Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Розробка вебсайту для пошуку тим часового житла з використанням MySQL та PHP

Виконав: студент IV курсу, групи СН-41
спеціальності 122 Комп’ютерні науки
(шифр і назва спеціальності)

_____ Васильців А.З
(підпис) (прізвище та ініціали)

Керівник _____ Шимчук Г.В.
(підпис) (прізвище та ініціали)

Нормоконтроль _____ Шимчук Г.В.
(підпис) (прізвище та ініціали)

Завідувач кафедри _____ Боднарчук І.О.
(підпис) (прізвище та ініціали)

Рецензент _____ Тимощук Д.І.
(підпис) (прізвище та ініціали)

Тернопіль 2025

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
«____» _червня_ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Бакалавр
(назва освітнього ступеня)

за спеціальністю 122 Комп'ютерні науки
(шифр і назва спеціальності)

Студенту Васильціву Андрію Зеновійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка вебсайту для пошуку тимчасового житла з використанням MySQL та PHP.

Керівник роботи Шимчук Григорій Валерійович, старший викладач кафедри КН
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «*07*» *травня* 2025 року № *4/7-444*

2. Термін подання студентом завершеної роботи 23 червня 2025 р.

3. Вихідні дані до роботи Інтернет джерела про технології розробки вебсайту для пошуку Тимчасового жита з використанням MYSQL та PHP

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. РОЗДІЛ 1. Аналіз предметної області та постановка завдання на розробку вебсайту Для пошуку тимчасового житла з використанням mysql та php. 1.1 аналіз предметної області 1.2 Аналіз наявних рішень. 1.3 Формування вимог вебсайту для пошуку тимчасового житла з Використанням mysql та PHP1.4 Основні учасники та їх функції на вебсайті для пошуку тим-Часового житла. Розділ 2. Проєктування та розробка вебсайту для пошуку тимчасового житла з використанням mysql та php. 2.1 моделювання та опис рівнів архітектури вебсайту для Пошуку тимчасового житла . Розділ 3. Встановлення, налаштування, експлуатація і супровід вебсайту для пошуку тимчасовіого житла. 3.1 Вибір хостинг-провайдера, його налаштування та тестування для вебсайту для пошуку тимчасового житла. 3.2 Валідація та тестування вебсайту для пошуку тимчасового житла. 3.3 Експлуатація вебсайту для пошуту тимчасового житла. 3.4 Заходи забезпечення захисту вебсайту для пошуку тимчасового житла Розділ 4. Безпека життєдіяльності, основи охорони праці. Перелік джерел. Висновки.Додатки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1.Титульний слайд. 2.Актуальність роботи 3. Мета і завдання дослідження 4. Аналіз предметної області. 5. Технологічний стек 6. Архітектура вебсайту 7. База даних та її Структура. 8. Функціональність системи. 15. Безпека та захист. 16. Тестування та валідація 17. Висновки та результати

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи охорони праці	Сенчишин В.С., доцент кафедри МТ		

7. Дата видачі завдання 07 травня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	07.05.2025	
2.	Підбір та опрацювання літературних джерел для розробки вебсайту з пошуку тимчасового житла	08.05.2025-24.05.2025	
3.	Опрацювання джерел по темі кваліфікаційної роботи	24.05.2025-26.05.2025	
4.	Виконання дослідження щодо розробки вебсайту		
	для пошуку тимчасового житла використанням		
	MYSQL та PHP		
5.	Розроблення веб сайту для пошуку тимчасового житла	26.05.2025-06.06.2025	
6.	Оформлення розділу «Аналіз предметної області та.	07.06.2025-08.06.2025	
	постановка завдання на розробку вебсайту для пошуку тимчасового житла з використанням mysql та php»		
7.	Оформлення розділу «Проєктування та розробка веб-	08.06.2025-09.06.2025	
	сайту для пошуку тимчасового житла з використанням mysql та php»		
8.	Оформлення розділу «Встановлення, налаштування,	09.05.2025-10.05.2025	
	експлуатація і супровід вебсайту для пошуку тимчасовіого житла»		
9.	Виконання завдання до підрозділу «Безпека життєдіяльності»	10.06.2025-10.06.2025	
10.	Виконання завдання до підрозділу «Основи охорони	10.06.2025-10.06.2025	
	праці»		
11.	Оформлення кваліфікаційної роботи	11.06.2025-11.06.2025	
12.	Нормоконтроль	12.06.2025-12.06.2025	
13.	Перевірка на плагіат	13.06.2025	
14.	Попередній захист кваліфікаційної роботи	20.06.2025	
15.	Захист кваліфікаційної роботи	24.06.2025	

Студент _____ Васильців А.З
(підпис) (прізвище та ініціали)

Керівник роботи _____ Шимчук Г.В
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Розробка вебсайту для пошуку тимчасового житла з використанням MySQL та PHP // Кваліфікаційна робота освітнього рівня «Бакалавр» // Васильців Андрій Зеновійович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра комп'ютерних наук, група СН-41 // Тернопіль, 2025 // С. – 114 , рис. – 51, табл. – 11, кресл. – 0 , додат. – 3 , бібліогр. – 56.

Ключові слова: тимчасове житло, вебсайт, база даних, пошук житла, оголошення, бронювання

Кваліфікаційна робота присвячена дослідженню процесу створення вебсайту для пошуку тимчасового житла з використанням MySQL та PHP. В першому розділі кваліфікаційної роботи описано аналіз предметної області, визначено актуальність проблеми для переселенців, сформульовано вимоги до функціоналу та безпеки, розглянуто існуючі рішення та обґрунтовано вибір технологій. Висвітлено структуру користувацьких ролей, сценарії взаємодії та принципи побудови інтерфейсу. Проаналізовано життєвий цикл розробки вебсайту.

В другому розділі кваліфікаційної роботи досліджено архітектуру платформи ShelterUA, подано концептуальну модель бази даних, діаграми кооперації, класів і станів, а також структуру каталогів і файлів проєкту. Досліджено функціональні схеми для основних сторінок і контролерів, що забезпечують прозорість логіки роботи та масштабованість сервісу.

В третьому розділі кваліфікаційної роботи описано процес розгортання, налаштування, тестування та експлуатації вебсайту. Проаналізовано результати валідації коду, тестування кросбраузерності й адаптивності, а також заходи із забезпечення захисту даних користувачів. Проведено комплексне тестування функціоналу платформи.

У четвертому розділі кваліфікаційної роботи проаналізовано питання працездатності людини-оператора та соціального значення охорони праці у контексті розробки та експлуатації вебсайту для пошуку тимчасового житла. Розкрито вплив зовнішніх і внутрішніх чинників на ефективність та безпеку праці розробників, а також важливість організації раціонального режиму праці й відпочинку. Особливу увагу приділено створенню комфортних і безпечних умов праці, що сприяє збереженню здоров’я, підвищенню продуктивності та мотивації працівників. Підкреслено, що впровадження сучасних заходів з охорони праці є необхідною умовою для сталого розвитку підприємства, підвищення якості життя персоналу та забезпечення надійної роботи інформаційної платформи.

ANNOTATION

Development of a Temporary Housing Search Website Using MySQL and PHP // Qualification work of the educational level «Bachelor» // Vasyltsiv Andrii Zenoviiovych // Ternopil Ivan Puluj National Technical University, Faculty of Computer and Information Systems and Software Engineering, Department of Computer Sciences, group SN-41 // Ternopil, 2025 // P. – 114, fig. – 51, tabl. – 11, slides. – 0 , annexes. – 3, references – 56.

Keywords: temporary housing, website, database, housing search, listing, booking

This qualification work is devoted to the study of the process of developing a website for temporary housing search using MySQL and PHP. The first section of the work describes the analysis of the subject area, determines the relevance of the problem for internally displaced persons, formulates requirements for functionality and security, reviews existing solutions, and substantiates the choice of technologies. The structure of user roles, interaction scenarios, and interface design principles are highlighted. The life cycle of website development is analyzed.

The second section of the qualification work investigates the architecture of the ShelterUA platform, presents the conceptual model of the database, cooperation, class and state diagrams, as well as the structure of project directories and files. Functional schemes for the main pages and controllers are explored, ensuring transparency of logic and scalability of the service.

The third section of the qualification work describes the process of deployment, configuration, testing, and operation of the website. The results of code validation, cross-browser and adaptive testing, as well as measures to ensure user data protection are analyzed. Comprehensive testing of the platform's functionality has been carried out.

In the fourth section of the qualification work, the issues of operator performance and the social significance of occupational safety are analyzed in the context of developing and operating a website for finding temporary accommodation. The influence of external and internal factors on the efficiency and safety of developers' work is revealed, as well as the importance of organizing a rational work and rest schedule. Special attention is paid to creating comfortable and safe working conditions, which contribute to preserving health, increasing productivity, and motivating employees. It is emphasized that the implementation of modern occupational safety measures is a necessary condition for the sustainable development of the enterprise, improving the quality of life of the staff, and ensuring the reliable operation of the information platform.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ (НЕПОТРІБНЕ ВИДАЛИТИ З НАЗВИ)

AJAX (англ. Asynchronous JavaScript and XML) – технологія асинхронного обміну даними між клієнтом і сервером без перезавантаження сторінки.

API (англ. Application Programming Interface) – програмний інтерфейс прикладного програмування.

CMS (англ. Content Management System) – система керування контентом.

CSS (англ. Cascading Style Sheets) – каскадні таблиці стилів.

DB, БД – база даних.

FAQ (англ. Frequently Asked Questions) – часто задавані питання.

HTML (англ. HyperText Markup Language) – мова розмітки гіпертексту.

HTTP (англ. HyperText Transfer Protocol) – протокол передачі гіпертексту.

ID – ідентифікатор.

JS (англ. JavaScript) – мова програмування для клієнтської частини веб-додатків.

Leaflet – бібліотека JavaScript для інтерактивних карт.

MySQL – система керування реляційними базами даних.

OAuth – відкритий протокол авторизації.

OpenStreetMap – відкритий картографічний сервіс.

PHPMailer – бібліотека для надсилання електронної пошти з PHP.

PHP (англ. Hypertext Preprocessor) – мова програмування для серверної частини веб-додатків.

SQL (англ. Structured Query Language) – мова структурованих запитів до бази даних.

SSL (англ. Secure Sockets Layer) – протокол захищеного з’єднання.

UI (англ. User Interface) – користувацький інтерфейс.

UX (англ. User Experience) – користувацький досвід.

VS Code (Visual Studio Code) – редактор коду.

ShelterUA – назва вебсайту для пошуку тимчасового житла.

Авторизація – процес перевірки прав доступу користувача.

Бронювання – процес подачі заявки на житло.

Відгук – оцінка або коментар користувача щодо об’єкта чи іншого користувача.

Головна сторінка – основна сторінка вебсайту після входу.

Карта – інтерактивний елемент для вибору або перегляду розташування об’єкта.

Користувач – особа, яка взаємодіє з вебсайтом.

Оголошення – інформація про доступне житло, розміщена власником.

Профіль – персональна сторінка користувача з його даними та активністю.

Сесія – механізм збереження стану користувача під час роботи з сайтом.

Токен – унікальний ідентифікатор для підтвердження дій або нарахування бонусів.

Форма – елемент інтерфейсу для введення та надсилання даних.

Чат – система обміну повідомленнями між користувачами.

ЗМІСТ

ВСТУП

Актуальність теми. Внаслідок війни в Україні мільйони людей були змушені покинути свої домівки та шукати тимчасове житло в інших регіонах країни. Проблема забезпечення переселенців житлом стала однією з найгостріших соціальних викликів сьогодення. Тому розробка вебсайту для пошуку тимчасового житла з використанням MySQL та PHP є актуальним напрямком сучасних досліджень в галузі інформаційних технологій. Такий вебсайт дозволяє швидко знаходити житло для внутрішньо переміщених осіб, полегшує комунікацію між власниками житла та тими, хто його потребує, і сприяє соціальній підтримці населення в умовах кризи.

Мета і задачі дослідження. Метою даної кваліфікаційної роботи освітнього рівня «Бакалавр» є підвищення якості послуг з пошуку тимчасового житла шляхом розробки сучасного вебсайту з використанням MySQL та PHP. Для досягнення поставленої мети потрібно виконати ряд завдань, зокрема:

– Проаналізувати стан досліджень в області веброзробки для пошуку житла;

– Визначити основні вимоги до функціоналу та безпеки вебсайту;

– Розробити архітектуру та структуру бази даних;

– Реалізувати основні модулі вебсайту для реєстрації користувачів, розміщення оголошень, пошуку житла, бронювання та обміну повідомленнями;

– Забезпечити захист персональних даних та безпеку користувачів;

– Провести тестування та оцінити ефективність роботи вебсайту.

Таким чином, у межах цієї роботи буде створено вебсайт, який дозволяє ефективно вирішувати проблему пошуку тимчасового житла для переселенців, забезпечуючи зручний та безпечний інтерфейс для користувачів.

Практичне значення одержаних результатів. Практичне значення одержаних результатів полягає у можливості впровадження розробленого вебсайту для реального використання переселенцями та власниками житла. Вебсайт може бути використаний як ефективний інструмент для організації

пошуку та надання тимчасового житла, що сприятиме оперативному вирішенню житлових питань у кризових ситуаціях.

Крім того, результати цієї роботи можуть бути використані як основа для створення подібних вебсайтів у соціальній сфері, а також для подальшого вдосконалення інформаційних систем, спрямованих на підтримку населення у надзвичайних ситуаціях. Вебсайт може бути адаптований для потреб інших категорій користувачів, наприклад, для студентів, туристів або людей, які шукають житло на короткий термін. Таким чином, розробка має значний потенціал для масштабування та інтеграції з іншими сервісами соціальної підтримки.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАВДАННЯ. НА РОЗРОБКУ ВЕБСАЙТУ ДЛЯ ПОШУКУ ТИМЧАСОВОГО ЖИТЛА З ВИКОРИСТАННЯМ MYSQL ТА PHP

1.1 Аналіз предметної області

В умовах війни в Україні питання пошуку тимчасового житла для переселенців стало надзвичайно актуальним. Багато людей змушені залишати свої домівки та шукати нове житло в інших регіонах країни. Традиційні способи пошуку житла, такі як оголошення на дошках чи в соціальних мережах, не завжди є ефективними, оскільки не забезпечують швидкого доступу до актуальної інформації та не гарантують безпеки користувачів.

Призначення вебсайту полягає у створенні сучасного онлайн-інструменту для пошуку тимчасового житла, який дозволяє переселенцям швидко знаходити доступні варіанти або заповнити заявку на бажане житло, а власникам житла – розміщувати свої пропозиції. Вебсайт автоматизує процеси пошуку, бронювання та комунікації між користувачами, що значно підвищує ефективність вирішення житлових питань у кризових ситуаціях.

Вебсайт орієнтований на дві основні категорії користувачів:

- Переселенці та особи, які шукають житло – можуть переглядати оголошення, залишати заявки на бронювання, спілкуватися з власниками, залишати відгуки.
- Власники житла – мають можливість розміщувати оголошення про доступне житло, переглядати заявки, спілкуватися з потенційними орендарями, отримувати відгуки.

Ключові функції вебсайту:

– Пошук житла за різними критеріями (місто, ціна, кількість кімнат, тип житла тощо).

– Можливість створення та редагування оголошень про житло.

– Оформлення заявок на бронювання житла та їх обробка.

– Система особистих повідомлень для безпечної комунікації між користувачами.

– Механізм відгуків і рейтингів для підвищення довіри та прозорості.

– Додавання оголошень і заявок до обраного для швидкого доступу.

– Система сповіщень про важливі події (нові заявки, зміна статусу, нові повідомлення тощо).

– Відстеження переглядів оголошень і заявок для аналітики.

– Наявність кількох типів відгуків

Всі ці функції реалізовані через відповідні сутності бази даних, що дозволяє ефективно зберігати, обробляти та аналізувати інформацію, а також забезпечувати безпечну та зручну взаємодію між усіма учасниками сервісу.

Таким чином, вебсайт для пошуку тимчасового житла з використанням MySQL та PHP є комплексним рішенням, яке враховує сучасні потреби переселенців та власників житла, забезпечує прозорість, безпеку та зручність у вирішенні житлових питань в умовах кризи.

1.2 Аналіз наявних рішень

На сучасному ринку існує низка вебсайтів, які дозволяють здійснювати пошук житла для оренди або купівлі. Серед найпопулярніших українських сервіси – OLX Нерухомість, DOM.RIA. Вони надають користувачам можливість розміщувати оголошення про житло, здійснювати пошук за різними критеріями, переглядати фотографії, читати відгуки та зв’язуватися з власниками.

Однак більшість із цих вебсайтів орієнтовані на комерційний ринок нерухомості та не враховують специфічних потреб переселенців, які шукають тимчасове житло в умовах війни. Такі сервіси часто не мають спеціальних фільтрів для пошуку безкоштовного або соціального житла, не забезпечують достатнього рівня безпеки для вразливих категорій населення, а також не завжди містять функціонал для швидкої комунікації між власниками та орендарями.

Деякі благодійні організації та волонтерські ініціативи створюють власні вебсайти або форми для збору заявок на житло, проте ці рішення зазвичай мають обмежений функціонал, не підтримують автоматизацію процесів пошуку, бронювання та зворотного зв’язку. Відсутність єдиної централізованої системи ускладнює пошук житла для переселенців та знижує ефективність допомоги.

Тому аналіз наявних рішень показує, що існує потреба у створенні спеціалізованого вебсайту для пошуку тимчасового житла, який би враховував особливості ситуації в Україні, забезпечував зручний інтерфейс, безпечну комунікацію, систему відгуків та можливість швидкого пошуку житла для переселенців.

1.2.1 OLX Нерухомість

OLX Нерухомість є однією з найпопулярніших платформ для пошуку житла в Україні. Вона має широку аудиторію, дозволяє користувачам розміщувати та переглядати оголошення, застосовувати базові фільтри за містом, ціною, кількістю кімнат, а також контактувати з власниками через вбудовану форму. Перевагою OLX є велика кількість пропозицій та простий інтерфейс, що робить платформу доступною для різних категорій користувачів [1].

Однак для переселенців та людей, які шукають тимчасове житло, ця платформа має низку суттєвих недоліків.

По-перше, на OLX відсутній окремий розділ або спеціальні фільтри для тимчасового житла саме для переселенців. Це ускладнює пошук відповідних варіантів у кризових ситуаціях, оскільки користувачам доводиться переглядати багато нерелевантних оголошень.

По-друге, користувачі не можуть залишати заявки на бажане житло, якщо не знаходять потрібного оголошення. Відсутня можливість створення запиту, який могли б побачити власники житла, що обмежує комунікацію між тими, хто шукає житло, та тими, хто може його надати.

Також варто зазначити, що OLX Нерухомість не має інтуїтивно зрозумілого та зручного інтерфейсу для переселенців, які шукають тимчасове житло. Детальніше можна побачити на Рисунку 1.1.

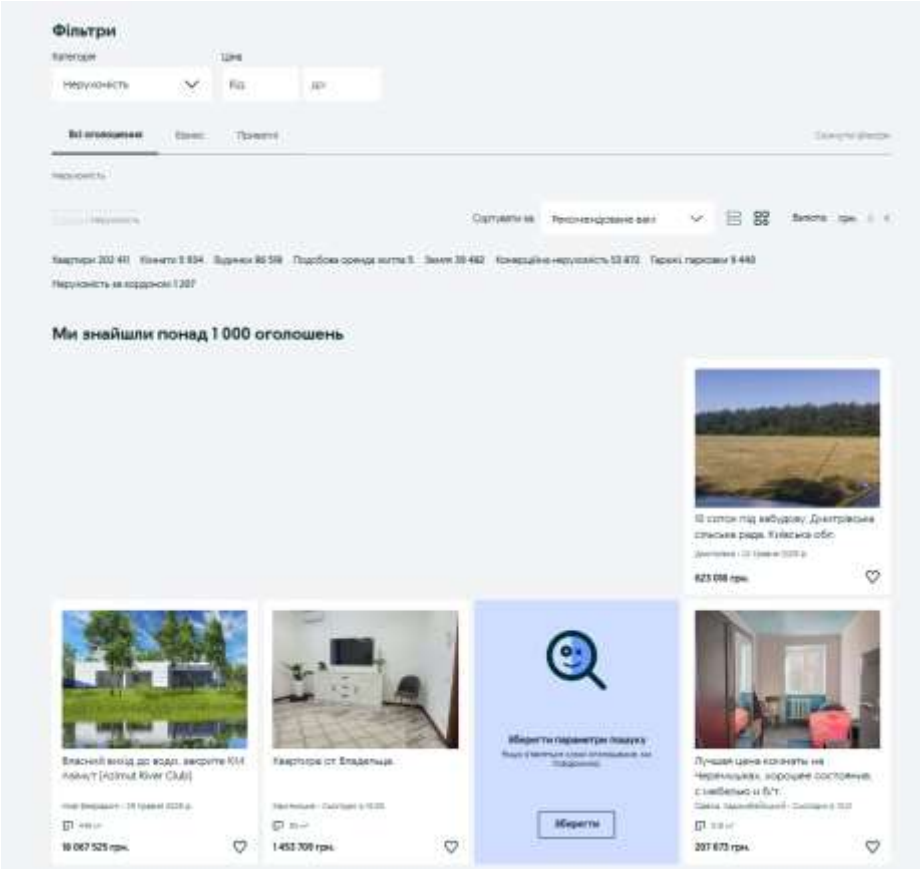


Рисунок 1.1 – Олх нерухомість

Більше того на платформі відсутня інтегрована карта, яка дозволяє швидко переглянути розташування квартири або будинку, що ускладнює вибір оптимального варіанту за місцем розташування.

1.2.2 DOM.RIA

DOM.RIA є одним із провідних українських вебсайтів для пошуку та розміщення оголошень про нерухомість [2]. Платформа пропонує широкий вибір квартир, будинків, кімнат для оренди та купівлі, а також надає користувачам можливість застосовувати розширені фільтри за містом, ціною, кількістю кімнат,

площею, типом житла тощо. Однією з переваг DOM.RIA є наявність перевірених оголошень із фотографіями, зробленими спеціальними агентами, що підвищує довіру до пропозицій. Також на сайті реалізована інтегрована карта, яка дозволяє переглядати розташування об’єктів на місцевості, що є зручним для користувачів [3].

Однак, незважаючи на сучасний інтерфейс та багатий функціонал, DOM.RIA не орієнтований спеціально на потреби переселенців та людей, які шукають тимчасове житло. Інтерфейс вебсайту можна побачити на Рисунку 1.2.

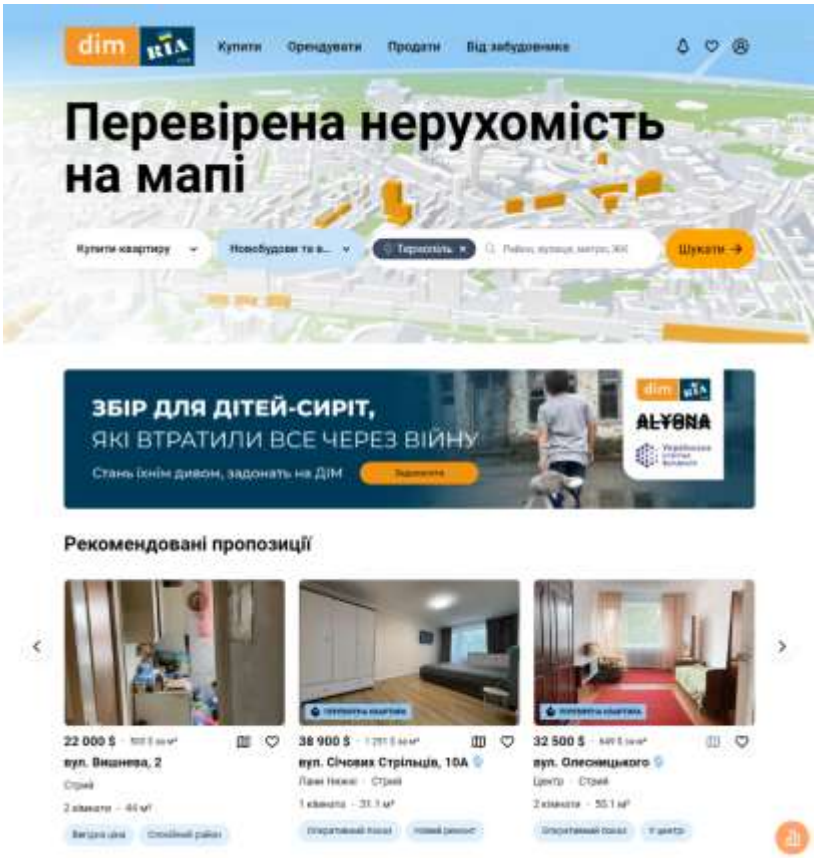


Рисунок 1.2 – DOM.RIA

На платформі відсутній окремий розділ або спеціальні фільтри для пошуку саме тимчасового житла для переселенців, що ускладнює процес пошуку відповідних варіантів у кризових ситуаціях.

Крім того, DOM.RIA не надає можливості залишати заявки на бажане житло, якщо користувач не знаходить потрібного оголошення. Відсутня функція

створення запиту, який могли б побачити власники житла, що обмежує комунікацію між тими, хто шукає житло, та потенційними орендодавцями.

Ще одним недоліком є те, що на DOM.RIA відсутня система відгуків про оголошення, що ускладнює оцінку надійності власників житла та якості пропозицій.

Таким чином, хоча DOM.RIA і є зручною платформою для пошуку нерухомості, вона не враховує специфічних потреб переселенців, які шукають тимчасове житло, і не забезпечує необхідного рівня гнучкості та зручності для цієї категорії користувачів.

1.2.3 Порівняння наявних рішень

OLX Нерухомість має простий інтерфейс та велику кількість оголошень, але не підтримує інтегровану карту, спеціальні фільтри для тимчасового житла та систему заявок. DOM.RIA пропонує сучасний дизайн, розширені фільтри та карту для перегляду об’єктів, але також не має окремого розділу для переселенців і не дозволяє залишати заявки на житло.

У розробці власного вебсайту для пошуку тимчасового житла будуть враховане найкраще – зручний інтерфейс, розширені фільтри та карта розташування житла. При цьому буде усунуто їхні недоліки: реалізовано спеціальні функції для переселенців, можливість створення заявок, систему відгуків і безпечну комунікацію між користувачами.

1.2.4 Обґрунтування вибору інтерфейсу користувача вебсайту для пошуку тимчасового житла

Після аналізу існуючих рішень та вивчення потреб цільової аудиторії було розроблено власну концепцію інтерфейсу користувача для платформи ShelterUA. Основними принципами стали простота, інтуїтивність та

відповідність сучасним стандартам UX/UI. Те як буде виглядати сторінка можна побачити на Рисунку 1.3

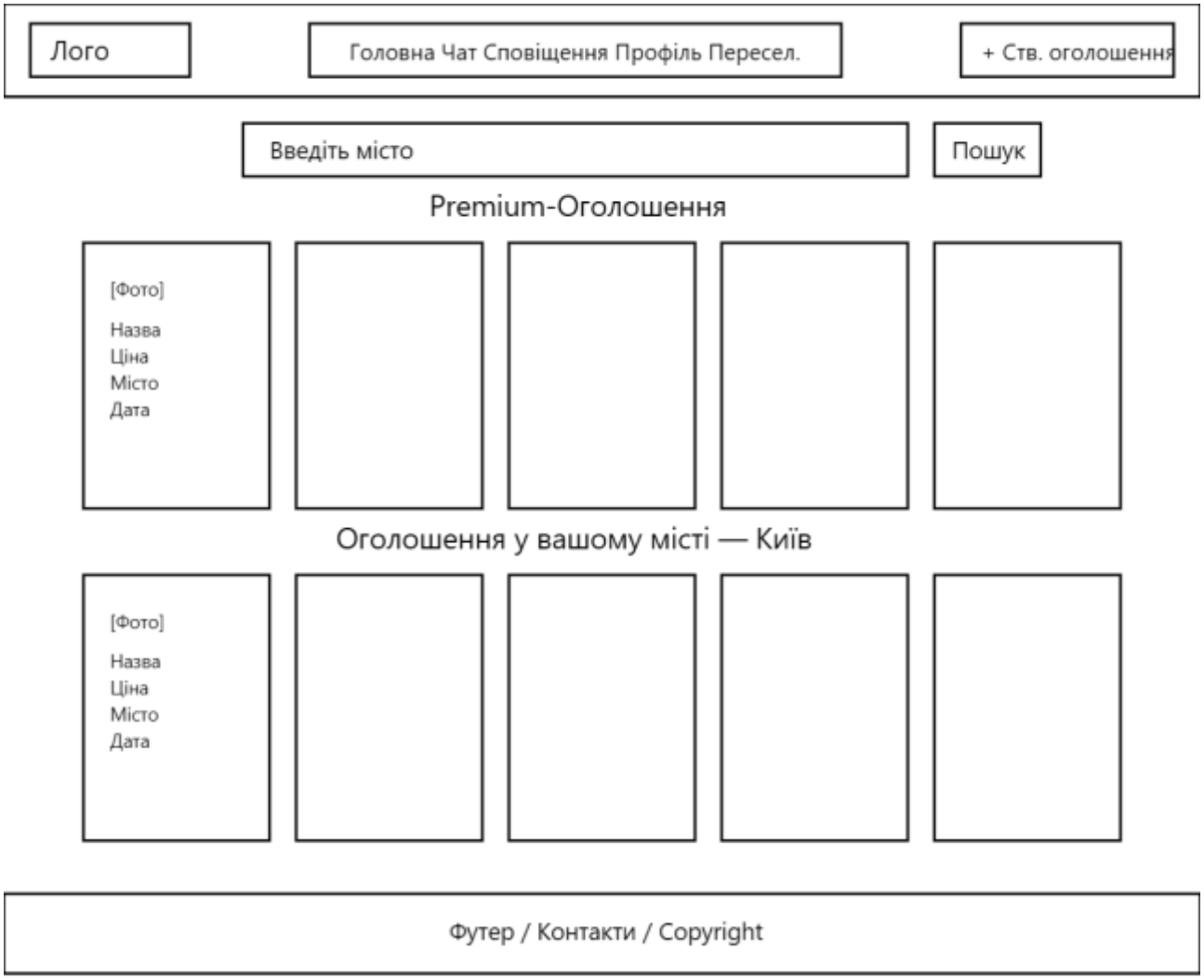


Рисунок 1.3 – Макет інтерфейсу вебсайту для пошуку тимчасового житла

Головна навігаційна панель (хедер) розташована у верхній частині сторінки, що відповідає звичному для більшості користувачів розташуванню меню на сучасних вебсайтах. Логотип платформи розміщено зліва на хедері – це підсилює впізнаваність бренду та дозволяє швидко повернутися на головну сторінку одним кліком.

Основний контент кожної сторінки розташований по центру, що забезпечує зручність сприйняття інформації незалежно від розміру екрана та пристрою користувача. Такий підхід дозволяє сфокусувати увагу на головних

елементах сторінки, спрощує навігацію та підвищує загальну зручність користування сайтом.

Для оформлення інтерфейсу обрано синьо-жовту кольорову гаму, яка асоціюється з національною ідентичністю України, викликає довіру та позитивні емоції у користувачів. Синій колір використовується для основних елементів навігації , жовтий – для акцентів, кнопок, сповіщень та виділення важливої інформації.

Такий дизайн був обраний на основі аналізу сучасних тенденцій веб-розробки, рекомендацій щодо зручності користування та результатів тестування прототипів серед представників цільової аудиторії [4]. Він забезпечує швидке освоєння інтерфейсу новими користувачами, підвищує ефективність взаємодії з платформою та сприяє формуванню позитивного користувацького досвіду.

1.2.5 Розробка логотипу

Для вебсайту пошуку тимчасового житла – ShelterUA було розроблено унікальний логотип, який відображає основну ідею сервісу – допомогу з пошуком житла в Україні, детальніше на Рисунку 1.4.

Рисунок 1.4 – Логотип вебсайту для пошуку тимчасового житла

Основними елементами логотипу є стилізований дах будинку у синьо-жовтих кольорах, що символізують національну ідентичність, захист і затишок.

Центральний елемент – будинок із вікнами, який підкреслює спрямованість платформи на вирішення житлових питань.

Вибір синьо-жовтої гами обумовлений асоціацією з державними кольорами України, що викликає довіру та підкреслює патріотичний характер проєкту. Округлі форми логотипу створюють відчуття безпеки та відкритості.

Логотип було створено власноруч із використанням графічного редактора на локальному комп’ютері. Остаточний варіант логотипу збережено за шляхом logo.png. Для створення логотипу було проведено аналіз графічних програм, детальніше у табличці 1.1.

Табличка 1.1 – Порівняння програмного забезпечення для розробки логотипу

Програма	**Платформа**	**Переваги**	**Недоліки**
Adobe Illustrator	Windows/Mac	Професійний, векторна графіка, багато інструментів	Платний, складний для новачків
Figma	Web/Windows/Mac	Безкоштовний тариф, спільна робота, простий інтерфейс	Менше функцій для складної графіки
CorelDRAW	Windows	Потужний, підтримка різних форматів	Платний, менш популярний
Canva	Web	Дуже простий, багато шаблонів, безкоштовний	Обмежені можливості редагування
Inkscape	Windows/Mac/Linux	Безкоштовний, векторна графіка	Менш інтуїтивний інтерфейс

Для створення логотипу було обрано Adobe Illustrator, оскільки він надає найширші можливості для роботи з векторною графікою, дозволяє створювати

якісні та масштабовані зображення, а також експортувати їх у різних форматах для подальшого використання на сайті [5].

1.3 Формування вимог вебсайту для пошуку тимчасового житла з використанням MySQL та PHP

Вимоги до вебсайту для пошуку тимчасового житла охоплюють широкий спектр функціональних та нефункціональних аспектів, які забезпечують зручність, безпеку та ефективність роботи сервісу [6]. Саме тому для проєкту було обрано назву ShelterUA, що підкреслює його основне призначення – надання допомоги у пошуку житла для переселенців та всіх, хто потребує тимчасового прихистку в Україні. Назва є лаконічною, легко запам’ятовується, асоціюється з безпекою та підтримкою, а також відображає національну спрямованість сервісу.

Однією з ключових функціональних вимог є система реєстрації та автентифікації користувачів. Вона передбачає можливість створення облікового запису з перевіркою електронної пошти та телефону, вхід через email/пароль або Google OAuth, відновлення паролю за допомогою email із підтвердженням токена, а також захист особистого кабінету від неавторизованого доступу.

Профіль користувача дозволяє переглядати та редагувати особисті дані, змінювати пароль, завантажувати або оновлювати фото профілю, переглядати власні оголошення, бронювання та обране, а також видаляти акаунт.

Оголошення (listings) реалізовані з можливістю створення детального опису, додавання до восьми фотографій, вказання ціни, адреси, координат на інтегрованій карті, а також розширених характеристик, таких як категорія, тип, кількість кімнат, площа, поверх, стан, наявність меблів, інтернету, паркування, дозволу на тварин чи куріння тощо. Користувачі можуть редагувати та видаляти свої оголошення, переглядати статистику переглядів, лайків і коментарів, додавати оголошення до обраного, користуватися преміум-можливостями

(виділення, підняття в топ, оплата), а також отримувати сповіщення про завершення терміну дії оголошення.

Функціонал бронювання (bookings) або створення запитів на житло включає детальний опис, бажані параметри, бюджет, координати, можливість редагування та видалення запиту, перегляд статистики, додавання до обраного, преміум-бронювання та автоматичне завершення терміну дії із відповідними сповіщеннями. Пошук і фільтрація реалізовані за містом, адресою, ціною, кількістю кімнат, площею, категорією, станом тощо, з інтерактивною картою для перегляду розташування об’єктів, автозаповненням міст, групуванням і сортуванням результатів.

Взаємодія між користувачами забезпечується вбудованим чатом для комунікації між власниками та переселенцями, можливістю надсилати повідомлення та файли, переглядати історію чатів і фільтрувати їх за роллю. Система відгуків і рейтингу дозволяє залишати відгуки про користувачів, оголошення та бронювання, використовувати зіркову систему оцінювання, переглядати середній рейтинг і залишати лише один відгук на об’єкт чи користувача.

Для стимулювання активності користувачів на платформі реалізовано систему токенів. За кожне створене оголошення користувач отримує токени, які може використати для активації преміум-статусу своїх оголошень або бронювань. Преміум-статус дозволяє виділити оголошення серед інших, підвищити його видимість та залучити більше потенційних орендарів. Вартість активації преміум-статусу становить 5 токенів. Якщо користувач не має достатньої кількості токенів, система повідомляє, скільки ще потрібно для активації преміум-можливостей.

Система сповіщень інформує про нові повідомлення, зміну статусу оголошення чи бронювання, спливання терміну дії, активацію преміум-можливостей, а також дозволяє переглядати, видаляти та відмічати сповіщення як прочитані. Для оголошень і бронювань передбачені стандартні та преміум-

плани з описом переваг, можливістю виділення, продовження терміну дії, підняття в топ, а також симуляцією оплати.

Інтерфейс сайту має бути інтуїтивно зрозумілим, адаптивним для різних ролей користувачів, підтримувати мобільну версію, зручну навігацію, вкладки, фільтри, сортування, а також модальні вікна для форм, карт і відгуків. Додаткові сторінки включають головну з описом цілей і переваг, сторінки «Про нас», «Послуги», «Контакти» (з формою зворотного зв'язку), «Політика конфіденційності», «FAQ», а також відображення популярних міст, інструкцій і кроків пошуку.

Окрему увагу приділено питанням безпеки: захист від несанкціонованого доступу, валідація та санітизація даних, захист від XSS і CSRF-атак, перевірка унікальності email та телефону.

Окрім функціональних вимог, існують також нефункціональні вимоги, які стосуються продуктивності, масштабованості, надійності, зручності використання, доступності, підтримки різних браузерів і пристроїв, а також можливості подальшого розвитку та інтеграції з іншими сервісами. Всі ці вимоги разом формують комплексну основу для створення сучасного, безпечного та зручного вебсайту для пошуку тимчасового житла [7dddddd].

1.4 Основні учасники та їх функції на вебсайті для пошуку тимчасового житла

У процесі розробки вебсайту для пошуку тимчасового житла важливо визначити основних учасників, які взаємодіють із системою, а також їхні функції. Це дозволяє чітко структурувати функціонал, забезпечити зручність користування та підвищити ефективність роботи вебсайту [8].

Основними учасниками вебсайту є:

1. Незареєстрований користувач

Це відвідувач, який не має облікового запису на вебсайті. Він може: переглядати загальний список оголошень про житло; ознайомлюватися з

описами, фотографіями та характеристиками житла; використовувати базові фільтри для пошуку; переглядати інформаційні сторінки (про сайт, контакти, політика конфіденційності, FAQ).

2. Зареєстрований користувач

Після реєстрації користувач отримує доступ до розширеного функціоналу вебсайту. І може обирати свою роль. Тобто «Власник житла» чи «Переселенець»

3. Власники житла

Ця категорія користувачів також є зареєстрованими, але їх основна функція розміщення оголошень про доступне житло. Вони можуть: створювати, редагувати та видаляти оголошення; переглядати заявки на свої оголошення; відповідати на повідомлення від потенційних орендарів; переглядати відгуки про свої оголошення; отримувати сповіщення про нові заявки та зміни статусу оголошень.

