалізувати специфічні елементи: Custom DivIcons (маркери у вигляді цінових тегів) та вектори (Polyline) для візуалізації відстаней до орієнтирів (корпусів Академії).
- Застосувати плагін react-leaflet-cluster для кластеризації маркерів-цінників, щоб уникнути їх візуального накладання при зміні масштабу.

1.4.3. Взаємодія з сервером, реальний час та безпека

Для забезпечення комунікації клієнтської частини з бекендом було обрано наступний стек:

- Axios: HTTP-клієнт для виконання REST API запитів. Він забезпечує зручну роботу з перехоплювачами (interceptors), що необхідно для автоматичного додавання токенів авторизації до кожного запиту.
- JWT-decode: бібліотека для безпечного розшифрування JSON Web Tokens на стороні клієнта, що дозволяє керувати сесіями користувачів та доступом до захищених маршрутів.
- SignalR (@microsoft/signalr): бібліотека для організації двостороннього зв'язку в реальному часі за протоколом WebSockets. Використання SignalR є критично важливим для модуля вбудованого чату, дозволяючи орендарям та орендодавцям миттєво обмінюватися повідомленнями без перезавантаження сторінки.

1.4.4. Маршрутизація та робота з медіа

- React Router DOM: стандартний інструмент для реалізації SPA-навігації, який дозволяє перемикатися між сторінками (карта, список оголошень, профіль, чат) без перезавантаження браузера, зберігаючи просторовий контекст користувача.
- Cloudinary: хмарний сервіс для зберігання та оптимізації медіафайлів. Оскільки оголошення про оренду містять велику кількість фотографій, делегування їх зберігання та автоматичного стисення сторонньому сервісу дозволяє знизити навантаження на основний сервер та пришвидшити завантаження зображень у клієнтському додатку.

Висновки до розділу 1

У першому розділі кваліфікаційної роботи проведено комплексний аналіз предметної області, досліджено наявні ринкові аналоги, а також обґрунтовано теоретичні та інженерно-технологічні засади проєктування клієнтської частини платформи «UniStay». Результати проведеного дослідження дозволяють зробити такі висновки:

- Ідентифіковано ключові проблеми ринку та сформовано вимоги до інтерфейсу. Дослідження специфіки пошуку житла в умовах малого студентського міста (на прикладі м. Острог) виявило гостру потребу

цільової аудиторії в оцінці просторового розташування об'єктів відносно Національного університету «Острозька академія». Визначено головні UX-недоліки традиційних платформ: високе когнітивне навантаження при просторовій орієнтації, відсутність інструментів наочного зіставлення варіантів та фрагментованість комунікації. На цій основі сформовано вимоги до платформи «UniStay», які передбачають інтерактивну карту з візуалізацією відстаней, модуль синхронного порівняння та інтегрований чат.

- Доведено неефективність загальнонаціональних агрегаторів для локального пошуку. На основі аналізу платформ OLX, Flatfy та DIM.RIA встановлено, що великі комерційні сервіси ігнорують специфіку малих міст із високим рівнем освітньої міграції, повертаючи нульові або нерелевантні результати пошуку в Острозі. Крім того, архітектура їхніх інтерфейсів обмежує користувача, використовуючи карту лише як статичний довідник, і не надає можливості для аналітичного side-by-side порівняння локацій чи ведення групових обговорень.
- Систематизовано теоретичні принципи UI/UX дизайну. В основу інтерфейсу «UniStay» покладено фундаментальні закони психології сприйняття. Застосування закону Міллера дозволило мінімізувати когнітивне навантаження шляхом розробки сторінки порівняння параметрів. На основі закону Фітца обґрунтовано необхідність кластеризації маркерів-цінників для уникнення їх накладання. Проєктування за принципом Single Page Application (SPA) забезпечило збереження просторового контексту, дозволяючи взаємодіяти з деталями оголошень у спливаючих вікнах (Popup) без повного перезавантаження сторінки та втрати фокуса карти.
- Обґрунтовано вибір прогресивного технологічного стека. Для реалізації Frontend-архітектури обрано бібліотеку React.js (через її компонентний підхід та ефективність Virtual DOM) у зв'язці з надшвидким інструментом збірки Vite. Картографічний модуль реалізовано на базі відкритої

бібліотеки Leaflet та тайлів OpenStreetMap, що дозволило створити безкоштовне, кастомізоване рішення з підтримкою плагіна кластеризації маркерів. Для організації обміну повідомленнями в реальному часі інтегровано бібліотеку SignalR, а роботу з великим масивом медіаданих делеговано хмарному сервісу Cloudinary, що сукупно забезпечує високу продуктивність, безпеку (JWT-decode, Axios Interceptors) та якісний користувацький досвід.

РОЗДІЛ 2

ПРОГРАМНО-ТЕХНІЧНА РЕАЛІЗАЦІЯ ТА UI/UX ДИЗАЙН КЛІЄНТСЬКОЇ ЧАСТИНИ ПЛАТФОРМИ «UNISTAY»

2.1. Архітектурна структура додатка та конфігурація середовища розробки

Розробка клієнтської частини платформи «UniStay» реалізована на основі сучасного інженерного підходу, який поєднує компонентно-орієнтовану архітектуру бібліотеки React.js (версії 19.1), строгу статичну типізацію мови TypeScript (версії 5.8) та надшвидке середовище збірки Vite (версії 6.3). Такий технологічний симбіоз дозволив створити гнучку, стійку до помилок та легку в масштабуванні структуру вебзастосунку.

2.1.1. Конфігурація середовища розробки та інструментів збірки

Традиційні інструменти збірки (наприклад, Webpack) під час розробки великих картографічних SPA-додатків часто стикаються із проблемою низької швидкості гарячого перезавантаження модулів (HMR), оскільки вони потребують повного перезбирання всього проєкту при зміні одного файлу. Для платформи «UniStay» цю проблему вирішено шляхом інтеграції інструменту Vite.

Конфігурація середовища базується на архітектурі нативних ES-модулів ("type": "module" у файлі конфігурації проєкту). Основні інженерні скрипти автоматизації розробки, закладені в систему, включають:

- npm run dev — запуск локального сервера розробки з підтримкою миттєвого відображення змін у браузері без втрати поточного стану компонентів карти чи чату.
- npm run build — оптимізована компіляція проєкту за допомогою збирача Rollup, що трансформує TypeScript та JSX-код у мініфіковані, очищені від зайвих логів та коментарів статичні файли (HTML, CSS, JS), готові до деплою на сервер.
- npm run lint — автоматизована статична перевірка якості та стилістики коду за допомогою утиліти ESLint (версії 9.25), що гарантує дотримання правил написання React-хуків та запобігає виникненню витоків пам'яті.

Використання TypeScript забезпечує розробку системи на рівні промислових стандартів. Статична типізація даних (опис інтерфейсів для оголошень житла, повідомлень у чаті та профілів користувачів) дозволяє виявляти помилки невідповідності типів ще на етапі написання коду, повністю нівелюючи ризик виникнення критичних помилок типу undefined у браузері кінцевого користувача.

2.1.2. Архітектурна структура та організація каталогу проєкту

Для забезпечення високої модульності, масштабованості та зручності паралельної розробки різних функціональних блоків, вихідний код платформи «UniStay» організовано за тематичними директоріями в каталозі src. Кожна папка виконує суворо визначену архітектурну роль, що відповідає принципу розділення відповідальності (Separation of Concerns).

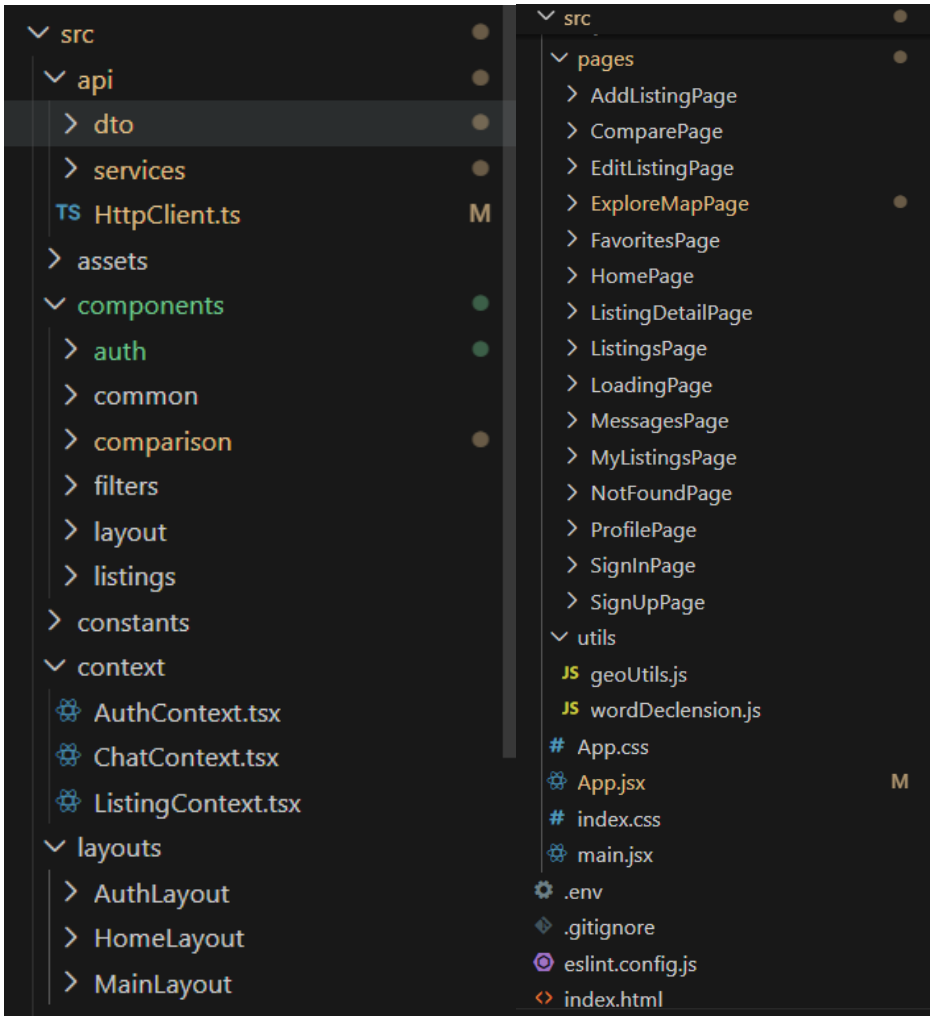


Рис. 2.1. Архітектурна структура папок клієнтської частини проєкту

На основі розробленої структури проєкту (рис. 2.1), організація каталогу має наступний вигляд:

- api/ — централізований вузол для взаємодії з серверною частиною. Містить конфігурацію базового клієнта HttpClient.ts та піддиректорії:
    - dto/ — опис структур даних (Data Transfer Objects), таких як ListingDto, ChatDto, UserDto, що гарантує цілісність даних при обміні з бекендом;
    - services/ — класи або функції для виконання конкретних запитів (наприклад, AuthService, ListingService), що інкапсулюють логіку HTTP-викликів.
- assets/ — статичні ресурси додатка, зокрема папка images для графічних елементів.
- components/ — бібліотека UI-компонентів, розділена на функціональні групи:
    - auth/ — компоненти, пов'язані з безпекою та входом, включаючи механізм захисту маршрутів ProtectedRoute.jsx;
    - common/ — атомарні перевикористовувані елементи інтерфейсу (Button, Input, Card, ConfirmModal);
    - comparison/ — спеціалізовані блоки для модуля порівняння житла (секції інфраструктури, цін, зручностей);
    - layout/ — композиційні елементи, такі як Navbar та ComparisonBar;
    - listings/ — компоненти для відображення карток оголошень та форм їх створення/редагування.
- constants/ — містить статичні конфігурації, наприклад, landmarks.js із координатами ключових об'єктів Острога (корпусів Академії) для відображення на карті.
- context/ — реалізація глобального управління станом (State Management) за допомогою React Context API. Включає AuthContext для сесій, ChatContext для повідомлень та ListingContext для роботи зі списками оголошень.

- layouts/ — високорівневі компоненти-шаблони (MainLayout, AuthLayout), що визначають загальну структуру сторінок (наявність хедеру, футеру чи бічних панелей).
- utils/ — допоміжні інструменти, зокрема geoUtils.js для математичних розрахунків на географічній площині та wordDeclension.js для коректного відмінювання українських слів в інтерфейсі.

Окрему увагу приділено організації директорії pages/, яка побудована за принципом самодостатності модулів. Кожна папка сторінки містить основний компонент, файл стилів (.module.css) та специфічні підкомпоненти:

- ExploreMapPage/ — включає MapController.jsx та AnalysisPanel для просторового аналізу.AddListingPage/ та EditListingPage/ — містять інструменти AddressPicker, AmenitySection та ImageUploader.
- ListingDetailPage/ — складається з модулів ListingGallery, ListingMainInfo, LocationMap та ReviewSection.
- MessagesPage/ — реалізує інтерфейс чату через компоненти ChatSidebar, ChatWindow та ChatMessage.

Файли в корені директорії src, такі як App.jsx та main.jsx, виконують роль точок входу, де відбувається ініціалізація маршрутизації та монтування React-дерева в DOM-структуру браузера. Така чітка ієрархія папок дозволяє уникнути дублювання коду та забезпечує швидку навігацію по проєкту при впровадженні нових функцій.

2.2. Клієнтська маршрутизація та реалізація захищених маршрутів

Для створення безшовного досвіду користувача (SPA — Single Page Application) у платформі «UniStay» впроваджено систему клієнтської маршрутизації на базі бібліотеки React Router DOM. Це дозволяє миттєво перемикатися між модулями системи без перезавантаження сторінки, зберігаючи при цьому стан додатка.

2.2.1. Архітектура маршрутизації та використання Layout-компонентів

У файлі App.jsx реалізовано ієрархічну структуру маршрутів, яка базується на використанні спеціалізованих компонентів-шаблонів (Layouts). Конфігурацію маршрутизації та структуру вкладених шляхів представлено у додатку А. Такий

підхід дозволяє винести спільні елементи інтерфейсу у батьківські компоненти, уникаючи їх дублювання на кожній сторінці:

- HomeLayout — забезпечує унікальну візуальну подачу головної сторінки;
- MainLayout — інтегрує компоненти Navbar та ComparisonBar для більшості функціональних сторінок;
- AuthLayout — фокусує увагу користувача на формах входу та реєстрації.

2.2.2. Механізм захисту приватних даних (ProtectedRoute)

Для забезпечення безпеки розроблено сервісний компонент ProtectedRoute, який виконує функцію високорівневого перехватника для маршрутів, що потребують автентифікації. Логіка роботи компонента базується на інтеграції з AuthContext. У разі відсутності авторизації компонент використовує об'єкт Maps для декларативного перенаправлення на сторінку /signin. Програмна реалізація компонента захисту представлена у додатку Б.

2.2.3. Ієрархія контекстних провайдерів

Для забезпечення сторінок даними в корені додатка розгорнуто каскад провайдерів стану: AuthProvider, ChatProvider та ListingProvider. Така послідовність гарантує, що сервіси завжди мають доступ до актуальних даних про сесію користувача.

2.3. Управління глобальним станом та логіка порівняння об'єктів нерухомості

У сучасних SPA-застосунках ефективне управління даними є критично важливим для забезпечення продуктивності. Для платформи «UniStay» замість використання громіздких зовнішніх бібліотек (наприклад, Redux) було обрано React Context API у поєднанні з хуком useCallback для оптимізації рендерингу. Центральним вузлом управління даними про житло є ListingContext.

2.3.1. Функціональна роль ListingProvider

Компонент ListingProvider (код наведено у додатку В) виконує роль єдиного джерела істини (Single Source of Truth) для всіх модулів, що працюють з оголошеннями. Його основними завданнями є:

- Синхронізація з API: автоматичне завантаження повного списку оголошень та ідентифікаторів «обраного» при ініціалізації додатка або зміні статусу авторизації.
- Управління життєвим циклом даних: методи addListing та deleteListing не лише взаємодіють із сервером, а й миттєво оновлюють локальний стан (UI State), що дозволяє користувачу бачити зміни без примусового оновлення сторінки.
- Реактивне відстеження «обраного»: логіка toggleFavorite реалізована з урахуванням асинхронності — стан оновлюється лише після підтвердження від сервера, що гарантує цілісність даних.

2.3.2. Алгоритмічна реалізація модуля порівняння

Однією з ключових функцій платформи є можливість порівняння об'єктів нерухомості (див. рис. 2.2). Логіка цього модуля інкапсульована в методі toggleCompare. Основними інженерними рішеннями тут є:

1. Обмеження вибірки: реалізовано програмне обмеження (валідація), яке дозволяє додавати до списку порівняння не більше двох об'єктів одночасно. Це продиктовано вимогами UX/UI для зручного відображення характеристик «пліч-о-пліч» на екранах мобільних пристроїв та ПК.
2. Обчислювальні властивості (Derived State): замість збереження повних об'єктів у списку порівняння, у стані зберігаються лише їхні ідентифікатори (compareIds). Масив об'єктів compareListings формується динамічно за допомогою методу filter. Це дозволяє уникнути дублювання даних у пам'яті та гарантує, що в порівнянні завжди відображається актуальна інформація (ціна, статус тощо).

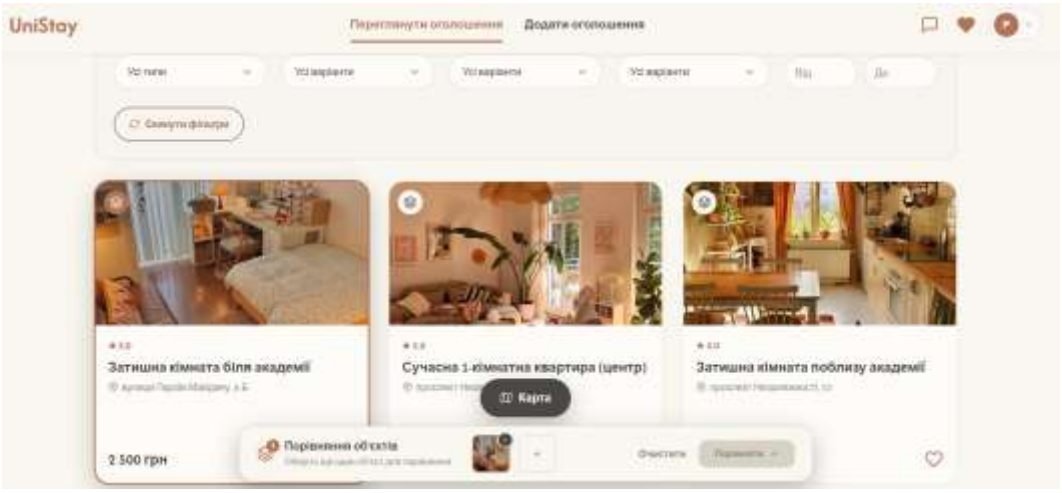


Рис. 2.2. Візуалізація додавання оголошення до списку порівняння

2.3.3. Оптимізація продуктивності через мемоїзацію

Для запобігання зайвих повторних запитів до сервера та зайвих циклів рендерингу використано хук useCallback. Це дозволяє стабілізувати посилання на функції fetchListings та fetchFavoriteIds, що є критичним при передачі цих функцій як залежностей в інші useEffect або дочірні компоненти.

Така архітектура управління станом забезпечує високу швидкість відгуку інтерфейсу та дозволяє легко масштабувати функціонал (наприклад, додавати нові фільтри або критерії порівняння) без перебудови всього додатка.

2.4. Розробка інтерактивного картографічного модуля та засобів візуального порівняння об'єктів

Ключовою особливістю розробленої платформи є система наочного аналізу об'єктів нерухомості, яка дозволяє користувачеві приймати рішення на основі точних геопросторових та технічних характеристик. Функціонал реалізовано через два взаємопов’язані інструменти: інтерактивну карту дослідження та сторінку деталізованого порівняння.

2.4.1. Реалізація геопросторового аналізу на базі Leaflet

Інтерактивна карта побудована з використанням бібліотеки React-Leaflet (див. рис. 2.3). Основні технічні рішення модуля включають:

- Ефективне відображення даних: використання MarkerClusterGroup дозволяє групувати об'єкти при віддаленні карти, що запобігає

візуальному нагромадженню та забезпечує чистоту інтерфейсу (див. рис. 2.3).

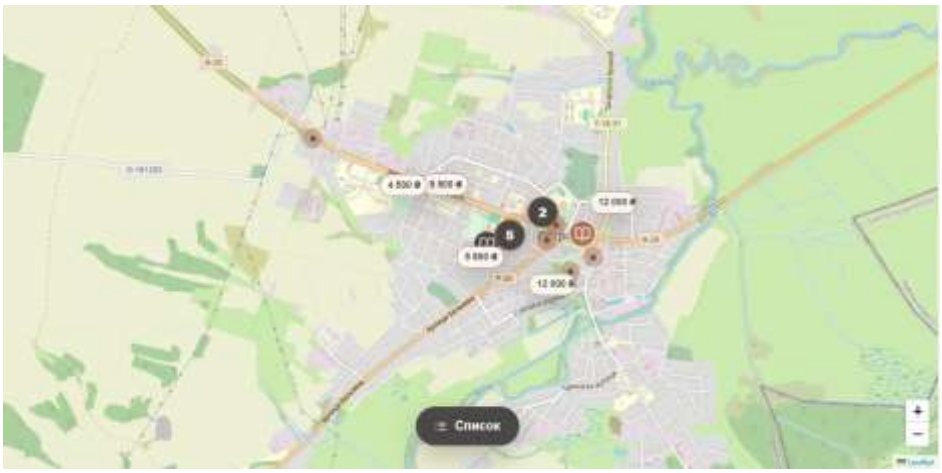


Рис. 2.3. Загальний вигляд модуля карти з використанням механізму кластеризації об'єктів

- Візуалізація інфраструктурних зв'язків: при виборі ключових об'єктів міста (наприклад, корпусів Острозької академії) система динамічно будує анімовані лінії (Polyline), що з’єднують їх із вибраними квартирами (див. рис. 2.4). Це дозволяє користувачу наочно оцінити відносну віддаленість об'єктів.

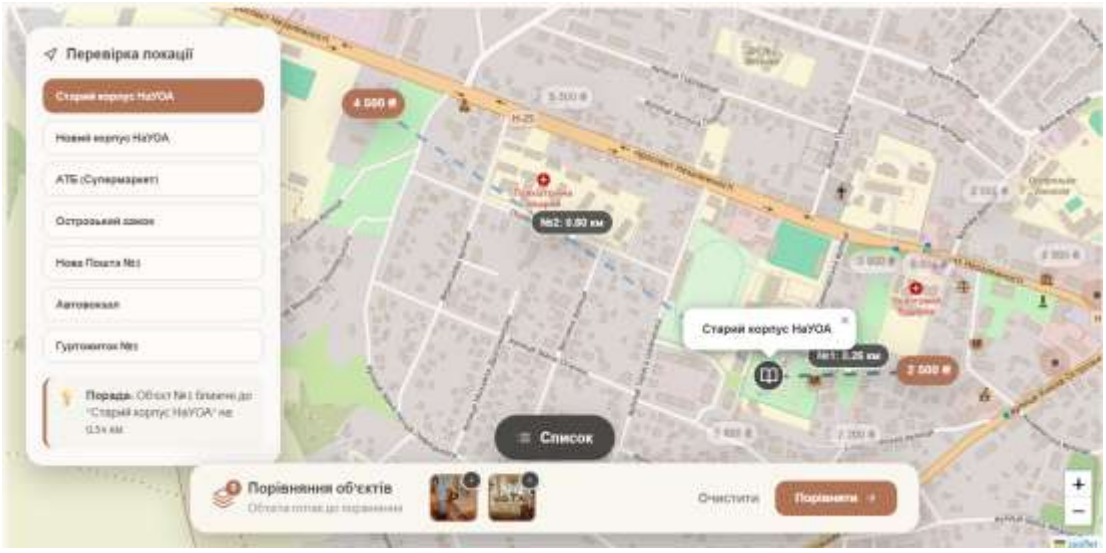


Рис. 2.4. Візуалізація аналізу відстані до ключових інфраструктурних об'єктів

- Математичне обчислення відстаней: розрахунок проводиться в реальному часі за допомогою тригонометричних функцій на основі координат (широти та довготи), що забезпечує користувача точними

даними у кілометрах без необхідності використання зовнішніх картографічних сервісів.

2.4.2. Механізм комплексного порівняння характеристик об’єктів

Сторінка порівняння виконує роль аналітичного центру, який структурує дані за чотирма векторами:

1. Фінансові умови: секція PriceSection візуалізує цінову різницю за допомогою порівняльної шкали. Система автоматично маркує вигіднішу пропозицію спеціальним індикатором, що полегшує швидке сприйняття інформації (див. рис. 2.5).