1. Переселенці та особи, які шукають житло

Ця категорія користувачів реєструється на вебсайті з метою пошуку тимчасового житла. Вони можуть: переглядати оголошення про доступне житло; використовувати фільтри для пошуку за різними критеріями; залишати заявки на бронювання житла; додавати оголошення до обраного; спілкуватися з власниками житла через вбудований чат; залишати відгуки про житло, власників та роботу вебсайту. На рисунку 1.5 показано взаємодії учасників вебсайту.

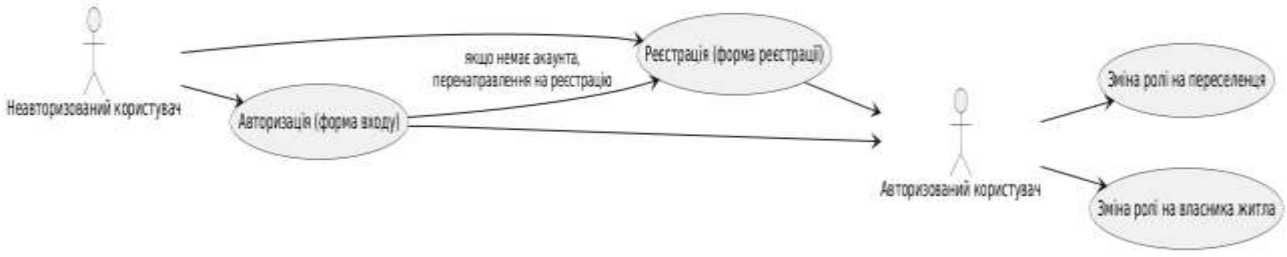


Рисунок 1.5 – Діаграма взаємодії користувача для сайту з тимчасового пошуку житла

Діаграма варіантів використання демонструє початкові сценарії взаємодії користувача з вебсайтом для пошуку тимчасового житла [9]. На ній представлені два типи акторів: «Неавторизований користувач» і «Авторизований користувач».

Неавторизований користувач може виконати одну з двох дій – перейти до форми реєстрації або до форми авторизації. У разі відсутності облікового запису спроба входу на сайт автоматично перенаправляє користувача на форму реєстрації. Після успішного створення облікового запису або авторизації користувач переходить у стан «Зареєстрований користувач».

Авторизований користувач з отриманим обліковим записом у будь-який момент може змінити свою роль на одну з двох доступних: «Переселенець» або «Власник житла». Перехід здійснюється через будь яку сторінку сайту, що забезпечує гнучкість у використанні сервісу й відповідає потребам користувачів у різних життєвих обставинах. Ця модель дозволяє чітко окреслити початкові точки входу в систему та адаптивність ролей зареєстрованих користувачів.

Для забезпечення безпеки та зручності користування вебсайтом важливо чітко визначити, які сторінки та функції доступні різним категоріям користувачів. Матриця доступу дозволяє наочно представити розподіл прав доступу між незареєстрованими, зареєстрованими користувачами, власниками житла та переселенцями (див. таблиця А.1.1).

Аналіз матриці доступу показує, що кожна роль на сайті має власний набір можливостей, що сприяє ефективній організації роботи сервісу, підвищує рівень безпеки та забезпечує індивідуальний підхід до потреб різних користувачів.

1.5 Варіанти використання вебсайту для пошуку тимчасового житла з використанням MySQL та PHP

Діаграма варіантів використання (use case diagram) ілюструє основні сценарії взаємодії користувачів із вебсайтом для пошуку тимчасового житла. Вона відображає можливості для різних типів користувачів – неавторизованого,

авторизованого, власника житла та переселенця – та демонструє, які функції доступні кожній ролі на різних етапах роботи із системою (див. рис. Б.1.1).

Дана діаграма дозволяє чітко побачити, як організована навігація та розподіл функціоналу між різними категоріями користувачів. Вона допомагає визначити, які дії є спільними, а які – специфічними для певної ролі, що сприяє подальшому проектуванню зручного та безпечного інтерфейсу вебсайту [9].

1.6 Аналіз підходів до розробки вебсайту

У сучасній веб-розробці існує декілька основних підходів до створення сайтів, кожен з яких має свої особливості, переваги та недоліки. Найбільш поширеними є використання готових систем управління контентом (CMS), застосування онлайн-конструкторів сайтів, а також розробка вебсайту з нуля із використанням мов програмування та фреймворків.

Готові CMS, такі як WordPress, Joomla чи Drupal, дозволяють швидко створити сайт із базовим функціоналом завдяки великій кількості шаблонів і плагінів. Вони ідеально підходять для невеликих проєктів, блогів, корпоративних сайтів або інтернет-магазинів із типовою структурою. Основними перевагами такого підходу є швидкість розгортання, наявність великої спільноти підтримки та відносна простота адміністрування. Проте, при спробі реалізувати складну бізнес-логіку або інтегрувати унікальні сервіси, CMS часто виявляються недостатньо гнучкими. Додаткові модулі можуть конфліктувати між собою, а внесення змін у структуру бази даних або логіку роботи сайту потребує значних зусиль [10].

Онлайн-конструктори сайтів (наприклад, Wix, Tilda, Weblium) орієнтовані на користувачів без глибоких технічних знань. Вони дозволяють створювати прості сайти за допомогою візуального редактора, використовуючи готові блоки та шаблони. Основна перевага такого підходу – це максимальна простота та швидкість запуску проєкту. Однак функціональні можливості конструкторів обмежені, а реалізація складних сценаріїв взаємодії, інтеграція з зовнішніми

сервісами чи розробка індивідуального дизайну часто неможливі або потребують додаткових витрат [11].

Розробка вебсайту з нуля із використанням мов програмування (наприклад, PHP, JavaScript) та систем управління базами даних (MySQL, PostgreSQL) є найбільш гнучким і масштабованим підходом. Такий спосіб дозволяє повністю контролювати архітектуру проєкту, реалізовувати будь-які бізнес-процеси, інтегрувати сторонні сервіси, забезпечувати високий рівень безпеки та оптимізувати продуктивність. Недоліком цього підходу є більша тривалість розробки, необхідність залучення кваліфікованих спеціалістів та вищі початкові витрати. Проте саме цей підхід дозволяє створити унікальний продукт, який максимально відповідає вимогам замовника та може легко масштабуватися у майбутньому [12].

Для проєкту вебсайту пошуку тимчасового житла, який передбачає наявність різних ролей користувачів, складну систему фільтрації, інтеграцію з картою, систему відгуків, повідомлень та сповіщень, питання вибору підходу до розробки є особливо важливим. Необхідно враховувати не лише поточні вимоги, а й можливість подальшого розвитку та розширення функціоналу.

1.6.1 Обґрунтування вибору оптимального підходу до розробки

З урахуванням специфіки проєкту та аналізу існуючих підходів, оптимальним рішенням для розробки вебсайту пошуку тимчасового житла є створення системи з нуля із використанням PHP та MySQL. Такий вибір обумовлений необхідністю реалізації унікального функціоналу, який неможливо або дуже складно впровадити за допомогою готових CMS чи конструкторів сайтів.

Розробка з нуля дозволяє повністю контролювати структуру бази даних, логіку взаємодії між користувачами, забезпечити високий рівень безпеки персональних даних та захист від несанкціонованого доступу. Це особливо важливо для сервісу, який працює з чутливою інформацією та передбачає різні

сценарії взаємодії між переселенцями та власниками житла. Крім того, власна розробка дає змогу реалізувати складну систему пошуку та фільтрації, інтегрувати інтерактивну карту для зручного перегляду розташування об’єктів, впровадити систему відгуків, рейтингів, особистих повідомлень та сповіщень.

Ще однією перевагою такого підходу є можливість гнучко адаптувати інтерфейс під потреби цільової аудиторії, враховувати особливості користувацького досвіду та впроваджувати нові функції у відповідь на зміну ринку чи зворотний зв’язок від користувачів. У майбутньому це дозволить легко масштабувати систему, додавати нові модулі, інтегрувати сторонні сервіси або змінювати бізнес-логіку без суттєвих обмежень.

Отже, вибір розробки вебсайту з нуля із використанням PHP та MySQL є найбільш доцільним для даного проєкту. Це забезпечить максимальну відповідність функціоналу вимогам, високу продуктивність, безпеку та можливість подальшого розвитку сервісу для пошуку тимчасового житла.

1.6.2 Етапи життєвого циклу розробки та експлуатації вебсайту

Життєвий цикл вебсайту охоплює всі основні етапи – від початкового аналізу вимог до супроводу та подальшого розвитку системи. Кожен етап має свої завдання, результати та впливає на якість кінцевого продукту. Чітке дотримання етапів життєвого циклу дозволяє мінімізувати ризики, забезпечити відповідність функціоналу очікуванням користувачів та гарантувати стабільну роботу вебсайту в довгостроковій перспектив, детальніше Рисунок 1.6.

Опис етапів:

Аналіз вимог – на цьому етапі визначаються цілі проєкту, основні функції, ролі користувачів, особливості предметної області та вимоги до системи. Проводиться аналіз аналогів, збір побажань замовника, формулюється перелік функціональних і нефункціональних вимог.

Проектування архітектури – визначається структура бази даних, логіка взаємодії компонентів, розробляються макети інтерфейсу, обираються

технології та інструменти. Створюється технічна документація, що слугує основою для подальшої розробки.

Розробка (програмування) – відбувається безпосереднє написання коду, створення бази даних, реалізація функціоналу згідно з вимогами та архітектурою. Проводиться інтеграція окремих модулів у єдину систему.

Тестування – перевіряється коректність роботи всіх компонентів, виявляються та виправляються помилки, оцінюється зручність інтерфейсу, безпека та продуктивність системи. Проводиться як модульне, так і інтеграційне тестування.

Рисунок 1.6 – Життєвий цикл сайт з пошуку тимчасового житла

Впровадження (деплоймент) – готовий вебсайт розміщується на сервері, налаштовуються доменне ім’я, сертифікати безпеки, резервне копіювання. Система стає доступною для кінцевих користувачів.

Експлуатація та супровід – забезпечується стабільна робота вебсайту, моніторинг працездатності, регулярне оновлення програмного забезпечення, резервне копіювання даних, оперативне усунення можливих збоїв.

Модернізація та розвиток – відбувається додавання нового функціоналу, оптимізація існуючих процесів, врахування зворотного зв’язку від користувачів,

впровадження сучасних технологій для підвищення конкурентоспроможності та зручності сервісу.

Дотримання структурованого життєвого циклу розробки вебсайту дозволяє створити якісний, надійний та гнучкий продукт, який відповідає сучасним вимогам і може ефективно розвиватися відповідно до потреб користувачів [14].

1.7 Вибір технологічного стеку для розробки вебсайту

У цьому підрозділі обґрунтовується вибір основних технологій, які використовуються для створення вебсайту пошуку тимчасового житла. Враховуючи вимоги до функціональності, безпеки, масштабованості та зручності підтримки, було обрано класичний стек. PHP обрано як основну серверну мову завдяки її поширеності, гнучкості та великій кількості готових рішень для розробки вебдодатків. MySQL використовується як система керування базами даних через її надійність, швидкодію та простоту інтеграції з PHP.

Для клієнтської частини застосовуються HTML5, CSS3 та JavaScript, що забезпечує сучасний, адаптивний та зручний інтерфейс для користувачів різних пристроїв. Для покращення візуального сприйняття та зручності навігації використовується бібліотека іконок Font Awesome, яка дозволяє швидко додавати сучасні іконки до інтерфейсу без необхідності створювати власні графічні елементи. Для впізнаваності сайту використовується власне лого, favicon (images/logo.png), який відображається у вкладці браузера. Додатково, для реалізації інтерактивної карти може використовуватися бібліотека Leaflet.

Такий вибір технологій забезпечує ефективну розробку, легку підтримку та можливість подальшого масштабування вебсайту відповідно до зростаючих потреб користувачів [15].

1.8 Вибір та обґрунтування середовища розробки для вебсайту для пошуку тимчасового житла

Для ефективної розробки важливо обрати зручне, функціональне та гнучке середовище розробки. Основними критеріями вибору є підтримка PHP, інтеграція з системами контролю версій, наявність розширень для роботи з HTML, CSS, JavaScript, зручність налагодження, підтримка роботи з базами даних, а також швидкість і простота налаштування локального серверного оточення.

На початковому етапі локальна розробка та тестування вебсайту здійснювалися за допомогою програмного комплексу OpenServer [16], який забезпечує швидке розгортання серверного середовища (PHP, MySQL) на Windows. Для написання коду, організації структури проєкту та роботи з файлами було обрано редактор Visual Studio Code (VS Code), який відповідає всім сучасним вимогам до розробки PHP-проєктів [17]. Для вибору було побудовано таку порівняльну таблицю 1.2.

Таблиця 1.2 - Порівняльна таблиця середовищ розробки

Критерій	**VS Code**	**PhpStorm**	**Sublime Text**	**Notepad++**
Підтримка PHP	+	+	+	+
Підтримка HTML/CSS/JS	+	+	+	+
Інтеграція з Git	+	+	+	–
Розширення та плагіни	+	+	+	+
Термінал	+	+	–	–
Підсвічування синтаксису	+	+	+	+
Автодоповнення	+	+	+	–

Продовження таблиці 1.2

Вартість	Безкоштовно	Платно	Частково	Безкоштовно
Зручність	+	+	+	+
Підтримка БД	+	+	–	–

З огляду на порівняльний аналіз, для розробки вебсайту було обрано Visual Studio Code. Цей редактор є безкоштовним, має широку підтримку розширень для PHP, HTML, CSS, JavaScript, інтегрується з Git, містить вбудований термінал та зручний інтерфейс для роботи з файлами. Важливим фактором вибору стало й те, що я вже тривалий час працюю у VS Code, добре знайомий з його можливостями та налаштуваннями, що дозволяє максимально ефективно організувати робочий процес.

1.9 Висновок до першого розділу

У першому розділі було проведено комплексний аналіз предметної області, визначено актуальність проблеми пошуку тимчасового житла для переселенців в Україні та сформульовано основні вимоги до майбутнього вебсайту. Розглянуто існуючі рішення на ринку, такі як OLX Нерухомість, DOM.RIA, проаналізовано їхні переваги й недоліки, а також обґрунтовано необхідність створення спеціалізованого сервісу, орієнтованого саме на потреби переселенців.

У процесі аналізу було визначено основні категорії користувачів, їхні ролі та функції на платформі, а також побудовано діаграми взаємодії та доступу до функціоналу. Сформовано детальний перелік функціональних і нефункціональних вимог, що охоплюють питання реєстрації, профілю, роботи з оголошеннями та бронюваннями, пошуку, комунікації, відгуків, сповіщень, безпеки та зручності інтерфейсу.

Окремо розглянуто підходи до розробки вебсайтів, обґрунтовано вибір створення системи з нуля на основі PHP та MySQL, а також описано життєвий

цикл розробки проєкту. Вибір технологічного стеку та середовища розробки підтверджено порівняльним аналізом, що дозволяє забезпечити гнучкість, масштабованість і відповідність сучасним стандартам.

Таким чином, результати першого розділу створюють надійну теоретичну та методологічну основу для подальшого проєктування і реалізації вебсайту для пошуку тимчасового житла, який максимально враховує потреби цільової аудиторії та сучасні вимоги до функціоналу, безпеки й зручності використання.

РОЗДІЛ 2. ПРОЄКТУВАННЯ ТА РОЗРОБКА ВЕБСАЙТУ ДЛЯ ПОШУКУ ТИМЧАСОВОГО ЖИТЛА З ВИКОРИСТАННЯМ MYSQL ТА PHP

2.1 Моделювання та опис рівнів архітектури вебсайту для пошуку тимчасового житла

Архітектура вебсайту для пошуку тимчасового житла реалізована у вигляді трирівневої структури, що чітко розділяє функціональні обов’язки між різними компонентами системи [18]. На клієнтському рівні знаходиться браузер користувача, який взаємодіє з інтерфейсом сайту, розробленим із використанням HTML, CSS та JavaScript. Важливими елементами цього рівня є інтерактивна карта (Leaflet) для відображення розташування житла та бібліотека іконок Font Awesome, що підвищує зручність і сучасність інтерфейсу. Саме тут користувачі здійснюють пошук, перегляд оголошень, надсилання заявок, спілкування та інші дії.

Серверний рівень, реалізований на PHP, відповідає за обробку всіх запитів від клієнта. Тут функціонують окремі модулі: модуль оголошень, модуль бронювань, модуль чату, модуль сповіщень та модуль користувачів. Всі ці компоненти координуються контролером запитів, який приймає дані від інтерфейсу, викликає відповідні модулі, обробляє бізнес-логіку та формує відповіді. Генератор HTML-відповідей забезпечує створення сторінок або даних, які повертаються користувачу.

На рівні даних розташована база даних MySQL, у якій зберігається вся інформація про користувачів, оголошення, бронювання, повідомлення, сповіщення тощо. Серверні модулі взаємодіють із базою даних через SQL-запити, отримують необхідні дані, обробляють їх і передають назад для відображення на клієнтському рівні [20]. На Рисунку 2.1 показано діаграму архітектури вебсайту.

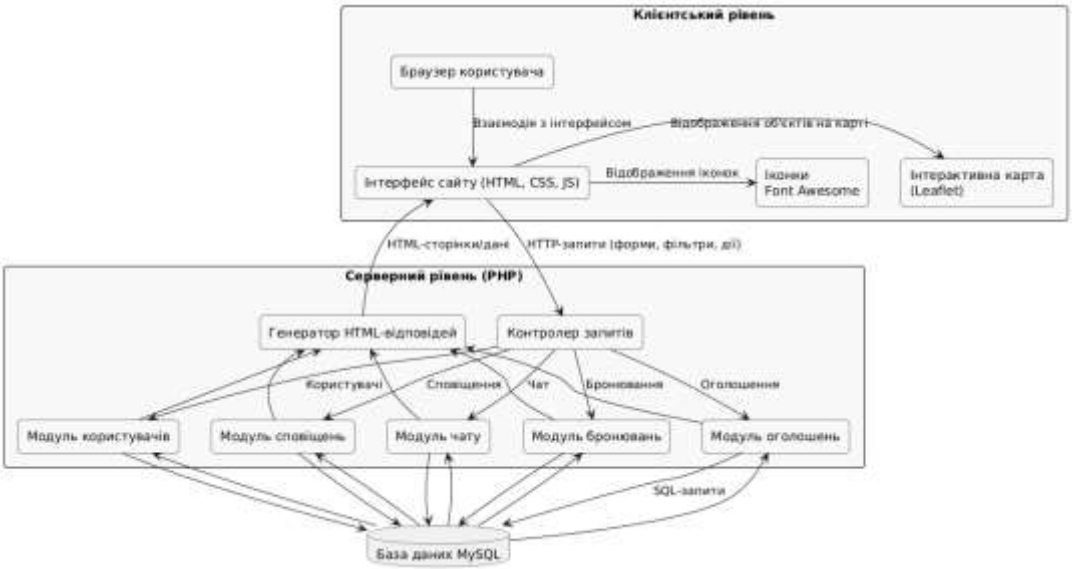


Рисунок 2.1 – Архітектура вебсайту для пошуку тимчасового житла

Така архітектура забезпечує масштабованість, безпеку та зручність супроводу вебсайту, а також дозволяє легко впроваджувати новий функціонал відповідно до потреб користувачів.

Також було розроблено діаграму послідовності [20], Рисунок 2.2 ілюструє основні сценарії взаємодії компонентів системи під час створення оголошення власником житла та формування запиту на бажане житло (бронювання) переселенцем.

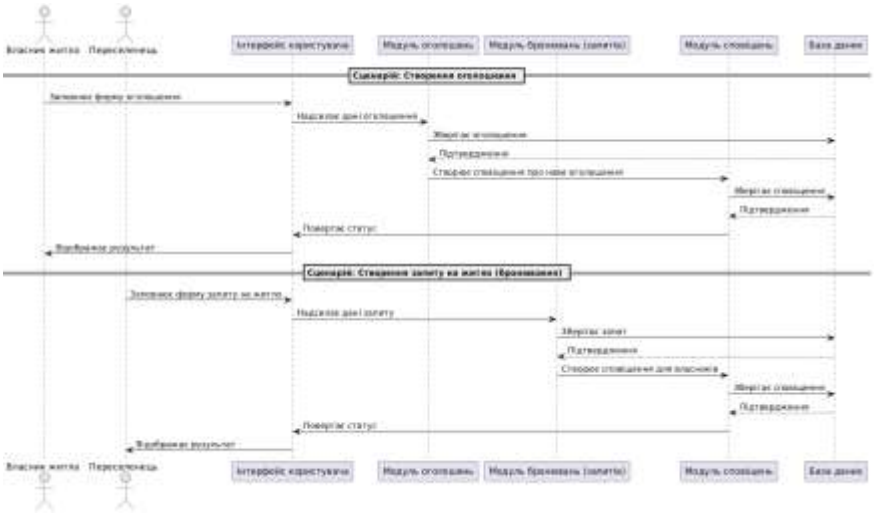


Рисунок 2.2 – Діаграма послідовності створення оголошення/бронювання для пошуку тимчасового житла

На ній відображено послідовність обміну повідомленнями між користувачем, інтерфейсом, відповідними модулями, системою сповіщень та базою даних. Це дозволяє наочно побачити, як саме реалізується логіка додавання нових оголошень і запитів у системі).

Для ілюстрації процесу взаємодії під час залишення відгуку про оголошення було побудовано діаграму кооперації на Рисунку 2.3. Вона демонструє, як користувач через інтерфейс передає дані відгуку до відповідного модуля, який зберігає їх у базі даних та повертає результат операції

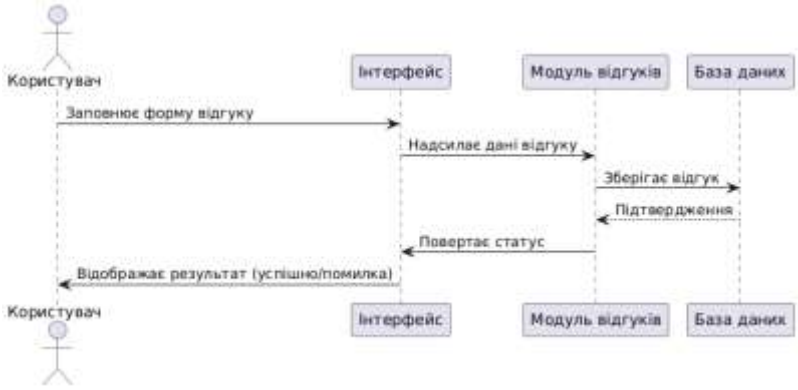


Рисунок 2.3 – Діаграма послідовності для відгуків

Для наочного відображення логіки додавання оголошення до обраного створено діаграму кооперації, детальніше на Рисунку 2.4. На ній показано, як користувач ініціює цю дію через інтерфейс, а система обробляє запит, зберігає інформацію у базі даних і повідомляє користувача про результат.

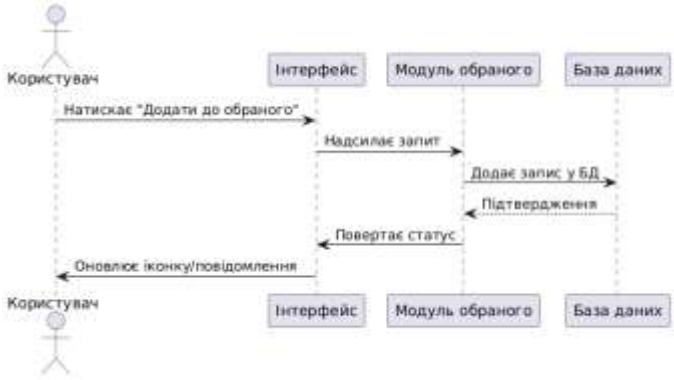


Рисунок 2.4 – Діаграма послідовності для обраних

Для демонстрації процесу надсилання повідомлення у чаті побудовано відповідну діаграму кооперації на Рисунку 2.5. Вона відображає, як користувач вводить повідомлення, інтерфейс передає його до серверного модуля чату, який зберігає дані у базі та оновлює історію чату для відображення нового повідомлення.

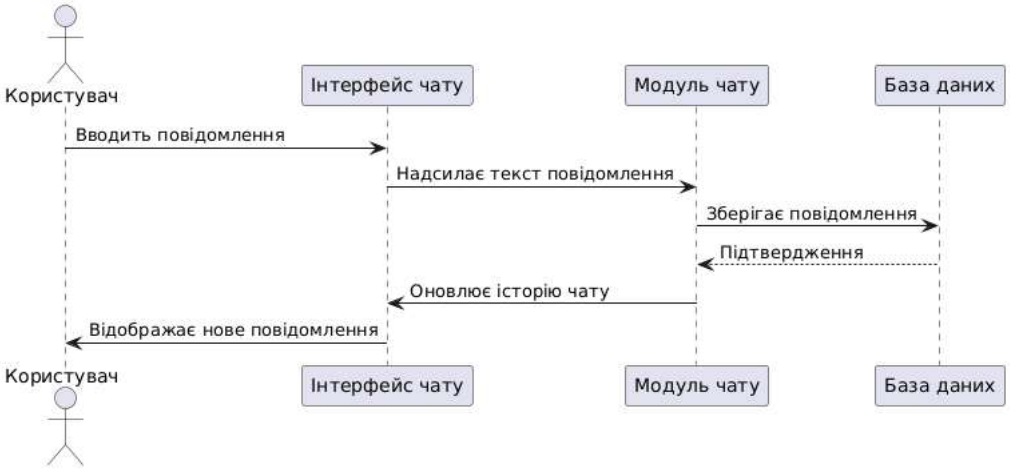


Рисунок 2.5 – Діаграма послідовності для повідомлень

Діаграми кооперації для цих функцій допоможуть наочно продемонструвати логіку взаємодії між користувачем, інтерфейсом та серверними модулями для ключових сценаріїв вашого сайту. Це підвищить зрозумілість архітектури та полегшить подальший супровід і розвиток проєкту.

2.2 Проектування структури вебсайту для пошуку тимчасового житла

Структура вебсайту ShelterUA побудована так, щоб забезпечити інтуїтивну навігацію для різних категорій користувачів: гостей, переселенців та власників житла [21]. Головна сторінка є центральною точкою входу, з якої користувач може перейти до основних розділів: пошуку житла, реєстрації/входу, інформаційних сторінок, профілю, оголошень, бронювань, чату, сповіщень тощо. Діаграму структури вебсайту можна побачити на Рисунку 2.6.

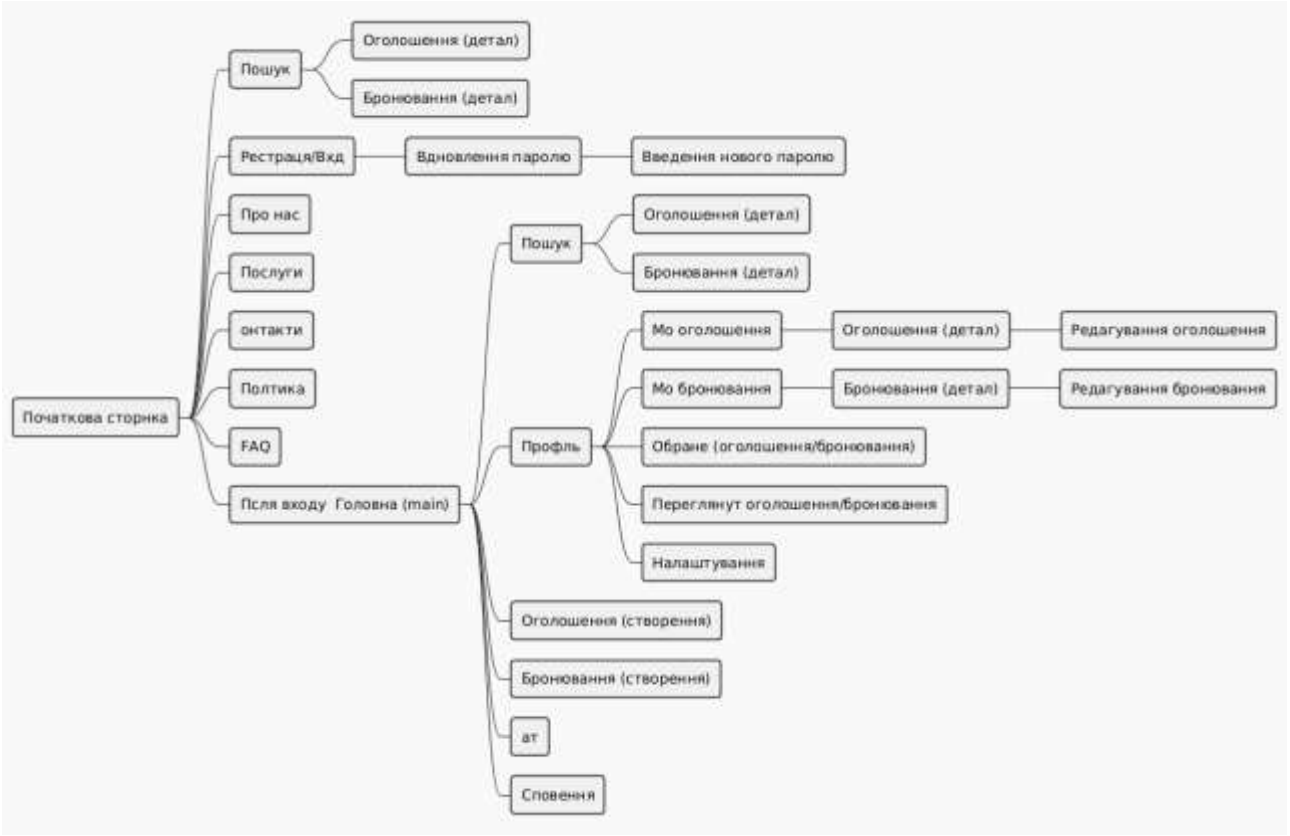


Рисунок 2.6 – Структура вебсайту для пошуку тимчасового житла

Структура сайту побудована так, щоб користувач міг інтуїтивно переходити між основними розділами залежно від свого статусу – гість чи авторизований користувач. Початкова сторінка є центральною точкою входу для всіх відвідувачів. З неї можна перейти до сторінки пошуку, де доступний перегляд детальної інформації про оголошення та запити на житло (бронювання). Також з початкової сторінки користувач має доступ до сторінок реєстрації та входу, а у випадку втрати паролю – до процедури його відновлення, яка складається з двох етапів: запиту на відновлення та введення нового паролю.

Окрім цього, з початкової сторінки можна перейти до інформаційних розділів: «Про нас», «Послуги», «Контакти», «Політика» та «FAQ». Після успішної авторизації користувач потрапляє на головну сторінку для зареєстрованих (main), яка відкриває доступ до персонального профілю, створення нових оголошень і бронювань, чату для спілкування та системи сповіщень.

У профілі користувача зібрані всі особисті дані, а також доступ до власних оголошень, бронювань, обраного, переглянутих об’єктів і налаштувань. З профілю можна перейти до детального перегляду своїх оголошень і бронювань, а також відредагувати їх у разі потреби. Така структура забезпечує зручну навігацію, швидкий доступ до основних функцій і дозволяє користувачу легко орієнтуватися на сайті незалежно від його ролі.

2.3 Визначення основних сутностей та розробка бази даних вебсайту для пошуку тимчасового житла

Визначення основних сутностей та їх характеристик є важливим кроком для структурування бази даних та забезпечення правильного функціонування вебсайту для пошуку тимчасового житла. Це дозволяє ефективно зберігати і обробляти інформацію про житло, користувачів, відгуки, бронювання та взаємодію між учасниками сервісу [22].

Розпочнемо з користувача (users). Опис сутності наведено в таблиці 2.1. Призначена для зберігання інформації про всіх користувачів вебсайту. Вона містить особисті дані, контактну інформацію, роль користувача та інші параметри, необхідні для ідентифікації та авторизації.

Таблиця 2.1 – Опис сутності (users)

Поле	Тип даних	Опис
user_id	int(11)	Унікальний ідентифікатор користувача
email	varchar(191)	Електронна пошта
password_hash	varchar(255)	Хеш паролю
first_name	varchar(100)	Ім'я
last_name	varchar(100)	Прізвище
phone	varchar(20)	Телефон

Продовження таблиці 2.1

profile_picture	varchar(255)	Шлях до фото профілю
location	varchar(255)	Місто/адреса
role	enum('user','admin')	Роль користувача
tokens	int(11)	Токени

Наступна сутність – Оголошення (listings). Містить всі оголошення про доступне житло, які розміщують власники. Вона зберігає детальну інформацію про кожен об’єкт, включаючи характеристики, адресу, ціну, фото та статус, детальніше в таблиці А.1.1.

Bookings (Запити/бронювання). Використовується для зберігання заявок на оренду або купівлю житла. Вона дозволяє фіксувати всі параметри запиту, статус, дати та інші важливі характеристики, детальніше в таблиці А.1.2.

Messages (Повідомлення) забезпечує можливість приватного спілкування між користувачами вебсайту. Вона зберігає текст повідомлень, відправника, одержувача, а також дату та час надсилання, детальніше в таблиці 2.2.

Таблиця 2.2 – Опис сутності (Messages)

Поле	**Тип даних**	**Опис**
message_id	int(11)	Унікальний ідентифікатор повідомлення
sender_id	int(11)	Відправник
receiver_id	int(11)	Одержувач
message	text	Текст повідомлення
sent_at	datetime	Дата та час надсилання
listing_id	int(11)	Оголошення (якщо є)
booking_id	int(11)	Запит (якщо є)
file_path	varchar(255)	Шлях до вкладення

SiteReviews (Відгуки про сайт) призначений для зберігання відгуків користувачів про сам вебсайт. Це дозволяє отримувати зворотний зв’язок щодо якості сервісу та його функціоналу. Детальніше про сутність наведено в таблиці 2.3.

Таблиця 2.3 – Опис сутності (SiteReviews)

Поле	Тип даних	Опис
user_id	int(11)	Унікальний ідентифікатор користувача
email	varchar(191)	Електронна пошта
password_hash	varchar(255)	Хеш паролю
first_name	varchar(100)	Ім'я
last_name	varchar(100)	Прізвище
phone	varchar(20)	Телефон
profile_picture	varchar(255)	Шлях до фото профілю
location	varchar(255)	Місто/адреса
role	enum('user','admin')	Роль користувача

UserReviews (Відгуки про користувача) містить відгуки, які користувачі залишають один про одного. Вона допомагає формувати репутацію кожного учасника сервісу. Детальніше в таблиці 2.4

Таблиця 2.4 – Опис сутності (UserReviews)

Поле	Тип даних	Опис
review_id	int(11)	Унікальний ідентифікатор
user_id	int(11)	Користувач, про якого відгук
reviewer_id	int(11)	Користувач, що залишив відгук

Продовження таблиці 2.4

rating	int(11)	Оцінка
comment	text	Текст відгуку
created_at	datetime	Дата створення

ListingReviews (Відгуки про оголошення), використовується для зберігання відгуків про конкретні оголошення. Це дозволяє оцінювати якість житла та досвід взаємодії з власником. В таблиці 2.5 наведено опис сутності.

Таблиця 2.5 – Опис сутності (ListingReviews)

Поле	**Тип даних**	**Опис**
review_id	int(11)	Унікальний ідентифікатор
listing_id	int(11)	Оголошення, про яке відгук
user_id	int(11)	Користувач, що залишив відгук
rating	int(11)	Оцінка
comment	text	Текст відгуку
created_at	datetime	Дата створення

FavoritesListings / FavoritesBookings (Обране) дозволяють користувачам додавати оголошення або запити до списку обраного для швидкого доступу в майбутньому, для бронювань буде booking_id. Опис сутностей в таблиці 2.6.

Таблиця 2.6 – Опис сутності (FavoritesListings / FavoritesBookings)

Поле	**Тип даних**	**Опис**
favorite_id	int(11)	Унікальний ідентифікатор
user_id	int(11)	Користувач
listing_id	int(11)	Оголошення
created_at	datetime	Дата додавання

ListingViews / BookingViews (Перегляди) фіксують кількість переглядів оголошень та запитів, що дозволяє аналізувати популярність об’єктів та інтерес користувачів, для бронювань замість listing_id буде booking_id. Детальніше в таблиці 2.7

Таблиця 2.7 – Опис сутності (ListingViews / BookingViews)

Поле	Тип даних	Опис
id	int(11)	Унікальний ідентифікатор
listing_id	int(11)	Оголошення
user_id	int(11)	Користувач
view_count	int(11)	Кількість переглядів
last_viewed_at	datetime	Дата останнього перегляду

Notifications (Сповіщення) використовується для зберігання системних сповіщень, які інформують користувачів про важливі події, зміни статусу оголошень чи бронювань. В таблиці 2.8 наведено опис сутності.

Окрім відображення сповіщень у профілі користувача, всі важливі сповіщення автоматично надсилаються на електронну пошту користувача за допомогою бібліотеки PHPMailer.

Таблиця 2.8 – Опис сутності (Notifications)

Поле	Тип даних	Опис
id	int(11)	Унікальний ідентифікатор
user_id	int(11)	Користувач
type	enum('published', 'premium','expire_soon', 'expired','renewed')	Тип сповіщення
listing_id	int(11)	Оголошення (якщо є)
booking_id	int(11)	Запит (якщо є)

Продовження таблиці 2.8

message	varchar(512)	Текст сповіщення
created_at	datetime	Дата створення
is_read	tinyint(1)	Прочитано/не прочитано

І password_resets (Відновлення паролю) зберігає токени для відновлення доступу до облікового запису у разі втрати паролю. Детальний опис сутності з відновлення пароля можна побачити в таблиці 2.9.

Таблиця 2.9 – Опис сутності (password_resets)

Поле	**Тип даних**	**Опис**
reset_id	int(11)	Унікальний ідентифікатор
token	varchar(64)	Токен для відновлення
expires_at	datetime	Термін дії токена
email	varchar(255)	Електронна пошта

Концептуальна модель бази даних для вебсайту пошуку тимчасового житла відображає основні сутності, їх атрибути та взаємозв’язки між ними. Вона є фундаментом для фізичної реалізації структури БД та забезпечує цілісність, узгодженість і ефективність зберігання даних. Основними сутностями є: користувачі, оголошення, запити/бронювання, повідомлення, відгуки, сповіщення, обране, перегляди, а також допоміжні таблиці для відновлення паролю [23].

Кожна сутність має власний унікальний ідентифікатор (наприклад, user_id, listing_id, booking_id), а зв’язки між ними реалізовані через зовнішні ключі. Наприклад, оголошення та запити пов’язані з користувачами, відгуки можуть стосуватися як користувачів, так і оголошень чи бронювань, а повідомлення – містити посилання на конкретне оголошення або запит. Система обраного та переглядів дозволяє фіксувати інтереси й активність користувачів, а сповіщення інформують про важливі події. На Рисунку 2.7. наведена ER-Діаграма вебсайту

для пошуку тимчасового житла. Така модель дозволяє гнучко розширювати функціонал сайту, забезпечує масштабованість і підтримку різних сценаріїв взаємодії.