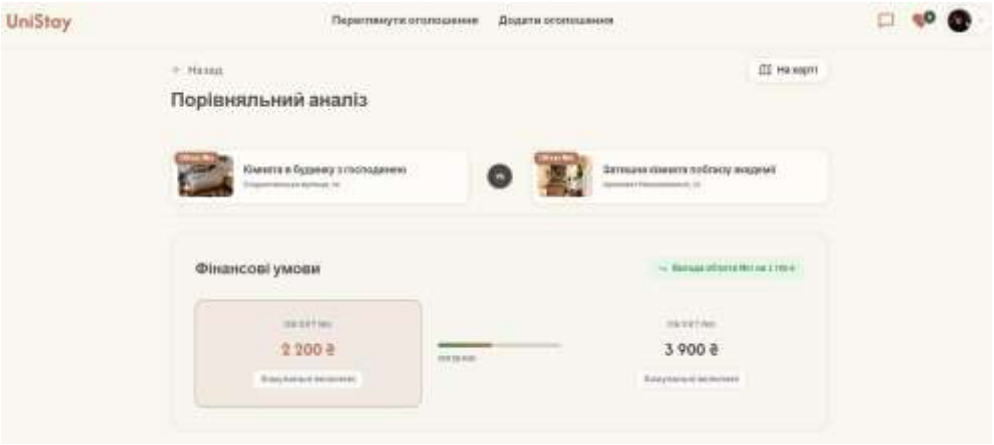


Рис. 2.5. Інтерфейс порівняння фінансових умов

2. Інфраструктурна перевага: у компоненті InfrastructureSection реалізовано систему порівняльних індикаторів (progress bars). Об’єкт, що знаходиться ближче до обраної точки, виділяється акцентним кольором, що робить аналіз локації інтуїтивно зрозумілим.
3. Диференціація зручностей: логіка порівняння базується на групуванні характеристик. Користувач одразу бачить, які зручності є спільними, а які — унікальними для кожного варіанту, що є критичним при виборі між схожими за ціною об'єктами.
4. Часова актуальність: модуль аналізує дату публікації оголошень, інформуючи користувача про термін перебування пропозиції на ринку.

2.4.3. Оптимізація користувацького досвіду (UX)

При розробці інтерфейсу основну увагу було приділено створенню «user-friendly» середовища через систему візуальних підказок та продуману колірну схему:

- Колірне акцентування «переможця»: для полегшення швидкого аналізу в системі впроваджено логіку автоматичного підсвічування переваг. В блоках порівняння цін та інфраструктури об'єкт, що має кращі показники (нижча ціна або менша відстань до обраного об’єкту), маркується акцентним кольором (winnerFill). Це дозволяє користувачеві на підсвідомому рівні зчитувати вигіднішу пропозицію, не вчитуючись у кожну цифру (див. рис. 2.6).
- Інформаційна підтримка (Contextual Hints): оскільки користувач може потрапити на сторінку порівняння безпосередньо зі списку оголошень, мимохідь карти, у блоці «Інфраструктура» реалізовано банер-пояснення (mapBanner). Він не лише інформує про можливість візуалізації відстаней на карті, а й слугує швидким переходом (Call-to-Action), що забезпечує безперервність користувацького досвіду (див. рис. 2.6).

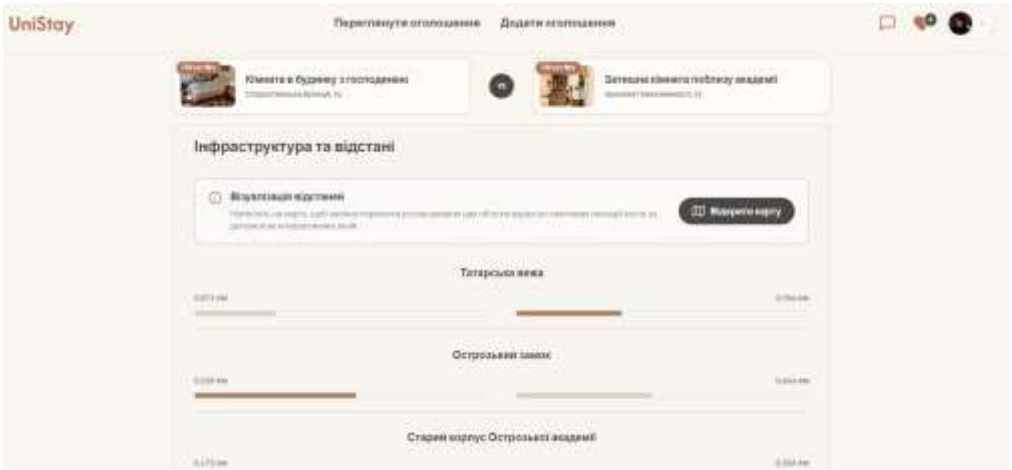


Рис. 2.6. Інтерфейс порівняння інфраструктури та відстані за допомогою ацентування кольором та контекстних підказок.

- Візуальна ієрархія та стан фокусу: на інтерактивній карті використано ефект «приглушення» неактивних об'єктів. Коли користувач аналізує конкретну пару квартир, решта маркерів набувають стану priceTagUnfocused (зменшується яскравість та насиченість), що допомагає сфокусувати увагу на головному завданні порівняння.
- Інтерактивний зворотний зв'язок: використання анімованих ліній (animatedLine) та тултипів, що з’являються при взаємодії, створює

відчуття "живого" інтерфейсу. Користувач отримує підтвердження кожної своєї дії, що робить процес складного вибору житла більш комфортним та зрозумілим.

2.5. Інтеграція комунікаційного сервісу в реальному часі

Для забезпечення швидкої комунікації між користувачами платформи було розроблено модуль обміну повідомленнями, що базується на технології SignalR. Це дозволило створити інтерактивне середовище для обговорення об'єктів нерухомості з підтримкою групових та приватних чатів.

2.5.1. Архітектура сторінки повідомлень та керування станом

Центральним елементом модуля є компонент MessagesPage, який координує роботу чатів через спеціалізований ChatContext. На основі аналізу коду можна виділити такі архітектурні рішення:

- Динамічна ініціалізація: при завантаженні сторінки сервіс ChatService.getUserChats() отримує список доступних діалогів. Якщо користувач переходить до повідомлень безпосередньо з оголошення, система автоматично активує потрібний чат через обробку location.state.
- Сортування та фільтрація: компонент ChatSidebar реалізує useMemo для ефективного сортування чатів за часом останнього оновлення (updatedAt) та фільтрації за пошуковим запитом користувача.
- Синхронізація з'єднання: використання SignalR-хабу дозволяє підписуватися на події друку тексту (NotifyTyping), що візуалізується в інтерфейсі через typingIndicator.

2.5.2. Функціонал інтерактивного вікна чату

Компонент ChatWindow виконує роль основного інтерфейсу взаємодії (див. рис. 2.7). В ньому реалізовано:

- Повний цикл обробки повідомлень (CRUD): на відміну від базових чатів, розроблена система дозволяє користувачеві не лише надсилати, а й редагувати та видаляти повідомлення. Оновлення інтерфейсу відбувається миттєво через оптимістичне оновлення стейту setActiveChatMessages.

- Автоматична навігація: використання useRef та useEffect забезпечує "м'яке" прокручування (scrollIntoView) до останнього повідомлення при отриманні нових даних.
- Візуальна диференціація: компонент ChatMessage розділяє повідомлення на "свої" та "чужі", відображаючи аватари відправників, мітки про редагування (editedAt) та точний час відправлення.

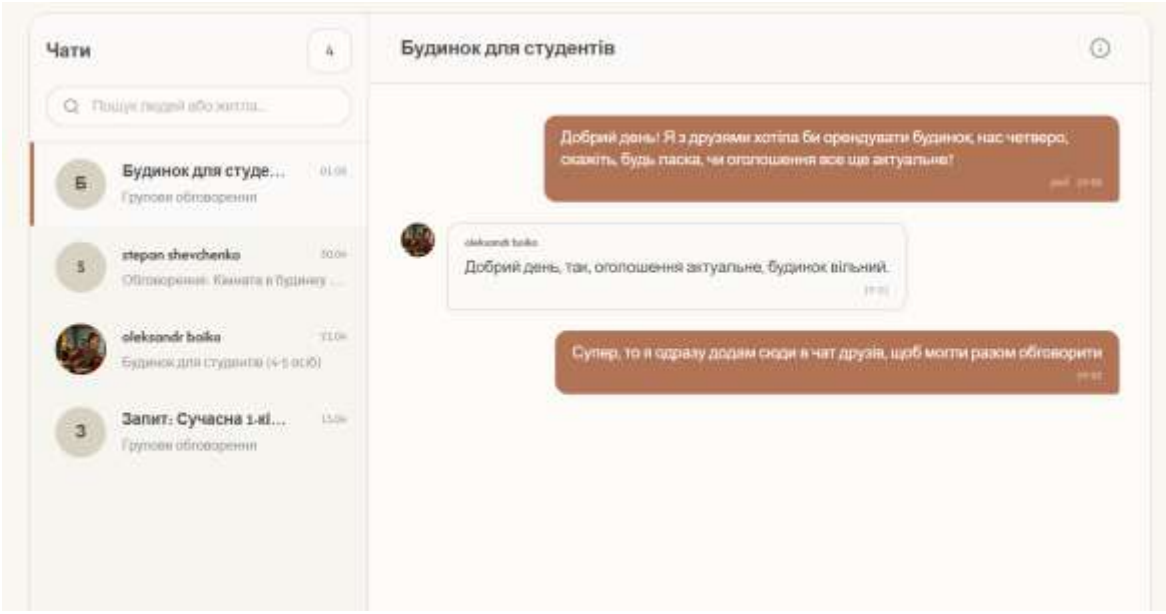
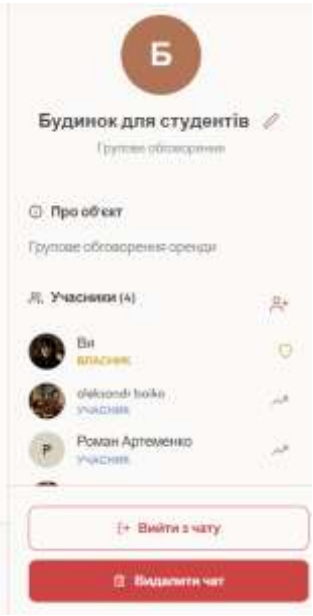


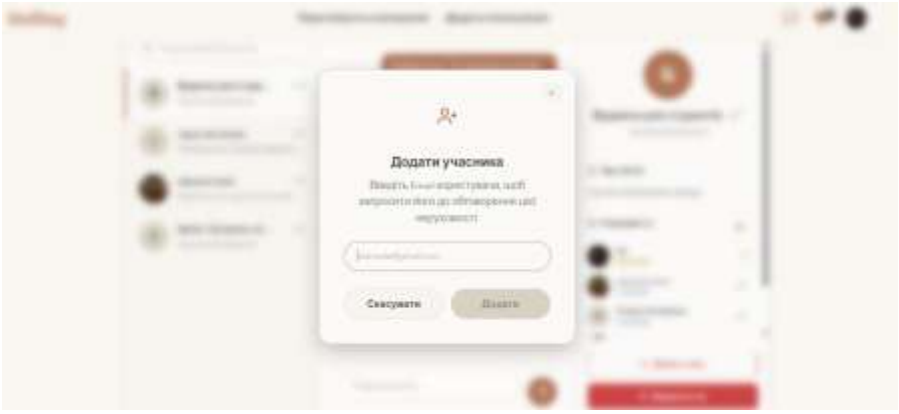
Рис. 2.7. Інтерфейс модуля обміну повідомленнями

2.5.3. Керування доступом та адміністрування чатів

Особливістю реалізації є компонент ChatDetails (див. рис. 2.8 – 2.10), який перетворює звичайний чат на інструмент спільного управління:

- Рольова модель: система розрізняє ролі "Власник", "Адмін" та "Учасник". Власник чату має ексклюзивні права на видалення всієї переписки або призначення адміністраторів через метод changeMemberRole.
- Керування учасниками: реалізовано модальні вікна AddMemberModal та EditChatModal, які дозволяють запрошувати нових користувачів за Email-адресою, змінювати назву об'єкта обговорення та його опис.
- Гнучкість типів: логіка компонента адаптується під тип чату (Private або Group). Для приватних діалогів інтерфейс автоматично підтягує дані співрозмовника, а для групових — відображає спільну інформацію про об'єкт нерухомості.





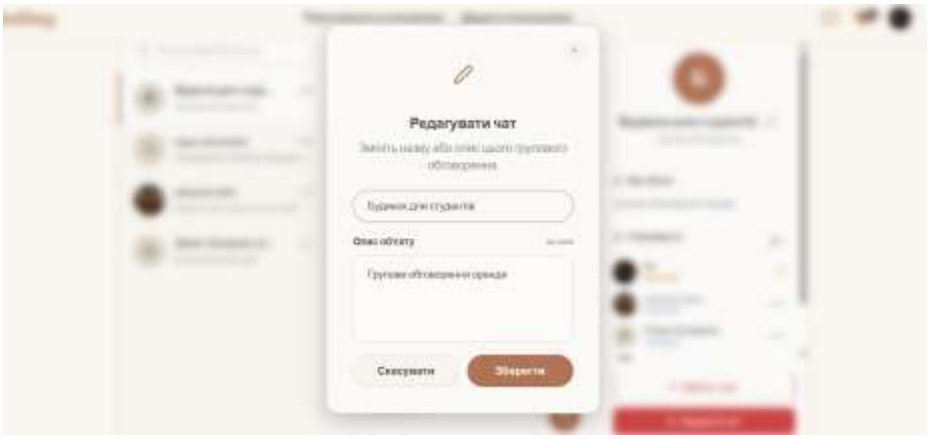


Рис. 2.8 – 2.10. Модуль адміністрування учасників та налаштувань групового чату

2.6. Оптимізація UI/UX та реалізація компонентів базового функціоналу

В межах розробки інтерфейсу користувача основну увагу було приділено створенню інтуїтивно зрозумілого шляху користувача (User Journey) при роботі з каталогом нерухомості. Основний акцент зроблено на декомпозиції інтерфейсу та забезпеченні інтерактивності основних функціональних блоків (див. рис. 2.11).

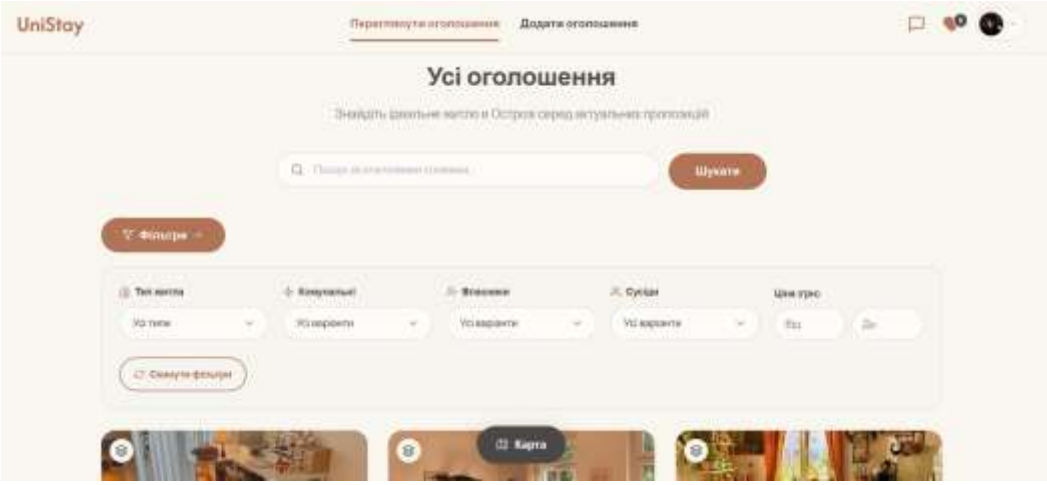
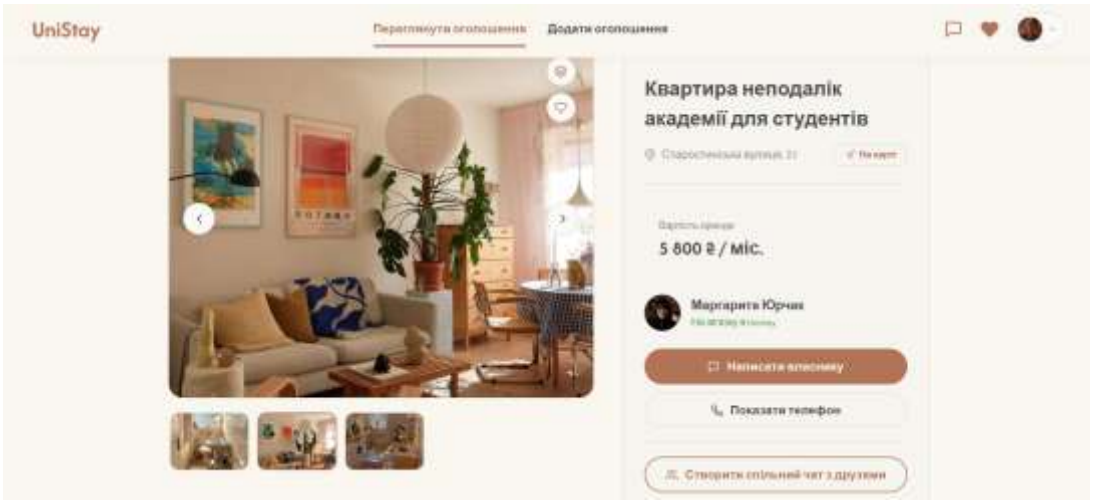
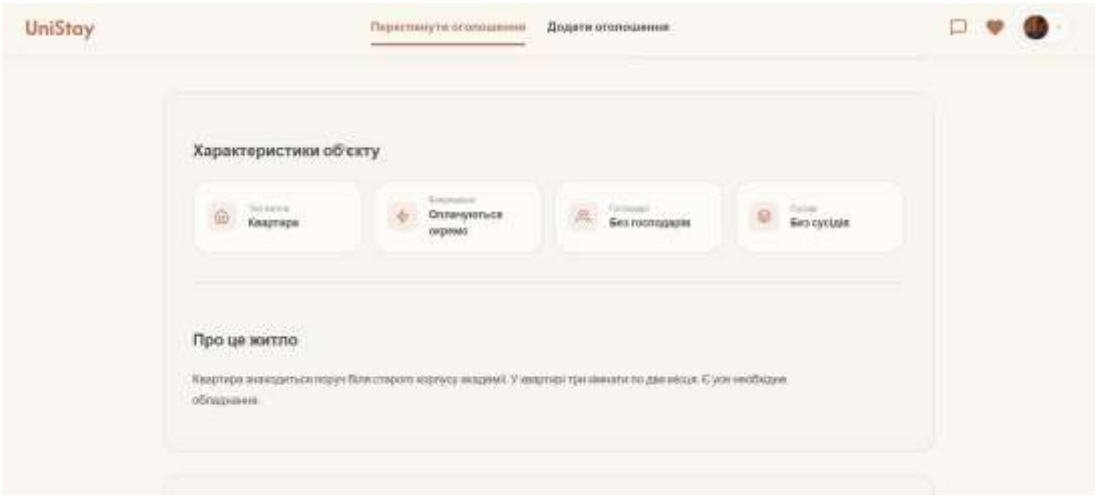


Рис. 2.11. Інтерфейс списку оголошень з реалізованою системою фільтрації

2.6.1. Архітектура сторінки перегляду оголошення (ListingDetailsPage)

Сторінка детальної інформації реалізована як композитний вузол, що складається з набору незалежних компонентів. Це дозволяє оптимізувати рендеринг та полегшує підтримку коду (див. рис. 2.12 – 2.15).





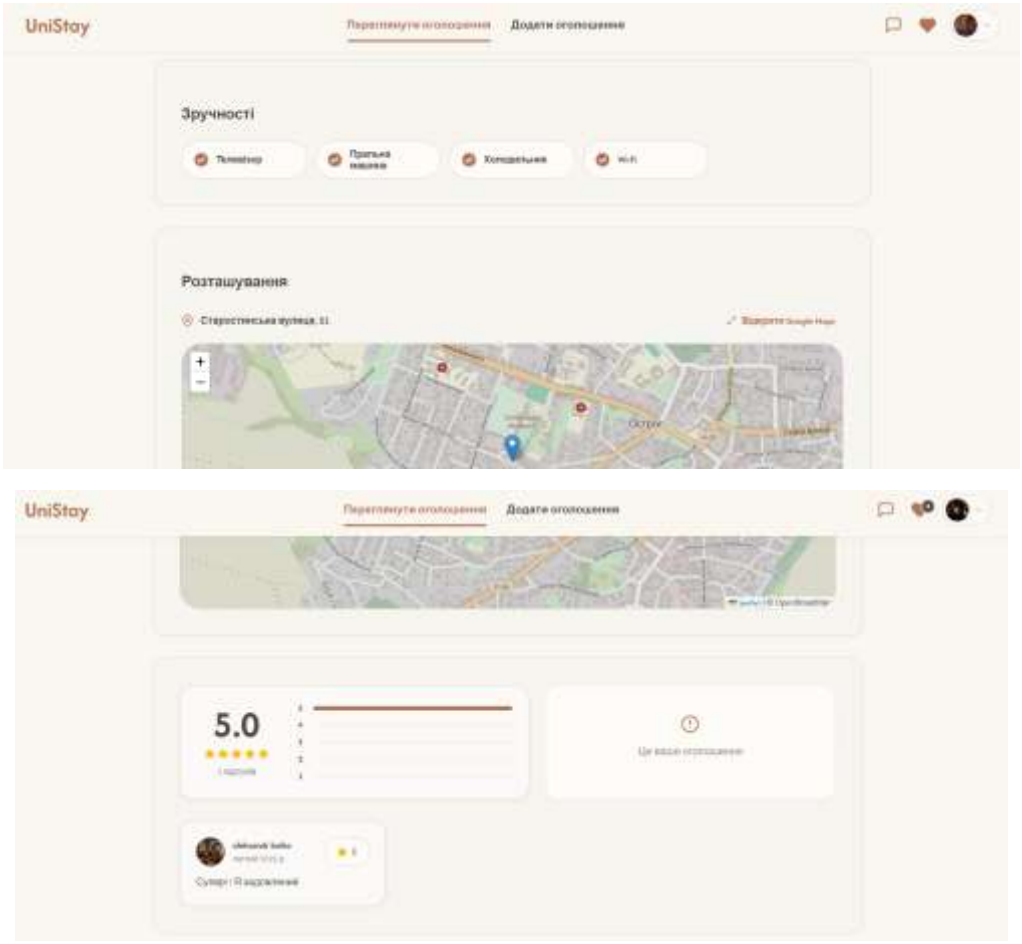
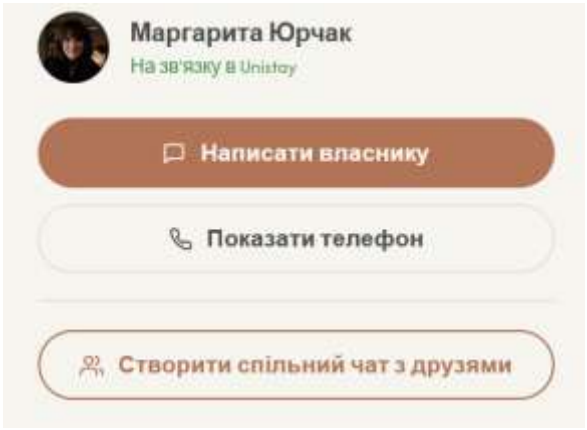


Рис. 2.12 – 2.15. Структура сторінки деталей оголошення

- ListingGallery: компонент реалізує інтерактивну галерею з можливістю перемикання зображень та візуальною індикацією статусу «Обране». Оптимізація UX досягається шляхом використання миттєвого фідбеку при взаємодії з кнопками «Favorite» та «Compare».
- ListingFeatures та AmenityDisplay: дані компоненти відповідають за структурування технічних параметрів (тип житла, умови проживання, сусіди) та зручностей. Використання сітки (Grid) та іконографіки дозволяє користувачу швидко проводити візуальний аудит об'єкта.

2.6.2. Комунікаційні інтерфейси та логіка взаємодії

Ключовим елементом UX-стратегії є забезпечення швидкого зв'язку між студентом та орендодавцем. Для цього було розроблено компонент ListingMainInfo, який інтегрує в собі декілька сценаріїв взаємодії (див. рис. 2.16-2.17).



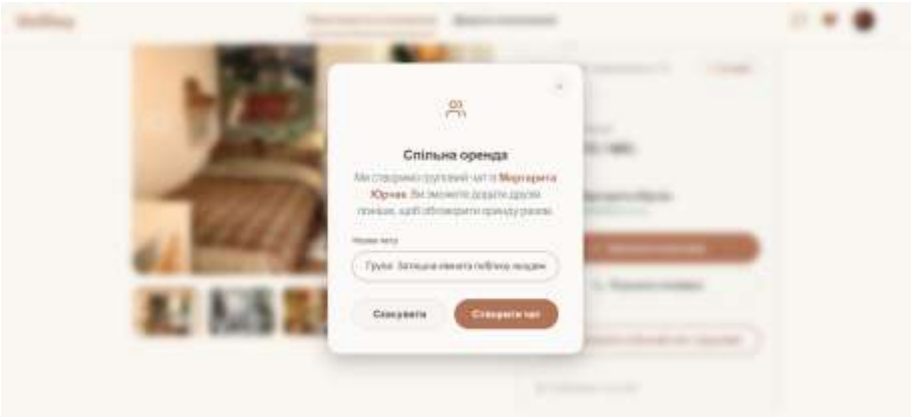


Рис. 2.16 – 2.17. Інтерфейсні елементи ініціації комунікації з орендодавцем

В межах цього компонента реалізовано наступні оптимізації:

- Контекстна перевірка: система аналізує приналежність оголошення поточному користувачу, автоматично блокуючи можливість створення діалогу із самим собою.
- Групова взаємодія: реалізовано унікальний для ринку нерухомості функціонал створення групових чатів безпосередньо зі сторінки об'єкта. Це дозволяє групі студентів спільно обговорювати умови оренди з власником, що значно скорочує цикл прийняття рішення.