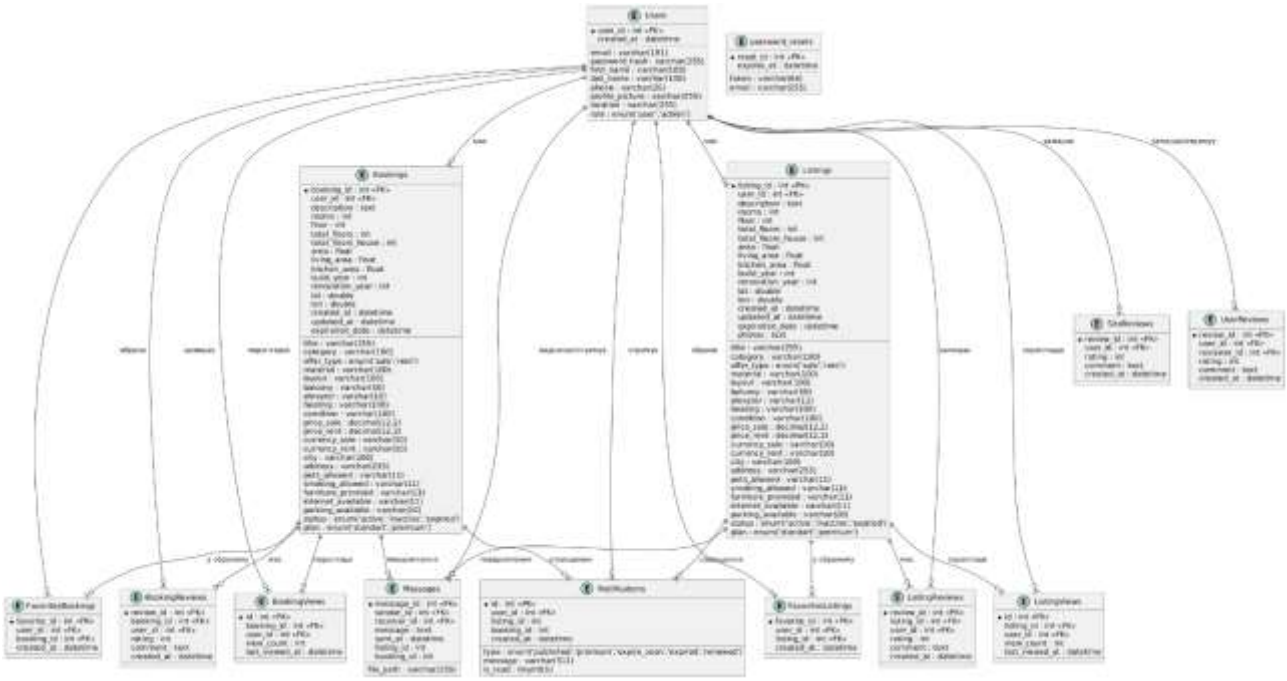


Рисунок 2.7 – ER-діаграма вебсайту для пошуку тимчасового житла

На ER-діаграмі відображено всі основні сутності, їх атрибути та ключові зв’язки. Користувач може створювати оголошення, запити, залишати відгуки, надсилати повідомлення, додавати об’єкти до обраного, переглядати їх, отримувати сповіщення та відновлювати пароль. Оголошення і запити мають свої відгуки, перегляди, повідомлення, можуть бути у списку обраного та пов’язані зі сповіщеннями. Така модель забезпечує повноцінну підтримку всіх бізнес-процесів сайту ShelterUA [24].

2.4 Проектування поведінки вебсайту для пошуку тимчасового житла

Структура файлів вебсайту ShelterUA побудована за принципами модульності, розділення відповідальності та зручності підтримки. Кожен файл або група файлів відповідає за окремий аспект функціонування платформи:

навігацію, автентифікацію, роботу з оголошеннями та бронюваннями, взаємодію з базою даних, відображення повідомлень, а також обробку дій користувача через контролери [25]. Такий підхід дозволяє легко масштабувати систему, впроваджувати новий функціонал і забезпечувати безпеку та стабільність роботи сервісу.

Навігаційні сторінки

- index.php – Початкова сторінка сайту з описом платформи, відгуками, популярними містами, навігацією до основних розділів.
- main.php – Головна сторінка для авторизованих користувачів із персоналізованим контентом, швидким доступом до профілю, оголошень, бронювань, чату та сповіщень.
- about.php, services.php, contact.php, policy.php, faq.php – Інформаційні сторінки з описом сервісу, послуг, контактною інформацією, політикою конфіденційності та відповідями на часті питання.
- search.php – Сторінка пошуку житла та запитів із фільтрами, інтерактивною картою, списком результатів.
- object.php, booking.php – Детальні сторінки оголошення та бронювання з усією інформацією, фото, відгуками, картою, кнопками для взаємодії.

Сторінки автентифікації та профілю

- auth.php, login.php – Форми для реєстрації та входу користувачів, включаючи соціальну авторизацію через Google.
- register.php, login_process.php, google_auth.php, google_callback.php – Серверна логіка для реєстрації, входу, авторизації через Google.
- reset_password.php, new_password.php – Сторінки для відновлення та зміни паролю.
- send_reset_link.php, new_password_process.php – Обробка запитів на відновлення паролю та встановлення нового паролю.

- profile.php, user.php – Особистий кабінет користувача та публічний профіль з інформацією, оголошеннями, бронюваннями, відгуками.

Керування оголошеннями та бронюваннями

- create_listing.php, edit.php, add_listing.php, update_listing.php, delete_listing.php, deactivate_listing.php, activate_listing.php, renew_listing.php, activate_premium_listing.php – Файли для створення, редагування, видалення, активації, деактивації, продовження та преміум-активації оголошень.
- request_booking.php, edit_booking.php, add_booking.php, update_booking.php, delete_booking.php, deactivate_booking.php, activate_booking.php, renew_booking.php, activate_premium_booking.php – Аналогічні файли для роботи з бронюваннями (запитами на житло).
- update_listing_coords.php, update_booking_coords.php – Оновлення координат об’єктів на карті.

4. Робота з базою даних

- db.php – Основний файл підключення до бази даних MySQL, встановлення кодування, timezone.

Файли шаблонів та допоміжні

- alert.php – Функції для встановлення та відображення повідомлень (alert) користувачу у вигляді спливаючих вікон.
- styles/alert.css, styles/styles.css, styles/styleindex.css, styles/about.css, ads.css – CSS-стилі для оформлення сторінок, повідомлень, карток, форм тощо.

6. Контролери та обробники дій (action/)

- update_profile.php, update_account.php, delete_account.php – Оновлення та видалення профілю, зміна email/паролю.
- add_favorite_listing.php, remove_favorite_listing.php, add_favorite_booking.php, remove_favorite_booking.php – Додавання та видалення оголошень і бронювань з обраного.

- submit_review.php, submit_user_review.php, submit_listing_review.php, submit_booking_review.php – Додавання відгуків про сайт, користувача, оголошення, бронювання.
- send_message.php, send_chat_message.php, get_messages.php, get_chats.php – Відправка повідомлень через форму зворотного зв’язку та вбудований чат, отримання історії листування.
- notifications.php, delete_notification.php – Відображення та видалення сповіщень користувача.
- get_listing_reviews.php, get_listing_likes.php, get_booking_likes.php, get_user_rating_and_photo.php, get_listing_views.php – Отримання відгуків, лайків, переглядів, рейтингу користувачів та об’єктів для динамічного відображення на сторінках.

Такий розподіл файлів забезпечує логічну структуру проєкту, спрощує навігацію для розробників, дозволяє швидко знаходити потрібний функціонал і підтримувати вебсайт у актуальному стані. Кожен файл має чітко визначену роль у системі, що підвищує надійність і масштабованість ShelterUA [26].

Для повного розуміння архітектури та логіки роботи вебсайту важливо не лише знати структуру бази даних і загальну організацію файлів, а й чітко уявляти, як саме відбувається взаємодія між окремими сторінками та контролерами. Кожна сторінка сайту виконує свою роль у процесі роботи сервісу: відображає дані, приймає введення користувача, ініціює запити до серверних скриптів (контролерів) та отримує відповіді для подальшого відображення чи обробки.

Щоб краще проілюструвати ці процеси, ми розглянемо функціональні діаграми для кожної ключової сторінки. На цих діаграмах показано, які контролери використовуються тією чи іншою сторінкою, як відбувається обмін даними, і яку роль відіграє кожен елемент у загальній логіці роботи сайту. Далі для кожної діаграми буде наведено детальний опис, що пояснює призначення сторінки, її основні функції та взаємодію з контролерами [27].

Розпочнемо з розгляду початкової сторінки сайту – index.php. На цій сторінці користувач бачить основну інформацію про платформу та відгуки інших користувачів. Діаграма нижче ілюструє, які саме контролери та допоміжні файли використовує index.php для своєї роботи, а також як відбувається обробка та додавання нових відгуків. На рисунку 2.8 можна побачити функціональну схему початкової сторінки.

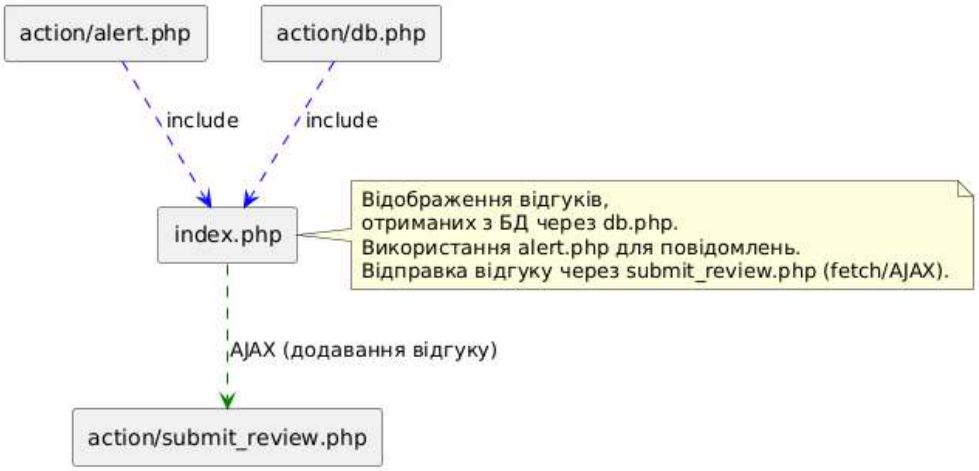


Рисунок 2.8 – Функціональна схема початкової сторінки для пошуку тимчасового житла

Дана діаграма демонструє, як працює головна сторінка сайту – index.php. При завантаженні цієї сторінки відбувається підключення двох основних допоміжних файлів: alert.php та db.php. Перший відповідає за відображення різноманітних повідомлень для користувача (наприклад, про успішне виконання дії чи помилку), а другий забезпечує підключення до бази даних і отримання відгуків, які потім відображаються на сторінці.

Коли користувач залишає відгук, його дані не надсилаються через стандартне оновлення сторінки, а відправляються асинхронно (через AJAX) до контролера submit_review.php. Цей контролер приймає дані, додає новий відгук до бази даних і повертає відповідь, яку обробляє інтерфейс сторінки. Таким чином, головна сторінка інтерактивно працює з відгуками: отримує їх з бази

даних для відображення, дозволяє додавати нові без перезавантаження сторінки та інформує користувача про результат дії через систему повідомлень.

Розглянемо функціональну схему сторінки контактів – contact.php. На цій сторінці користувач може переглянути контактну інформацію платформи та скористатися формою для надсилання повідомлення адміністрації. Діаграма на Рисунку 2.9 ілюструє, які саме контролери та допоміжні файли використовує contact.php для своєї роботи, а також як відбувається обробка повідомлень,

Рисунок 2.9 – Функціональна схема сторінки контактів для пошуку тимчасового житла

Дана діаграма демонструє, що при завантаженні сторінки contact.php підключається файл alert.php, який відповідає за відображення повідомлень для користувача (наприклад, про успішне надсилання повідомлення чи помилку). Коли користувач заповнює та надсилає форму зворотного зв’язку, дані цієї форми передаються методом POST на контролер send_message.php. Саме цей контролер обробляє повідомлення та надсилає його на пошту підтримки за допомогою бібліотеки PHPMailer, а також повертає відповідь для відображення результату дії на сторінці. Таким чином, сторінка контактів забезпечує зручний канал зв’язку між користувачем і службою підтримки платформи.

Наступним розглянемо функціональну схему сторінок автентифікації та відновлення паролю – auth.php, login.php, reset_password.php та new_password.php. Ці сторінки забезпечують реєстрацію, вхід, соціальну авторизацію, а також процес відновлення паролю для користувачів платформи. Діаграма на Рисунку 2.10 ілюструє, які саме контролери та допоміжні файли використовуються для кожної дії, а також як відбувається обробка даних на різних етапах автентифікації

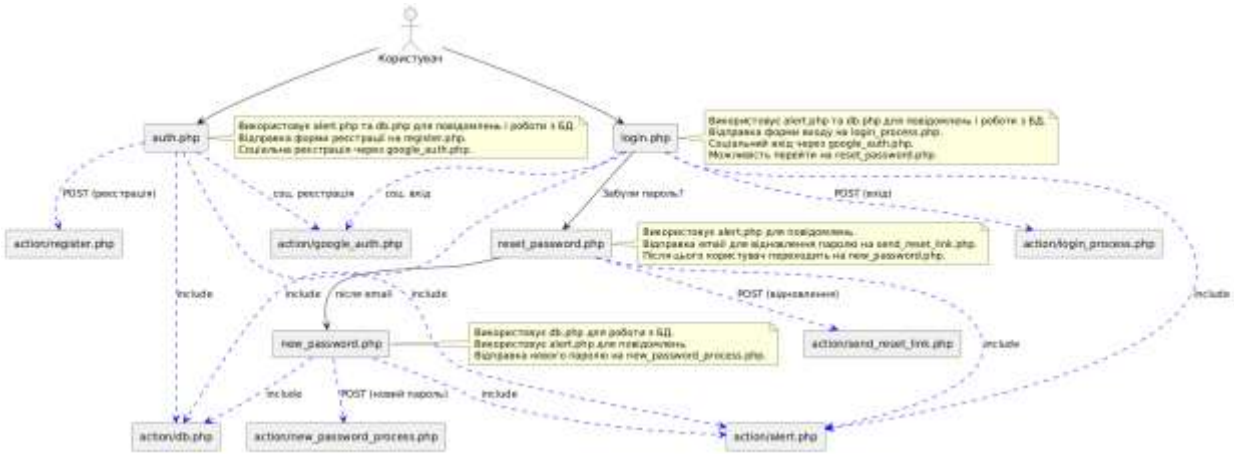


Рисунок 2.10 – Функціональна схема сторінок автентифікації для пошуку тимчасового житла

Дана діаграма демонструє, що сторінки auth.php та login.php підключають файли alert.php і db.php для роботи з повідомленнями та базою даних. На сторінці auth.php користувач може зареєструватися через стандартну форму (дані надсилаються на action/register.php) або скористатися соціальною реєстрацією через Google (action/google_auth.php). Аналогічно, на сторінці login.php можна увійти через стандартну форму (action/login_process.php) або через Google, а також перейти до відновлення паролю (reset_password.php).

Сторінка reset_password.php дозволяє користувачу надіслати запит на відновлення паролю – дані надсилаються на action/send_reset_link.php, який відправляє email з посиланням для зміни паролю. Після цього користувач переходить на сторінку new_password.php, де може встановити новий пароль. Ця

сторінка також підключає alert.php і db.php, а сам новий пароль надсилається на new_password_process.php для збереження у базі даних.

Таким чином, діаграма відображає повний цикл автентифікації та відновлення доступу: від реєстрації та входу до відновлення паролю, із зазначенням усіх контролерів, які беруть участь у цих процесах.

Розглянемо функціональну схему головної робочої сторінки користувача – main.php на Рисунку 2.11. На цій сторінці користувач може переглядати доступні оголошення та бронювання, а також додавати їх у список обраного або видаляти з нього. Діаграма нижче ілюструє, які саме контролери та допоміжні файли використовує main.php для своєї роботи, а також як відбувається обробка дій з обраним

Рисунок 2.11 – Функціональна схема головної сторінки для пошуку тимчасового житла

Діаграма демонструє, що при завантаженні сторінки main.php підключаються файли alert.php та db.php. Перший відповідає за відображення повідомлень для користувача, другий – за роботу з базою даних. Для додавання або видалення оголошень і бронювань з обраного використовуються AJAX-запити до відповідних контролерів: add_favorite_listing.php, remove_favorite_listing.php, add_favorite_booking.php та remove_favorite_booking .php. Це дозволяє користувачу оперативно керувати списком обраного без перезавантаження сторінки, а всі результати дій відображаються через систему повідомлень.

Перейдемо до сторінки обміну повідомленнями – messenger.php. На цій сторінці користувач може переглядати список своїх чатів, читати історію повідомлень та надсилати нові повідомлення іншим користувачам. Діаграма на Рисунку 2.12 ілюструє, які саме контролери використовує messenger.php для своєї роботи, а також як відбувається обробка повідомлень

Рисунок 2.12 – Функціональна схема сторінки обміну повідомленнями для пошуку тимчасового житла

Дана діаграма демонструє, що messenger.php взаємодіє з трьома основними контролерами через AJAX-запити: get_chats.php (отримання списку чатів), get_messages.php (отримання історії повідомлень) та send_chat_message.php (надсилання нового повідомлення). Для роботи з базою даних усі ці контролери підключають db.php. Така організація дозволяє користувачу в реальному часі отримувати та надсилати повідомлення без перезавантаження сторінки, забезпечуючи зручний і швидкий обмін інформацією між користувачами платформи.

Наступна сторінка схема сторінки сповіщень – notifications.php. На цій сторінці користувач може переглядати всі свої сповіщення, а також видаляти непотрібні повідомлення. Діаграма на Рисунку 2.13 ілюструє, які саме контролери використовує notifications.php для своєї роботи, а також як відбувається обробка сповіщень.

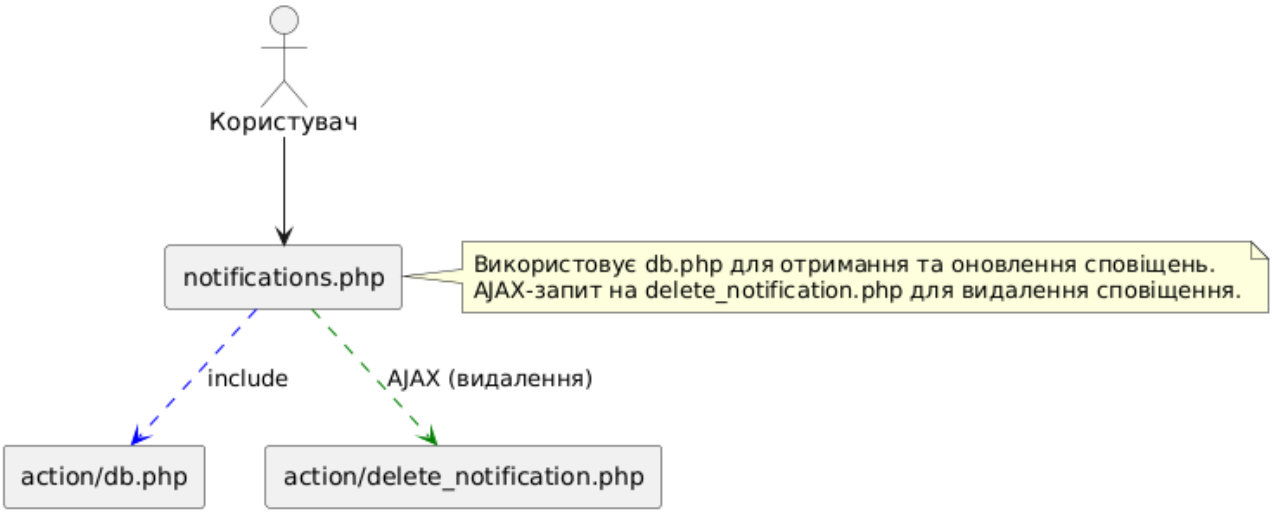


Рисунок 2.13 – Функціональна схема сторінки сповіщень для пошуку тимчасового житла

Дана діаграма демонструє, що notifications.php підключає файл db.php для отримання та оновлення списку сповіщень з бази даних. Для видалення окремих сповіщень використовується AJAX-запит до контролера delete_notification.php, який обробляє запит на видалення та оновлює список сповіщень без перезавантаження сторінки. Для надсилання email-сповіщень використовується бібліотека PHPMailer, яка забезпечує надійну доставку повідомлень про важливі події на електронну пошту користувачів.

Схема на рисунку 2.14 відображає функціональну організацію сторінки профілю – profile.php. Тут користувач отримує доступ до всіх основних налаштувань свого акаунта, а також може керувати власними оголошеннями та бронюваннями. Контролери згруповані у три стовпці: перший відповідає за зміну профілю та акаунта (update_profile.php, update_account.php, delete_account.php, logout.php), другий – за дії з оголошеннями (деактивація, активація, поновлення, преміум-активація, видалення з обраного), третій – за аналогічні операції з бронюваннями.

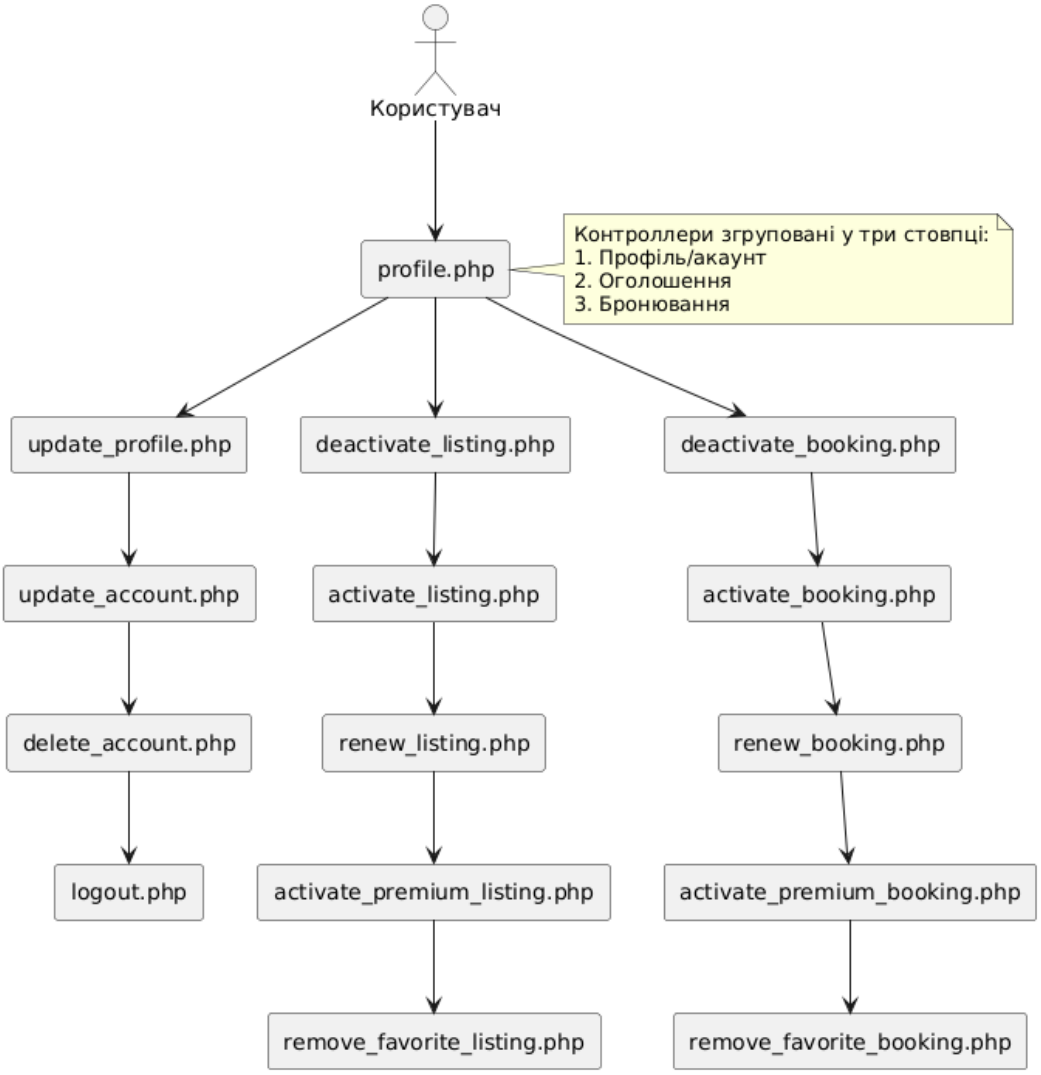


Рисунок 2.14 – Функціональна схема сторінки профілю для пошуку тимчасового житла

На цій діаграмі видно, що всі основні дії користувача з профілем, оголошеннями та бронюваннями реалізовані через окремі контролери. Наприклад, для зміни особистих даних використовується ланцюжок update_profile.php → update_account.php → delete_account.php → logout.php. Для керування оголошеннями та бронюваннями передбачені окремі гілки, які дозволяють деактивувати, активувати, поновлювати, підключати преміум-статус або видаляти з обраного відповідний об’єкт. Користувач може активувати преміум-статус для свого оголошення або бронювання, витративши 5 токенів, які він отримує за активність на платформі (наприклад, за створення нових

оголошень). Якщо токенів недостатньо, система повідомляє користувача про необхідну кількість для активації преміум-можливості.

Діаграма на рисунку 2.15 ілюструє процес створення оголошень і бронювань на платформі для пошуку тимчасового житла. Користувач може скористатися сторінкою create_listing.php для додавання нового оголошення або request_booking.php для створення запиту на бронювання. Обидві сторінки підключають допоміжні файли alert.php (для відображення повідомлень) та db.php (для роботи з базою даних).

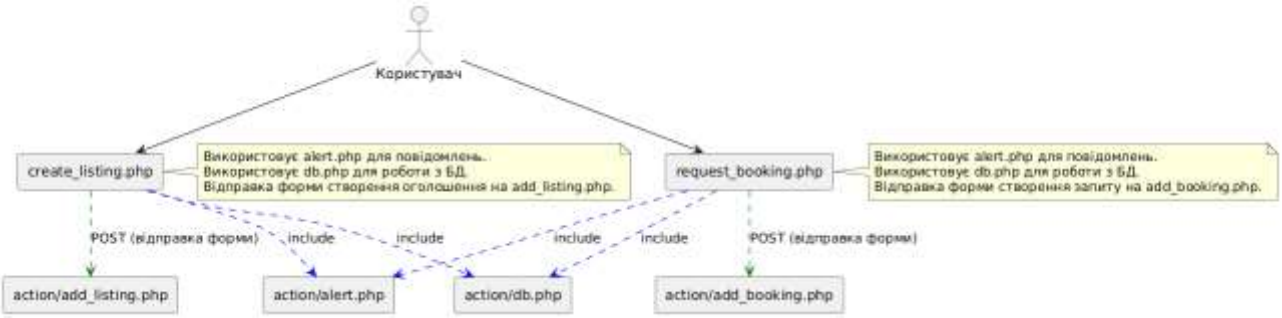


Рисунок 2.15 – Функціональна схема створення оголошень і бронювань для пошуку тимчасового житла

На сторінці create_listing.php користувач заповнює форму з інформацією про нове оголошення. Після натискання кнопки відправки дані передаються методом POST на контролер add_listing.php, який обробляє створення оголошення в базі даних. Аналогічно, на сторінці request_booking.php користувач заповнює форму для створення запиту на бронювання, а дані надсилаються на контролер add_booking.php. Усі результати дій (успіх чи помилка) відображаються через систему повідомлень, що забезпечує зручний і зрозумілий процес створення нових оголошень і бронювань на платформі.

Наступна діаграма на рисунку 2.16 відображає процес редагування оголошень і бронювань на платформі для пошуку тимчасового житла. Користувач може скористатися сторінкою edit.php для оновлення інформації про оголошення або edit_booking.php для редагування даних бронювання. Обидві

сторінки підключають допоміжні файли alert.php (для відображення повідомлень) та db.php (для отримання й оновлення даних з бази даних).

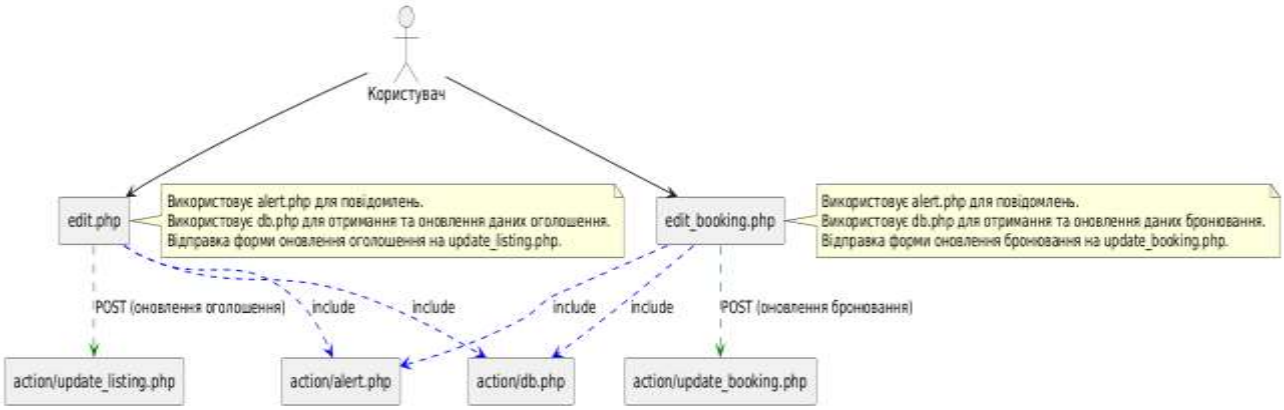


Рисунок 2.16 – Функціональна схема редагування оголошень і бронювань для пошуку тимчасового житла

На сторінці edit.php користувач змінює дані оголошення у відповідній формі. Після натискання кнопки збереження зміни передаються методом POST на контролер update_listing.php, який обробляє оновлення оголошення в базі даних. Аналогічно, на сторінці edit_booking.php користувач редагує дані бронювання, а зміни надсилаються на контролер update_booking.php. Усі результати дій (успіх чи помилка) відображаються через систему повідомлень, що забезпечує зручний і зрозумілий процес редагування оголошень і бронювань на платформі.

Функціональність пошуку на платформі реалізована через сторінку search.php, яка надає користувачам можливість знаходити та взаємодіяти з оголошеннями та бронюваннями. Ця сторінка інтегрує широкий спектр функцій, включаючи роботу з обраним, відстеження переглядів, лайків та відгуків. Усі контролери організовані у дві групи для зручного управління різними типами контенту. На рисунку 2.17 можна побачити функціональну схему сторінки.

Архітектура сторінки search.php базується на підключенні db.php для роботи з базою даних та використанні AJAX-запитів для динамічної взаємодії з контролерами.

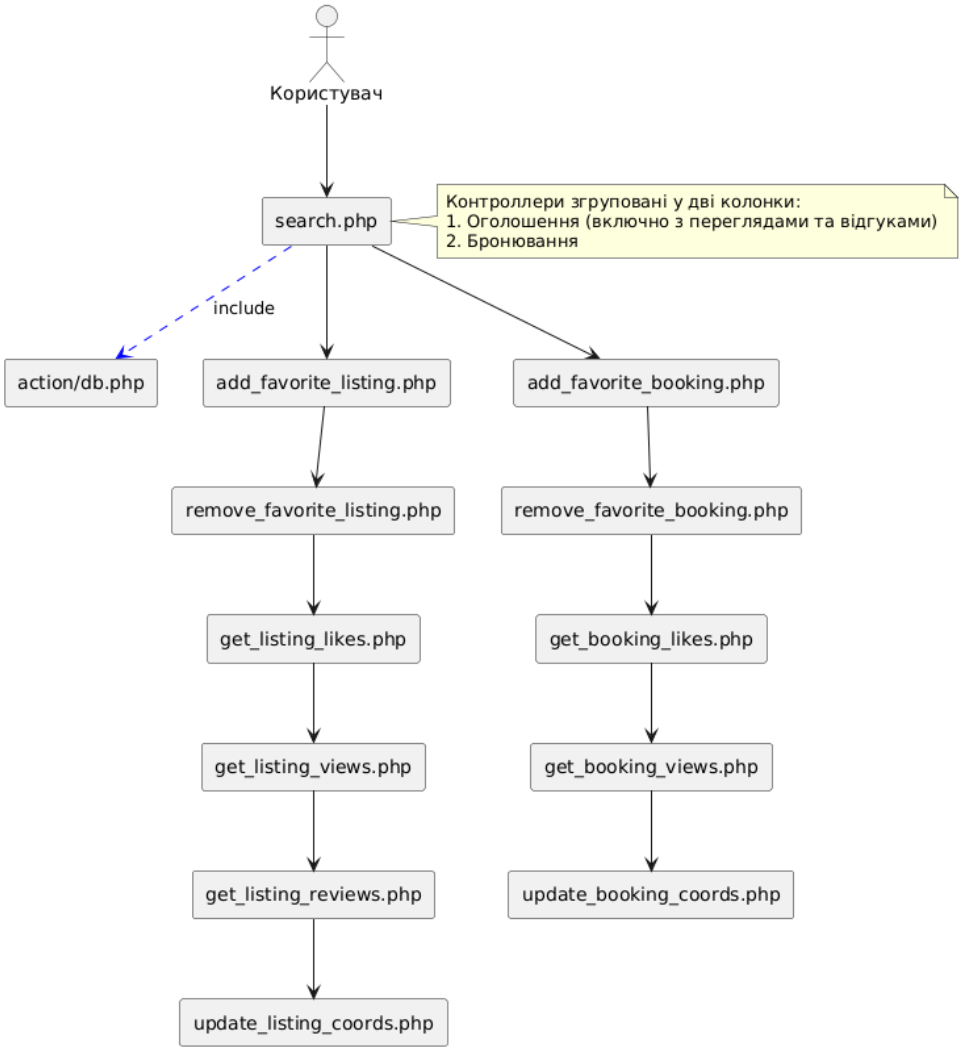


Рисунок 2.17 – Функціональна схема сторінки пошуку для пошуку тимчасового житла

Перша колонка контролерів призначена для роботи з оголошеннями: додавання/видалення з обраного (add_favorite_listing.php, remove_favorite_listing.php), отримання статистики лайків і переглядів (get_listing_likes.php, get_listing_views.php), завантаження відгуків (get_listing_reviews.php) та оновлення географічних координат (update_listing_coords.php). Друга колонка містить аналогічні контролери для бронювань, що забезпечує симетричну функціональність та уніфікований підхід до обробки різних типів контенту на платформі.

Сторінки детального перегляду є ключовими елементами платформи, що дозволяють користувачам отримати повну інформацію про конкретне

оголошення або бронювання. Object.php та booking.php забезпечують всебічну взаємодію з контентом, включаючи можливість додавання в обране, перегляд статистики, читання та додавання відгуків, а також спілкування з автором через внутрішній чат. На Рисунку 2.18 можна детальніше ознайомитися з функціональною схемою сторінки для перегляду оголошення чи бронювання.

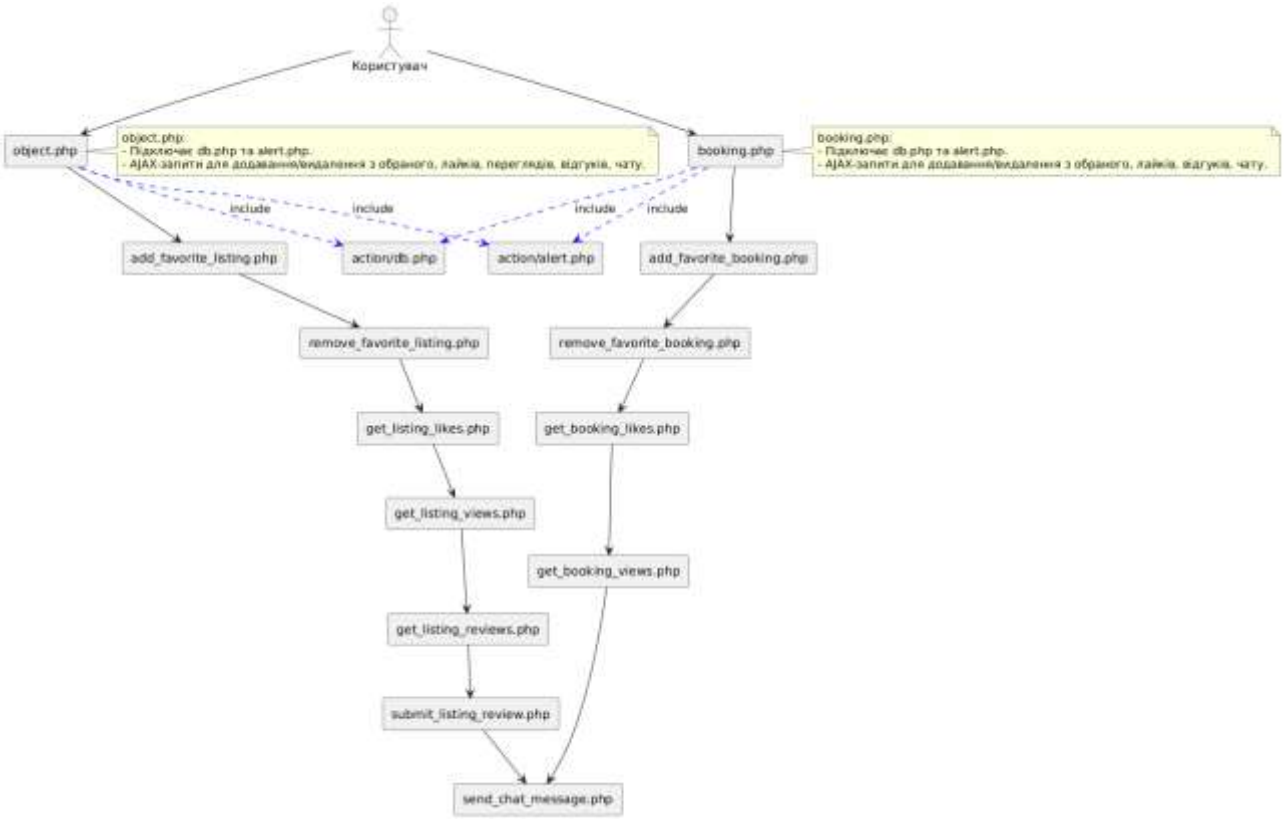


Рисунок 2.18 – Функціональна схема сторінок детального перегляду оголошень та бронювань для пошуку тимчасового житла

Обидві сторінки мають спільну архітектуру та підключають базові модулі db.php і alert.php для роботи з даними та повідомленнями. Сторінка object.php використовує ланцюжок контролерів для оголошень: від додавання/видалення з обраного до отримання лайків, переглядів, відгуків, їх додавання та завершується можливістю надіслати повідомлення автору через send_chat_message.php. Сторінка booking.php реалізує подібну функціональність для бронювань, але з урахуванням специфіки цього типу контенту. Така організація забезпечує

повноцінну взаємодію користувача з обраним об'єктом та можливість отримати всю необхідну інформацію для прийняття рішення.