2.6.3. Керування контентом орендодавця

Для користувачів, які публікують оголошення, розроблено інтерфейс MyListingsPage, що забезпечує повний цикл керування об'єктами (CRUD) (див. рис. 2.18).

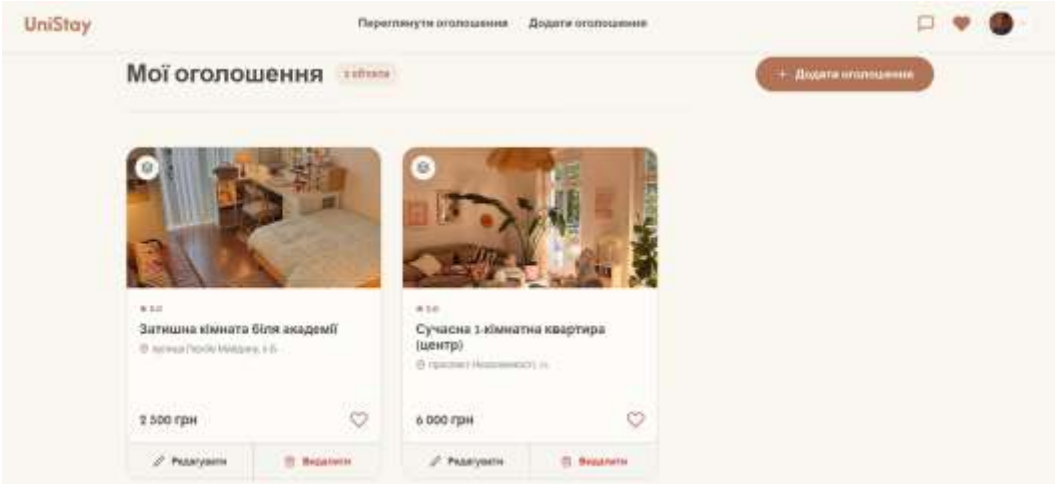


Рис. 2.18. Панель керування власними оголошеннями користувача

Важливим аспектом безпеки UX є використання модальних вікон підтвердження для деструктивних дій (див. рис. 2.19).

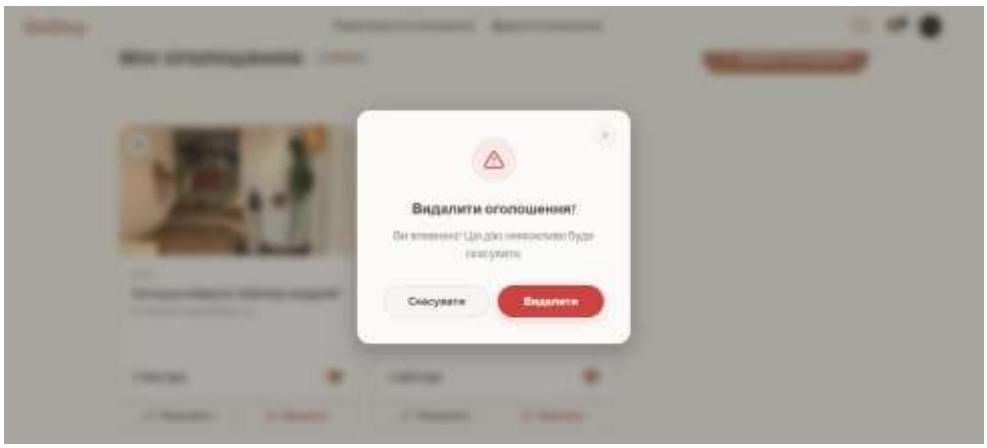


Рис. 2.9. Реалізація механізму запобігання випадковому видаленню даних

При реалізації сторінки редагування (EditListingPage) було використано підхід «контрольованих форм», де стан інтерфейсу синхронізується з моделлю даних об'єкта, що забезпечує цілісність інформації перед її відправкою на сервер.

2.6.4. Система соціального доказу та зворотного зв'язку

Важливою частиною UX є компонент ReviewSection, який забезпечує прозорість взаємодії між користувачами (див. рис. 2.20 – 2.21). Реалізація включає декілька рівнів обробки даних:

1. Математична обробка (ReviewStats): за допомогою хука useMemo реалізовано динамічний розрахунок середнього рейтингу та відсоткового розподілу оцінок (1-5 зірок), що дозволяє візуалізувати репутацію об'єкта.

2. Інтерактивна форма (ReviewForm): реалізовано складний інтерфейс оцінювання з підтримкою ефектів наведення (hover) на зірки, вибором текстових міток (наприклад, "Чудово") та лічильником символів для коментаря.
3. Рольова модель доступу: логіка компонента передбачає три стани:
   - Гість: отримує повідомлення про необхідність авторизації.
   - Власник: отримує повідомлення, що не може оцінювати власне оголошення.
   - Авторизований орендар: отримує доступ до форми публікації.
4. Компонент ReviewCard: забезпечує персоналізацію відгуків. У разі відсутності завантаженого аватара користувача, алгоритм автоматично генерує літерний ініціал на основі імені (метод getInitial), що підтримує естетичну цілісність інтерфейсу.

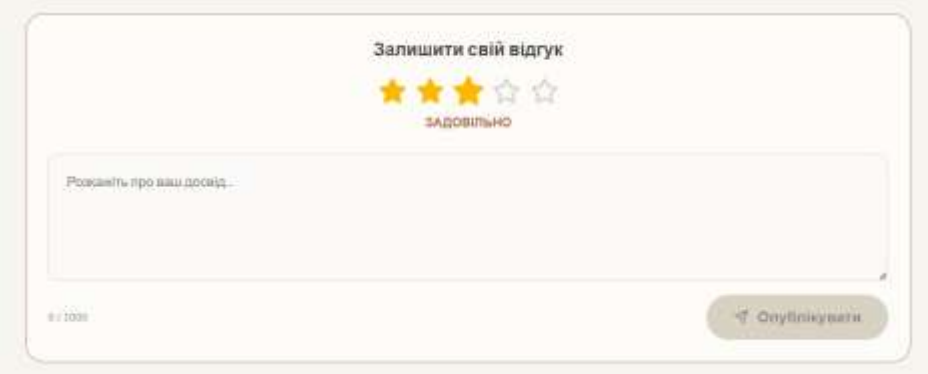


Рис. 2.20 – 2.21. Реалізація системи відгуків

2.7. Розробка допоміжних інтерфейсів та обробка виняткових станів системи

Для забезпечення безперервності та цілісності користувацького досвіду (UX) було розроблено низку допоміжних інтерфейсів. Основна мета цих компонентів — мінімізувати когнітивне навантаження на користувача у моменти очікування або виникнення помилок.

2.7.1. Персоналізація профілю користувача

Компонент ProfilePage реалізує функціонал керування особистими даними. Технічна особливість сторінки полягає у використанні jwt-decode для первинної ідентифікації користувача на основі токена, після чого дані синхронізуються з API (див. рис. 2.22).

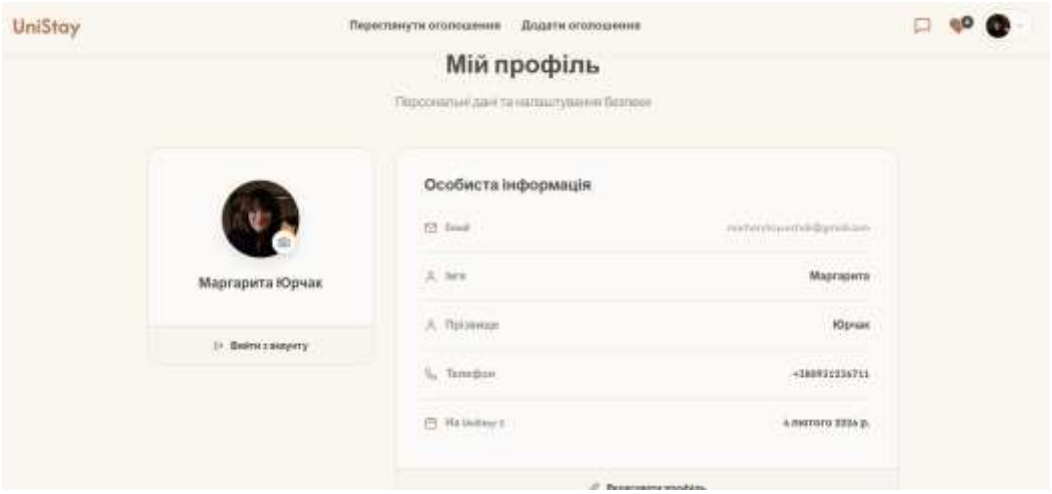


Рис. 2.22. Інтерфейс персонального профілю користувача

У профілі впроваджено:

- Інтерактивне оновлення фото: реалізовано через пряме завантаження у хмарне сховище з миттєвим відображенням стану завантаження (isUploading).
- Режим редагування «Inline»: перемикання між режимом перегляду та формою введення дозволяє зберігати чистоту інтерфейсу.

2.7.2. Обробка виняткових станів (404 та Loading)

Для обробки ситуацій, коли сторінка не знайдена або дані завантажуються, розроблено кастомні компоненти NotFoundPage та LoadingPage. Замість стандартних системних повідомлень використано стилізовані ілюстрації будинків, створені засобами CSS.

- NotFoundPage: містить чіткий заклик до дії (Call to Action) — кнопку повернення на головну, що запобігає тупиковим ситуаціям у навігації (див. рис. 2.23).

Рис. 2.23. Стилізований інтерфейс сторінки 404

- LoadingPage: використовує анімації @keyframes (bounce, progressAnim) та імітацію вмикання світла у вікні («lightSwitch»), що створює ефект «живого» застосунку та зменшує суб'єктивне сприйняття часу очікування (див. рис. 2.24).

Рис. 2.24. Стилізований інтерфейс стану завантаження

2.7.3. UX-дизайн порожніх станів (Empty States)

Порожні стани розглядаються не як помилка, а як можливість для онбордингу. Розроблено універсальний компонент .emptyState, який адаптується під різні контексти:

1. Обрані оголошення: якщо список фаворитів порожній, система пояснює призначення розділу та пропонує перейти до каталогу (див. рис. 2.25).

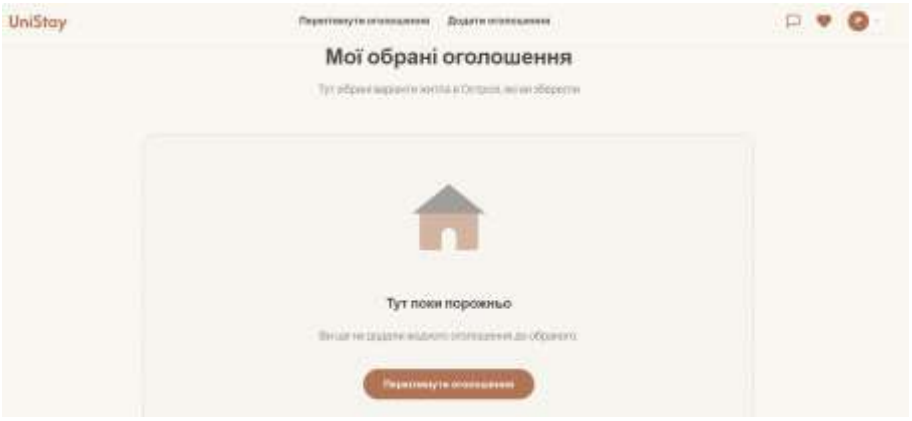


Рис. 2.25. Сторінка обраних оголошень при порожньому списку

2. Власні оголошення: для нових орендодавців передбачено акцентну кнопку «Додати перше оголошення», що стимулює цільову дію (див. рис. 2.16).

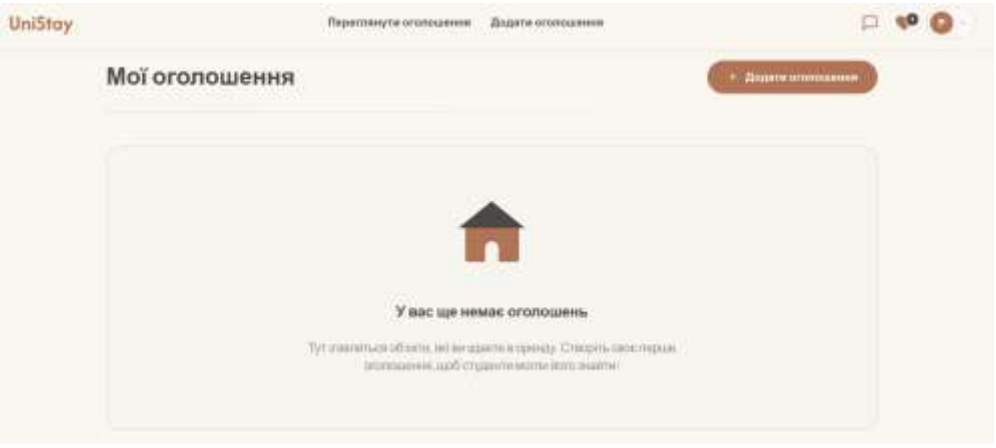


Рис. 2.26. Сторінка власних оголошень при порожньому списку

Використання пунктирних рамок (dashed border) та приглушеної прозорості ілюстрацій візуально підкреслює відсутність контенту, зберігаючи при цьому структуру сторінки.