Також доцільним є побудувати діаграма станів для оголошень та бронювань Вона є важливим інструментом для розуміння життєвого циклу оголошень та бронювань на платформі пошуку тимчасового житла. Вона демонструє всі можливі стани, через які проходить контент від моменту створення до його остаточного видалення, а також умови переходів між цими станами [28]. Візуалізація на Рисунку 2.19 допомагає зрозуміти логіку роботи системи та процеси модерації контенту .

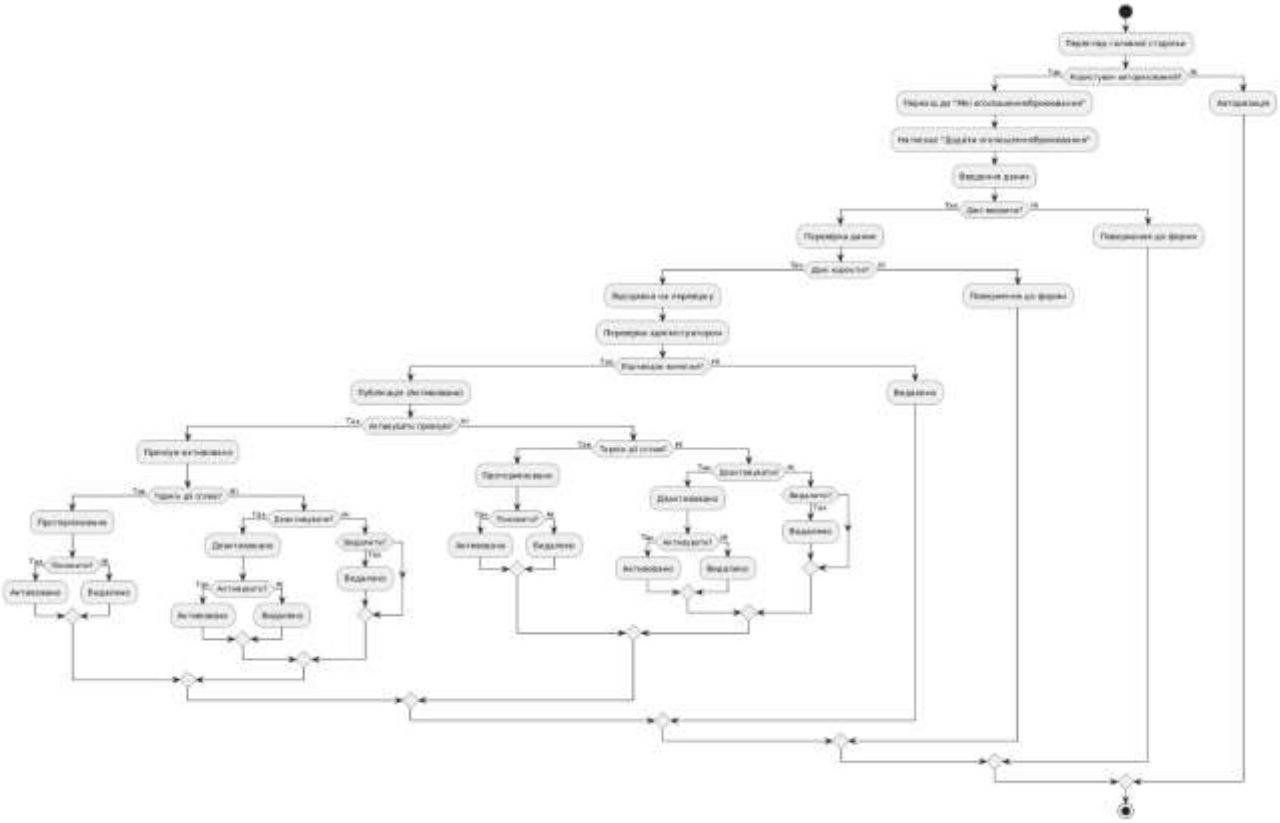


Рисунок 2.19 – Діаграма станів оголошень/бронювань для платформи пошуку тимчасового житла

Діаграма відображає повний життєвий цикл оголошення або бронювання, починаючи з перегляду головної сторінки користувачем. Процес розпочинається з перевірки авторизації – неавторизованим користувачам необхідно пройти автентифікацію перед створенням контенту. Після успішного входу користувач

переходить до створення оголошення або бронювання, заповнює необхідні дані та відправляє їх на перевірку адміністратором.

Центральною частиною діаграми є стан "Публікація (Активовано)", з якого можливі різні переходи залежно від дій користувача або системних процесів. Користувач може активувати преміум-статус для підвищення видимості свого контенту, деактивувати оголошення при необхідності або видалити його повністю. Система автоматично переводить контент у стан "Протерміновано" після закінчення терміну дії, але надає можливість поновлення.

Діаграма також показує, що з будь-якого активного стану можна повернутися до попереднього або перейти до видалення, що забезпечує гнучкість управління контентом. Такий підхід гарантує користувачам повний контроль над своїми оголошеннями та бронюваннями, а платформі – ефективну систему модерації та управління життєвим циклом контенту [29].

2.5. Діаграма класів вебсайту для пошуку тимчасового житла

Діаграма класів є фундаментальним елементом об'єктно-орієнтованого проектування, що надає статичний погляд на структуру системи. Вона демонструє основні сутності (класи), їх атрибути, методи та взаємозв'язки між ними. У контексті вебсайту для пошуку тимчасового житла – ShelterUA діаграма класів допомагає візуалізувати архітектуру системи, показуючи як різні компоненти взаємодіють між собою для забезпечення функціональності пошуку та оренди тимчасового житла [30].

Діаграма класів платформи ShelterUA (рисунок Б1.2) структурована у три основні рівні архітектури: рівень даних (моделі), рівень логіки (контролери) та рівень утиліт.

Рівень даних включає основні бізнес-сутності системи. Центральним класом є User, який містить інформацію про користувачів платформи та їх ролі в системі. Класи Listing та Booking представляють два основні типи контенту – оголошення про оренду житла та запити на бронювання відповідно. Ці класи

мають схожу структуру полів, що відображає їх функціональну подібність, але різні бізнес-цілі.

Система відгуків реалізована через спеціалізовані класи: ListingReview, BookingReview, UserReview та SiteReview, що дозволяє користувачам оцінювати різні аспекти платформи. Функціональність обраного контенту забезпечується класами FavoritesListing та FavoritesBooking, а відстеження активності користувачів – через ListingView та BookingView.

Комунікаційна підсистема представлена класом Message, який забезпечує обмін повідомленнями між користувачами з можливістю прикріплення файлів. Клас Notification відповідає за системні сповіщення користувачів про важливі події.

Рівень контролерів представлений PHP-класами, що відповідають за обробку HTTP-запитів та бізнес-логіку. Кожен контролер спеціалізується на конкретній функціональній області: AuthController керує автентифікацією, ProfileController – профілями користувачів, ListingController та BookingController – відповідним контентом, MessengerController – системою повідомлень.

Рівень утиліт містить допоміжні класи загального призначення: DatabaseConnection для роботи з базою даних, AlertSystem для відображення повідомлень користувачу, FileUploader для завантаження файлів та EmailService для розсилки електронної пошти.

Взаємозв'язки між класами відображають реальні бізнес-процеси: користувач може створювати оголошення та бронювання, залишати відгуки, додавати контент в обране, обмінюватися повідомленнями з іншими користувачами. Контролери використовують моделі даних для реалізації функціональності, а утилітарні класи надають технічну підтримку.

Діаграма класів демонструє добре структуровану архітектуру платформи ShelterUA з чітким розділенням відповідальності між компонентами.

Особливістю архітектури є симетричність у роботі з оголошеннями та бронюваннями – для кожного типу контенту реалізований повний набір функцій (перегляд, редагування, додавання в обране, відгуки), що створює уніфіковий

користувацький досвід. Система відгуків та оцінок забезпечує довіру між користувачами, а вбудований месенджер сприяє ефективній комунікації [31].

2.5 Проектування структури каталогів вебсайту для пошуку тимчасового житла

Структура каталогів сайту ShelterUA спроектована для чіткого розділення функціональних зон, забезпечення безпеки, масштабованості та зручності підтримки коду, детальніше на Рисунку 2.20.

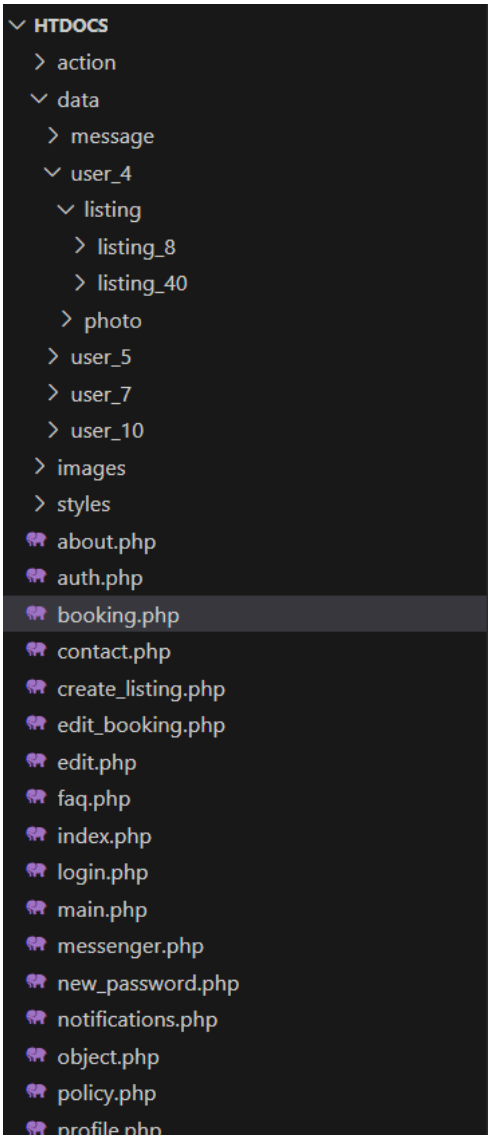


Рисунок 2.20 – Структура каталогів вебсайту для пошуку тимчасового житла

В корені проекту розташовані основні файли сторінок (наприклад, index.php, main.php, login.php, profile.php), що відповідають за відображення ключових розділів платформи. Всі дії, пов’язані з обробкою запитів, роботою з базою даних, авторизацією, надсиланням повідомлень чи іншою бізнес-логікою, винесені у підкаталог action. Це дозволяє централізовано керувати всіма серверними сценаріями, спрощує їхню підтримку та підвищує безпеку, оскільки прямий доступ до цих скриптів ззовні обмежується лише необхідними випадками [32].

Для зберігання користувацьких даних, таких як фотографії профілів, оголошень чи повідомлень, використовується каталог data. Всередині нього створюються окремі підкаталоги для кожного користувача (user_{id}), а також для повідомлень (message). Це дозволяє ізолювати файли різних користувачів, уникати конфліктів і забезпечувати швидкий доступ до потрібних ресурсів. Фотографії оголошень і профілів зберігаються у відповідних підпапках, що спрощує їхню організацію та резервне копіювання.

Статичні ресурси, такі як стилі CSS, розміщуються у папці styles, а зображення – у каталозі images. Це відповідає загальноприйнятим практикам веб-розробки, дозволяє ефективно кешувати ресурси браузером і легко підключати їх до різних сторінок сайту. Така організація також полегшує командну роботу, оскільки кожен розробник швидко знаходить потрібні файли за їхнім призначенням.

Загалом, структура каталогів ShelterUA забезпечує логічне розділення коду, статичних ресурсів і користувацьких даних, що сприяє зручності розробки, масштабуванню проекту та підтримці високого рівня безпеки.

2.6 Висновок до другого розділу

У другому розділі було детально розглянуто процес проектування та розробки вебсайту ShelterUA для пошуку тимчасового житла з використанням MySQL та PHP. Проведено моделювання архітектури системи, що базується на

трирівневій структурі, яка забезпечує чітке розділення обов’язків між клієнтською частиною, серверною логікою та рівнем даних. Такий підхід гарантує масштабованість, безпеку та гнучкість у подальшому розвитку платформи.

Було спроектовано інтуїтивну структуру вебсайту, яка враховує потреби різних категорій користувачів і забезпечує зручну навігацію між основними розділами. Особливу увагу приділено визначенню основних сутностей та розробці структури бази даних, що дозволяє ефективно зберігати й обробляти інформацію про користувачів, оголошення, бронювання, повідомлення, відгуки, сповіщення та інші важливі об’єкти системи.

Важливим етапом стало проектування поведінки вебсайту, розробка функціональних діаграм для ключових сторінок і контролерів, що забезпечує прозорість логіки роботи сервісу та спрощує його підтримку. Окремо було розглянуто структуру каталогів і файлів, яка відповідає принципам модульності, безпеки та зручності командної роботи.

Загалом, результати проектування, описані у цьому розділі, створюють надійну основу для реалізації сучасного, функціонального та масштабованого вебсайту ShelterUA, що відповідає актуальним вимогам користувачів і забезпечує ефективний пошук тимчасового житла.

РОЗДІЛ 3. ВСТАНОВЛЕННЯ, НАЛАШТУВАННЯ, ЕКСПЛУАТАЦІЯ І СУПРОВІД ВЕБСАЙТУ ДЛЯ ПОШУКУ ТИМЧАСОВІОГО ЖИТЛА

3.1 Вибір хостинг-провайдера, його налаштування та тестування для вебсайту для пошуку тимчасового житла

У процесі розробки вебсайту для пошуку тимчасового житла початкове тестування та налагодження виконувалося на локальному сервері OpenServer. Це дозволило швидко перевіряти функціональність сайту, вносити зміни та працювати без підключення до інтернету [16]. Однак для забезпечення доступу до сайту з будь-якої точки світу та тестування в реальних умовах виникла необхідність вибору зовнішнього хостингу.

Для порівняння у таблиці 3.1 було розглянуто кілька популярних хостинг-провайдерів, які підтримують PHP та MySQL:

Таблиця 3.1 – Порівняння хостинг-провайдерів

Хостинг	**Тип**	**Дисковий простір**	**MySQL**	**SSL**	**Вартість**
InfinityFree	Безкошт.	Необмежено*	Так	Так	0 грн/міс
Hostinger	Платний	50 ГБ+	Так	Так	від 49 грн/міс
FreeHosting.com	Безкошт.	10 ГБ	Так	Так	0 грн/міс
Ukraine.com.ua	Платний	5-50 ГБ	Так	Так	від 60 грн/міс

Для розгортання сайту було обрано InfinityFree як оптимальний варіант для тестового та навчального проєкту. Цей хостинг є повністю безкоштовним, дозволяє розміщувати як файли сайту, так і базу даних MySQL, підтримує PHP, надає безкоштовний SSL-сертифікат для захищеного з’єднання, не містить сторонньої реклами. InfinityFree дозволяє швидко розгорнути сайт, протестувати

його роботу в реальних умовах, забезпечити доступність для користувачів і працювати з базою даних через phpMyAdmin. Серед обмежень – політика справедливого використання ресурсів (CPU, RAM), але для невеликого або навчального проєкту цього цілком достатньо [33].

Для коректної роботи сайту на хостингу необхідно виконати низку налаштувань. Спочатку потрібно завантажити всі файли сайту у кореневу директорію хостингу htdocs Далі через панель керування InfinityFree слід створити базу даних, після чого імпортувати до неї структуру та початкові дані за допомогою phpMyAdmin. У файлі підключення до бази даних (db.php) потрібно вказати коректні параметри: адресу сервера, ім’я користувача, пароль та назву бази даних, отримані у панелі хостингу. Приклад підключення можна побачити у лістингу 3.1

Лістинг 3.1 – db.php

```
<?php
ini_set('display_errors', 1);
error_reporting(E_ALL);
$servername = "сервер";
$username = "імя";
$password = "пароль";
$dbname = "бд";
$conn = new mysqli($servername, $username, $password, $dbname);
$conn->set_charset('utf8mb4');
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
date_default_timezone_set('Europe/Kiev');
return $conn;
?>
```

Цей файл потрібно підключати на всіх сторінках, які працюють з базою даних, за допомогою require_once 'db.php';.

Важливо також перевірити права доступу до файлів і папок: для файлів рекомендується встановити права 644, а для папок – 755, щоб забезпечити безпеку та доступність ресурсу. Для захисту даних користувачів необхідно увімкнути SSL-сертифікат у налаштуваннях домену, що дозволить використовувати захищене з'єднання https. Слід переконатися, що на хостингу використовується актуальна версія PHP (рекомендовано 7.4 або вище), а також налаштувати часовий пояс відповідно до вашого регіону, Europe/Kiev.

Додатково потрібно впевнитися, що всі шляхи до файлів і зображень у коді відповідають реальній структурі директорій на хостингу, щоб уникнути помилок при завантаженні ресурсів. За необхідності можна додати файл .htaccess для налаштування правил URL, редиректів або обмеження доступу до окремих директорій. Виконання цих налаштувань забезпечить стабільну та безпечну роботу сайту на обраному хостингу.

Таким чином, InfinityFree став оптимальним вибором для першого етапу публічного розміщення вебсайту, оскільки дозволяє повноцінно розміщувати як сам сайт, так і базу даних, забезпечуючи всі необхідні можливості для роботи та тестування.

3.2 Валідація та тестування вебсайту для пошуку тимчасового житла

Валідація та тестування вебсайту для пошуку тимчасового житла є важливими етапами розробки, оскільки дозволяють виявити та усунути помилки, підвищити якість і безпеку ресурсу. Валідація коду допомагає переконатися у відповідності HTML, CSS та PHP сучасним стандартам. Для цього можна скористатися такими онлайн-інструментами, як W3C Validator для перевірки HTML і CSS, а також PHP Code Checker для аналізу PHP-коду.

У межах цього етапу буде здійснена перевірка коректності коду сайту, тестування кросбраузерності для забезпечення однакового відображення у різних браузерах, а також перевірка адаптивності інтерфейсу на різних

пристроях – комп’ютерах, планшетах і смартфонах. Це гарантує зручність та доступність платформи для всіх користувачів.

3.2.1 Валідація коду вебсайту для пошуку тимчасового житла

Валідація коду є важливим етапом розробки сучасного вебсайту, оскільки дозволяє переконатися у відповідності розмітки та стилів міжнародним стандартам, а також підвищити якість і стабільність роботи ресурсу. Для перевірки HTML- та CSS-коду було використано сервіс W3C Validator, який автоматично аналізує сторінки на наявність синтаксичних помилок, невідповідностей стандартам та потенційних проблем із відображенням. У рамках валідації були перевірені основні сторінки сайту: головна, сторінка пошуку, профіль користувача, сторінки оголошень та бронювань. Результати перевірки наведено на рисунку 3.1, де видно, що основні сторінки відповідають стандартам W3C і не містять критичних помилок [34].

Рисунок 3.1 – Перевірка помилок за допомогою сервісу W3C Validator

Для аналізу серверної частини сайту було використано онлайн-інструмент PHP Code Checker, який дозволяє перевірити PHP-код на наявність синтаксичних помилок, попередити про використання застарілих конструкцій та підвищити безпеку коду [35]. Основні PHP-файли сайту були завантажені для перевірки, результати якої також наведено на рисунку 3.2.

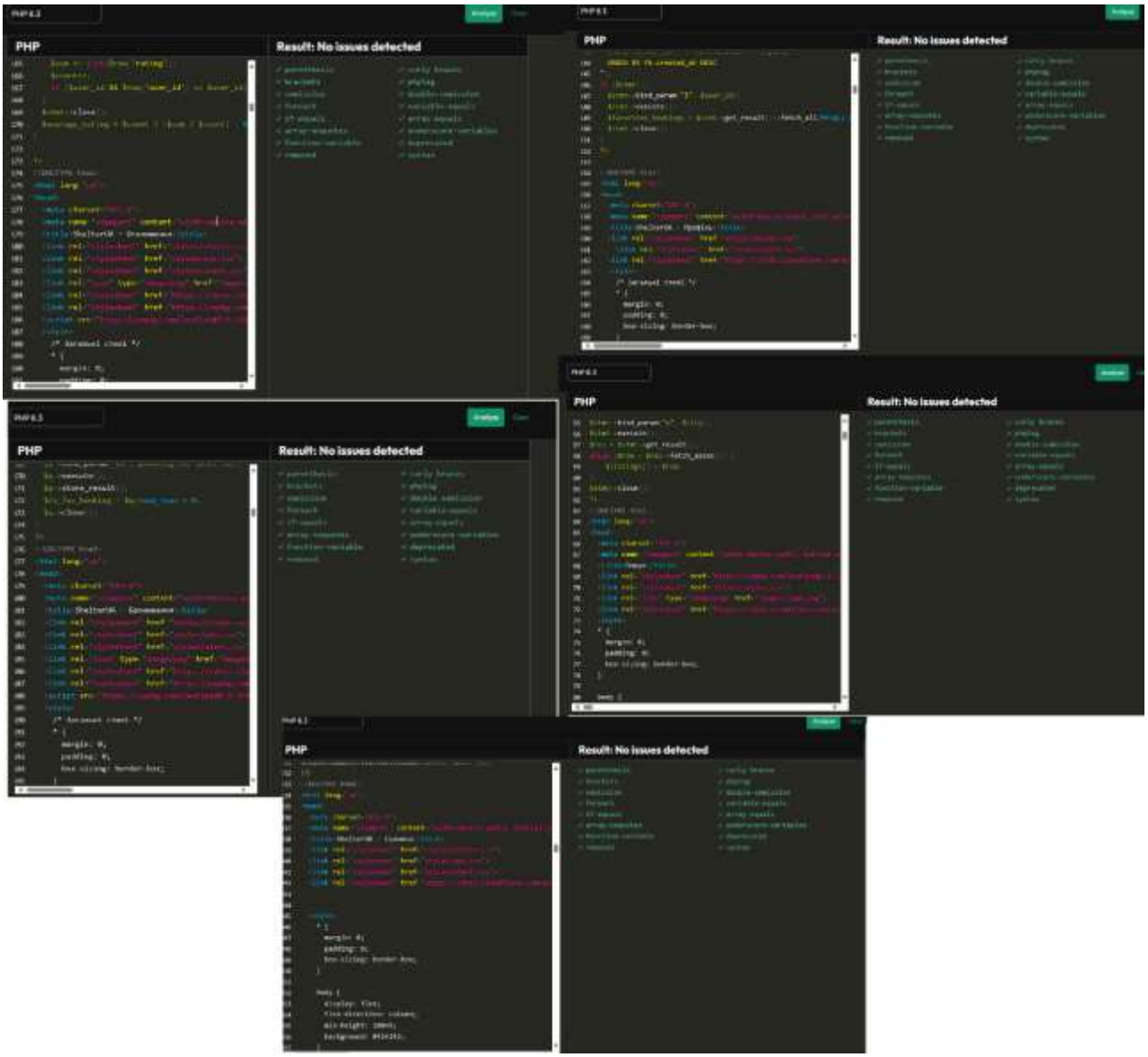


Рисунок 3.2 – Перевірка помилок за допомогою сервісу PHP Code Checker

Перевірка показала відсутність критичних помилок у коді, що свідчить про його якість та відповідність сучасним вимогам.

3.2.2 Кросбраузерність вебсайту дял пошуку тимчасового житла

Кросбраузерність вебсайту для пошуку тимчасового житла є важливою характеристикою, що забезпечує коректне відображення та функціонування ресурсу у різних сучасних браузерах. Для перевірки кросбраузерності основні сторінки сайту були протестовані у Google Chrome та Microsoft Edge. В результаті тестування встановлено, що всі ключові елементи інтерфейсу відображаються коректно та працюють однаково у всіх зазначених браузерах.

На рисунку 3.3 наведено приклади відображення головної сторінки сайту у різних браузерах.

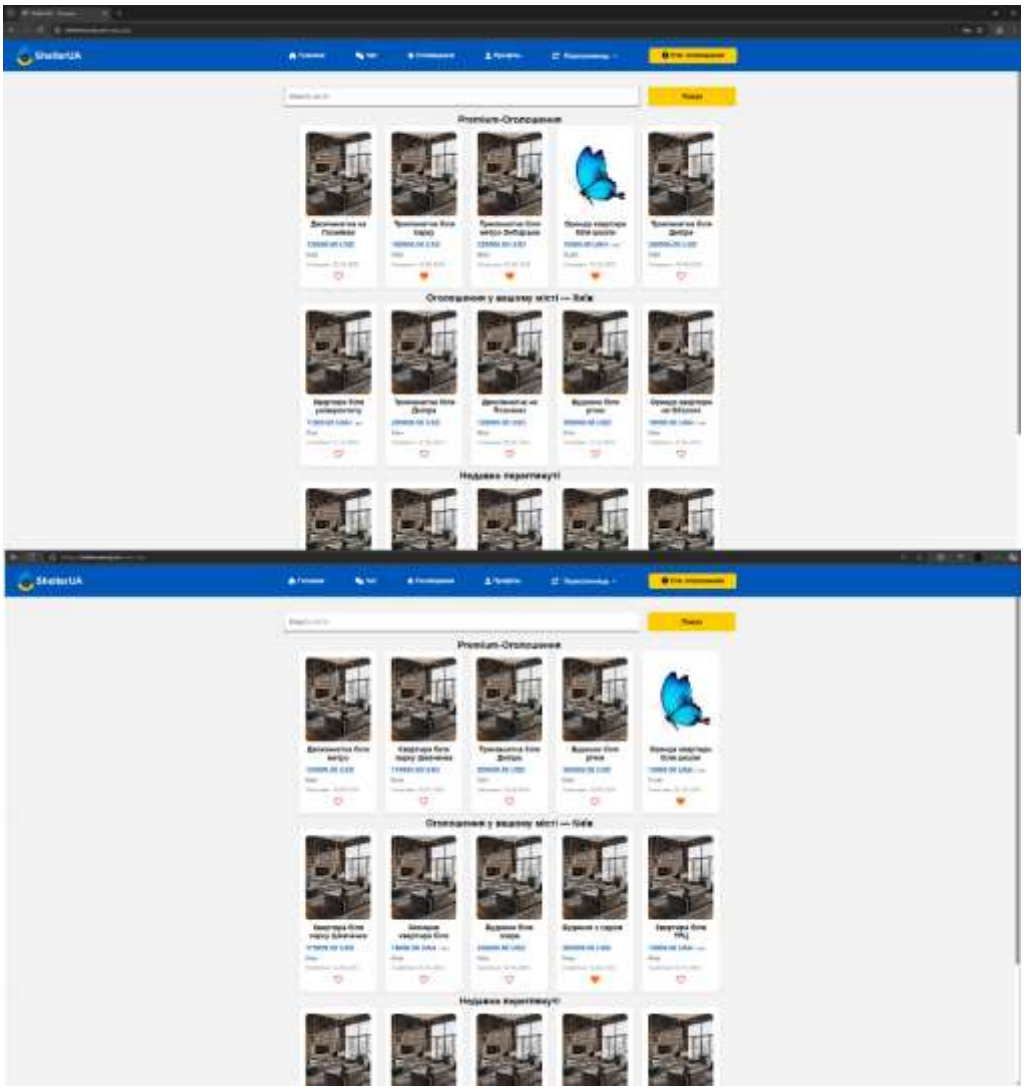


Рисунок 3.3 – Порівняння відображення головної сторінки вебсайту для пошуку тимчасового житла

На рисунку 3.4 наведено приклади відображення пошукової сторінки сайту у різних браузерах.

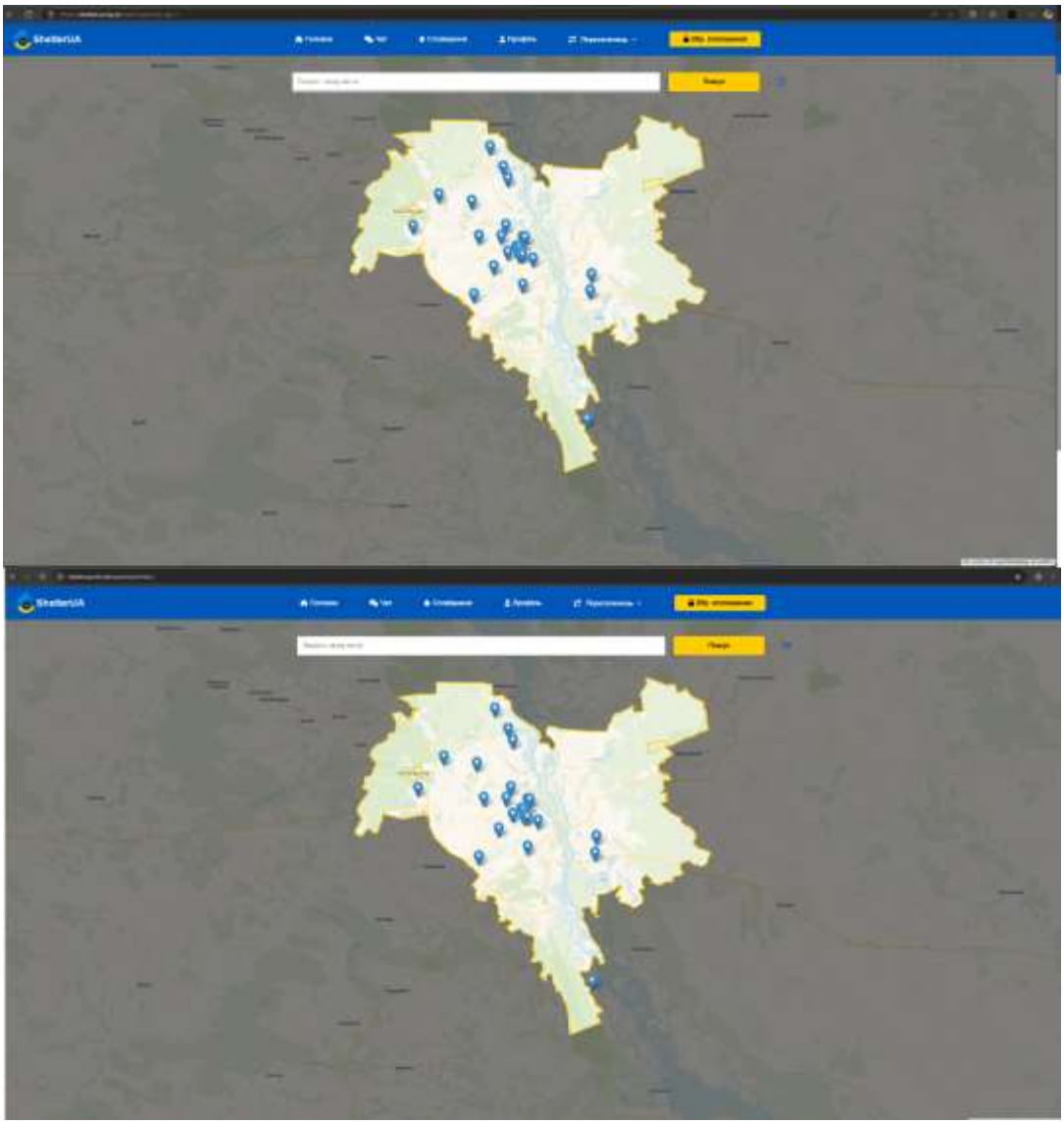


Рисунок 3.4 – Порівняння відображення пошукової сторінки вебсайту для пошуку тимчасового житла

І на останок на рисунку 3.5 наведено приклади відображення сторінки оголошення сайту у різних браузерах.

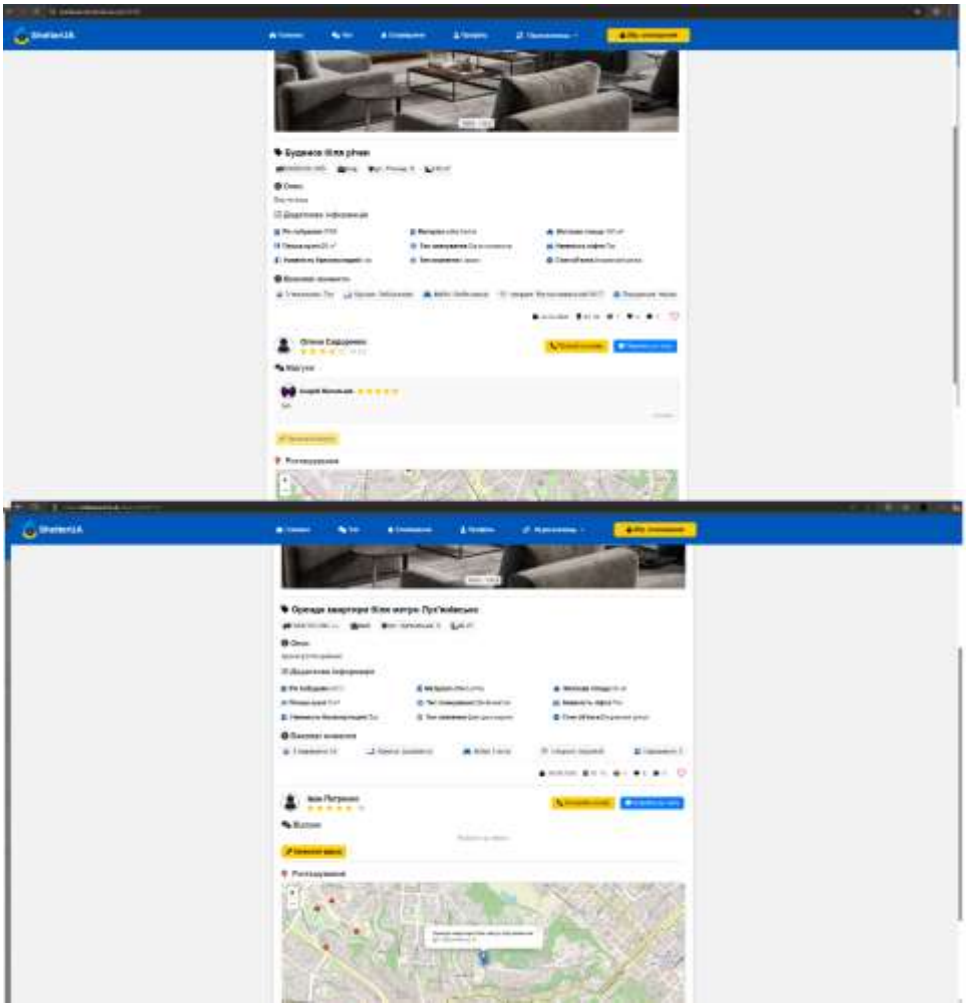


Рисунок 3.5 – Порівняння відображення сторінки оголошення вебсайту для пошуку тимчасового житла

За аналогічним принципом було проведено тестування інших основних сторінок вебсайту. У всіх випадках інтерфейс коректно відображався у різних браузерах, а функціональні елементи працювали відповідно до вимог технічного завдання. Жодних критичних розбіжностей у відображенні чи роботі сайту виявлено не було, що підтверджує високий рівень кросбраузерної сумісності розробленого ресурсу.

3.2.3 Адаптивність вебсайту для пошуку тимчасового житла

Адаптивність вебсайту для пошуку тимчасового житла є ключовою вимогою сучасної розробки, оскільки значна частина користувачів взаємодіє з

платформою з мобільних пристроїв та планшетів. Для забезпечення коректного відображення і зручності користування на різних екранах було застосовано адаптивну верстку з використанням медіа-запитів CSS. Основні сторінки сайту, були протестовані на різних пристроях і в емуляторах мобільних браузерів. Результати тестування, наведені на рисунку 3.6 [36].

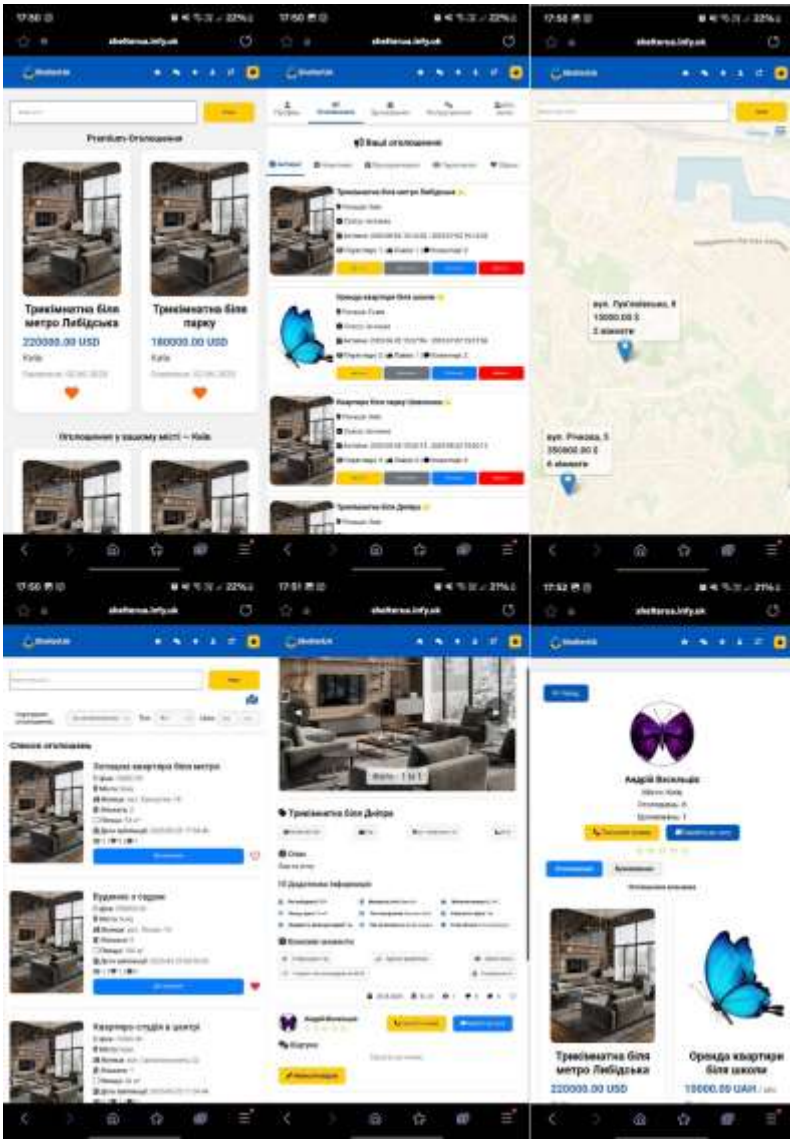


Рисунок 3.6 – Результат тестування адаптивності вебсайту для пошуку тимчасового житла

Результати засвідчують, що всі елементи інтерфейсу – меню, форми, картки оголошень, кнопки та повідомлення – коректно масштабується та залишаються зручними для взаємодії незалежно від розміру екрана. Таким

чином, сайт є повністю адаптивним і забезпечує позитивний користувацький досвід як на комп’ютерах, так і на мобільних пристроях.

3.3 Експлуатація вебсайту для пошуту тимчасового житла

Експлуатація вебсайту для пошуку тимчасового житла охоплює комплекс заходів, спрямованих на забезпечення стабільної, безпечної та ефективної роботи ресурсу після його впровадження. На цьому етапі важливо не лише підтримувати працездатність платформи, а й забезпечувати її подальший розвиток, оновлення та адаптацію до потреб користувачів. У цьому підрозділі буде детально розглянуто всі основні етапи роботи з вебсайтом [37].

Початкова сторінка вебсайту ShelterUA виконує роль головного навігаційного вузла та першого контакту користувача із сервісом, більш детально можна побачити на Рисунку 3.7. Вона розроблена таким чином, щоб забезпечити доступ до всіх основних функцій і розділів ресурсу, а також створити позитивне перше враження про платформу. На цій сторінці розміщено логотип та назву сервісу, головне меню з посиланнями на ключові розділи, а також інформаційні блоки, які знайомлять відвідувача з можливостями сайту.

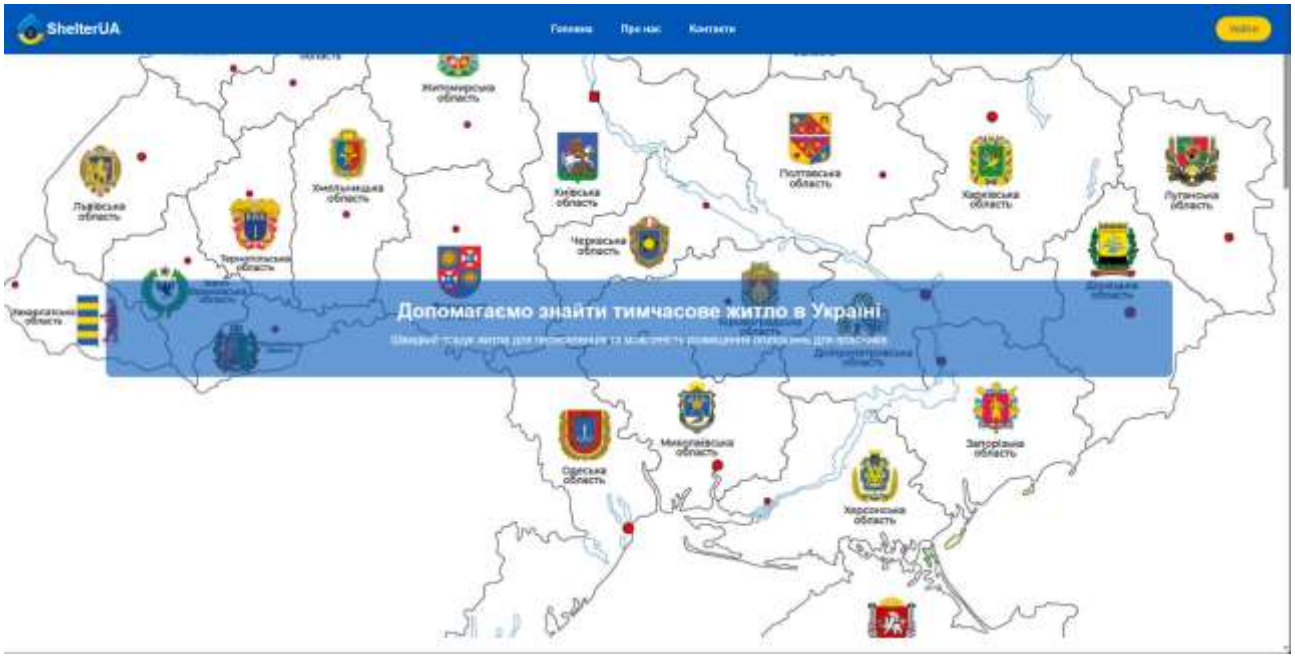


Рисунок 3.7 – Початкова сторінка вебсайту для пошуку тимчасового житла

На початковій сторінці для користувача окремо виділені посилання на сторінки реєстрації та входу, що дозволяє швидко створити новий акаунт або авторизуватися для отримання розширених можливостей. Крім того, у верхньому або нижньому меню розміщені посилання на інформаційні сторінки: «Про нас», «Послуги», «Контакти», «Політика конфіденційності» та «FAQ», які можна побачити у Додатку Б . Це забезпечує користувачам доступ до всієї необхідної довідкової інформації та підвищує довіру до ресурсу.

Важливою особливістю початкової сторінки є можливість залишати відгук про сайт. Для цього реалізовано спеціальну форму яку можна побачити на Рисунку 3.8, де користувач може вказати своє ім’я, електронну пошту, оцінку та текст відгуку.

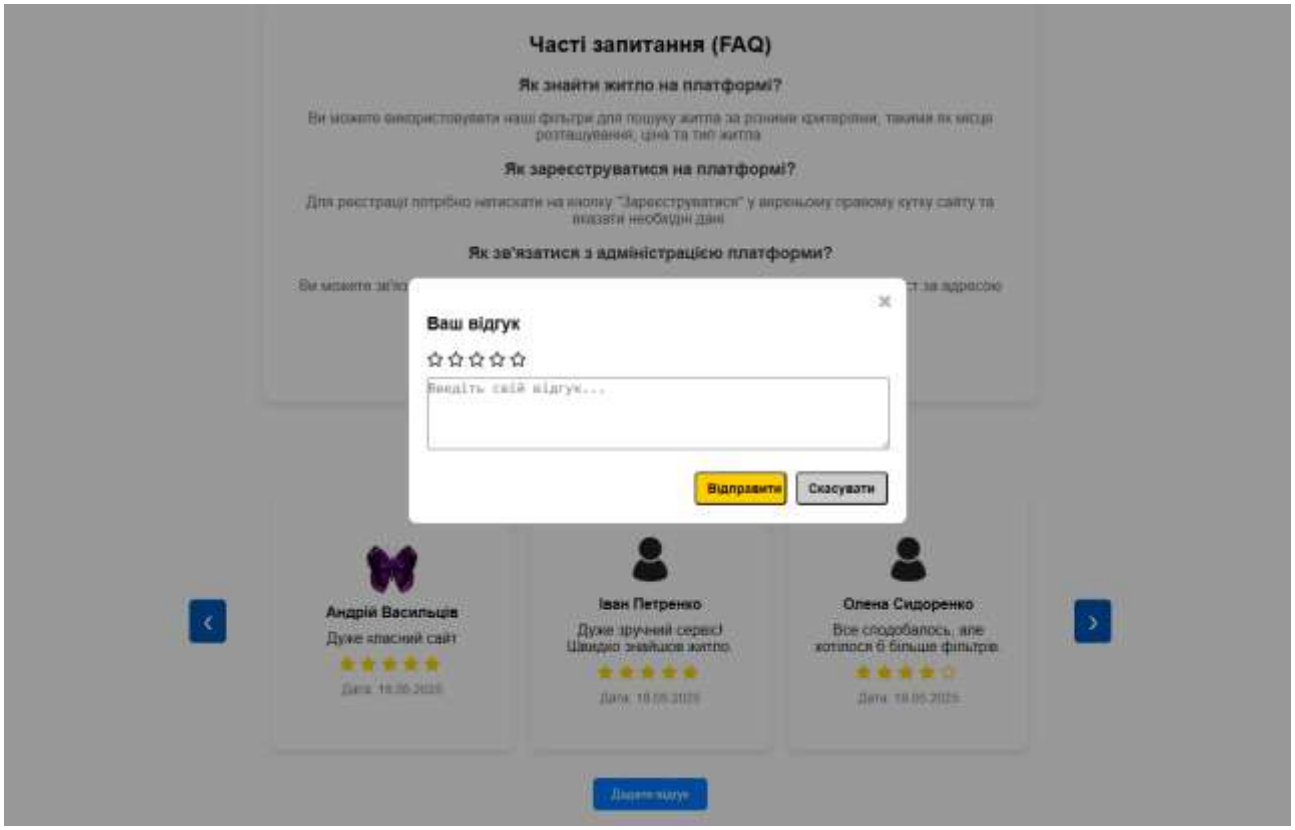


Рисунок 3.8 – Форма відгуку вебсайту для пошуку тимчасового житла

Після заповнення форми та натискання кнопки «Надіслати» дані передаються на сервер асинхронно, без перезавантаження сторінки, за допомогою AJAX-запиту. На сервері працює окремий PHP-контролер

submit_review.php, який приймає отримані дані, перевіряє їх на коректність і зберігає у базі даних. Після успішного додавання відгуку користувач отримує сповіщення про результат, а новий відгук одразу з'являється у списку відгуків на сторінці. Для реалізації цього функціоналу у коді використовуються такі компоненти: HTML-форма для введення відгуку, JavaScript-код для обробки події відправки та взаємодії з сервером, а також серверний PHP-контролер для обробки та збереження даних.

Ще однією важливою функцією, яка підвищує рівень сервісу та довіру користувачів до платформи, є форма зворотного зв'язку з підтримкою. На сторінці «Контакти» реалізовано спеціальну форму, за допомогою якої користувач може звернутися до адміністрації сайту з будь-яким питанням, пропозицією чи скаргою. Для цього необхідно вказати своє ім'я, електронну пошту та текст повідомлення. Після відправлення дані надсилаються на сервер, де обробляються відповідним PHP-контролером (наприклад, send_message.php).

Особливістю цієї форми є те, що після успішної відправки повідомлення користувач автоматично отримує відповідь на вказану електронну пошту. У цьому листі міститься підтвердження отримання звернення та інформація про те, що найближчим часом з ним зв'яжеться представник служби підтримки. Така автоматична відповідь дозволяє користувачу бути впевненим, що його звернення не залишилося без уваги. Візуалізацію автоматичної відповіді наведено на рисунку 3.9 [38].

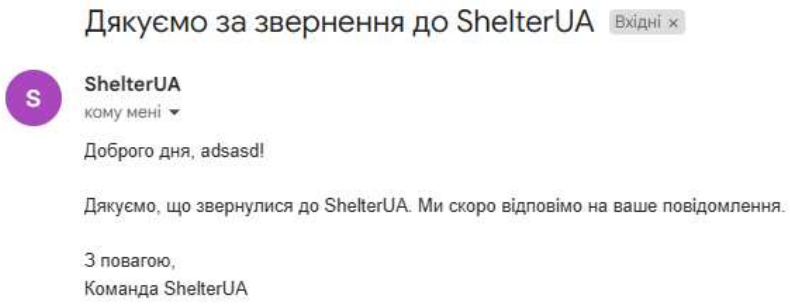


Рисунок 3.9 – Автоматична відповідь підтримки вебсайту для пошуку тимчасового житла

Згодом, після розгляду звернення, користувач отримує детальну відповідь на своє питання або пропозицію від адміністрації платформи.

Наступний етап користувача це реєстрація і авторизація, що є важливим етапом, оскільки саме після створення облікового запису користувач отримує доступ до розширених можливостей сервісу: створення та перегляду власних оголошень, бронювань, ведення особистого профілю, спілкування через чат та отримання сповіщень. На сайті реалізовано два основних способи реєстрації: традиційний (через email і пароль) та спрощений – за допомогою облікового запису Google.

Особливу увагу варто приділити саме реєстрації через Google, оскільки цей спосіб значно підвищує зручність для користувача, скорочує час на заповнення форми та зменшує ймовірність помилок при введенні даних. Для цього використовується протокол OAuth 2.0, який дозволяє безпечно отримати базову інформацію про користувача (ім’я, email, фото профілю) без передачі пароля стороннім сервісам [39]. Процес реєстрації через Google складається з кількох етапів: спочатку користувач натискає кнопку «Увійти через Google», після чого його перенаправляє на сторінку авторизації Google. Після підтвердження доступу Google повертає на сайт спеціальний код авторизації. Візуальне представлення форм для входу та реєстрації, включаючи кнопку для авторизації через Google, наведено на рисунку 3.10.

Далі цей код обробляється у файлі google_callback.php, який можна побачити у Додатку В. На цьому етапі сайт надсилає запит до серверів Google для отримання access_token, а потім – для отримання даних користувача. Якщо email користувача вже існує у базі даних, система виконує вхід під цим обліковим записом; якщо ні – створює новий профіль із отриманими даними. Вся необхідна інформація (ID користувача, email, ім’я, фото профілю) зберігається у сесії, що дозволяє ідентифікувати користувача на всіх сторінках сайту. Після успішної авторизації користувач автоматично перенаправляється на головну сторінку.

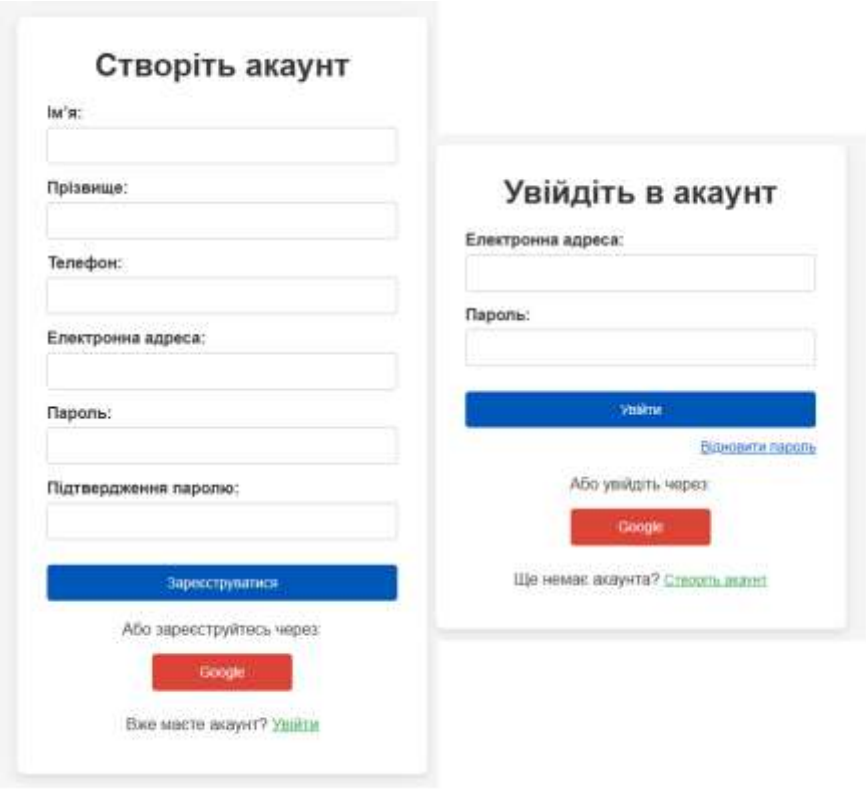


Рисунок 3.10 – Форми реєстрації за допомогою Google вебсайту для пошуку тимчасового житла

Після успішної реєстрації та входу на платформу користувач отримує доступ до всіх основних функцій сайту. Однак у процесі експлуатації можуть виникати ситуації, коли користувач забуває свій пароль або втрачає доступ до облікового запису. Для вирішення таких питань на сайті реалізовано зручний і безпечний механізм відновлення пароля, який дозволяє швидко відновити доступ до персонального кабінету. Далі розглянемо, як саме відбувається процедура відновлення пароля на платформі ShelterUA, а також які етапи проходить користувач під час цього процесу.

Якщо користувач забув свій пароль, він може скористатися спеціальною формою на сторінці відновлення пароля, де необхідно ввести адресу електронної пошти, з якою було зареєстровано обліковий запис. Після відправлення запиту система генерує унікальний токен для відновлення та надсилає на вказану електронну пошту лист із посиланням для зміни пароля.

Користувач отримує електронний лист, у якому міститься індивідуальне посилання для переходу на сторінку встановлення нового пароля. Перейшовши за цим посиланням, користувач може ввести новий пароль, який після підтвердження буде збережено у базі даних. Такий підхід гарантує, що лише власник електронної пошти може змінити пароль до свого облікового запису. Візуалізація цього процесу наведена на рисунку 3.11, де показано приклад електронного листа з посиланням для відновлення пароля, який отримує користувач [40].

Рисунок 3.11 – Електронний лист для відновлення паролю вебсайту для пошуку тимчасового житла

Головна сторінка платформи ShelterUA відкривається для користувача після успішної авторизації або реєстрації. Вона є центральним елементом взаємодії зареєстрованого користувача із сервісом та забезпечує швидкий доступ до основних функцій і персоналізованого контенту. На головній сторінці відображаються актуальні оголошення про доступне житло, персональні рекомендації, а також блоки з інформацією про нові бронювання, сповіщення та повідомлення.

Інтерфейс головної сторінки структуровано таким чином, щоб користувач міг легко переходити до розділів «Пошук», «Мої оголошення», «Мої бронювання», «Обране», «Профіль», «Чат» та «Сповіщення». Для зручності навігації передбачено верхнє меню, де розміщені відповідні посилання. Візуалізацію головної сторінки можна побачити на Рисунку 3.12.

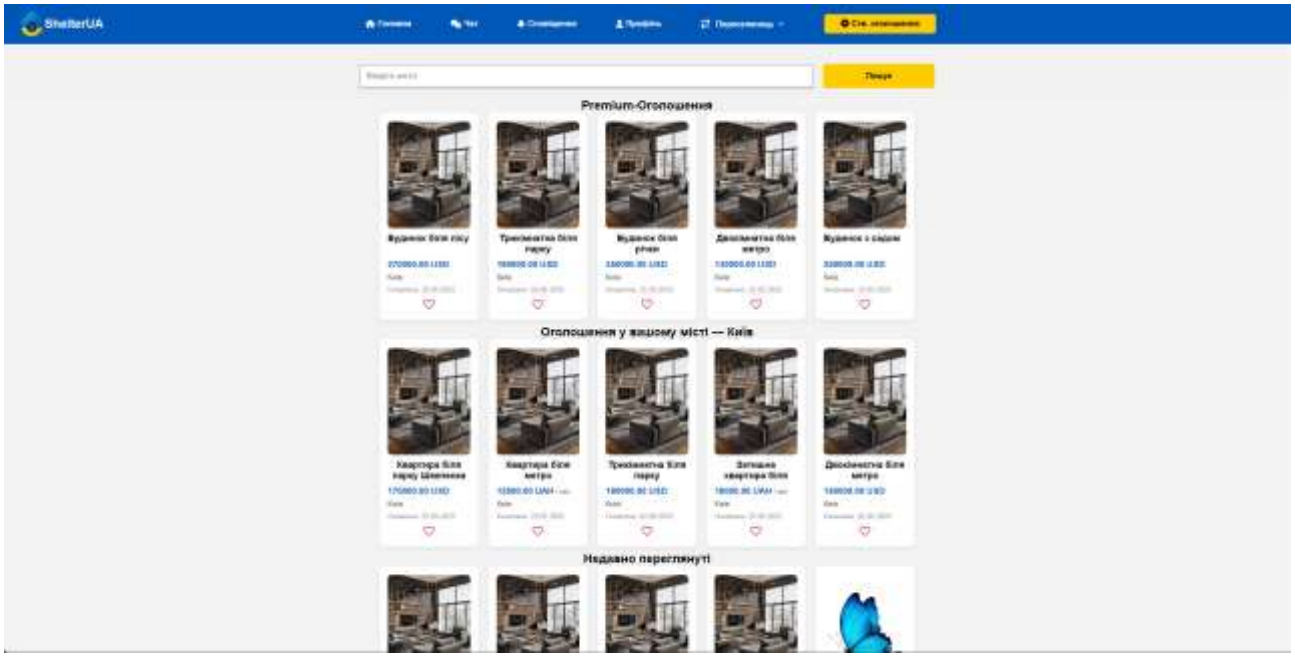


Рисунок 3.12 – Головна сторінка вебсайту для пошуку тимчасового житла

З функціональної точки зору головна сторінка дозволяє не лише переглядати актуальні пропозиції, а й швидко створювати нові оголошення або запити на бронювання за допомогою відповідних кнопок. Також користувач може оперативно реагувати на сповіщення, переглядати нові повідомлення у чаті та переходити до редагування власного профілю.

Після ознайомлення з головною сторінкою користувач отримує можливість перейти до створення власного оголошення або подання запиту на бронювання житла. Для цього на головній сторінці передбачені відповідні кнопки, які ведуть до сторінок створення оголошення (create_listing.php) або створення бронювання (request_booking.php). На цих сторінках користувачу необхідно заповнити детальну інформацію про об’єкт або бажані параметри житла: заголовок, категорію, кількість кімнат, цінову інформацію, площу, опис, додаткові характеристики і тд.

Особливу увагу варто звернути на вибір місцезнаходження об’єкта. Для цього у формі передбачено інтерактивний елемент – кнопку «Вибрати координати на карті». Після її натискання відкривається модальне вікно з інтегрованою картою, реалізованою на основі бібліотеки Leaflet та даних OpenStreetMap. Користувач може знайти потрібну адресу через пошук або

просто клікнути на карту, після чого координати (широта і довгота) автоматично підставляються у приховані поля форми. Обрані координати відображаються для попереднього перегляду, що дозволяє переконатися у правильності вибору [41]. Візуалізацію цього процесу наведено на рисунку 3.13, де показано приклад вікна вибору координат на карті.

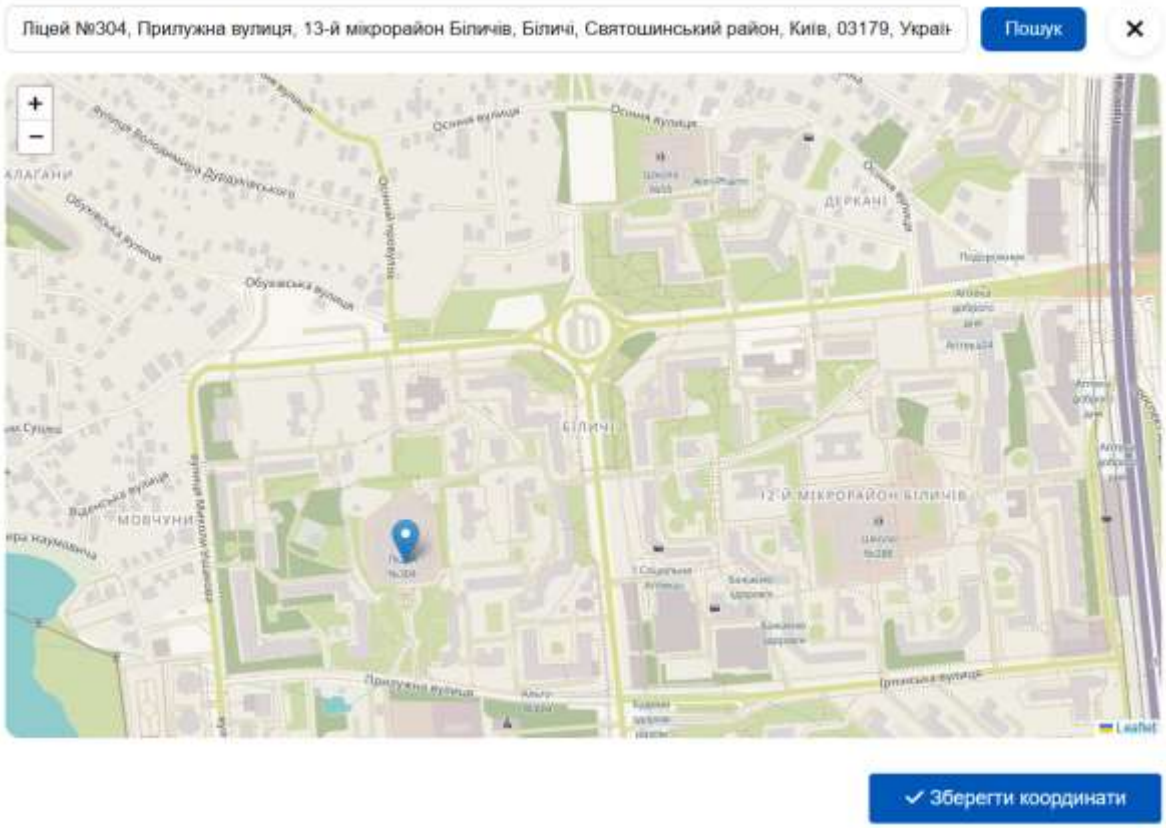


Рисунок 3.13 – Вибір координат оголошення/бронювання вебсайту для пошуку тимчасового житла

Надалі ці координати зберігаються разом з іншими даними оголошення або бронювання у базі даних. Вони використовуються для відображення об’єкта на карті під час пошуку, перегляду детальної інформації, а також для фільтрації та геолокаційного пошуку житла. Код обробників для додавання оголошеня/бронювання можна побачити в Додатку В.

Окрім основних функцій, система мотивації користувачів передбачає нарахування одного токена за кожне успішно створене оголошення або

бронювання. Після завершення процедури додавання нового об’єкта чи запиту, відповідна кількість токенів автоматично зараховується на рахунок користувача. Токени можуть бути використані для активації преміум-можливостей, підвищення видимості оголошень та відображення оголошень довший термін. Такий підхід стимулює активність користувачів і сприяє наповненню сайту актуальними пропозиціями.

Після створення оголошення або бронювання, а також накопичення певної активності на платформі, користувач отримує доступ до розширених можливостей особистого кабінету – сторінки профілю. Перехід до профілю здійснюється через відповідний пункт у головному меню, і саме тут зосереджено всі основні інструменти для керування персональними даними, оголошеннями, бронюваннями та налаштуваннями облікового запису.

У профілі користувач може переглядати та редагувати свої особисті дані, змінювати фото профілю, оновлювати контактну інформацію, а також налаштовувати місцезнаходження. На окремих вкладках відображаються всі створені оголошення та бронювання, які автоматично групуються за статусом: активні, неактивні, протерміновані, переглянуті та обрані. Для кожного оголошення чи бронювання доступні функції редагування, активації, деактивації, продовження терміну дії, видалення або підключення преміум-статусу. Всі ці можливості реалізовані у вигляді зручних кнопок і вкладок, що дозволяє швидко знаходити потрібний об’єкт і виконувати необхідні дії.

Окрему увагу приділено розділам «Обране» та «Переглянуті», де користувач може зберігати цікаві оголошення чи бронювання для швидкого доступу в майбутньому, а також переглядати історію своєї активності. У вкладці налаштувань можна змінити пароль, електронну пошту, а також видалити обліковий запис або вийти з системи. Всі ці функції спрямовані на забезпечення максимальної гнучкості та контролю над персональними даними й контентом користувача.

Візуалізацію інтерфейсу профілю, структуру вкладок та приклади відображення оголошень і бронювань наведено на рисунку 3.14.

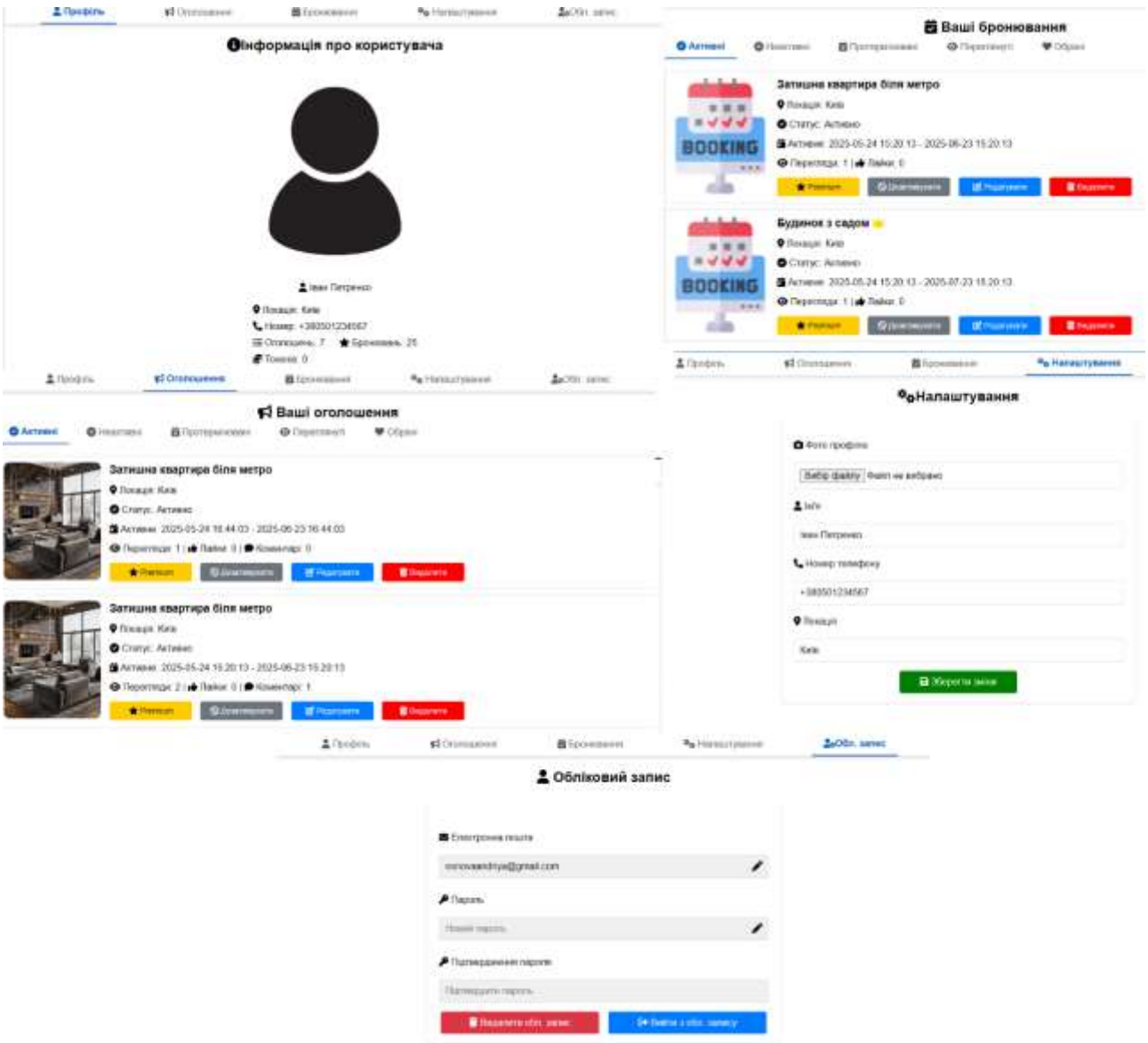


Рисунок 3.14 – Візуалізація інтерфейсу профілю вебсайту для пошуку тимчасового житла

Така організація особистого кабінету дозволяє користувачу ефективно керувати всіма аспектами своєї діяльності на платформі ShelterUA, забезпечує зручність, безпеку та прозорість взаємодії з сервісом.

Після можна перейти до системи сповіщень в ShelterUA, вони інформують користувача про всі ключові події, пов’язані з його оголошеннями, бронюваннями та взаємодією на сайті. Усі сповіщення зберігаються у спеціальному розділі, доступному через головне меню, де користувач може

переглядати історію повідомлень, видаляти їх або переходити до відповідних об’єктів (оголошень чи бронювань) для детальнішого ознайомлення. Приклад інтерфейсу сторінки сповіщень наведено на рисунку 3.15.

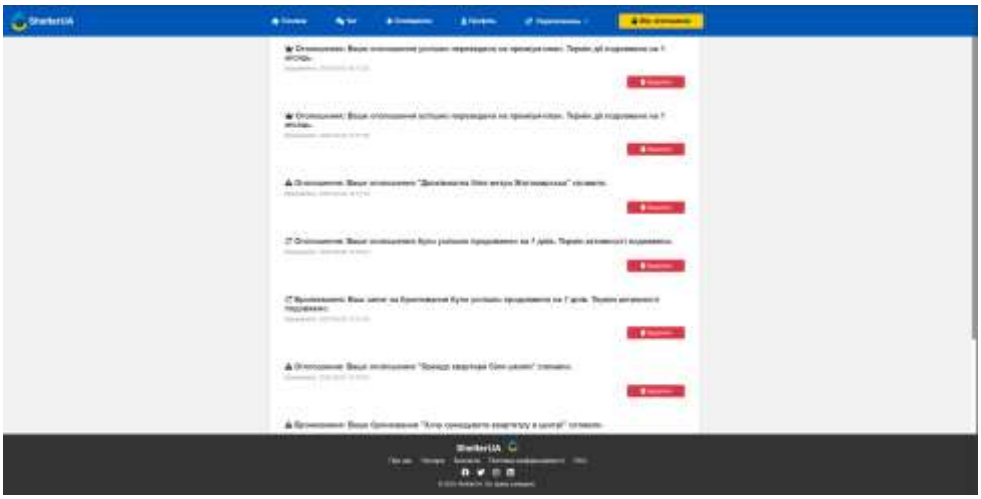


Рисунок 3.15 – Сповіщення на вебсайті для пошуку тимчасового житла

Особливістю реалізації є те, що всі важливі сповіщення, такі як зміна статусу оголошення, наближення дати завершення, активація преміум-статусу, автоматично дублюються на електронну пошту користувача. Це дозволяє оперативно отримувати інформацію навіть без постійного перебування на сайті. Приклад таких листів зі сповіщенням можна побачити на рисунку 3.16.

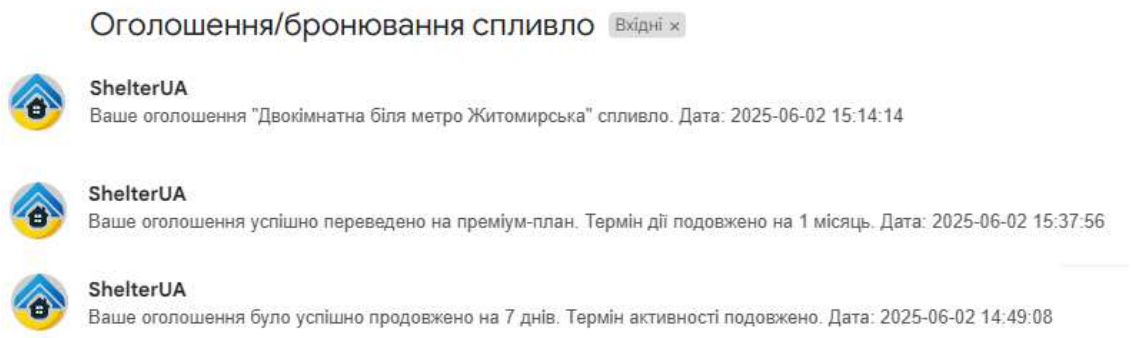


Рисунок 3.16 – Сповіщення на електронній пошті від вебсайту для пошуку тимчасового житла

Непрочитані сповіщення відображаються на головній сторінці та у меню навігації у вигляді іконки з червоним індикатором і кількістю нових повідомлень, що привертає увагу користувача та стимулює своєчасно реагувати на важливі події.

Наступним ключовим етапом взаємодії користувача з платформою є сторінка пошуку, яка реалізує сучасний підхід до геолокаційного відбору житла та бронювань. На цій сторінці користувач може обрати місто для пошуку, ввівши його назву у відповідне поле. Для зручності реалізовано автодоповнення назв міст України, що дозволяє швидко знайти потрібний населений пункт навіть при частковому введенні.

Після вибору міста система автоматично визначає його координати за допомогою сервісу Nominatim (OpenStreetMap) і відображає межі обраного міста на інтерактивній карті, використовуючи бібліотеку Leaflet [42]. Якщо місто не знайдено, система автоматично повертає користувача до Києва, про що виводиться відповідне повідомлення, код реалізації можна побачити у Додатку В у файлі search.php, функція initializeCity()). Детальніше можна побачити на Рисунку 3.17.

Рисунок 3.17 – Обведення міста вебсайту для пошуку тимчасового житла

Особливістю реалізації є те, що карта відображає не лише межі обраного міста, а й обводить кордони всієї України, залишаючи видимими лише ті ділянки, які належать до території країни, детальніше на Рисунку 3.18.

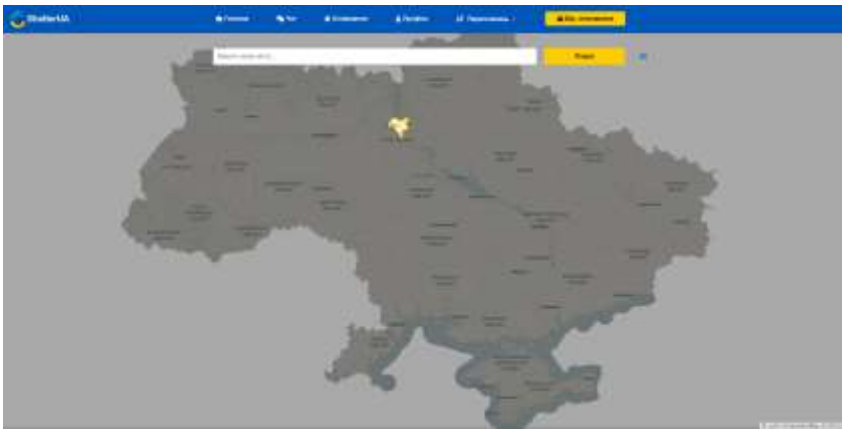


Рисунок 3.18 – Відображення тільки карти України на вебсайті для пошуку тимчасового житла

Це досягається шляхом накладання спеціальної маски, яка затемнює все за межами України, підвищуючи візуальну зручність та орієнтацію користувача. У випадку, якщо у базі є декілька міст з однаковою назвою (наприклад, «Львів»), система пропонує користувачу вибрати потрібний населений пункт із модального вікна, де для кожного варіанту вказано тип, область та іншу додаткову інформацію, Візуалізацію модального вікна можна побачити на Рисунку 3.19.

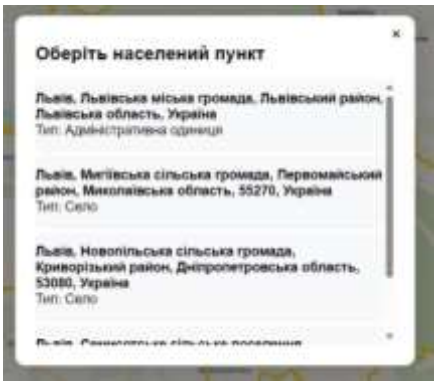


Рисунок 3.19 – Відображення модального вікна вибору міста на вебсайті для пошуку тимчасового житла

Також на карті відображаються маркери для кожного оголошення та бронювання, координати яких були вказані під час створення або автоматично визначені за адресою. Якщо декілька оголошень.бронювань мають однакові координати (наприклад, розташовані на одній вулиці чи в одному будинку), вони групуються в один маркер і відповідно ціна і кількість кімнат відображаються в діапазоні від меншого до більшого, детальніше на Рисунку 3.20.