2.8. Клієнтська безпека, обробка JWT-токенів та взаємодія з API

Для забезпечення надійної взаємодії клієнтської частини застосунку з серверним API було розроблено архітектурний шар сервісів та механізм автоматизованого керування станом автентифікації.

2.8.1. Автоматизація HTTP-запитів та Interceptors

В основі мережевої взаємодії лежить кастомний клас HttpClient (додаток Г), побудований на базі бібліотеки Axios. Основною перевагою такої реалізації є використання перехоплювачів (Interceptors), які виконують наступні завдання:

- Request Interceptor: автоматично витягує JWT-токен із локального сховища браузера (localStorage) та додає його до заголовка Authorization: Bearer кожного запиту. Це позбавляє необхідності вручну прописувати заголовки в кожному окремому сервісі.
- Response Interceptor: реалізує централізовану обробку помилок. У разі отримання коду 401 (Unauthorized), що свідчить про завершення терміну дії сесії, система автоматично очищує дані користувача та перенаправляє його на сторінку входу.

2.8.2. Обробка JWT-токенів та керування станом (AuthContext)

Для збереження даних користувача в межах усієї сесії розроблено глобальний контекст AuthContext. Ключовою технологією тут є використання бібліотеки jwt-decode. Процес автентифікації на боці клієнта працює за наступним алгоритмом:

1. Після успішного входу сервер повертає зашифрований JWT-токен.
2. Метод extractDataFromToken проводить декомпозицію токена для витягування ідентифікатора (nameid), ролі користувача (role) та імені користувача.
3. Отримані дані записуються в стан React (State), що дозволяє миттєво адаптувати інтерфейс (наприклад, приховувати кнопки редагування для осіб, які не є власниками оголошення).

2.8.3. Рівень сервісів (Service Layer)

Для відокремлення бізнес-логіки від компонентів відображення реалізовано шар сервісів (наприклад, ListingService). Кожен сервіс є набором асинхронних методів, які взаємодіють з екземпляром HttpClient.

Така структура забезпечує високу типізацію даних (за допомогою TypeScript DTO) та дозволяє легко змінювати кінцеві точки API в одному місці, не змінюючи код самих сторінок.

2.9. Інтеграція з хмарним сховищем Cloudinary для оптимізації медіаконтенту

Для зберігання та обслуговування графічного контенту (фотографій об'єктів нерухомості та аватарів користувачів) замість зберігання файлів на власному сервері було обрано платформу Cloudinary. Це рішення дозволило зняти навантаження з бекенду та забезпечити високу швидкість завантаження зображень через мережу доставки контенту (CDN).

2.9.1. Технічна реалізація завантаження зображень

У системі впроваджено механізм Unsigned Uploads, що дозволяє клієнтській частині застосунку безпечно завантажувати файли безпосередньо в хмару без передачі секретних ключів через фронтенд.

Процес завантаження складається з таких етапів:

1. Підготовка даних: формується об'єкт FormData, до якого додається файл та upload_preset (у моєму випадку — listing_images).
2. Прямий запит до API: за допомогою методу fetch дані відправляються на кінцеву точку Cloudinary: https://api.cloudinary.com/v1_1/${CLOUD_NAME}/image/upload.
3. Отримання результату: після успішного завантаження хмарний сервіс повертає об'єкт з параметром secure_url.
4. Синхронізація з базою даних: отримане посилання передається на власний бекенд для прив'язки до профілю користувача або оголошення.

2.9.2. Компонент ImageUploader та мультикадрове завантаження

Для модуля створення оголошень було розроблено спеціалізований компонент ImageUploader, який підтримує вибір декількох файлів одночасно.

- Асинхронна обробка: оскільки завантаження декількох зображень високої якості може тривати певний час, реалізовано стан завантаження (isUploading), який візуально блокує інтерфейс та відображає індикатор (FiLoader), запобігаючи помилкам користувача.
- Управління чергою: після завантаження кожного окремого файлу його URL додається до масиву imageUrls, а користувач має можливість

миттєво видалити некоректне фото ще до фінального збереження оголошення (див. рис. 2.27).

Рис. 2.27. Інтерфейс завантаження медіаконтенту при створенні оголошення.

2.9.3. Оптимізація та безпека медіаданих

Використання протоколу HTTPS (secure_url) гарантує безпечну передачу зображень. Крім того, Cloudinary автоматично оптимізує розмір файлів під формат пристрою користувача, що суттєво економить трафік. У профілі користувача (ProfilePage) цей механізм дозволяє миттєво оновити аватар без необхідності повного перезавантаження сторінки, що покращує показник чуйності інтерфейсу.

Висновки до розділу 2

У другому розділі було проведено повний цикл програмно-технічної реалізації клієнтської частини платформи «UniStay», результати якого дозволяють зробити наступні висновки:

1. Обґрунтовано та впроваджено сучасний технологічний стек, що базується на поєднанні бібліотеки React 19, мови TypeScript та інструменту збірки Vite. Це забезпечило високу швидкість розробки (через механізм HMR) та мінімізацію помилок на етапі виконання завдяки строгій статичній типізації даних.
2. Розроблено модульну архітектуру проєкту, яка базується на чіткому розподілі відповідальності між API-сервісами, глобальними контекстами та UI-компонентами. Використання каскаду провайдерів (Auth, Chat,

Listing) дозволило створити єдине джерело істини для даних, що забезпечило синхронну роботу всіх модулів системи.

3. Реалізовано унікальний картографічний модуль на базі бібліотеки Leaflet, який став ключовим інструментом візуального аналізу нерухомості в м. Острог. Впровадження алгоритмів розрахунку відстаней у реальному часі та динамічної побудови ліній зв’язку (Polyline) дозволило користувачам наочно оцінювати інфраструктурну перевагу об'єктів відносно навчальних корпусів.
4. Спроектовано та впроваджено систему інтерактивного порівняння, яка автоматизує процес вибору житла за допомогою колірного акцентування переваг (схема winnerFill) та візуальних індикаторів. Це дозволило знизити когнітивне навантаження на користувача та пришвидшити прийняття рішення про оренду.
5. Створено комунікаційне середовище в реальному часі з використанням технології SignalR. Розроблений модуль чатів підтримує не лише приватне листування, а й групову взаємодію з розширеною рольовою моделлю управління, що є критично важливим для студентської аудиторії при колективному пошуку житла.
6. Вирішено завдання оптимізації медіаконтенту та безпеки, шляхом інтеграції з хмарним сервісом Cloudinary та реалізації захищених маршрутів на основі JWT-авторизації. Використання Axios Interceptors дозволило автоматизувати обробку токенів та централізовано керувати сесіями користувачів.
7. Приділено значну увагу UI/UX дизайну виняткових станів (сторінки 404, Loading, Empty States). Застосування анімованих CSS-компонентів та контекстних підказок дозволило створити «живий» інтерфейс, який зберігає залученість користувача навіть за відсутності контенту або під час очікування відповіді сервера.

Таким чином, розроблена клієнтська частина платформи є повноцінним високонавантаженим SPA-застосунком, який відповідає сучасним стандартам продуктивності, безпеки та зручності використання.

РОЗДІЛ 3

ВЕРИФІКАЦІЯ РОБОТИ ТА ПРАКТИЧНЕ ЗАСТОСУВАННЯ РОЗРОБЛЕНОГО МОДУЛЯ КАРТИ

3.1. Функціональне тестування ключових сценаріїв взаємодії

Метою тестування є перевірка відповідності розробленого функціоналу вимогам технічного завдання та підтвердження коректності роботи алгоритмів у критичних сценаріях використання. Оскільки платформа «UniStay» орієнтована на інтерактивність, основну увагу було приділено перевірці модуля карти, системи порівняння та механізмів автентифікації.

3.1.1. Тестування модуля інтерактивної карти та геоаналітики

Для перевірки картографічного модуля було проведено серію тестів на основі реальних координат об'єктів нерухомості в м. Острог. Результати тестування наведено у таблиці 3.1.

Таблиця 3.1

Результати тестування картографічного модуля

Сценарій тестування	**Очікуваний результат**	**Фактичний результат**	**Статус**
Масштабування карти (Zoom out)	Маркери оголошень мають групуватися у кластери з відображенням кількості об'єктів.	При зменшенні масштабу спрацьовує MarkerClusterGroup, маркери групуються коректно.	**Пройдено**
Вибір інфраструктурного об'єкта (корпус академії)	Поява анімованої лінії до обраних квартир та відображення відстані.	Система динамічно будує Polyline та виводить відстань у кілометрах з точністю до десятих.	**Пройдено**
Клік на неактивний маркер ціни	Зміна стану маркера на активний та фокусування карти на об'єкті.	Маркер змінює колір, спрацьовує ефект приглушення для інших об'єктів.	**Пройдено**

3.1.2. Верифікація алгоритму порівняння об'єктів

Критичним аспектом тестування логіки порівняння була перевірка обмежень, закладених у ListingContext, та коректність обчислювальних властивостей стану.

1. Тест ліміту вибірки: при спробі додати третій об'єкт до списку порівняння, система ігнорує дію або виводить попередження (Toast-повідомлення). Тестування підтвердило, що стан compareIds не розширюється більше ніж на два елементи, що запобігає перевантаженню інтерфейсу сторінки порівняння.
2. Тест динамічного оновлення цін: при зміні ціни власником у базі даних, завдяки реактивності ListingProvider, інформація в модулі порівняння оновлюється автоматично без необхідності повторного додавання об'єкта до списку.
3. Перевірка візуальних індикаторів: підтверджено, що логіка winnerFill коректно визначає об'єкт з меншою ціною та меншою відстанню, зафарбовуючи відповідні блоки акцентним кольором.

3.1.3. Тестування комунікаційних інтерфейсів та безпеки

Перевірка модуля повідомлень (SignalR) та системи доступу проводилася шляхом симуляції дій двох різних користувачів у різних сесіях браузера:

- Синхронізація чату: повідомлення, надіслане орендарем, миттєво з'являється у вікні чату орендодавця без перезавантаження сторінки.
- Захист маршрутів: спроба отримати доступ до сторінки /messages або /profile неавторизованим користувачем (шляхом прямого введення URL) призводить до спрацювання ProtectedRoute та перенаправлення на сторінку входу.

Таким чином, результати функціонального тестування підтвердили стабільність роботи клієнтської частини платформи та повну працездатність інтерактивних елементів карти й модулів аналітичного порівняння.

3.2. Опис інтерфейсу користувача та сценарії експлуатації системи

Для забезпечення ефективної взаємодії користувача з платформою «UniStay» було розроблено інтуїтивно зрозумілий інтерфейс, що мінімізує час на вивчення функціоналу. Нижче наведено опис основних сценаріїв експлуатації системи, які розкривають практичну цінність розробленого модуля карти та порівняння.

3.2.1. Сценарій «Пошук на карті та аналіз локації»

Цей сценарій є базовим для цільової аудиторії (студентів), яким важливо знайти житло на певній відстані від навчальних корпусів.

1. Навігація до карти: користувач переходить у розділ «Карта», натискаючи відповідну кнопку при перегляді списку усіх оголошень.
2. Взаємодія з маркерами: на інтерактивній карті відображаються актуальні пропозиції. Користувач може використовувати зум для детального огляду районів міста (спрацьовує механізм кластеризації).
3. Вибір орієнтира: у бічній панелі або безпосередньо на карті користувач обирає ключовий об'єкт (наприклад, «Новий корпус НаУОА»).
4. Аналіз відстані: при кліку на маркер житла система автоматично будує анімовану лінію та розраховує дистанцію. Це дозволяє миттєво оцінити час на дорогу до університету.

3.2.2. Сценарій «Комплексне порівняння обраних варіантів»

Коли користувач вагається між двома об'єктами, він використовує аналітичний модуль порівняння.

1. Формування списку: користувач натискає іконку порівняння на картках двох вподобаних оголошень. Нижній бар (Comparison Bar) відображає поточний стан вибірки.
2. Перехід до аналітики: натисканням кнопки «Порівняти» користувач переходить на спеціалізовану сторінку.
3. Візуальний аудит:
    - у секції «Ціна» система автоматично підсвічує дешевший варіант.
    - У секції «Інфраструктура» порівнюються відстані до стратегічних точок міста.
    - У блоці «Зручності» виділяються унікальні переваги (наприклад, наявність пральної машини в одному об'єкті та її відсутність в іншому).