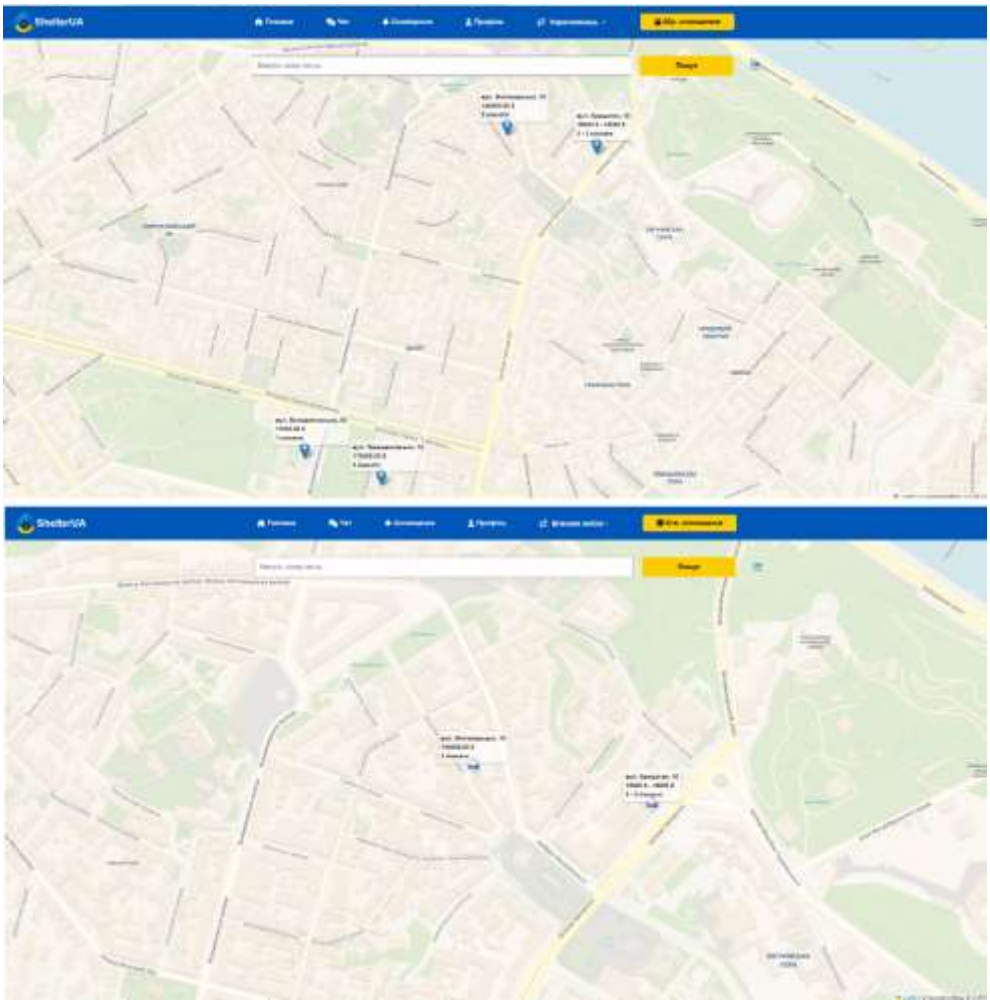


Рисунок 3.20 – Відображення маркерів з оголошеннями/бронюваннями на вебсайті для пошуку тимчасового житла

При натисканні на такий маркер відкривається детальна інформація або список усіх об’єктів, що знаходяться за цією адресою. Для кожного маркера реалізовано підпис із назвою вулиці, діапазоном цін та кількістю кімнат, який з’являється при достатньому масштабі карти. Аналогічний підхід застосовується

і для бронювань: їх маркери мають власний дизайн, а при натисканні відкривається детальна інформація про запит на житло, те як виглядає список об’єктів можна побачити на Рисунку 3.21.

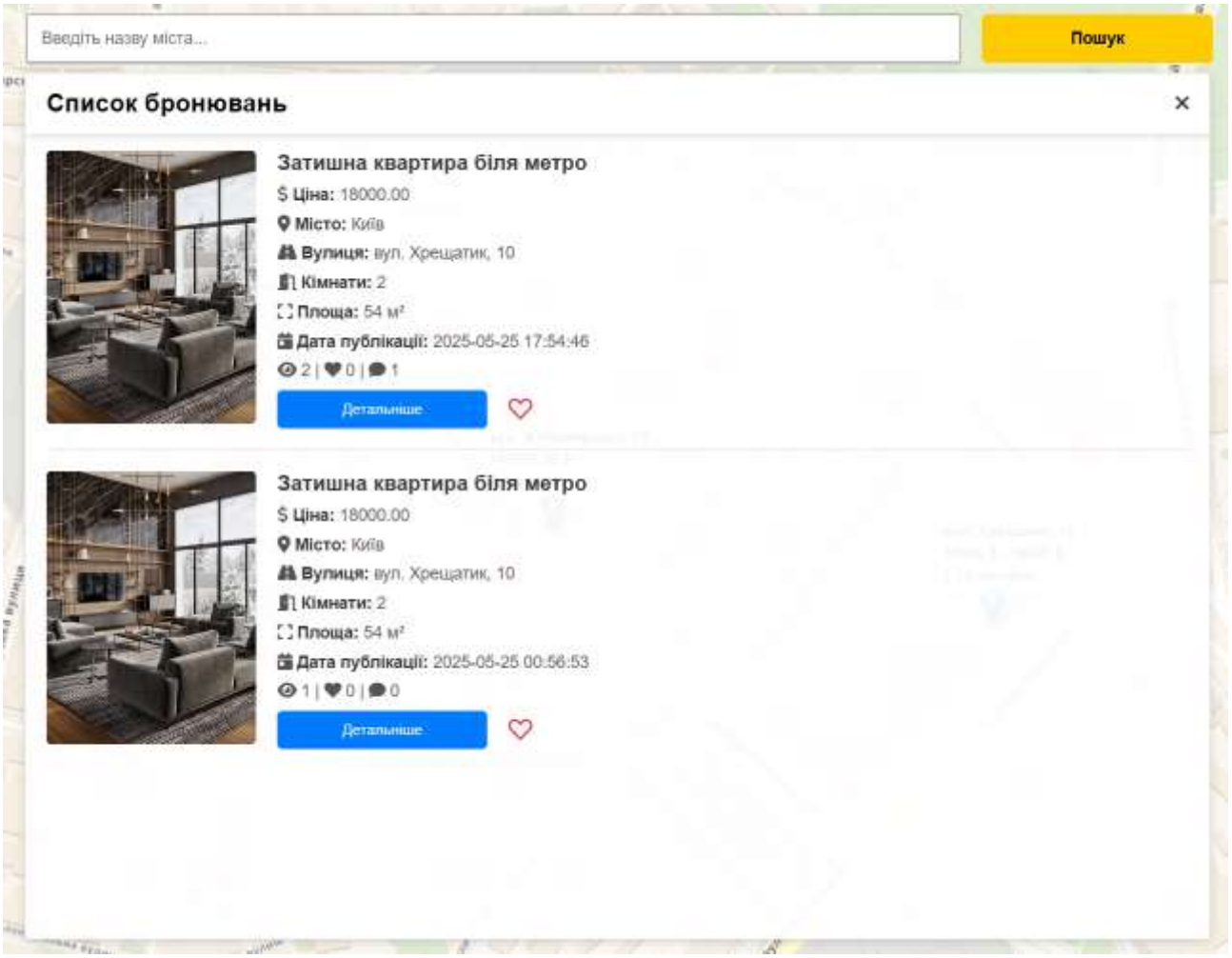


Рисунок 3.21 – Відображення списку об’єктів з оголошеннями/бронюваннями після натискання на маркер на вебсайті для пошуку тимчасового житла

Такий функціонал забезпечує не лише зручність пошуку, а й дозволяє користувачу швидко зорієнтуватися у просторі, оцінити пропозиції в конкретному районі та прийняти обґрунтоване рішення щодо вибору житла чи подання запиту на бронювання. Але для підвищення зручності користування платформою було реалізовано додаткову можливість переглядати всі оголошення або бронювання не лише на карті, а й у вигляді структурованого списку

Відображення у форматі списку дає змогу не лише переглядати детальну інформацію про кожен об’єкт, а й застосовувати різноманітні фільтри та сортування – наприклад, за ціною, кількістю кімнат, датою додавання чи іншими параметрами. Відображення оголошень в списку можна побачити на Рисунку 3.22.

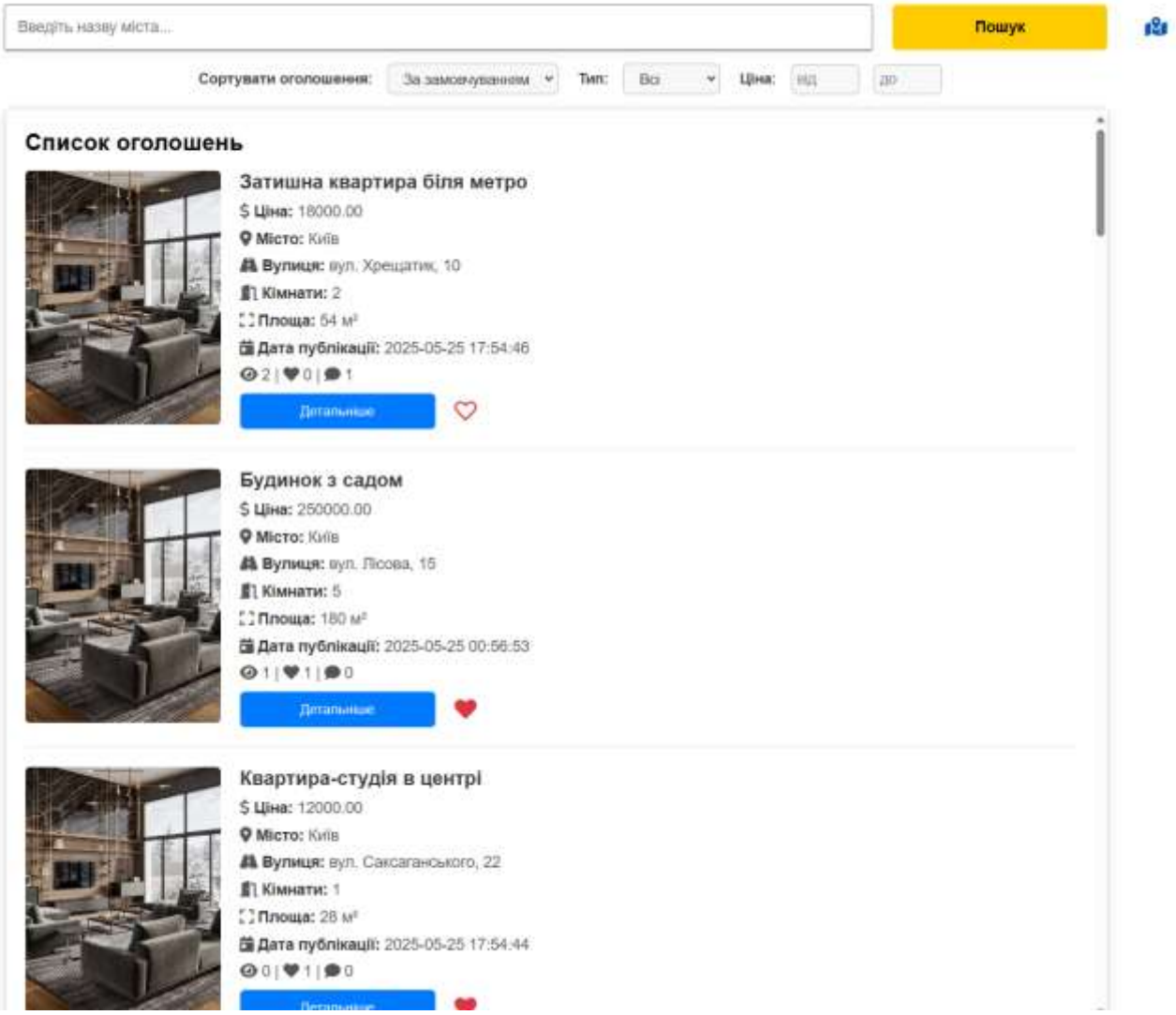


Рисунок 3.22 – Відображення оголошень в списку на вебсайті для пошуку тимчасового житла

Реалізація цього функціоналу детально представлена у коді сторінки пошуку (див. Додаток В), де передбачено перемикач між режимом карти та списку, а також механізми фільтрації й сортування результатів.

Після того як користувач визначився з оголошенням або бронюванням, яке його зацікавило, він може перейти на відповідну детальну сторінку для ознайомлення з усією інформацією про об’єкт, детальніше можна побачити на Рисунку 3.23.

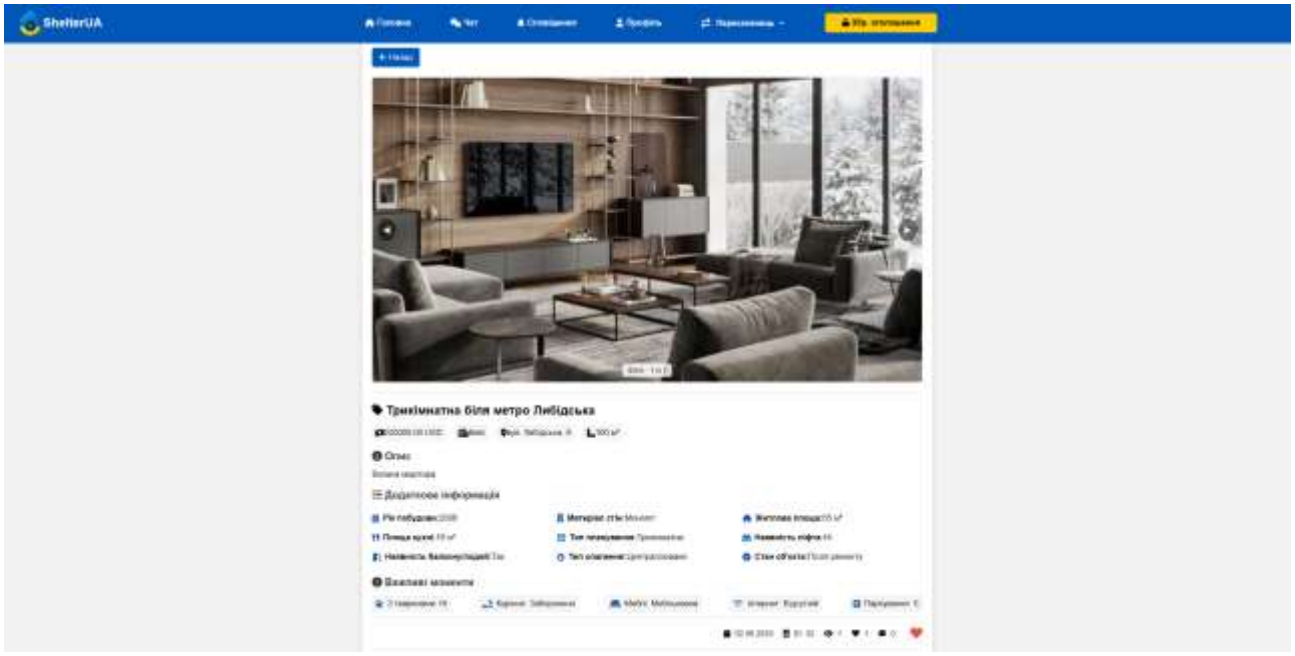


Рисунок 3.23 – Вигляд сторінки з оголошенням на вебсайті для пошуку тимчасового житла

На сторінці оголошення або бронювання користувач бачить повну інформацію про об’єкт: заголовок, тип, категорію, адресу, площу, ціну, опис, додаткові характеристики (матеріал стін, планування, стан, наявність меблів, інтернету, паркування тощо). Всі ці дані структуровано у вигляді зручних блоків, що дозволяє швидко знайти потрібну інформацію. Окремо відображаються важливі моменти, такі як можливість проживання з тваринами, дозволеність куріння, а також інші особливості об’єкта. Для зручності платформа надає можливість додати оголошення чи бронювання до списку обраного, щоб не загубити цікаву пропозицію та швидко повернутися до неї у майбутньому. Додати або видалити об’єкт з обраного можна за допомогою спеціальної іконки у вигляді серця, яка розташована у блоці метаданих оголошення чи бронювання

Важливою складовою є фотогалерея, яка реалізована у вигляді інтерактивної каруселі. Користувач може переглядати всі фотографії, завантажені власником, що дає змогу оцінити стан житла та його особливості.

Під основною інформацією також реалізовано функціонал відгуків. Користувач може ознайомитися з відгуками інших відвідувачів, які вже скористалися цією пропозицією, а також залишити власний відгук, оцінивши об’єкт за п’ятибальною шкалою та написавши коментар. Для цього передбачено окрему кнопку, що відкриває модальне вікно з формою для введення тексту та вибору рейтингу. Всі відгуки відображаються у хронологічному порядку а нижче розміщено інтерактивну карту, на якій відображається точне розташування об’єкта. Для цього використовуються координати, вказані під час створення оголошення або бронювання, а сама карта реалізована на основі бібліотеки Leaflet з використанням даних OpenStreetMap. Це дозволяє користувачу зорієнтуватися у місцевості та оцінити зручність розташування. Розміщення карти та відгуків можна побачити на Рисунку 3.24.

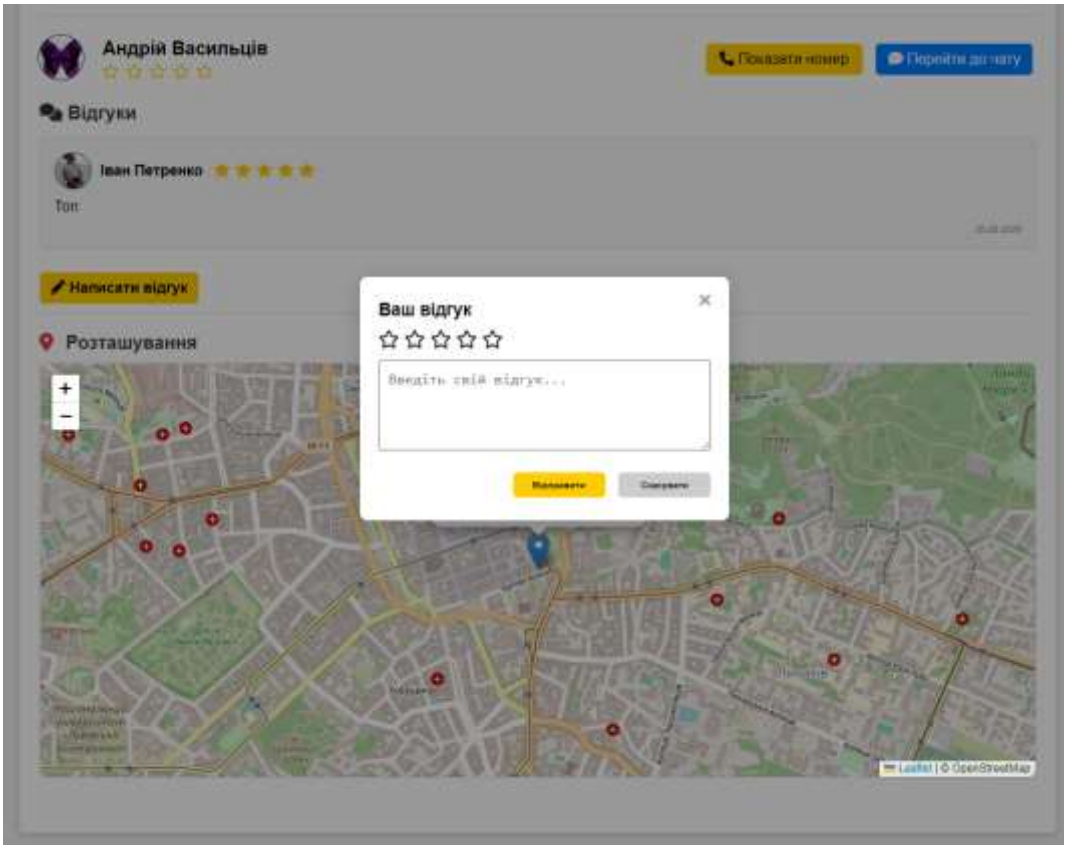


Рисунок 3.24 – Вікно відгуку та карта вебсайту для пошуку тимчасового житла

Окремий блок на сторінці присвячено власнику оголошення або бронювання. Тут можна побачити фото власника, його ім’я, середній рейтинг, а також перейти до публічного профілю власника для ознайомлення з іншими його оголошеннями чи відгуками. Додатково передбачені кнопки для перегляду номера телефону власника (який відкривається лише після натискання) та для переходу до чату, що дозволяє оперативно зв’язатися з власником безпосередньо через платформу.

Для сторінки бронювання інтерфейс організовано майже ідентично до сторінки оголошення: користувач також бачить усю основну інформацію про запит на житло, розташування на карті, контактні дані та блок із власником. Водночас, на відміну від оголошення, на сторінці бронювання відсутні фотогалерея та розділ із відгуками, оскільки такі елементи є неактуальними для запитів на житло. Основний акцент зроблено на деталях бронювання, умовах, датах та можливості зв’язатися з автором запиту, те як виглядає сторінка бронювання можна побачити на Рисунку 3.25.

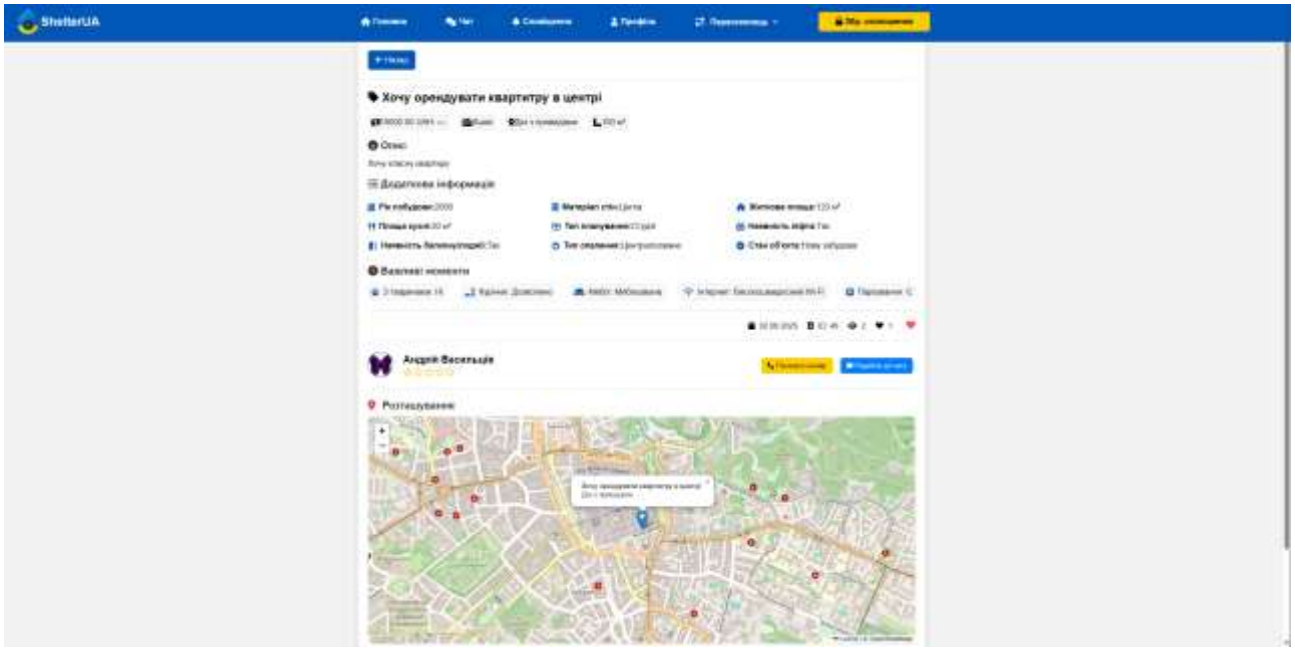


Рисунок 3.25 – Сторінка бронювання вебсайту для пошуку тимчасового житла

Далі варто зазначити, як уже було згадано раніше користувач має можливість переглянути публічний профіль іншого користувача, детальніше на

Рисунку 3.24. У цьому профілі можна залишити відгук про користувача, ознайомитися з усіма його оголошеннями та бронюваннями, переглянути контактний номер телефону (який відкривається за запитом), а також перейти до чату для безпосереднього спілкування.

Отже після ознайомлення з деталями оголошення або бронювання, а також із профілем власника, користувач має можливість перейти до особистого чату для безпосереднього спілкування. Перехід до чату здійснюється за допомогою спеціальної кнопки на сторінці оголошення, бронювання або у профілі користувача. При натисканні цієї кнопки система автоматично створює новий чат між користувачем і власником об’єкта (або відкриває вже існуючий діалог), якщо такий ще не був створений. Водночас реалізовано захист від створення чату із самим собою: якщо користувач намагається ініціювати діалог із власним обліковим записом, кнопка буде неактивною.

Інтерфейс чату організовано у вигляді двох основних розділів: «Шукаю житло» та «Здаю житло», що дозволяє швидко перемикатися між діалогами залежно від ролі користувача у кожній конкретній розмові. У списку чатів відображаються всі активні діалоги, згруповані за співрозмовниками та пов’язаними оголошеннями чи бронюваннями. При виборі чату відкривається історія переписки, де повідомлення відправника та отримувача чітко розділені за кольором і розташуванням. Детальніше як це виглядає від лиця двох різних користувачів можна побачити на Рисунку 3.26.

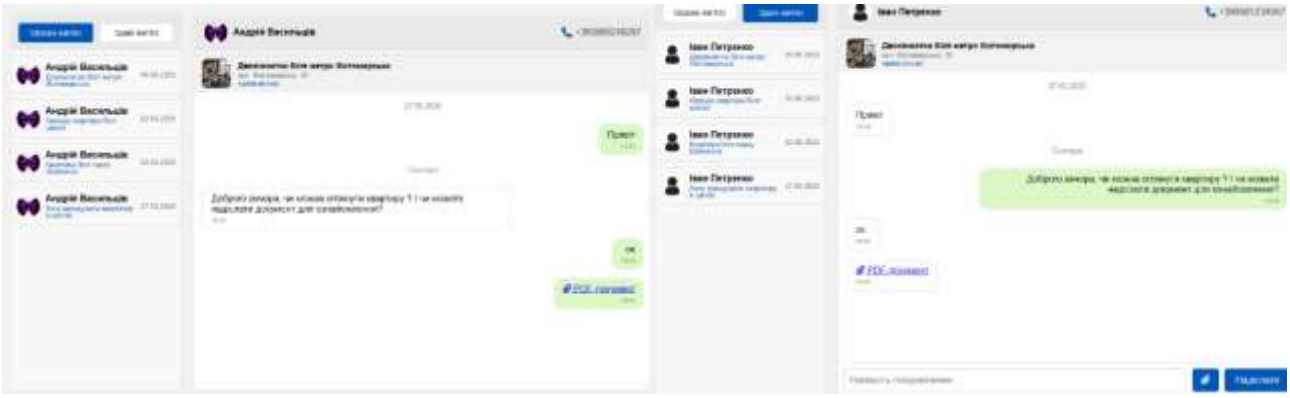


Рисунок 3.26 – Сторінка чату вебсайту для пошуку тимчасового житла

Користувач може надсилати як текстові повідомлення, так і файли – наприклад, фотографії, документи чи інші вкладення, що можуть бути корисними для уточнення деталей угоди. Всі файли зберігаються у спеціальному каталозі та доступні для перегляду або завантаження безпосередньо з інтерфейсу чату, код реалізації чату можна побачити в додатку В. Повідомлення надсилаються у реальному часі, а інтерфейс автоматично оновлюється для обох співрозмовників [43].

3.4 Заходи забезпечення захисту вебсайту для пошуку тимчасового житла

У процесі розробки вебсайту для пошуку тимчасового житла було впроваджено комплекс заходів для забезпечення його захисту та безпеки даних користувачів. По-перше, реалізовано захист від несанкціонованого доступу до персональних даних шляхом використання сесій, перевірки прав доступу та обмеження доступу до окремих сторінок лише для авторизованих користувачів. Всі критично важливі операції, такі як створення, редагування чи видалення оголошень і бронювань, супроводжуються додатковими перевірками на стороні сервера.

Для захисту від SQL-ін’єкцій у всіх запитах до бази даних використовуються підготовлені вирази (prepared statements), що унеможливлює виконання шкідливого коду через поля введення. Валідація та фільтрація даних користувача здійснюється як на клієнтській, так і на серверній стороні, що дозволяє запобігти XSS-атакам та іншим видам ін’єкцій. Для захисту від CSRF-атак у формах використовуються унікальні токени, які перевіряються при кожній критичній операції [44].

Особливу увагу приділено безпечному зберіганню паролів: усі паролі хешуються за допомогою сучасних криптографічних алгоритмів перед збереженням у базі даних. Для підвищення безпеки облікових записів реалізовано механізм відновлення пароля через email із використанням

унікальних токенів, а також можливість авторизації через Google, що додатково знижує ризики компрометації паролів.

Варто зазначити, що частина заходів безпеки забезпечується також на рівні хостинг-провайдера InfinityFree. Зокрема, передбачено обмеження на розмір і тип завантажуваних файлів, а також перевірку їхнього вмісту для запобігання завантаженню шкідливого ПЗ. Хостинг автоматично застосовує SSL-сертифікати для захищеного з'єднання (HTTPS), здійснює базовий захист від DDoS-атак, обмежує доступ до серверних ресурсів та ізолює акаунти користувачів один від одного [45].

3.5 Висновки до третього розділу

У третьому розділі було комплексно розглянуто питання розгортання, тестування, експлуатації та захисту вебсайту для пошуку тимчасового житла. На першому етапі було обґрунтовано вибір хостинг-провайдера: після порівняння декількох популярних сервісів для розміщення сайту було обрано InfinityFree як оптимальний безкоштовний варіант для навчального та тестового проєкту. Проведено налаштування серверного середовища, створення бази даних, підключення до неї через окремий конфігураційний файл, а також виконано всі необхідні дії для забезпечення стабільної та безпечної роботи ресурсу на хостингу.

Особливу увагу приділено валідації та тестуванню вебсайту. За допомогою сервісів W3C Validator та PHP Code Checker було перевірено основні сторінки на відповідність сучасним стандартам, що дозволило виявити та усунути потенційні помилки. Проведено тестування кросбраузерності та адаптивності інтерфейсу, результати якого підтвердили коректну роботу сайту у різних браузерах і на різних пристроях. Це гарантує зручність та доступність платформи для широкого кола користувачів.

У розділі також детально описано основні етапи експлуатації вебсайту: від початкової сторінки, реєстрації та авторизації до роботи з профілем, створення

оголошень, бронювань, перегляду сповіщень, пошуку та комунікації через чат. Кожен функціональний блок супроводжувався візуалізаціями, що дозволяє чітко уявити логіку роботи платформи та взаємодію користувача з інтерфейсом.

Окремий підрозділ було присвячено заходам забезпечення захисту вебсайту. Реалізовано комплексну систему безпеки: захист від SQL-ін’єкцій, XSS- та CSRF-атак, безпечне зберігання паролів, перевірку прав доступу, обмеження на завантаження файлів, а також використання SSL-сертифікатів. Частину захисних функцій додатково забезпечує хостинг-провайдер InfinityFree, що підвищує загальний рівень безпеки платформи.

Загалом, результати, отримані у третьому розділі, свідчать про те, що вебсайт для пошуку тимчасового житла розгорнуто, протестовано та захищено відповідно до сучасних вимог. Це створює надійну основу для подальшого розвитку платформи, її масштабування та впровадження нових функціональних можливостей.

РОЗДІЛ 4. БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

4.1 Працездатність людини – оператора

Під працездатністю оператора розуміють його здатність виконувати професійні завдання з необхідною якістю у визначений термін. Цей показник є комплексною характеристикою, що відображає можливість людини ефективно виконувати свої функції протягом робочого часу. Працездатність визначає не лише продуктивність праці, а й рівень безпеки виконання завдань, особливо у випадках, коли оператор відповідає за складні або відповідальні процеси. Вона формується під впливом багатьох факторів, які умовно поділяють на зовнішні та внутрішні.

До зовнішніх чинників, що впливають на працездатність оператора, належать обсяг і форма інформації, яку він отримує під час роботи, ергономічність робочого місця, характер взаємодії у колективі, а також умови навколишнього середовища. Наприклад, надмірна кількість інформації або її складна структура можуть призводити до перевантаження, що негативно впливає на якість виконання завдань. Невідповідність робочого місця ергономічним вимогам може спричинити швидке настання втоми, а несприятливий мікроклімат або шум – зниження концентрації уваги. Важливу роль відіграє і психологічний клімат у колективі: підтримка з боку колег та керівництва сприяє підвищенню працездатності, тоді як конфлікти або напружені стосунки можуть її знижувати [46].

Внутрішні чинники включають рівень професійної підготовки оператора, його фізичну тренованість, стан здоров’я та емоційну стійкість. Високий рівень підготовки дозволяє швидко орієнтуватися у складних ситуаціях, приймати оптимальні рішення та ефективно використовувати наявні ресурси. Фізична тренованість забезпечує витривалість, а емоційна стійкість допомагає зберігати працездатність навіть у стресових умовах. Особисті якості, такі як

відповідальність, уважність, здатність до самоконтролю, також мають значний вплив на ефективність роботи оператора.

У процесі трудової діяльності оператор проходить різні функціональні стани, які визначають рівень його працездатності (рис. 10). Ці стани змінюються протягом робочого циклу і залежать від інтенсивності навантаження, тривалості роботи, а також від індивідуальних особливостей організму. Відстеження змін функціонального стану дозволяє своєчасно виявляти ознаки втоми та вживати заходів для її попередження.

Виділяють чотири основні фази працездатності: адаптація до роботи, стійка працездатність, субкомпенсація та втома, детальніше на Рисунку 4.1. Тривалість кожної з цих фаз залежить від підготовленості оператора, характеру виконуваних завдань, а також від умов праці. Розуміння особливостей кожної фази дозволяє організувати робочий процес таким чином, щоб максимально використовувати періоди високої працездатності та мінімізувати негативні наслідки втоми.

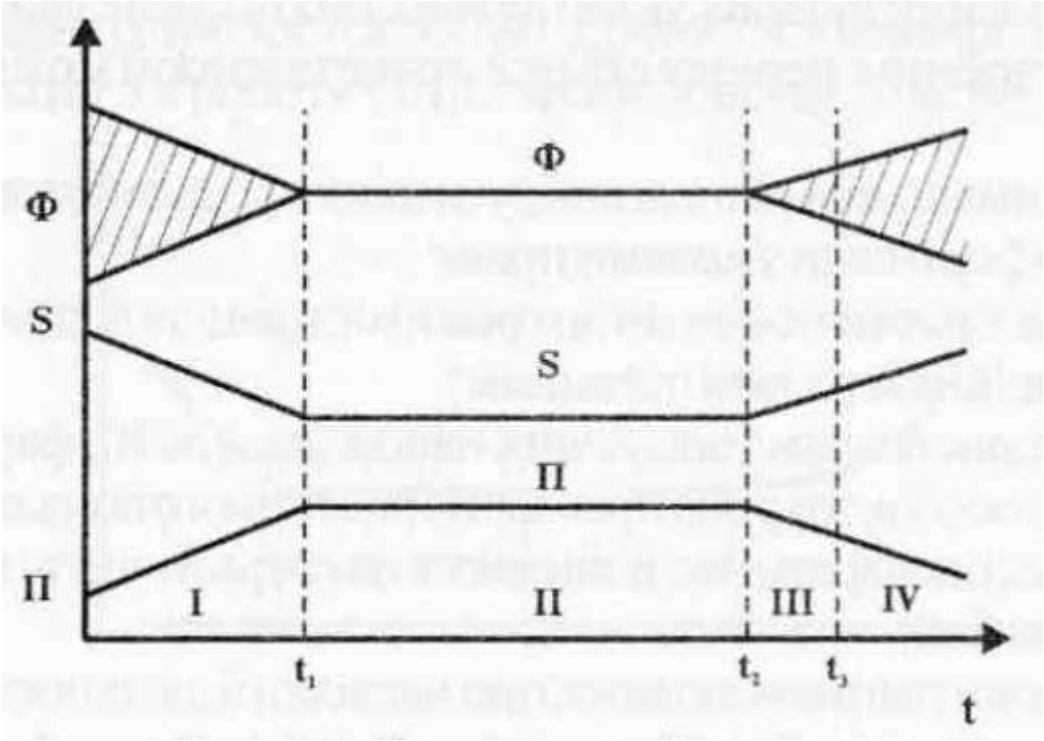


Рисунок 4.1 – Фази працездатності людини

Ф – показник функціонального стану;

Б – помилки роботи;

П – продуктивність праці.

Фаза адаптації ($0–t_1$) – це період, коли організм пристосовується до умов праці. У цей час відбувається налаштування фізіологічних систем на виконання конкретних завдань, формується оптимальний рівень уваги та концентрації. Тривалість цієї фази визначається інтенсивністю роботи, складністю завдань і рівнем підготовки оператора. Скоротити період адаптації дозволяє попереднє тренування, виконання фізичних вправ, адаптація органів чуття до умов праці, а також короткочасне вирішення складних завдань перед початком зміни. Це сприяє швидшому досягненню оптимального функціонального стану.

Фаза стійкої працездатності ($t_1–t_2$) характеризується максимальною якістю виконання завдань при оптимальному функціонуванні організму. У цей період оператор демонструє найвищу продуктивність, мінімальну кількість помилок та стабільний рівень уваги. Тривалість цієї фази залежить від інтенсивності роботи: при динамічній діяльності вона значно довша, ніж при статичній. Емоційний стан оператора суттєво впливає на тривалість стійкої працездатності: позитивні емоції, такі як впевненість у власних силах, задоволення від роботи, подовжують цей період, тоді як негативні – скорочують його.

Підтримати стійку працездатність дозволяють оптимальний рівень психофізіологічної напруги, комфортні умови праці, раціональне чергування праці та відпочинку, емоційне розвантаження, а також інформування про результати роботи. Використання легких стимуляторів (кава, чай) може тимчасово підвищити працездатність, однак надмірне застосування фармакологічних засобів або транквілізаторів призводить до зниження психічної активності та уповільнення реакцій. Важливо забезпечити належний контроль за станом оператора, щоб своєчасно виявляти ознаки перевтоми.

Фаза субкомпенсації ($t_2–t_3$) є початком розвитку втоми. У цей період якість роботи ще залишається на високому рівні, але досягається за рахунок підвищеного напруження функцій організму. Оператор може відчувати

зниження уваги, збільшення кількості помилок, зменшення швидкості реакцій. Якщо не вжити заходів для відновлення працездатності, субкомпенсація переходить у фазу втоми.

Фаза втоми характеризується помітним зниженням якості виконання завдань та погіршенням функціонального стану оператора. Ознаками втоми є зміни частоти пульсу, дихання, а також зниження чутливості органів зору і слуху. Втома може призвести до серйозних помилок, зниження безпеки праці та навіть до виникнення аварійних ситуацій. Тому важливо своєчасно організовувати відпочинок та проводити профілактичні заходи для попередження розвитку втоми.

Після завершення робочого циклу необхідна фаза відновлення працездатності (відпочинку), тривалість якої може варіюватися від кількох хвилин до декількох діб залежно від навантаження, індивідуальних особливостей організму та умов праці. Ефективний відпочинок дозволяє повністю відновити функціональний стан оператора, підвищити його готовність до виконання наступних завдань та знизити ризик виникнення професійних захворювань.

В результаті випливає, що працездатність оператора визначається здатністю виконувати завдання якісно і вчасно. На неї впливають зовнішні (умови праці, інформаційне навантаження, мікроклімат) та внутрішні (підготовка, здоров’я, емоційний стан) чинники. У процесі роботи виділяють фази адаптації, стійкої працездатності, субкомпенсації та втоми. Для підтримки високої працездатності важливо організовувати раціональний режим праці та відпочинку, створювати комфортні умови та слідкувати за станом оператора.

4.2 Соціальне значення охорони праці

Соціальне значення охорони праці є надзвичайно важливим для сучасного суспільства, оскільки воно безпосередньо впливає на ефективність функціонування виробничої сфери, збереження здоров'я працівників та підвищення якості життя населення. Охорона праці сприяє постійному вдосконаленню умов праці, підвищенню їх безпеки, зниженню рівня виробничого травматизму та професійних захворювань. Це, у свою чергу, забезпечує стабільний розвиток економіки та соціальної сфери держави.

Одним із ключових аспектів соціального значення охорони праці є її вплив на зростання продуктивності праці. Покращення умов праці, впровадження сучасних засобів захисту, оптимізація режимів праці та відпочинку дозволяють зменшити кількість виробничих травм і мікротравм, скоротити внутрішньозмінні простої, а також запобігти передчасному стомленню працівників. Усі ці заходи сприяють більш ефективному використанню робочого часу, що позитивно позначається на загальних виробничих показниках підприємства.

Збереження трудових ресурсів є ще одним важливим соціальним аспектом охорони праці. Покращення умов праці сприяє зміцненню здоров'я працівників, подовженню їхнього трудового життя, підвищенню професійної активності та виробничого стажу. Працівники, які працюють у безпечних і комфортних умовах, мають вищий рівень мотивації, частіше підвищують свою кваліфікацію та професійну майстерність, що позитивно впливає на загальний рівень розвитку трудового потенціалу країни.

Важливим наслідком ефективної системи охорони праці є збільшення сукупного національного продукту. Це досягається завдяки підвищенню продуктивності праці, збереженню та розвитку трудових ресурсів, зниженню витрат на лікування травм і професійних захворювань, а також зменшенню втрат, пов'язаних із нещасними випадками на виробництві. Таким чином, охорона

праці є невід’ємною складовою соціально-економічного розвитку держави, що забезпечує стабільність, добробут і здоров’я населення [50].

Отже, соціальне значення охорони праці полягає у створенні безпечних і здорових умов праці, що сприяє підвищенню продуктивності, збереженню трудових ресурсів і зростанню національного добробуту. Ефективна система охорони праці позитивно впливає на економічний розвиток країни та якість життя населення.

4.3 Висновки до четвертого розділу

У четвертому розділі було розглянуто питання працездатності людини-оператора та соціального значення охорони праці є фундаментальними для забезпечення ефективної, безпечної та результативної діяльності у будь-якій виробничій сфері. Працездатність оператора визначає його здатність якісно та своєчасно виконувати професійні завдання, що напряму залежить від комплексу зовнішніх і внутрішніх чинників, таких як умови праці, інформаційне навантаження, рівень підготовки, стан здоров’я та емоційна стійкість. Раціональна організація робочого процесу, оптимальне чергування праці та відпочинку, а також створення комфортного мікроклімату дозволяють підтримувати високий рівень працездатності, знижувати ризик виникнення втоми та помилок, а також підвищувати загальну безпеку праці.

Соціальне значення охорони праці полягає у створенні безпечних і здорових умов для всіх учасників виробничого процесу. Це сприяє зростанню продуктивності, збереженню трудових ресурсів, підвищенню мотивації працівників і, як наслідок, збільшенню національного добробуту. Ефективна система охорони праці не лише знижує рівень виробничого травматизму та професійних захворювань, а й забезпечує стабільний соціально-економічний розвиток держави, підвищує якість життя населення та сприяє формуванню позитивного іміджу підприємства.

Таким чином, інтеграція сучасних підходів до організації праці, впровадження ефективних заходів з охорони праці та постійне вдосконалення умов роботи є запорукою успішної діяльності підприємств, збереження здоров’я працівників і сталого розвитку суспільства в цілому.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи було досягнуто основної мети – створення сучасного, безпечного та функціонального вебсайту для пошуку тимчасового житла, який відповідає актуальним соціальним потребам та вимогам до якості, зручності й захищеності. Всі етапи розробки, від аналізу предметної області до впровадження та тестування, були реалізовані комплексно та системно, що дозволило отримати практично значущий результат.

В першому розділі кваліфікаційної роботи освітнього рівня «Бакалавр» подано комплексний аналіз предметної області пошуку тимчасового житла в Україні, визначено актуальність проблеми для переселенців, сформульовано основні функціональні та нефункціональні вимоги до майбутнього вебсайту, а також обґрунтовано вибір технологічного стеку та середовища розробки. Розглянуто існуючі рішення на ринку, такі як OLX Нерухомість і DOM.RIA, проаналізовано їхні переваги й недоліки, а також визначено потребу у створенні спеціалізованого сервісу для переселенців. Висвітлено структуру користувацьких ролей, сценарії взаємодії, принципи побудови інтерфейсу та особливості організації життєвого циклу розробки вебсайту.

В другому розділі кваліфікаційної роботи досліджено архітектуру вебсайту ShelterUA, що базується на трирівневій структурі з чітким розділенням клієнтської, серверної та рівня даних. Обґрунтовано вибір основних сутностей бази даних, розроблено концептуальну модель, діаграми кооперації, класів та станів, а також структуру каталогів і файлів проєкту. Сформовано функціональні схеми для ключових сторінок і контролерів, що забезпечують прозорість логіки роботи сервісу, зручність підтримки та масштабованість платформи.

В третьому розділі кваліфікаційної роботи розроблено повний цикл розгортання, налаштування, тестування та експлуатації вебсайту для пошуку тимчасового житла. Запропоновано оптимальний вибір хостинг-провайдера, виконано налаштування серверного середовища, організовано підключення до бази даних і забезпечено стабільну роботу ресурсу. Спроєктовано та реалізовано валідацію коду, тестування кросбраузерності й адаптивності, а також детально

описано всі етапи взаємодії користувача з платформою: від реєстрації, створення оголошень і бронювань до роботи з профілем, сповіщеннями, пошуком і чатом. Протестовано функціонал на відповідність сучасним стандартам, що підтверджено результатами валідації та тестування.

У розділі «Безпека життєдіяльності, основи охорони праці» висвітлено питання інформаційно-психологічної безпеки користувачів вебсайту, ризики шахрайства та емоційного стресу, а також професійні ризики для веб-розробників, які працюють із соціально значущими продуктами. Окреслено комплекс заходів для мінімізації технічних і психологічних загроз, підвищення рівня довіри до платформи, а також профілактики емоційного вигорання та підтримки психоемоційного стану розробників.

Загалом, результати кваліфікаційної роботи свідчать про успішне вирішення поставлених завдань, створення сучасного, безпечного та функціонального вебсайту для пошуку тимчасового житла, який відповідає актуальним потребам цільової аудиторії та сучасним вимогам до якості, безпеки й зручності використання.

ПЕРЕЛІК ДЖЕРЕЛ

1. OLX [Електронний ресурс]. – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/OLX (дата звернення: 02.06.2025).

2. DOM.RIA [Електронний ресурс]. – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/DOM.RIA (дата звернення: 02.06.2025).

3. Переваги DOM.RIA [Електронний ресурс]. – Режим доступу до ресурсу: https://dom.ria.com/uk/ (дата звернення: 01.06.2025).

4. Приклади дизайну веб-інтерфейсу [Електронний ресурс]. – Режим доступу до ресурсу: https://stfalcon.com/uk/blog/post/web-ui-design-examples (дата звернення: 07.06.2025).

5. Adobe Illustrator для початківців: можливості, плюси та мінуси [Електронний ресурс]. – Режим доступу до ресурсу: https://goit.global/ua/articles/adobe-illustrator-dlia-pochatkivtsiv-mozhlyvosti-pliusy-ta-minusy/ (дата звернення: 07.06.2025).1

6. 10 етапів створення сайту – від ідеї до запуску [Електронний ресурс]. – Режим доступу до ресурсу: https://wezom.com.ua/ua/blog/etapy-razrabotki-sajta?utm_source=chatgpt.com (дата звернення: 07.06.2025).

7. Вимоги до програмного забезпечення [Електронний ресурс]. – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/%D0%92%D0%B8%D0%BC%D0%BE%D0%B3%D0%B8_%D0%B4%D0%BE_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%B7%D0%B0%D0%B1%D0%B5%D0%B7%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%BD%D1%8F (дата звернення: 07.06.2025).

8. Кінцевий посібник щодо ролей користувачів WordPress [Електронний ресурс]. – Режим доступу до ресурсу: https://www.dreamhost.com/blog/uk/roli-koristuvachiv-wordpress/?utm_source=chatgpt.com (дата звернення: 07.06.2025).

9. Діаграма варіантів використання UML [Електронний ресурс]. – Режим доступу до ресурсу: https://uk.techcodeview.com/uml-use-case-diagram (дата звернення: 07.06.2025)

10. Що таке CMS сайту? [Електронний ресурс]. – Режим доступу до ресурсу: https://frontend.lviv.ua/shho-take-cms-top-5-cms-dlya-rozrobky-sajtu (дата звернення: 07.06.2025).

11. 22 кращих конструктори сайтів: повний огляд [Електронний ресурс]. – Режим доступу до ресурсу: https://hostiq.ua/blog/ukr/site-builders/ (дата звернення: 07.06.2025).

12. Цікаво, як створити вебсайт з нуля? Ось повний підручник [Електронний ресурс]. – Режим доступу до ресурсу: https://www.dreamhost.com/blog/uk/posibnik-dlia-pochatkivtsiv-z-vebsaitiv/ (дата звернення: 07.06.2025).

13. Чому самостійне створення сайту – це гарна ідея [Електронний ресурс]. – Режим доступу до ресурсу: https://tods-blog.com.ua/news/itcomp/samostijne-stvorennya-sajtu/ (дата звернення: 07.06.2025).

14. Етапи життєвого циклу розробки ПЗ [Електронний ресурс]. – Режим доступу до ресурсу: https://icstudio.online/post/etapi-zhittyevogo-ciklu-rozrobki-pz?utm_source=chatgpt.com (дата звернення: 07.06.2025).

15. Як вибрати правильний технологічний стек для вашого проекту [Електронний ресурс]. – Режим доступу до ресурсу: https://redstone.agency/blog/yak-vybraty-pravylnyi-tekhnolohichnyi-stek-dlia-vashoho-proektu/ (дата звернення: 07.06.2025).

16. Open Server [Електронний ресурс]. – Режим доступу до ресурсу: https://armedsoft.com/ua/blog/nalashtuvannya-open-server (дата звернення: 07.06.2025).

17. Найкращий редактор коду Visual Studio Code [Електронний ресурс]. – Режим доступу до ресурсу: https://dou.ua/forums/topic/47918/ (дата звернення: 07.06.2025).

18. Клієнт-сервер архітектура [Електронний ресурс]. – Режим доступу до ресурсу: https://buklib.net/books/23148 (дата звернення: 07.06.2025).

19. Collaboration Diagrams | Unified Modeling Language (UML) [Електронний ресурс]. – Режим доступу до ресурсу: https://www.geeksforgeeks.org/collaboration-diagrams-unified-modeling-languageuml/ (дата звернення: 07.06.2025).

20. Діаграми взаємодії, співпраці та послідовності з прикладами [Електронний ресурс]. – Режим доступу до ресурсу: https://www.guru99.com/uk/interaction-collaboration-sequence-diagrams-examples.html (дата звернення: 07.06.2025).

21. Як побудувати структуру сайту: від А до Я з прикладами [Електронний ресурс]. – Режим доступу до ресурсу: https://sendpulse.ua/blog/website-structure-with-examples?utm_source=chatgpt.com (дата звернення: 07.06.2025).

22. Основні етапи проєктування баз даних [Електронний ресурс]. – Режим доступу до ресурсу: https://javarush.com/ua/quests/lectures/ua.questhibernate.level17.lecture01 (дата звернення: 07.06.2025).

23. Що таке моделювання даних? Типи (концептуальний, логічний, фізичний) [Електронний ресурс]. – Режим доступу до ресурсу: https://www.guru99.com/uk/data-modelling-conceptual-logical.html (дата звернення: 07.06.2025).

24. Entity–relationship model [Електронний ресурс]. – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Entity%E2%80%93relationship_model (дата звернення: 07.06.2025).

25. Проектування вебсайта [Електронний ресурс]. – Режим доступу до ресурсу: https://sites.znu.edu.ua/webprog/lect/1194.ukr.html (дата звернення: 07.06.2025)

26. Структура файлів [Електронний ресурс]. – Режим доступу до ресурсу: http://htmlbook.in.ua/file-structure/ (дата звернення: 07.06.2025).

27. Функціональна схема [Електронний ресурс]. – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Functional_flow_block_diagram (дата звернення: 07.06.2025).

28. UML для бізнес-моделювання: для чого потрібні діаграми процесів [Електронний ресурс]. – Режим доступу до ресурсу: https://evergreens.com.ua/ua/articles/uml-diagrams.html (дата звернення: 07.06.2025).

29. Створення схеми діаграми станів UML [Електронний ресурс]. – Режим доступу до ресурсу: https://support.microsoft.com/uk-ua/office/%D1%81%D1%82%D0%B2%D0%BE%D1%80%D0%B5%D0%BD%D0%BD%D1%8F-%D1%81%D1%85%D0%B5%D0%BC%D0%B8-%D0%B4%D1%96%D0%B0%D0%B3%D1%80%D0%B0%D0%BC%D0%B8-%D1%81%D1%82%D0%B0%D0%BD%D1%96%D0%B2-uml-2e46fd66-e861-4e8c-9188-36255395ebf3 (дата звернення: 07.06.2025).

30. Підручник з діаграми класів UML: абстрактний клас із прикладами [Електронний ресурс]. – Режим доступу до ресурсу: https://www.guru99.com/uk/uml-class-diagram.html (дата звернення: 07.06.2025).

31. Що таке моделювання даних? [Електронний ресурс]. – Режим доступу до ресурсу: https://www.guru99.com/uk/data-modelling-conceptual-logical.html (дата звернення: 07.06.2025).

32. Проектування структури вебсайту [Електронний ресурс]. – Режим доступу до ресурсу: https://wezom.com.ua/ua/blog/proektirovanie-struktury-web-sajta (дата звернення: 07.06.2025).

33. Огляд Infinityfree [Електронний ресурс]. – Режим доступу до ресурсу:https://googiehost.com/uk/%D0%B1%D0%BB%D0%BE%D0%B7%D1%96/infinityfree-%D0%BE%D0%B3%D0%BB%D1%8F%D0%B4/ (дата звернення: 07.06.2025).

34. W3C Markup Validation Service [Електронний ресурс]. – Режим доступу до ресурсу: https://validator.w3.org/about.html (дата звернення: 07.06.2025).

35. PHP Code Checker [Електронний ресурс]. – Режим доступу до ресурсу: https://www.bairesdev.com/tools/phpcodechecker/ (дата звернення: 07.06.2025).

36. Що таке адаптивний дизайн сайту та як його зробити [Електронний ресурс]. – Режим доступу до ресурсу: https://hostiq.ua/blog/ukr/adaptive-design/ (дата звернення: 07.06.2025).

37. Етапи створення веб сайтів [Електронний ресурс]. – Режим доступу до ресурсу: https://webtune.com.ua/statti/web-rozrobka/etapy-stvorennya-veb-sajtiv/ (дата звернення: 07.06.2025).

38. PHPMailer – A full-featured email creation and transfer class for PHP [Електронний ресурс]. – Режим доступу до ресурсу: https://github.com/PHPMailer/PHPMailer (дата звернення: 07.06.2025).

39. Using OAuth 2.0 for Web Server Applications [Електронний ресурс]. – Режим доступу до ресурсу: https://developers.google.com/identity/protocols/oauth2/webserver#:~:text=OAuth%202.0%20allows%20users%20to%20share%20specific%20data,users%20to%20store%20files%20in%20their%20Google%20Drives (дата звернення: 07.06.2025).

40. bin2hex. Create reset link [Електронний ресурс]. – Режим доступу до ресурсу: https://www.php.net/manual/en/function.bin2hex.php (дата звернення: 07.06.2025).

41. leaflet – an open-source JavaScript library for mobile-friendly interactive maps [Електронний ресурс]. – Режим доступу до ресурсу: https://leafletjs.com/ (дата звернення: 07.06.2025).

42. Nominatim Manual [Електронний ресурс]. – Режим доступу до ресурсу: https://nominatim.org/release-docs/develop/ (дата звернення: 07.06.2025).

43. Build Live Chat System with Ajax, PHP & MySQL [Електронний ресурс]. – Режим доступу до ресурсу: https://www.phpzag.com/build-live-chat-system-with-ajax-php-mysql/ (дата звернення: 07.06.2025).

44. PHP MySQL Prepared Statements [Електронний ресурс]. – Режим доступу до ресурсу: https://www.w3schools.com/php/php_mysql_prepared _statements.asp (дата звернення: 07.06.2025).

45. Security in InfinityFree [Електронний ресурс]. – Режим доступу до ресурсу: https://forum.infinityfree.com/t/how-to-get-free-ssl-https-on-infinityfree/49323 (дата звернення: 07.06.2025).

46. Бедрій Я.І. Основи охорони праці : навч. посіб. 4-е вид. перероб. і доп. — Тернопіль : Навчальна книга – Богдан, 2018. — 240 с. — Розділ 1.2.3. Працездатність людини-оператора.

47. Запорожець О.І. Безпека життєдіяльності. Підручник, 2-е видання, Центр учбової літератури, 2020. 448 с.

48. Мелех Л.В. Безпека життєдіяльності та охорона праці: навч. посіб. / Мелех Л.В. – Львів: ЛДУ внутрішніх справ. 2022. 219 с.

49. Безпека життєдіяльності: навч. посіб. / Т.Є. Стиценко, Г.В. Пронюк, Н.М. Сердюк, І.І. Хондак. Харків: ХНРУЕ, 2018. 336 с.

50. Гандзюк М.П., Желібо Є.П., Халімовський М.О. Основи охорони праці: Підручник. 5-е вид. / За ред. М.П. Гандзюка. – К.: Каравела, 2011. – 384 с.

51. Leshchyshyn, Y., Scherbak, L., Nazarevych, O., Gotovych, V., Tymkiv, P., & Shymchuk, G. (2019, May). Multicomponent Model of the Heart Rate Variability Change-point. In 2019 IEEE XVth International Conference on the Perspective Technologies and Methods in MEMS Design (MEMSTECH) (pp. 110-113). IEEE.

52. Lytvynenko, I., Lupenko, S., Nazarevych, O., Shymchuk, G., & Hotovych, V. (2021, September). Mathematical model of gas consumption process in the form of cyclic random process. In 2021 IEEE 16th International Conference on Computer Sciences and Information Technologies (CSIT) (Vol. 1, pp. 232-235). IEEE.

53. Kozlovskyi, V., Balanyuk, Y., Martyniuk, H., Nazarevych, O., Scherbak, L., & Shymchuk, G. (2022, April). Information Technology for Estimating City Gas Consumption During the Year. In 2022 International Conference on Smart Information Systems and Technologies (SIST) (pp. 1-4). IEEE.

54. Lytvynenko, I., Lupenko, S., Kunanets, N., Nazarevych, O., Shymchuk, G., & Hotovych, V. (2021, November). Simulation of gas consumption process based on the mathematical model in the form of cyclic random process considering the scale factors. In 1st International Workshop on Information Technologies: Theoretical and Applied Problems, ITTAP 2021.

55. Kunanets, N., Pasichnyk, V., Bodnarchuk, I., Martsenko, S., Matsiuk, O., Matsiuk, A., ... & Shymchuk, H. (2019). Information system for visual analyzer disease diagnostics. In CEUR Workshop Proceedings (pp. 43-56).

56. Lupenko, S., Lytvynenko, I., Nazarevych, O., Shymchuk, G., & Hotovych, V. (2021, December). Approach to gas consumption process forecasting on the basis of a mathematical model in the form of a random cyclic process. In Proceedings of the International Conference „Advanced applied energy and information technologies 2021”, 2021 (pp. 213-219). TNTU, Zhytomyr «Publishing house „Book-Druk “» LLC.

ДОДАТКИ

Додаток А

Таблиця А.1.1 – Опис сутності (Listings)

Поле	Тип даних	Опис
listing_id	int(11)	Унікальний ідентифікатор оголошення
user_id	int(11)	Власник оголошення
title	varchar(255)	Назва оголошення
description	text	Опис
category	varchar(100)	Категорія (apartment, house тощо)
offer_type	enum('sale','rent')	Тип пропозиції (продаж/оренда)
rooms	int(11)	Кількість кімнат
floor	int(11)	Поверх
total_floors	int(11)	Поверховість квартири
total_floors_house	int(11)	Поверховість будинку
area	float	Загальна площа
living_area	float	Житлова площа
kitchen_area	float	Площа кухні
build_year	int(11)	Рік побудови
renovation_year	int(11)	Рік ремонту
material	varchar(100)	Матеріал будинку
layout	varchar(100)	Планування
balcony	varchar(50)	Балкон
elevator	varchar(11)	Ліфт
heating	varchar(100)	Опалення
condition	varchar(100)	Стан
price_sale	decimal(12,2)	Ціна продажу
price_rent	decimal(12,2)	Ціна оренди
currency_sale	varchar(10)	Валюта продажу
currency_rent	varchar(10)	Валюта оренди
city	varchar(100)	Місто
address	varchar(255)	Адреса
lat	double	Широта
lon	double	Довгота
pets_allowed	varchar(11)	Дозвіл на тварин
smoking_allowed	varchar(11)	Дозвіл на куріння
furniture_provided	varchar(11)	Меблі
internet_available	varchar(11)	Інтернет
parking_available	varchar(20)	Паркінг
created_at	datetime	Дата створення
updated_at	datetime	Дата оновлення
status	enum('active','inactive','expired')	Статус оголошення
plan	enum('standart','premium')	Тип плану
expiration_date	datetime	Дата завершення
photos	text	Фото (шляхи до файлів через кому)

Таблиця А.1.2 – Опис сутності (Booking)

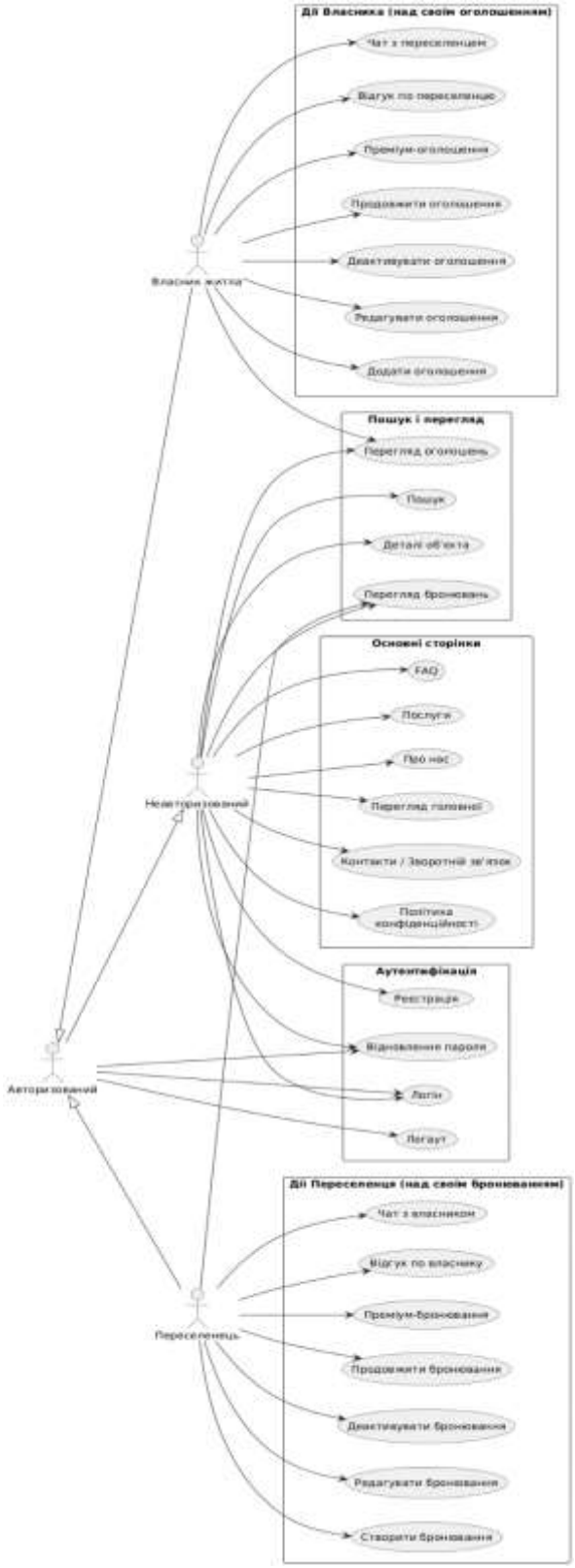
Поле	Тип даних	Опис
booking_id	int(11)	Унікальний ідентифікатор бронювання
user_id	int(11)	Користувач, що створив запит
title	varchar(255)	Назва запиту
description	text	Опис
category	varchar(100)	Категорія
offer_type	enum('sale','rent')	Тип пропозиції
rooms	int(11)	Кількість кімнат
floor	int(11)	Поверх
total_floors	int(11)	Поверховість будинку
total_floors_house	int(11)	Поверховість будинку (альтернативне)
area	float	Загальна площа
living_area	float	Житлова площа
kitchen_area	float	Площа кухні
build_year	int(11)	Рік побудови
renovation_year	int(11)	Рік ремонту
material	varchar(100)	Матеріал будинку
layout	varchar(100)	Планування
balcony	varchar(50)	Балкон
elevator	varchar(10)	Ліфт
heating	varchar(100)	Опалення
condition	varchar(100)	Стан
price_sale	decimal(12,2)	Ціна продажу
price_rent	decimal(12,2)	Ціна оренди
currency_sale	varchar(10)	Валюта продажу
currency_rent	varchar(10)	Валюта оренди
city	varchar(100)	Місто
address	varchar(255)	Адреса
lat	double	Широта
lon	double	Довгота
pets_allowed	varchar(11)	Дозвіл на тварин
smoking_allowed	varchar(11)	Дозвіл на куріння
furniture_provided	varchar(11)	Меблі
internet_available	varchar(11)	Інтернет
parking_available	varchar(20)	Паркінг
created_at	datetime	Дата створення
updated_at	datetime	Дата оновлення
status	enum('active','inactive','expired')	Статус запиту
plan	enum('standart','premium')	Тип плану
expiration_date	datetime	Дата завершення

Таблиця А.1.3 – Матриця доступу користувачів до сторінок вебсайту

Сторінка	Не автор.	Авторизований	Власник	Переселенець
Головна	+	+	+	+
Про нас	+	+	+	+
Послуги	+	+	+	+
FAQ	+	+	+	+
Політика конфіденційності	+	+	+	+
Контакти	+	+	+	+
Пошук	+	+	+	+
Перегляд оголошень	+	+	+	+
Перегляд бронювань	+	+	+	+
Деталі об’єкта	+	+	+	+
Реєстрація	+	–	–	–
Логін	+	–	–	–
Відновлення пароля	+	+	+	+
Логаут	–	+	+	+
Улюблені	–	+	+	+
Відгуки	–	+	+	+
Сповіщення	–	+	+	+
Профіль	–	+	+	+
Додати оголошення	–	–	+	–
Редагувати оголошення	–	–	+	–
Деактивувати оголошення	–	–	+	–
Продовжити оголошення	–	–	+	–
Преміум-оголошення	–	–	+	–
Відгук по переселенцю	–	–	+	–
Чат з переселенцем	–	–	+	–
Створити бронювання	–	–	–	+
Редагувати бронювання	–	–	–	+
Деактивувати бронювання	–	–	–	+
Продовжити бронювання	–	–	–	+
Преміум-бронювання	–	–	–	+
Відгук по власнику	–	–	–	+
Чат з власником	–	–	–	+

Додаток Б

Діаграма варіантів використання вебсайту для пошуку тимчасового житла



Діаграма класів вебсайту для пошуку тимчасового житла

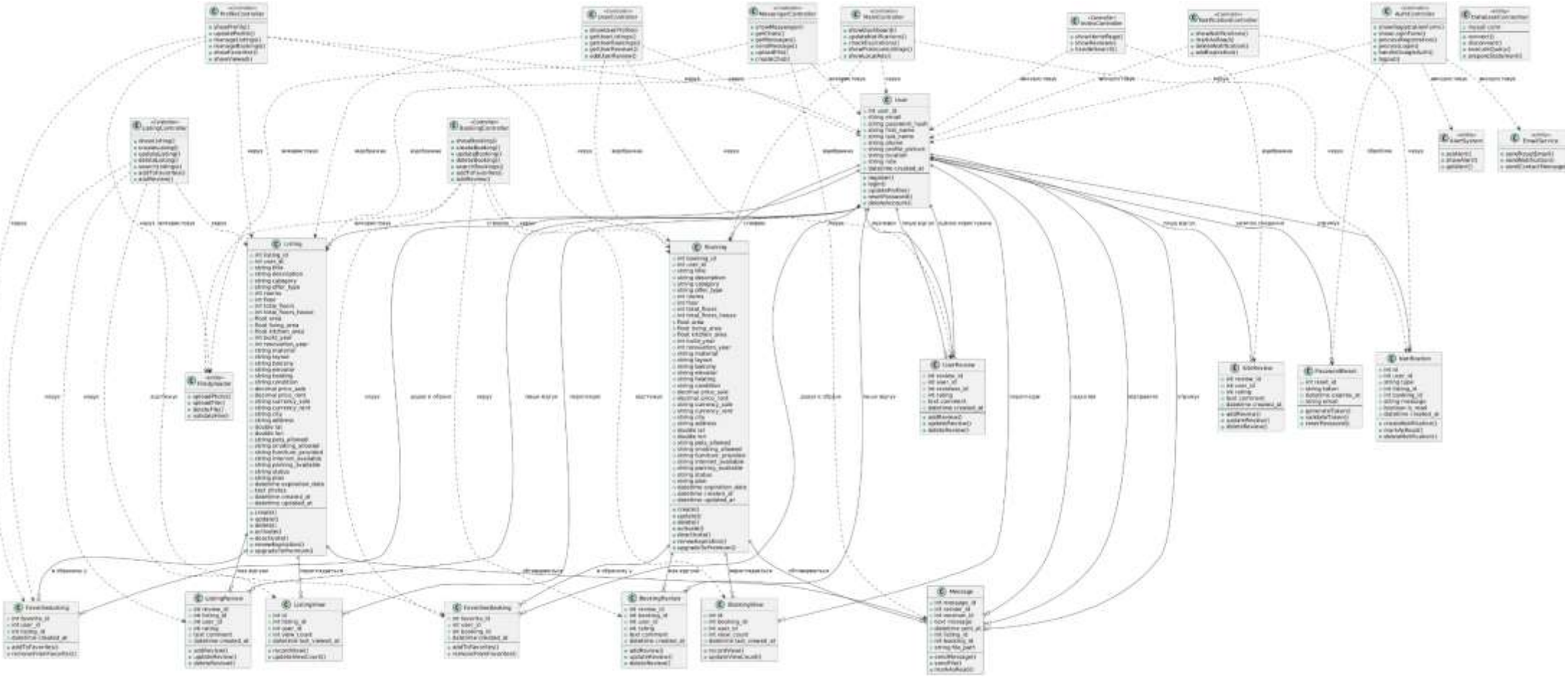
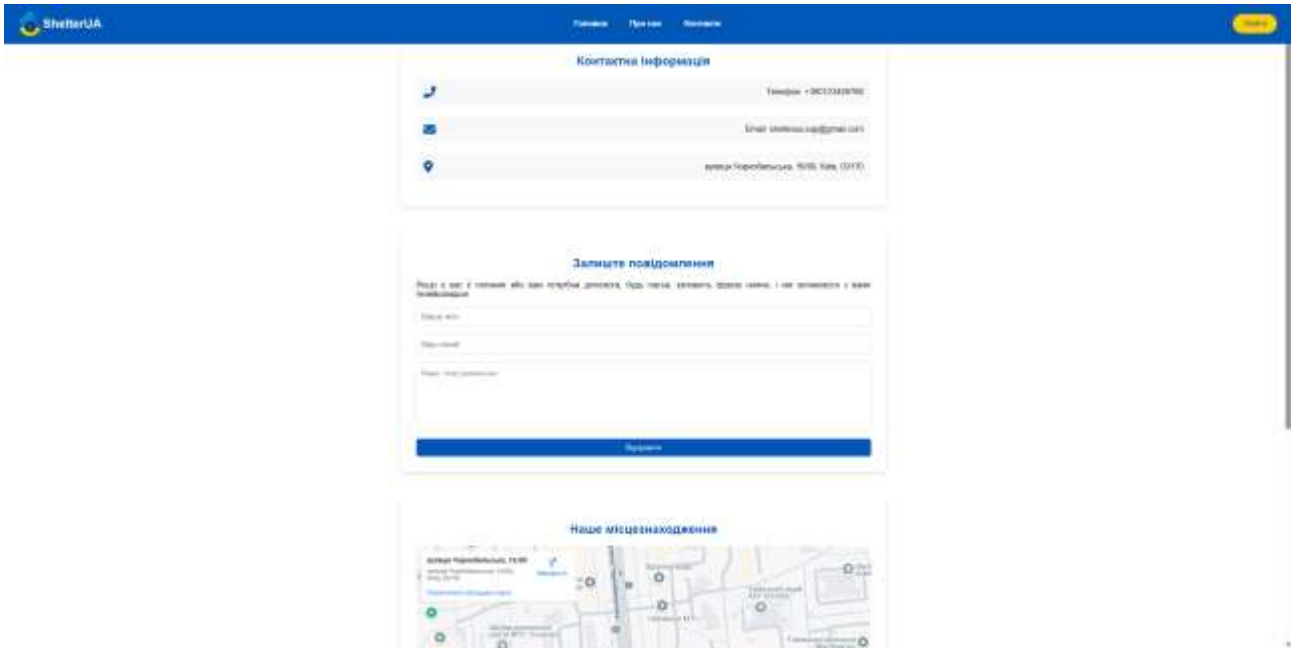


Рисунок Б.1.2 - Діаграма класів вебсайту для пошуку тимчасового житла

Про Нас

Контакти



Послуги

Політика конфіденційності

FAQ

Додаток В

google_callback.php

```
<?php
session_start();
require_once 'db.php';
require_once 'alert.php';

$client_id = '';
$client_secret = '';
$redirect_uri                                                    =
'http://courses.free.nf/action/google_callback.php';

if (!isset($_GET['code'])) {
    setAlert("Від Google не прийшов код авторизації", "error");
    header("Location: /login.php");
    exit;
}

// 1. Отримуємо access_token
$token_url = 'https://oauth2.googleapis.com/token';

$data = [
    'code' => $_GET['code'],
    'client_id' => $client_id,
    'client_secret' => $client_secret,
    'redirect_uri' => $redirect_uri,
    'grant_type' => 'authorization_code'
];

$options = [
    'http' => [
        'header'      =>   "Content-type:   application/x-www-form-
urlencoded",
        'method'  => 'POST',
        'content' => http_build_query($data)
    ]
];

$context  = stream_context_create($options);
$response = file_get_contents($token_url, false, $context);
$result = json_decode($response, true);

if (!isset($result['access_token'])) {
    setAlert("Помилка отримання access_token", "error");
    header("Location: /login.php");
    exit;
}

$access_token = $result['access_token'];

// 2. Отримуємо дані користувача
```

```
$user_info =
file_get_contents('https://www.googleapis.com/oauth2/v1/userinfo?a
ccess_token=' . $access_token);
$user = json_decode($user_info, true);

$email = $user['email'] ?? null;
$first_name = $user['given_name'] ?? null;
$last_name = $user['family_name'] ?? null;
$profile_picture = $user['picture'] ?? null;

// 3. Перевіряємо чи є такий email у БД
$stmt = $conn->prepare("SELECT user_id FROM Users WHERE email = ?");
$stmt->bind_param("s", $email);
$stmt->execute();
$res = $stmt->get_result();

if ($res->num_rows === 0) {
    $stmt = $conn->prepare("INSERT INTO Users (email, password_hash,
first_name, last_name, phone, profile_picture, role, created_at)
                            VALUES (?, NULL, ?, ?, ?, ?, 'user',
NOW())");
    $stmt->bind_param("sssss", $email, $first_name, $last_name,
$phone, $profile_picture);
    $stmt->execute();
    $user_id = $stmt->insert_id;
} else {
    $row = $res->fetch_assoc();
    $user_id = $row['user_id'];
}

// 4. Зберігаємо в сесію
$_SESSION['user_id'] = $user_id;
$_SESSION['user_email'] = $email;
$_SESSION['user_name'] = $first_name;
$_SESSION['user_role'] = 'user';
$_SESSION['profile_picture'] = $profile_picture;

setAlert("Успішний вхід через Google!", "success");
header("Location: /main.php");
exit;
?>
```

Send_message.php

```
<?php
require_once 'alert.php';
session_start();

ini_set('display_errors', 1);
ini_set('display_startup_errors', 1);
error_reporting(E_ALL);

use PHPMailer\PHPMailer\PHPMailer;
use PHPMailer\PHPMailer\Exception;

require 'PHPMailer/Exception.php';
require 'PHPMailer/PHPMailer.php';
require 'PHPMailer/SMTP.php';

$mail = new PHPMailer(true);

try {
    $userEmail = $_POST['email'] ?? '';
    $userName = $_POST['name'] ?? 'Користувач';
    $userMessage = $_POST['message'] ?? '';

    if (!filter_var($userEmail, FILTER_VALIDATE_EMAIL)) {
        throw new Exception('Невірна email адреса');
    }

    $mail->isSMTP();
    $mail->Host       = 'smtp.gmail.com';
    $mail->SMTPAuth   = true;
    $mail->Username   = 'shelterua.sup@gmail.com';
    $mail->Password   = 'jnwy fnwj xhjm wcmc';
    $mail->SMTPSecure = PHPMailer::ENCRYPTION_SMTPS;
    $mail->Port       = 465;

    $mail->CharSet = 'UTF-8';

    $mail->setFrom('shelterua.sup@gmail.com', 'ShelterUA');
    $mail->clearAddresses();
    $mail->clearReplyTos();
    $mail->addAddress('shelterua.sup@gmail.com');
    $mail->addReplyTo($userEmail, $userName);

    $mail->Subject = 'Повідомлення від користувача з сайту ShelterUA';
```

```
                $mail->Body = "Повідомлення від: $userName <$userEmail>\n\nТекст:\n$userMessage";

            $mail->send();

            $mail->clearAddresses();
            $mail->clearReplyTos();
            $mail->addAddress($userEmail, $userName);
            $mail->setFrom('shelterua.sup@gmail.com', 'ShelterUA');
            $mail->Subject = 'Дякуємо за звернення до ShelterUA';
             $mail->Body = "Доброго дня, $userName!\n\nДякуємо, що звернулися до ShelterUA. Ми скоро відповімо на ваше повідомлення.\n\nЗ повагою,\nКоманда ShelterUA";

            $mail->send();

              setAlert('Ваше повідомлення успішно надіслано! Ми скоро з вами зв\'яжемося.', 'success');

            header('Location: ../contact.php');
            exit;
        } catch (Exception $e) {
              setAlert('Сталася помилка при відправці листа: ' . $mail->ErrorInfo, 'error');
            header('Location: ../contact.php');
            exit;
        }
```