4. Прийняття рішення: На основі підсвічених «переваг» (winnerFill) користувач робить раціональний вибір.

3.2.3. Сценарій «Ініціація комунікації та групове обговорення»

Після вибору об'єкта користувач переходить до стадії прямої взаємодії з орендодавцем.

1. Контакт: на сторінці деталей оголошення користувач має можливість відкрити приватний чат з власником.
2. Груповий чат: якщо студент планує орендувати житло разом з друзями, він ініціює створення групового чату. Це дозволяє додати товаришів до обговорення конкретного об'єкта, де всі учасники бачать історію повідомлень та технічні характеристики житла в контексті чату.
3. Уточнення деталей: за допомогою SignalR-з'єднання користувачі отримують миттєві відповіді, що критично в період активного сезону пошуку житла перед початком навчального року.

3.3. Аналіз продуктивності та оцінка зручності інтерфейсу

Завершальним етапом розробки стала оцінка якісних показників системи, а саме швидкодії клієнтської частини та функціональності інтерфейсу. Оскільки додаток насичений інтерактивними картами та динамічними даними, важливо було переконатися, що архітектурні рішення забезпечують комфортну роботу користувача.

3.3.1. Технічні показники продуктивності

Завдяки використанню середовища збірки Vite та мемоїзації компонентів через хуки useMemo та useCallback, було досягнуто високих показників продуктивності:

- Швидкість первинного завантаження: равдяки оптимізації Rollup та розділенню коду (code-splitting), об’єм початкового JavaScript-бандла мінімізовано, що дозволяє сторінці карти ставати інтерактивною менш ніж за 1.5 секунди при середній швидкості з'єднання.
- Ефективність рендерингу карти: дикористання MarkerClusterGroup дозволило стабілізувати частоту кадрів (FPS) на рівні 60 кадр/сек навіть

при відображенні понад 50 об’єктів на обмеженій ділянці карти. Це нівелює ефект «зависання» інтерфейсу при панорамуванні мапи.

- Оптимізація запитів: застосування шару сервісів (HttpClient) та кешування станів у ListingContext дозволило скоротити кількість повторних запитів до API при перемиканні між сторінкою каталогу та сторінкою порівняння.

3.3.2. Оцінка UI/UX та юзабіліті

Для оцінки зручності було проведено комплексний аналіз користувацького досвіду, який підтвердив наступні переваги:

1. Зниження когнітивного навантаження: впровадження логіки winnerFill (автоматичне підсвічування кращих характеристик) дозволяє користувачу приймати рішення на основі візуальних підказок, а не лише шляхом порівняння цифр. Це робить процес вибору житла інтуїтивно зрозумілим.
2. Візуальна ієрархія: використання стану priceTagUnfocused на мапі ефективно керує увагою користувача, виділяючи об’єкти, що порівнюються, та приглушуючи другорядний контент.
3. Запобігання помилкам: система модальних вікон підтвердження для видалення оголошень та обмеження в модулі порівняння (не більше 2-х об'єктів) створюють «безпечне» середовище, де користувач захищений від випадкових деструктивних дій або перевантаження екрана зайвою інформацією.
4. Зворотний зв’язок (Feedback): анімовані лінії на карті створюють відчуття «живої» системи, що миттєво реагує на дії користувача.

Підсумовуючи, обрані технологічні рішення дозволили створити продукт, який поєднує високу обчислювальну потужність із сучасними вимогами до зручності користувацьких інтерфейсів.

Висновки до розділу 3

У третьому розділі було проведено верифікацію та оцінку розробленої платформи «UniStay», що дозволило сформулювати такі висновки:

1. Проведено успішне функціональне тестування основних модулів системи. Перевірка картографічного сервісу, логіки порівняння та системи реального часу (SignalR) підтвердила повну відповідність фактичних результатів очікуваним сценаріям.
2. Розроблено сценарії експлуатації, які демонструють практичну цінність системи для кінцевого користувача (студента). Доведено, що поєднання геопросторового аналізу та аналітичної таблиці порівняння суттєво спрощує процес прийняття рішення про оренду житла.
3. Підтверджено високу продуктивність застосунку, досягнуту завдяки сучасним інструментам збірки та оптимізації рендерингу React-компонентів. Система демонструє стабільну роботу навіть при активній взаємодії з важкими графічними елементами карти.
4. Аналіз результатів впровадження UI/UX рішень засвідчив, що впроваджені інтерактивні елементи (анімовані лінії, колірне акцентування, контекстні підказки) створюють комфортне середовище для користувача та підвищують конкурентоспроможність платформи на ринку подібних сервісів.

ВИСНОВКИ

Результатом виконання даної кваліфікаційної роботи стала розробка та програмна реалізація клієнтської частини спеціалізованої вебплатформи «UniStay», що спрямована на оптимізацію процесу пошуку та аналізу орендованого житла в межах локального студентського містечка. У ході дослідження було встановлено, що сучасний ринок нерухомості перенасичений універсальними пошуковими сервісами, які через свою архітектурну надлишковість не здатні задовольнити специфічні потреби студентської аудиторії. Зокрема, існуючі сервіси ігнорують потребу у швидкій оцінці просторової віддаленості житла від навчальних локацій та не пропонують інструментів для колективного обговорення варіантів. Це зумовило необхідність створення принципово нового інтерфейсного рішення, де центральним елементом виступає інтерактивна аналітика.

Проєктування платформи базувалося на створенні максимально інтуїтивного користувацького шляху (User Journey), де кожен етап — від авторизації до укладання домовленості в чаті — супроводжується візуальними підказками та динамічним зворотним зв'язком. Використання компонентно-орієнтованого підходу бібліотеки React 19 у поєднанні зі строгою типізацією TypeScript дозволило побудувати масштабовану архітектуру, стійку до критичних помилок. Завдяки впровадженню інструменту збірки Vite було досягнуто високих показників швидкодії, що забезпечує миттєве відображення змін на сторінці без перезавантаження браузера, зберігаючи при цьому цілісність сесії користувача.

Особливу увагу в роботі приділено картографічному модулю, побудованому на основі бібліотеки Leaflet. Розроблений інструментарій дозволяє не лише бачити об'єкти на мапі, а й проводити детальну оцінку розташування через побудову наочних ліній зв’язку та автоматичний розрахунок відстаней. Для того, щоб карта залишалася читабельною навіть при великій концентрації оголошень в одному районі, було застосовано механізм групування маркерів. Це дозволило зберегти візуальний порядок та легкість навігації, допомагаючи студенту швидко орієнтуватися в районах міста незалежно від кількості доступних пропозицій.

Аналітичний блок платформи був розширений за рахунок модуля синхронного порівняння характеристик об’єктів. Впроваджена логіка автоматизованого підсвічування переваг за фінансовими та інфраструктурними критеріями дозволяє користувачу приймати раціональні рішення, спираючись на автоматично згенеровані підказки системи. Такий підхід суттєво знижує навантаження на пам'ять користувача, оскільки вся необхідна інформація для зіставлення виводиться на єдиний екран у структурованому вигляді.

Комунікаційна складова проєкту реалізована через інтеграцію технології SignalR, що забезпечило миттєвий обмін повідомленнями та підтримку групових чатів. Це рішення є унікальним для сервісів нерухомості, оскільки воно дозволяє групі орендарів спільно керувати процесом вибору, призначати адміністраторів діалогів та вести перемовини з орендодавцем у захищеному середовищі. Безпека взаємодії підкріплена використанням JWT-токенів та централізованою обробкою HTTP-запитів через систему перехоплювачів Axios, що гарантує конфіденційність приватної переписки та даних профілю.

Додатково було вирішено питання оптимізації важкого медіаконтенту шляхом інтеграції з хмарним сервісом Cloudinary, що дозволило значно пришвидшити завантаження фотографій об'єктів та забезпечити їх коректне відображення на різних типах пристроїв. Проведене тестування та аналіз виняткових станів інтерфейсу (таких як сторінки завантаження та помилок) підтвердили високу зручність системи. Таким чином, розроблена платформа «UniStay» є комплексним програмним продуктом, який демонструє ефективність поєднання сучасних вебтехнологій з індивідуальним підходом до потреб конкретної соціальної групи, пропонуючи новий рівень зручності у сфері цифровізації послуг оренди житла.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. OLX.ua — сервіс оголошень України : офіційний сайт. URL: https://www.olx.ua (дата звернення: 12.04.2026).
2. Flatfy від ЛУН : офіційний сайт. URL: https://flatfy.ua (дата звернення: 12.04.2026).
3. DIM.RIA — перевірена нерухомість України : офіційний сайт. URL: https://dom.ria.com (дата звернення: 13.04.2026).
4. Нілсен Я. 10 евристик юзабіліті для дизайну інтерфейсу користувача. *NN/g Nielsen Norman Group*. URL: https://www.nngroup.com/articles/ten-usability-heuristics/ (дата звернення: 15.04.2026).
5. Офіційна документація React : бібліотека JavaScript для створення інтерфейсів. URL: https://react.dev (дата звернення: 18.03.2026).
6. TypeScript: JavaScript із синтаксисом для типів : офіційна документація. URL: https://www.typescriptlang.org/docs/ (дата звернення: 18.03.2026).
7. Vite: наступне покоління інструментів для фронтенд-розробки. URL: https://vitejs.dev (дата звернення: 20.03.2026).
8. Leaflet: відкрита JavaScript-бібліотека для інтерактивних карт. URL: https://leafletjs.com (дата звернення: 22.03.2026).
9. Документація React Leaflet: компоненти React для карт Leaflet. URL: https://react-leaflet.js.org (дата звернення: 02.04.2026).
10. Вступ до SignalR : офіційна документація Microsoft. URL: https://learn.microsoft.com/en-us/aspnet/core/signalr/introduction (дата звернення: 06.04.2026).
11. Axios: HTTP-клієнт на основі Promise для браузера та node.js. URL: https://axios-http.com/docs/intro (дата звернення: 05.04.2026).
12. Документація Cloudinary: рішення для управління зображеннями та відео. URL: https://cloudinary.com/documentation (дата звернення: 07.04.2026).
13. Специфікація JSON Web Token (JWT). URL: https://jwt.io/introduction/ (дата звернення: 07.04.2026).

14. React Router: декларативна маршрутизація для React. URL: https://reactrouter.com (дата звернення: 08.04.2026).
15. OpenStreetMap: відкриті картографічні дані : офіційний сайт. URL: https://www.openstreetmap.org (дата звернення: 12.04.2026).
16. **MDN Web Docs** : документація з технологій веб-розробки (HTML, CSS, JS). URL: https://developer.mozilla.org (дата звернення: 14.04.2026).
17. **Lucide React** : бібліотека відкритих іконок для React-застосунків. URL: https://lucide.dev/guide/packages/lucide-react (дата звернення: 14.04.2026).
18. **ESLint** : інструмент для статичного аналізу коду JavaScript та TypeScript. URL: https://eslint.org (дата звернення: 15.04.2026).
19. **Rollup.js** : збирач модулів для JavaScript, що використовується у Vite. URL: https://rollupjs.org (дата звернення: 15.04.2026).
20. **PostCSS** : інструмент для трансформації стилів за допомогою JS-плагінів. URL: https://postcss.org (дата звернення: 16.04.2026).
21. **CSS Modules** : специфікація для створення модульного та безпечного CSS. URL: https://github.com/css-modules/css-modules (дата звернення: 16.04.2026).
22. **Web Content Accessibility Guidelines (WCAG) 2.1** : настанови з доступності веб-вмісту. URL: https://www.w3.org/TR/WCAG21/ (дата звернення: 17.04.2026).
23. **Leaflet.markercluster** : плагін для кластеризації маркерів у Leaflet. URL: https://github.com/Leaflet/Leaflet.markercluster (дата звернення: 17.04.2026).
24. **React Hook Form** : бібліотека для керування станом форм у React. URL: https://react-hook-form.com (дата звернення: 18.04.2026).
25. **Cloudinary Upload Presets** : документація з налаштування завантаження медіафайлів. URL: https://cloudinary.com/documentation/upload_presets (дата звернення: 18.04.2026).
26. **SignalR Hubs** : посібник зі створення хабів для передачі даних у реальному часі. URL: https://learn.microsoft.com/en-us/aspnet/core/signalr/hubs (дата звернення: 19.04.2026).

27. **Google Fonts** : бібліотека типографіки для веб-інтерфейсів. URL: https://fonts.google.com (дата звернення: 20.04.2026).
28. **React Context API** : офіційне керівництво з управління станом додатка. URL: https://react.dev/learn/passing-data-deeply-with-context (дата звернення: 20.04.2026).
29. **HTTP-заголовки авторизації (Bearer Token)** : документація MDN. URL: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Authorization (дата звернення: 21.04.2026).
30. **Can I Use** : сервіс перевірки підтримки веб-технологій браузерами. URL: https://caniuse.com (дата звернення: 21.04.2026).
31. **Prettier** : інструмент для автоматичного форматування коду. URL: https://prettier.io (дата звернення: 22.04.2026).
32. **jwt-decode** : бібліотека для розкодування JWT на стороні клієнта. URL: https://github.com/auth0/jwt-decode (дата звернення: 23.04.2026).
33. **React Virtual DOM** : опис алгоритму порівняння та оновлення інтерфейсу. URL: https://legacy.reactjs.org/docs/faq-internals.html (дата звернення: 23.04.2026).
34. **WebSockets API** : стандарт протоколу для повнодуплексного зв’язку. URL: https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API (дата звернення: 24.04.2026).
35. **CSS Flexbox Guide** : повне керівництво з побудови гнучких макетів. URL: https://css-tricks.com/snippets/css/a-guide-to-flexbox/ (дата звернення: 24.04.2026).
36. **Framer Motion** : бібліотека для анімацій у React. URL: https://www.framer.com/motion/ (дата звернення: 25.04.2026).
37. **Browserslist** : конфігурація для підтримки цільових версій браузерів. URL: https://github.com/browserslist/browserslist (дата звернення: 25.04.2026).
38. **National University of Ostroh Academy** : офіційний вебсайт (як джерело геоданих корпусів). URL: https://www.oa.edu.ua (дата звернення: 26.04.2026)

Додаток А

Програмна реалізація конфігурації маршрутів та глобальної структури застосунку

```
import React from "react";
import { Routes, Route } from "react-router-dom";
import "./App.css";

// Layouts
import MainLayout from "./layouts/MainLayout/MainLayout";
import AuthLayout from "./layouts/AuthLayout/AuthLayout";
import HomeLayout from "./layouts/HomeLayout/HomeLayout";

// Pages
import HomePage from "./pages/HomePage/HomePage";
import ListingsPage from "./pages/ListingsPage/ListingsPage";
import ListingDetailPage from "./pages/ListingDetailPage/ListingDetailsPage";
import AddListingPage from "./pages/AddListingPage/AddListingPage";
import EditListingPage from "./pages/EditListingPage/EditListingPage";
import FavoritesPage from "./pages/FavoritesPage/FavoritesPage";
import MyListingsPage from "./pages/MyListingsPage/MyListingsPage";
import SignInPage from "./pages/SignInPage/SignInPage";
import SignUpPage from "./pages/SignUpPage/SignUpPage";
import NotFoundPage from "./pages/NotFoundPage/NotFoundPage";
import ExploreMapPage from "./pages/ExploreMapPage/ExploreMapPage";
import ComparePage from "./pages/ComparePage/ComparePage";
import ProfilePage from "./pages/ProfilePage/ProfilePage";
import MessagesPage from "./pages/MessagesPage/MessagesPage"; // ДОДАНО

// Context & Protection
import { AuthProvider } from "./context/AuthContext";
import { ListingProvider } from "./context/ListingContext";
import { ChatProvider } from "./context/ChatContext";
import ProtectedRoute from "./components/auth/ProtectedRoute";

function App() {
  return (
    <AuthProvider>
      <ChatProvider>
        <ListingProvider>
          <Routes>
            {/* 1. Головна сторінка */}
            <Route element={<HomeLayout />}>
              <Route path="/" element={<HomePage />} />
            </Route>

            {/* 2. Всі інші сторінки */}
            <Route element={<MainLayout />}>
              <Route path="listings" element={<ListingsPage />} />
              <Route
                path="listings/:listingId"
                element={<ListingDetailPage />}
              />
              <Route path="explore-map" element={<ExploreMapPage />} />
              <Route path="compare" element={<ComparePage />} />

              {/* Захищені маршрути */}
              <Route
                path="add-listing"
                element={
                  <ProtectedRoute>
                    <AddListingPage />
                  </ProtectedRoute>
                }
```

Продовження додатку А

```
  />
  <Route
    path="edit-listing/:listingId"
    element={
      <ProtectedRoute>
        <EditListingPage />
      </ProtectedRoute>
    }
  />
  <Route
    path="favorites"
    element={
      <ProtectedRoute>
        <FavoritesPage />
      </ProtectedRoute>
    }
  />
  <Route
    path="my-listings"
    element={
      <ProtectedRoute>
        <MyListingsPage />
      </ProtectedRoute>
    }
  />
  <Route
    path="profile"
    element={
      <ProtectedRoute>
        <ProfilePage />
      </ProtectedRoute>
    }
  />

  {/* Сторінка повідомлень */}
  <Route
    path="messages"
    element={
      <ProtectedRoute>
        <MessagesPage />
      </ProtectedRoute>
    }
  />
</Route>

{/* 3. Авторизація */}
<Route
  path="/signin"
  element={
    <AuthLayout>
      <SignInPage />
    </AuthLayout>
  }
/>
<Route
  path="/signup"
  element={
    <AuthLayout>
      <SignUpPage />
    </AuthLayout>
  }
/>
```

Продовження додатку А

```
            {/* 4. Сторінка 404 */}
            <Route path="*" element={<NotFoundPage />} />
          </Routes>
        </ListingProvider>
      </ChatProvider>
    </AuthProvider>
  );
}

export default App;
```

Додаток Б

Програмна реалізація компонента контролю доступу ProtectedRoute

```
import React from "react";
import { Navigate } from "react-router-dom";
import { useAuth } from "../../context/AuthContext";

const ProtectedRoute = ({ children }) => {
  const { isAuthenticated, loading } = useAuth();

  if (loading) {
    return <div>Завантаження авторизації...</div>;
  }

  if (!isAuthenticated) {
    return <Navigate to="/signin" replace />;
  }

  return children;
};

export default ProtectedRoute;
```

Додаток В

Програмна реалізація модуля управління станом та бізнес-логіки обробки оголошень

```
import React, { createContext, useContext, useState, useEffect, useCallback, ReactNode }
from 'react';
import { ListingService } from '../api/services/ListingService';
import { FavoriteService } from '../api/services/FavoriteService';
import { useAuth } from './AuthContext';
import { ListingDto, CreateListingDto } from '../api/dto/ListingDto';

interface ListingContextType {
    listings: ListingDto[];
    favoriteListings: ListingDto[];
    favoriteIds: string[];
    compareIds: string[];
    compareListings: ListingDto[];
    loading: boolean;
    error: string | null;
    fetchListings: () => Promise<void>;
    fetchFavoriteIds: () => Promise<void>;
    toggleFavorite: (listingId: string) => Promise<void>;
    toggleCompare: (listingId: string) => void;
    clearCompare: () => void;
    clearFavorites: () => Promise<void>;
    addListing: (dto: CreateListingDto) => Promise<ListingDto>;
    deleteListing: (id: string) => Promise<void>;
}

const ListingContext = createContext<ListingContextType | undefined>(undefined);

export const ListingProvider: React.FC<{ children: ReactNode }> = ({ children }) => {
    const { isAuthenticated } = useAuth();
    const [listings, setListings] = useState<ListingDto[]>([]);
    const [favoriteIds, setFavoriteIds] = useState<string[]>([]);
    const [compareIds, setCompareIds] = useState<string[]>([]);
    const [loading, setLoading] = useState<boolean>(false);
    const [error, setError] = useState<string | null>(null);

    const fetchListings = useCallback(async () => {
        setLoading(true);
        setError(null);
        try {
            const data = await ListingService.getAll();
            setListings(data);
        } catch (err) {
            setError("Не вдалося завантажити оголошення");
        } finally {
            setLoading(false);
        }
    }, []);

    const fetchFavoriteIds = useCallback(async () => {
        if (!isAuthenticated) {
            setFavoriteIds([]);
            return;
        }
        try {
            const data = await FavoriteService.getMyFavorites();
            setFavoriteIds(data.map((fav: any) => (fav.listingId || fav.id).toString()));
        } catch (err) {
            console.error("Помилка завантаження обраних:", err);
```

Продовження додатку В

```
}
    }, [isAuthenticated]);

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

    useEffect(() => {
        fetchFavoriteIds();
    }, [fetchFavoriteIds, isAuthenticated]);

    const toggleFavorite = async (listingId: string) => {
        if (!isAuthenticated) {
            alert("Будь ласка, увійдіть в акаунт.");
            return;
        }

        const idStr = listingId.toString();
        const isFavorited = favoriteIds.includes(idStr);

        try {
            if (isFavorited) {
                try {
                    await FavoriteService.removeFromFavorites(idStr);
                } catch (err: any) {
                    if (err.response?.status !== 404) throw err;
                }
                setFavoriteIds(prev => prev.filter(id => id !== idStr));
            } else {
                await FavoriteService.addToFavorites(idStr);
                setFavoriteIds(prev => [...prev, idStr]);
            }
        } catch (err) {
            console.error("Помилка toggle favorite:", err);
        }
    };

    const toggleCompare = (listingId: string) => {
        const idStr = listingId.toString();
        setCompareIds(prev => {
            if (prev.includes(idStr)) {
                return prev.filter(id => id !== idStr);
            }
            if (prev.length >= 2) {
                alert("Ви можете порівняти лише 2 оголошення одночасно.");
                return prev;
            }
            return [...prev, idStr];
        });
    };

    const clearCompare = () => setCompareIds([]);

    const clearFavorites = async () => {
        try {
            await Promise.allSettled(
                favoriteIds.map(id => FavoriteService.removeFromFavorites(id))
            );
            setFavoriteIds([]);
        } catch (err) {
            console.error("Помилка при очищенні обраного:", err);
        }
    };
    const value = {
```

Продовження додатку В

```
        listings,
        favoriteListings: listings.filter(l => favoriteIds.includes(l.id.toString())),
        favoriteIds,
        compareIds,
        compareListings: listings.filter(l => compareIds.includes(l.id.toString())),
        loading,
        error,
        fetchListings,
        fetchFavoriteIds,
        toggleFavorite,
        toggleCompare,
        clearCompare,
        clearFavorites,
        addListing: async (dto: CreateListingDto) => {
            const created = await ListingService.create(dto);
            setListings(prev => [created, ...prev]);
            return created;
        },
        deleteListing: async (id: string) => {
            await ListingService.delete(id);
            setListings(prev => prev.filter(l => l.id.toString() !== id.toString()));
        }
    };

    return (
        <ListingContext.Provider value={value}>
            {children}
        </ListingContext.Provider>
    );
};

export const useListings = () => {
    const context = useContext(ListingContext);
    if (context === undefined) {
        throw new Error("useListings must be used within a ListingProvider");
    }
    return context;
};
```

Додаток Г

Програмна реалізація механізмів мережевої взаємодії та централізованої обробки HTTP-запитів

```
export const useListings = () => {
    const context = useContext(ListingContext);
    if (context === undefined) {
        throw new
```

```
import axios, { AxiosError, AxiosInstance, AxiosRequestConfig } from "axios";


class HttpClient {
  private axiosInstance: AxiosInstance;


  constructor(configs: AxiosRequestConfig) {
    this.axiosInstance = axios.create({
      baseURL: import.meta.env.VITE_API_URL || "https://localhost:7190/api",
      timeout: configs.timeout || 15000,
      headers: {
        "Content-Type": "application/json",
        Accept: "application/json",
        ...configs.headers,
      },
    });
    this.initInterceptors();
  }


  private initInterceptors() {
    this.axiosInstance.interceptors.request.use(
      (config) => {
        const token = localStorage.getItem("token");
        if (token) {
          config.headers["Authorization"] = `Bearer ${token}`;
        }
        return config;
      },
      (error) => Promise.reject(error),
    );
this.axiosInstance.interceptors.response.use(
```

Продовження додатку Г

```
    (response) => response,
    (error) => {
      if (error instanceof AxiosError) {
        if (error.response?.status === 401) {
          localStorage.removeItem("token");
          if (!window.location.pathname.includes("/signin")) {
            window.location.href = "/signin";
          }
        }
        const backendMessage =
          error.response?.data?.Message || error.response?.data?.message;

        if (backendMessage) {
          error.message = backendMessage;
        }
      }
      return Promise.reject(error);
    },
  );
}

public async get<T>(url: string, config?: AxiosRequestConfig): Promise<T> {
  const response = await this.axiosInstance.get<T>(url, config);
  return response.data;
}

public async post<T, D = unknown>(
  url: string,
  data: D,
  config?: AxiosRequestConfig,
): Promise<T> {
  const response = await this.axiosInstance.post<T, any, D>(
    url,
    data,
    config,
```

Продовження додатку Г

```
    );
    return response.data;
  }

  public async put<T, D = unknown>(
    url: string,
    data: D,
    config?: AxiosRequestConfig,
  ): Promise<T> {
    const response = await this.axiosInstance.put<T, any, D>(url, data, config);
    return response.data;
  }

  public async delete<T>(url: string, config?: AxiosRequestConfig): Promise<T> {
    const response = await this.axiosInstance.delete<T>(url, config);
    return response.data;
  }
}

const api = new HttpClient({});
export default api;
```