Submit_rewiev

```
<?php
session_start();
require_once 'db.php';
require_once 'alert.php';

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $user_id = $_SESSION['user_id'] ?? null;
    $rating = isset($_POST['rating']) ? (int)$_POST['rating'] : 0;
    $comment = trim($_POST['comment'] ?? '');

    if (!$user_id) {
        setAlert('Користувач не авторизований.', 'error');
        exit;
    }

    if ($rating < 1 || $rating > 5) {
        setAlert('Оберіть рейтинг від 1 до 5.', 'error');
        exit;
    }

    if (strlen($comment) < 3) {
        setAlert('Коментар занадто короткий.', 'error');
        exit;
    }

     $checkStmt = $conn->prepare("SELECT review_id FROM SiteReviews WHERE user_id = ?");
    $checkStmt->bind_param('i', $user_id);
    $checkStmt->execute();
    $checkStmt->store_result();

    if ($checkStmt->num_rows > 0) {
        setAlert('Ви вже залишили відгук. Дякуємо!', 'error');
        $checkStmt->close();
        exit;
    }
    $checkStmt->close();

     $stmt = $conn->prepare("INSERT INTO SiteReviews (user_id, rating, comment, created_at) VALUES (?, ?, ?, NOW())");
    if ($stmt) {
        $stmt->bind_param('iis', $user_id, $rating, $comment);
        if ($stmt->execute()) {
```

```
            setAlert('Дякуємо за ваш відгук!', 'success');
        } else {
            setAlert('Помилка при збереженні відгуку.', 'error');
        }
        $stmt->close();
    } else {
        setAlert('Помилка при підготовці запиту.', 'error');
    }
} else {
    setAlert('Невірний запит.', 'error');
}
?>
```

add_listing.php

```
<?php
require_once 'db.php';
require_once 'alert.php';
session_start();

if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    $user_id = $_SESSION['user_id'];
    // Збираємо дані з форми
    $title = $_POST['title'] ?? null;
    $description = $_POST['description'] ?? null;
    $category = $_POST['category'] ?? null;
    $offer_type = $_POST['offer_type'] ?? null;
    $rooms = $_POST['rooms'] ?? null;
    $floor = $_POST['floor'] ?? null;
    $total_floors = $_POST['total_floors'] ?? null;
    $total_floors_house = $_POST['total_floors_house'] ?? null;
    $area = $_POST['area'] ?? null;
    $living_area = $_POST['living_area'] ?? null;
    $kitchen_area = $_POST['kitchen_area'] ?? null;
    $build_year = $_POST['build_year'] ?? null;
    $renovation_year = $_POST['renovation_year'] ?? null;
    $material = $_POST['material'] ?? null;
    $layout = $_POST['layout'] ?? null;
    $balcony = $_POST['balcony'] ?? null;
    $elevator = $_POST['elevator'] ?? null;
    $heating = $_POST['heating'] ?? null;
    $condition = $_POST['condition'] ?? null;
    $price_sale = $_POST['price_sale'] ?? null;
    $price_rent = $_POST['price_rent'] ?? null;
    $currency_sale = $_POST['currency_sale'] ?? null;
    $currency_rent = $_POST['currency_rent'] ?? null;
    $city = $_POST['city'] ?? null;
    $address = $_POST['address'] ?? null;
    $pets_allowed = $_POST['pets_allowed'] ?? null;
    $smoking_allowed = $_POST['smoking_allowed'] ?? null;
    $furniture_provided = $_POST['furniture_provided'] ?? null;
    $internet_available = $_POST['internet_available'] ?? null;
    $parking_available = $_POST['parking_available'] ?? null;

    $lat = $_POST['lat'] ?? null;
    $lon = $_POST['lon'] ?? null;

    $created_at = date('Y-m-d H:i:s');
    $updated_at = date('Y-m-d H:i:s');
    $status = 'active';
    $plan = 'standart';
    $expiration_date = $_POST['expiration_date'] ?? date('Y-m-d
    H:i:s', strtotime('+1 week'));

    $sql = "INSERT INTO Listings (
```

```
    user_id, title, description, category, offer_type, rooms,
floor, total_floors, total_floors_house,
    area, living_area, kitchen_area, build_year,
renovation_year, material, layout,
    balcony, elevator, heating, `condition`, price_sale,
price_rent,
    currency_sale, currency_rent, city, address, lat, lon,
    pets_allowed, smoking_allowed, furniture_provided,
internet_available, parking_available,
    created_at, updated_at, status, plan, expiration_date,
photos
) VALUES (
    ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?
)";

$stmt = $conn->prepare($sql);
// поки photos порожнє
$emptyPhotos = '';
$stmt->bind_param(
    "issssiiiidddiissssssddssssddsssssssssss",
    $user_id, $title, $description, $category, $offer_type,
$rooms, $floor, $total_floors, $total_floors_house,
    $area, $living_area, $kitchen_area, $build_year,
$renovation_year, $material, $layout,
    $balcony, $elevator, $heating, $condition, $price_sale,
$price_rent,
    $currency_sale, $currency_rent, $city, $address, $lat,
$lon,
    $pets_allowed, $smoking_allowed, $furniture_provided,
$internet_available, $parking_available,
    $created_at, $updated_at, $status, $plan,
$expiration_date, $emptyPhotos
);

if (!$stmt->execute()) {
    $stmt->close();
    setAlert('Помилка при додаванні оголошення', 'error');
    header("Location: ../profile.php?tab=announcements");
    exit;
}

$listing_id = $conn->insert_id;
$stmt->close();

$msg = 'Ваше оголошення успішно додано.';
$now = date('Y-m-d H:i:s');
$notif = $conn->prepare("INSERT INTO Notifications (user_id,
type, listing_id, message, created_at) VALUES (?, 'added', ?,
?, ?)");
$notif->bind_param('iiss', $user_id, $listing_id, $msg, $now);
$notif->execute();
$notif->close();
```

```
    $stmt = $conn->prepare("UPDATE Users SET tokens = tokens + 1
    WHERE user_id = ?");
    $stmt->bind_param("i", $user_id);
    $stmt->execute();
    $stmt->close();

    $photoPaths = [];
    if (!empty($_FILES['photos']['name'][0])) {
        $photos = $_FILES['photos'];
        $targetDir = __DIR__ .
    "/../data/user_{$user_id}/listing/listing_{$listing_id}/";
        if (!is_dir($targetDir)) {
            mkdir($targetDir, 0777, true);
        }
        foreach ($photos['name'] as $index => $photo) {
            $ext = pathinfo($photo, PATHINFO_EXTENSION);
            $filename = 'photo_' . ($index + 1) . '.' . $ext;
            $targetFile = $targetDir . $filename;
            if (move_uploaded_file($photos['tmp_name'][$index],
    $targetFile)) {
                $photoPaths[] =
    "data/user_{$user_id}/listing/listing_{$listing_id}/" .
    $filename;
            }
        }
    }

    $photosList = !empty($photoPaths) ? implode(',', $photoPaths)
    : 'images/object.jpg';

    $updateSql = "UPDATE Listings SET photos = ? WHERE listing_id
    = ?";
    $updateStmt = $conn->prepare($updateSql);
    $updateStmt->bind_param('si', $photosList, $listing_id);
    $updateStmt->execute();
    $updateStmt->close();

    setAlert('Оголошення успішно додано', 'success');
    header("Location: ../profile.php?tab=announcements");
    exit;
}
  ?>
```

add_booking.php

```
<?php
ini_set('display_errors', 1);
error_reporting(E_ALL);

require_once 'db.php';
require_once 'alert.php';
session_start();

if ($_SERVER['REQUEST_METHOD'] !== 'POST') {
    setAlert('Неправильний запит.', 'error');
    header('Location: ../main.php');
    exit;
}

$user_id = $_SESSION['user_id'] ?? null;
if (!$user_id) {
    setAlert('Ви повинні бути залоговані, щоб залишати запити.',
'error');
    header('Location: ../login.php');
    exit;
}

$title               = $_POST['title']               ?? null;
$category            = $_POST['category']            ?? null;
$rooms               = $_POST['rooms']               ?? null;
$floor               = $_POST['floor']               ?? null;
$total_floors        = $_POST['total_floors']        ?? null;
$total_floors_house  = $_POST['total_floors_house']  ?? null;
$offer_type          = $_POST['offer_type']          ?? null;
$price_sale          = $_POST['price_sale']          ?? null;
$currency_sale       = $_POST['currency_sale']       ?? null;
$price_rent          = $_POST['price_rent']          ?? null;
$currency_rent       = $_POST['currency_rent']       ?? null;
$area                = $_POST['area']                ?? null;
$build_year          = $_POST['build_year']          ?? null;
$renovation_year     = $_POST['renovation_year']     ?? null;
$material            = $_POST['material']            ?? null;
$description         = $_POST['description']         ?? null;
$living_area         = $_POST['living_area']         ?? null;
$kitchen_area        = $_POST['kitchen_area']        ?? null;
$layout              = $_POST['layout']              ?? null;
$balcony             = $_POST['balcony']             ?? null;
$elevator            = $_POST['elevator']            ?? null;
$heating             = $_POST['heating']             ?? null;
$condition           = $_POST['condition']           ?? null;
$city                = $_POST['city']                ?? null;
$address             = $_POST['address']             ?? null;
$pets_allowed        = $_POST['pets_allowed']        ?? null;
$smoking_allowed     = $_POST['smoking_allowed']     ?? null;
$furniture_provided  = $_POST['furniture_provided']  ?? null;
$internet_available  = $_POST['internet_available']  ?? null;
$parking_available   = $_POST['parking_available']   ?? null;
```

```
$lat = $_POST['lat'] ?? null;
$lon = $_POST['lon'] ?? null;

$created_at = date('Y-m-d H:i:s');
$updated_at = date('Y-m-d H:i:s');
$status = 'active';
$plan = 'standart';
$expiration_date = $_POST['expiration_date'] ?? date('Y-m-d H:i:s',
strtotime('+1 week'));

$sql = "INSERT INTO Bookings (
    user_id, title, description, category, offer_type, rooms, floor,
total_floors, total_floors_house,
    area, living_area, kitchen_area, build_year, renovation_year,
material, layout,
    balcony, elevator, heating, `condition`, price_sale, price_rent,
    currency_sale, currency_rent, city, address, lat, lon,
    pets_allowed, smoking_allowed, furniture_provided,
internet_available, parking_available,
    created_at, updated_at, status, plan, expiration_date
) VALUES (
    ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?
)";

$stmt = $conn->prepare($sql);
if (!$stmt) {
    setAlert('Помилка підготовки запиту.', 'error');
    header('Location: ../request_booking.php.php');
    exit;
}
$stmt->bind_param(
    "issssiiiidddiissssssddssssddsssssssssss",
    $user_id, $title, $description, $category, $offer_type, $rooms,
$floor, $total_floors, $total_floors_house,
    $area, $living_area, $kitchen_area, $build_year,
$renovation_year, $material, $layout,
    $balcony, $elevator, $heating, $condition, $price_sale,
$price_rent,
    $currency_sale, $currency_rent, $city, $address, $lat, $lon,
    $pets_allowed, $smoking_allowed, $furniture_provided,
$internet_available, $parking_available,
    $created_at, $updated_at, $status, $plan, $expiration_date
);

if ($stmt->execute()) {
    $booking_id = $conn->insert_id;
    $msg = 'Ваш запит на бронювання успішно додано.';
    $now = date('Y-m-d H:i:s');
    $notif = $conn->prepare("INSERT INTO Notifications (user_id,
type, booking_id, message, created_at) VALUES (?, 'added', ?, ?,
?)");
```

```
    $notif->bind_param('iiss', $user_id, $booking_id, $msg, $now);
    $notif->execute();
    $notif->close();
    setAlert('Ваш запит успішно збережено! Ми з вами зв’яжемося.',
'success');
} else {
    setAlert('Не вдалося зберегти запит. Спробуйте пізніше.',
'error');
}
$stmt->close();

header('Location: ../profile.php?tab=bookings');
exit;
```

Search.php

```
                                                    <script
src="https://unpkg.com/leaflet@1.9.3/dist/leaflet.js"
crossorigin=""></script>
        <script>
          var map = L.map('map', {
            center: [50.4501, 30.5234],
            zoom: 13,
            minZoom: 7,
            maxZoom: 18,
            zoomControl: false
          });
          L.tileLayer('https://{s}.basemaps.cartocdn.com/rastertile
s/voyager/{z}/{x}/{y}{r}.png', {
            attribution: '© OpenStreetMap, © CARTO'
          }).addTo(map);

          // Функція завантаження кордонів України через Nominatim
з затемненням за межами України
          async function loadUkraineBorders() {
                                        const          url          =
'https://nominatim.openstreetmap.org/search?format=json&polygon_ge
ojson=1&q=Україна';
            try {
             const response = await fetch(url, { headers: { 'Accept':
'application/json' } });
              const data = await response.json();
              if (data && data.length > 0) {
                const ukraine = data[0];
                if (ukraine.geojson) {
                  const geojsonData = ukraine.geojson;
                  const ukraineBoundary = L.geoJSON(geojsonData, {
                    style: function () {
                           return  {  color:  '#a9a9a9',  weight:  5,
fillOpacity: 0 };
                    }
                  }).addTo(map);


                  const outerRing = [[-180, 90], [180, 90], [180, -
90], [-180, -90], [-180, 90]];
                  let innerRing;
                 // Визначаємо внутрішнє кільце – геометрію України
                  if (geojsonData.type === "Polygon") {
                    innerRing = geojsonData.coordinates[0];
                  } else if (geojsonData.type === "MultiPolygon") {
                    innerRing = geojsonData.coordinates[0][0];
                  }

                  const maskGeoJSON = {
                    "type": "Polygon",
                    "coordinates": [outerRing, innerRing]
```

```
                };

                // Додаємо маску, яка затемнює все, що знаходиться
за межами України
                L.geoJSON(maskGeoJSON, {
                  style: function () {
                    return {
                      fillColor: 'darkgray',
                      fillOpacity: 1,
                      stroke: false
                    };
                  },
                  interactive: false
                }).addTo(map);

                const bounds = ukraineBoundary.getBounds();
                map.setMaxBounds(bounds);
              }
            } else {
              console.error("Не знайдено кордонів України.");
            }
          } catch (error) {
              console.error("Помилка при завантаженні кордонів
України:", error);
          }
        }
        loadUkraineBorders();

        loadUkraineBorders();
        const cityName = <?= json_encode("{$city}, Україна") ?>;
        const cityShort = <?= json_encode($city) ?>;
        // markersData з PHP (додаємо lat/lon)
                  let     markersData     =     <?php     echo
json_encode(array_map(function($l) {
           $offerType = $l['offer_type'] ?? ($l['offerType'] ??
'');
          $price = '';
          if ($offerType === 'sale') {
            $price = $l['price_sale'] ?? '';
          } elseif ($offerType === 'rent') {
            $price = $l['price_rent'] ?? '';
          } else {
            $price = $l['price_rent'] ?? $l['price_sale'] ?? '';
          }
          return [
            'title' => $l['title'],
            'street' => $l['address'] ?? ($l['street'] ?? ''),
            'price' => $price,
            'rooms' => $l['rooms'] ?? '',
            'city' => $l['city'],
            'area' => $l['area'] ?? '',
            'date' => $l['updated_at'] ?? $l['created_at'],
```

```
                    'photo'   =>   ($l['photos']   ?   explode(',',
$l['photos'])[0] : 'images/default.jpg'),
                'listing_id' => $l['listing_id'],
                'views' => $l['views'] ?? 0,
                'likes' => $l['likes'] ?? 0,
                'comments' => $l['comments'] ?? 0,
                'lat' => isset($l['lat']) ? (float)$l['lat'] : null,
                'lon' => isset($l['lon']) ? (float)$l['lon'] : null,
            ];
                },     $listings),     JSON_UNESCAPED_UNICODE     |
JSON_UNESCAPED_SLASHES); ?>;
                    let      bookingsData      =      <?php      echo
json_encode(array_map(function($b) {
             $offerType = $b['offer_type'] ?? ($b['offerType'] ??
'');
            $price = '';
            if ($offerType === 'sale') {
              $price = $b['price_sale'] ?? '';
            } elseif ($offerType === 'rent') {
              $price = $b['price_rent'] ?? '';
            } else {
              $price = $b['price_rent'] ?? $b['price_sale'] ?? '';
            }
            return [
              'title' => $b['title'],
              'street' => $b['address'] ?? ($b['street'] ?? ''),
              'price' => $price,
              'rooms' => $b['rooms'] ?? '',
              'city' => $b['city'],
              'area' => $b['area'] ?? '',
              'date' => $b['updated_at'] ?? $b['created_at'],
              'photo' => 'images/booking.png',
              'booking_id' => $b['booking_id'],
              'views' => $b['views'] ?? 0,
              'likes' => $b['likes'] ?? 0,
              'lat' => isset($b['lat']) ? (float)$b['lat'] : null,
              'lon' => isset($b['lon']) ? (float)$b['lon'] : null,
            ];
                },     $bookings),     JSON_UNESCAPED_UNICODE     |
JSON_UNESCAPED_SLASHES); ?>;

         async function initializeCity() {
          let cityToSearch = cityName;
          let triedFallback = false;
          while (true) {
                                        const          url          =
`https://nominatim.openstreetmap.org/search?format=json&polygon_ge
ojson=1&q=${encodeURIComponent(cityToSearch)}`;
            try {
             const response = await fetch(url, { headers: { 'Accept':
'application/json' } });
              let results = await response.json();
```

```
            // Відфільтровуємо, щоб НЕ показувати річки, озера і
гори
            results = results.filter(r => {
              if (r.type === 'river' || r.type === 'lake' || r.type
=== 'mountain') return false;
                                        if     (r.display_name     &&
r.display_name.match(/\bрічка\b|\briver\b|\bозеро\b|\blake\b|гора|
mountain/i)) return false;
              return true;
            });
           if (results && results.length > 0) {
            const selectedCities = results;
            if (selectedCities.length > 1) {
                                  if    (cityBoundaryLayers.length)
cityBoundaryLayers.forEach(l => map.removeLayer(l));
              cityBoundaryLayers = [];
             if (cityMaskLayer) { map.removeLayer(cityMaskLayer);
cityMaskLayer = null; }
              const bbox = selectedCities[0].boundingbox;
              const south = parseFloat(bbox[0]);
              const north = parseFloat(bbox[1]);
              const west = parseFloat(bbox[2]);
              const east = parseFloat(bbox[3]);
             const bounds = L.latLngBounds([south, west], [north,
east]);
              map.fitBounds(bounds);
              window.selectedCitiesForFocus = selectedCities;
              selectedCities.forEach((cityObj, idx) => {
               if (cityObj.geojson) {
                const color = '#ffcc00';
                                    const    cityBoundaryLayer    =
L.geoJSON(cityObj.geojson, {
                  style: function() {
                   return {
                    color: color,
                    weight: 3,
                    fill: false
                   };
                  }
                }).addTo(map);
                cityBoundaryLayers.push(cityBoundaryLayer);
                cityBoundaryLayer.bindTooltip(
                           `${cityObj.display_name}<br>Населення:
${cityObj.population || '-'}<br>importance: ${cityObj.importance ||
'-'}<br><button onclick=\"focusCity(${idx})\">Обрати</button>`,
                  {direction: 'top', sticky: true}
                );
               }
              });
              // Додаємо модальне вікно для вибору
              showCityChoiceModal(selectedCities);
              break;
            }
```

```
                const city = results[0];
                const bbox = city.boundingbox;
                const south = parseFloat(bbox[0]);
                const north = parseFloat(bbox[1]);
                const west = parseFloat(bbox[2]);
                const east = parseFloat(bbox[3]);
                 const bounds = L.latLngBounds([south, west], [north,
east]);
                map.fitBounds(bounds);
                if (city.geojson) {
                  const geojsonData = city.geojson;
                  const cityBoundaryLayer = L.geoJSON(geojsonData, {
                   style: function() {
                    return {
                      color: '#ffcc00',
                      weight: 3,
                      fill: false
                    };
                   }
                  }).addTo(map);
                  cityBoundaryLayers.push(cityBoundaryLayer);
                   const outerRing = [[-180, 90], [180, 90], [180, -
90], [-180, -90], [-180, 90]];
                  const innerRing = geojsonData.type === "Polygon"
                   ? geojsonData.coordinates[0]
                   : geojsonData.coordinates[0][0];
                  const maskGeoJSON = {
                   "type": "Polygon",
                   "coordinates": [outerRing, innerRing]
                  };
                  cityMaskLayer = L.geoJSON(maskGeoJSON, {
                   style: function() {
                    return {
                      fillColor: 'black',
                      fillOpacity: 0.5,
                      stroke: false
                    };
                   },
                   interactive: false
                  }).addTo(map);
                  cityBoundaryLayer.bringToFront();
                 }
                 break;
                } else if (!triedFallback) {
                 // fallback на Київ
                 showCityNotFoundAlertAndRedirect();
                 return;
                } else {
                 showCityNotFoundAlertAndRedirect();
                 return;
                }
               } catch (error) {
```

```
                console.error("Помилка при отриманні даних міста:",
error);
              // Якщо помилка – fallback на Київ
              if (!triedFallback) {
               cityToSearch = 'Київ, Україна';
               triedFallback = true;
              } else {
               const kyivCenter = [50.4501, 30.5234];
               map.setView(kyivCenter, 13);
               fetch('https://nominatim.openstreetmap.org/search?for
mat=json&polygon_geojson=1&q='     +     encodeURIComponent('Київ,
Україна'), { headers: { 'Accept': 'application/json' } })
                  .then(r => r.json())
                  .then(kyivResults => {
                     if (kyivResults && kyivResults.length > 0 &&
kyivResults[0].geojson) {
                   const geojsonData = kyivResults[0].geojson;
                    const cityBoundaryLayer = L.geoJSON(geojsonData,
{
                     style: function() {
                      return {
                       color: '#ffcc00',
                       weight: 3,
                       fill: false
                      };
                     }
                   }).addTo(map);
                   cityBoundaryLayers.push(cityBoundaryLayer);
                   const outerRing = [[-180, 90], [180, 90], [180, -
90], [-180, -90], [-180, 90]];
                   const innerRing = geojsonData.type === "Polygon"
                     ? geojsonData.coordinates[0]
                     : geojsonData.coordinates[0][0];
                   const maskGeoJSON = {
                     "type": "Polygon",
                     "coordinates": [outerRing, innerRing]
                   };
                   cityMaskLayer = L.geoJSON(maskGeoJSON, {
                     style: function() {
                      return {
                       fillColor: 'black',
                       fillOpacity: 0.5,
                       stroke: false
                      };
                     },
                     interactive: false
                   }).addTo(map);
                   cityBoundaryLayer.bringToFront();
                  }
                     console.log('Fallback to Kyiv: markersData',
markersData);
                     console.log('Fallback to Kyiv: bookingsData',
bookingsData);
```

```
                if (typeof addListingMarkers === 'function') {
                  addListingMarkers(markersData);
                } else {
                 console.warn('addListingMarkers is not defined');
                }
                if (typeof addBookingMarkers === 'function') {
                  addBookingMarkers(bookingsData);
                } else {
                 console.warn('addBookingMarkers is not defined');
                }
               });
              break;
             }
            }
           }
          }
```

```
      const favListings = <?php echo json_encode($fav_listings, JSON_UNESCAPED_UNICODE | JSON_UNESCAPED_SLASHES); ?>;
      const favBookings = <?php echo json_encode($fav_bookings, JSON_UNESCAPED_UNICODE | JSON_UNESCAPED_SLASHES); ?>;

      let markers = [];

      // Функція для обчислення Levenshtein distance (мінімальна кількість змін для перетворення одного рядка в інший)
      function levenshtein(a, b) {
        if (a.length === 0) return b.length;
        if (b.length === 0) return a.length;
        const matrix = [];
        for (let i = 0; i <= b.length; i++) matrix[i] = [i];
        for (let j = 0; j <= a.length; j++) matrix[0][j] = j;
        for (let i = 1; i <= b.length; i++) {
          for (let j = 1; j <= a.length; j++) {
            if (b.charAt(i - 1) === a.charAt(j - 1)) {
              matrix[i][j] = matrix[i - 1][j - 1];
            } else {
              matrix[i][j] = Math.min(
                matrix[i - 1][j - 1] + 1,
                matrix[i][j - 1] + 1,
                matrix[i - 1][j] + 1
              );
            }
          }
        }
        return matrix[b.length][a.length];
      }

      // Функція для fuzzy-пошуку найближчої вулиці серед результатів Nominatim
      function findClosestStreet(queryStreet, candidates) {
        let minDist = Infinity;
        let best = null;
```

```
        candidates.forEach(c => {
            const dist = levenshtein(queryStreet.toLowerCase(),
c.display_name.toLowerCase());
          if (dist < minDist) {
            minDist = dist;
            best = c;
          }
        });
        return {best, minDist};
      }

      // Функція для групування даних за вулицею та знаходження
діапазону ціни та кількості кімнат
      function groupByStreet(data) {
        const grouped = {};

        data.forEach(item => {
          if (!grouped[item.street]) {
            grouped[item.street] = {
                    minPrice: parseFloat(item.price.replace('  $',
'').replace(',', '')),
                    maxPrice: parseFloat(item.price.replace('  $',
'').replace(',', '')),
              minRooms: parseInt(item.rooms, 10),
              maxRooms: parseInt(item.rooms, 10),
              street: item.street,
              flats: []
            };
          }

          // Оновлення діапазону для ціни та кімнат
          grouped[item.street].flats.push(item);
             const price = parseFloat(item.price.replace('  $',
'').replace(',', ''));
          const rooms = parseInt(item.rooms, 10);
                            grouped[item.street].minPrice =
Math.min(grouped[item.street].minPrice, price);
                            grouped[item.street].maxPrice =
Math.max(grouped[item.street].maxPrice, price);
                            grouped[item.street].minRooms =
Math.min(grouped[item.street].minRooms, rooms);
                            grouped[item.street].maxRooms =
Math.max(grouped[item.street].maxRooms, rooms);
        });

        return Object.values(grouped);
      }

      function showObjectDetails(flats) {
          const container = document.getElementById('object-
container');
        const body = container.querySelector('.object-body');
        body.innerHTML = '';
```

```
        if (!Array.isArray(flats)) flats = [flats];
        flats.forEach(data => {
          const views = data.views || 0;
          const likes = data.likes || 0;
          const comments = data.comments || 0;
                                  const            isFav            =
favListings.includes(Number(data.listing_id));
          body.innerHTML += `
            <div class="object-list-item">
              <div class="object-list-photo">
                   <img  src="${data.photo}"  alt="Фото  оголошення"
onerror="this.onerror=null;this.src='images/default.jpg';">
              </div>
              <div class="object-list-details">
                <h2 class="object-list-title">${data.title}</h2>
                          <p><i   class="fas   fa-dollar-sign"></i>
<strong>Ціна:</strong> ${data.price}</p>
                        <p><i   class="fas   fa-map-marker-alt"></i>
<strong>Місто:</strong> ${data.city}</p>
                                <p><i    class="fas    fa-road"></i>
<strong>Вулиця:</strong> ${data.street}</p>
                             <p><i   class="fas   fa-door-open"></i>
<strong>Кімнати:</strong> ${data.rooms}</p>
                               <p><i    class="fas    fa-expand"></i>
<strong>Площа:</strong> ${data.area} м²</p>
                <p><i class="fas fa-calendar-day"></i> <strong>Дата
публікації:</strong> ${data.date}</p>
                <p>
                 <span class='object-eye'><i class="fas fa-eye"></i>
${views}</span> |
                          <span  class='object-like'  data-listing-
id="${data.listing_id}"><i        class="fas        fa-heart"></i>
${likes}</span> |
                     <span class='object-comment'><i class="fas fa-
comment"></i> ${comments}</span>
                </p>
                <div class="object-buttons">
                        <a  href="object.php?id=${data.listing_id}"
class="details-btn">Детальніше</a>
                 <button class="object-list-favorite favorite${isFav
?   '   favorited'   :   ''}"   data-listing-id="${data.listing_id}"
onclick="event.stopPropagation();
toggleFavoriteListing(${data.listing_id}, this)">
                   <i class="${isFav ? 'fas' : 'far'} fa-heart"></i>
                  </button>
                </div>
              </div>
            </div>
          `;
        });
        container.style.display = 'block';
      }
```

```
      document.getElementById('close-object-
btn').addEventListener('click', function () {
        document.getElementById('object-container').style.display =
'none';
      });

      // Функція для динамічного показу/приховування маркерів та їх
підписів залежно від зуму
      function updateLabelVisibility() {
        const zoomLevel = map.getZoom();
        const tooltipThreshold = 14;
        const markerThreshold = 10;

        listingMarkers.forEach(item => {
          if (zoomLevel <= markerThreshold) {
            item.marker.setOpacity(0);
            item.marker.closeTooltip();
          } else {
            item.marker.setOpacity(1);
            if (zoomLevel < tooltipThreshold) {
              item.marker.closeTooltip();
            } else {
              item.marker.openTooltip();
            }
          }
        });
        bookingMarkers.forEach(item => {

          if (zoomLevel <= markerThreshold) {
            item.marker.setOpacity(0);
            item.marker.closeTooltip();
          } else {
            item.marker.setOpacity(1);
            if (zoomLevel < tooltipThreshold) {
              item.marker.closeTooltip();
            } else {
              item.marker.openTooltip();
            }
          }
        });
      }

      // --- Приховати всі тултіпи при старті ---
      function hideAllTooltips() {
        listingMarkers.forEach(item => item.marker.closeTooltip());
        bookingMarkers.forEach(item => item.marker.closeTooltip());
      }
      map.on('zoomend', updateLabelVisibility);

      // Показати модальне вікно для вибору міста/селища
      function showCityChoiceModal(cities) {
        let modal = document.getElementById('city-choice-modal');
        if (!modal) {
```

```
          modal = document.createElement('div');
          modal.id = 'city-choice-modal';
          modal.style.position = 'fixed';
          modal.style.top = '0';
          modal.style.left = '0';
          modal.style.width = '100vw';
          modal.style.height = '100vh';
          modal.style.background = 'rgba(0,0,0,0.45)';
          modal.style.display = 'flex';
          modal.style.alignItems = 'center';
          modal.style.justifyContent = 'center';
          modal.style.zIndex = '9999';
          document.body.appendChild(modal);
        }
                    const              typeTranslations              =
{city:"Місто",town:"Містечко",village:"Село",hamlet:"Хутір",suburb
:"Передмістя",neighbourhood:"Мікрорайон",locality:"Населений
пункт",municipality:"Муніципалітет",borough:"Район
міста",county:"Округ        /        Повіт",state:"Область        /
Штат",region:"Регіон",district:"Район",quarter:"Квартал",city_dist
rict:"Міський                район",administrative:"Адміністративна
одиниця",province:"Провінція",department:"Департамент",state_distr
ict:"Округ
області",island:"Острів",islet:"Острівець",archipelago:"Архіпелаг"
,continent:"Континент",country:"Країна",residential:"Житловий
масив",farm:"Ферма",allotments:"Дачний
масив",isolated_dwelling:"Окрема
садиба",industrial:"Промзона",commercial:"Комерційна
зона",retail:"Торгова
зона",square:"Площа",street:"Вулиця",road:"Дорога",motorway:"Автом
агістраль",trunk:"Головна                    дорога",primary:"Основна
дорога",secondary:"Другорядна              дорога",tertiary:"Третинна
дорога",unclassified:"Не           класифіковано",service:"Службова
дорога",pedestrian:"Пішохідна              зона",footway:"Пішохідна
доріжка",path:"Стежка",cycleway:"Велодоріжка",railway:"Залізниця",
station:"Станція",platform:"Платформа",place:"Місце",forest:"Ліс",
wood:"Лісосмуга",meadow:"Луг",grass:"Трава",park:"Парк",garden:"Са
д",farmyard:"Двір
ферми",basin:"Басейн",reservoir:"Водосховище",lake:"Озеро",river:"
Річка",stream:"Потік",canal:"Канал",water:"Водойма",wetland:"Болот
о",beach:"Пляж",bay:"Затока",peninsula:"Півострів",cape:"Мис",moun
tain:"Гора",hill:"Пагорб",ridge:"Хребет",valley:"Долина",plateau:"
Плато",volcano:"Вулкан",glacier:"Льодовик",cemetery:"Кладовище",ae
rodrome:"Аеродром",airport:"Аеропорт",harbour:"Гавань",port:"Порт"
,marina:"Марина",ferry_terminal:"Поромний термінал"};

        modal.innerHTML = `
                            <div          style="background:#fff;max-
width:480px;width:90vw;padding:32px    24px    24px    24px;border-
radius:12px;box-shadow:0                  4px                  32px
rgba(0,0,0,0.18);position:relative;">
                                  <button        id="close-city-modal"
style="position:absolute;top:10px;right:16px;font-
```

```
size:22px;background:none;border:none;cursor:pointer;">&times;</bu
tton>
             <h2 style="margin-bottom:18px;font-size:22px;">Оберіть
населений пункт</h2>
            <div style="max-height:320px;overflow-y:auto;">
              ${cities.map((c, idx) => `
                 <div style='padding:10px 0;border-bottom:1px solid
#eee;'>
                  <b>${c.display_name}</b><br>
                    ${c.type ? `Тип: ${typeTranslations[c.type] ||
c.type}` : ''}
                  <br>
                  <button class="choose-city-btn" data-idx="${idx}"
style="margin-top:6px;padding:6px 18px;">Обрати</button>
                </div>
              `).join('')}
            </div>
          </div>
        `;
        modal.style.display = 'flex';
  document.getElementById('close-city-modal').onclick = function()
    {
    focusCity(0);
          modal.style.display = 'none';
        };
        modal.querySelectorAll('.choose-city-btn').forEach(btn => {
          btn.onclick = function() {
            const idx = +this.getAttribute('data-idx');
            focusCity(idx);
            modal.style.display = 'none';
          };
        });
      }

      let cityBoundaryLayers = [];
      let cityMaskLayer = null;

      window.focusCity = function(idx) {
        if (!window.selectedCitiesForFocus) return;
        const cityObj = window.selectedCitiesForFocus[idx];
        if (!cityObj || !cityObj.boundingbox) return;
        if (cityBoundaryLayers.length) cityBoundaryLayers.forEach(l
=> map.removeLayer(l));
        cityBoundaryLayers = [];
          if   (cityMaskLayer)   {   map.removeLayer(cityMaskLayer);
cityMaskLayer = null; }
        if (cityObj.geojson) {
          const cityBoundaryLayer = L.geoJSON(cityObj.geojson, {
            style: function() {
              return { color: '#ffcc00', weight: 3, fill: false };
                }
          }).addTo(map);
          cityBoundaryLayers.push(cityBoundaryLayer);
```

```
          const outerRing = [[-180, 90], [180, 90], [180, -90], [-
180, -90], [-180, 90]];
          let innerRing;
          if (cityObj.geojson.type === "Polygon") {
            innerRing = cityObj.geojson.coordinates[0];
          } else if (cityObj.geojson.type === "MultiPolygon") {
            innerRing = cityObj.geojson.coordinates[0][0];
          }
          if (innerRing) {
            const maskGeoJSON = {
              "type": "Polygon",
              "coordinates": [outerRing, innerRing]
            };
            cityMaskLayer = L.geoJSON(maskGeoJSON, {
              style: function() {
                 return { fillColor: 'darkgray', fillOpacity: 0.5,
stroke: false };
              },
              interactive: false
            }).addTo(map);
          }
        }
        const bbox = cityObj.boundingbox;
        const south = parseFloat(bbox[0]);
        const north = parseFloat(bbox[1]);
        const west = parseFloat(bbox[2]);
        const east = parseFloat(bbox[3]);
        const bounds = L.latLngBounds([south, west], [north, east]);
        map.fitBounds(bounds);
      };

      initializeCity().then(() => {
      });
        </script>
        <script>


          function hideAllPopups() {
              const  filterForm  =  document.getElementById("filter-
form");
            const filterContainer = document.getElementById("filter-
container-form");
            if (filterForm) filterForm.style.display = "none";
              if  (filterContainer)  filterContainer.style.display  =
"none";

            const objectContainer = document.getElementById("object-
container");
              if  (objectContainer)  objectContainer.style.display  =
"none";

            // Ховаємо спливаючі вікна з картки (якщо такі є)
```

```
            document.querySelectorAll(".flat-popup").forEach(popup
=> {
              popup.style.display = "none";
            });
          }

                                  document.getElementById("toggle-map-
btn").addEventListener("click", function () {
            const mapDiv = document.getElementById("map");
                               const      objectListContainer      =
document.getElementById("object-list-container");
                              const      bookingListContainer      =
document.getElementById("booking-list-container");
            const footer = document.getElementById("footer");
           const roleSwitcher = document.getElementById("user-role-
switcher");

            hideAllPopups();

            if (mapDiv.style.display === "none") {
              mapDiv.style.display = "block";
              objectListContainer.style.display = "none";
              bookingListContainer.style.display = "none";
              footer.style.display = "none";
              this.innerHTML = '<i class="fas fa-list"></i>';
            } else {
              mapDiv.style.display = "none";
              if (roleSwitcher.value === "buyer") {
                objectListContainer.style.display = "block";
                bookingListContainer.style.display = "none";
               this.innerHTML = '<i class="fa-solid fa-map-location-
dot"></i>';
              } else {
                objectListContainer.style.display = "none";
                bookingListContainer.style.display = "block";
                footer.style.display = "block";
               this.innerHTML = '<i class="fa-solid fa-map-location-
dot"></i>';
              }
            }
          });


      document.addEventListener("DOMContentLoaded", function () {
         const  roleSwitcher  =  document.getElementById("user-role-
switcher");
        const roleIcon = document.getElementById("role-icon");
        const objectListContainer = document.getElementById("object-
list-container");
                  const           bookingListContainer           =
document.getElementById("booking-list-container");
        const mapDiv = document.getElementById("map");
        const footer = document.getElementById("footer");
```

```
        const listTitle = document.getElementById("list-title");
         const  bookingTitle  =  document.getElementById("booking-
title");

        roleSwitcher.addEventListener("change", function () {
          // Оновлюємо заголовки і кнопки
          updateActionButton(this.value);
          if (mapDiv.style.display === "none") {
            if (this.value === "buyer") {
              objectListContainer.style.display = "block";
              bookingListContainer.style.display = "none";
                  if  (listTitle)  listTitle.textContent  =  "Список
оголошень";
            } else {
              objectListContainer.style.display = "none";
              bookingListContainer.style.display = "block";
               if (bookingTitle) bookingTitle.textContent = "Список
бронювань";
            }
          }
          hideAllMarkers();
          if (this.value === 'buyer') {
            addListingMarkers();
          } else {
            addBookingMarkers();
          }
        });
        // --- Додаємо перемикання ролі по кліку на іконку ---
        if (roleIcon && roleSwitcher) {
сторінки
          roleSwitcher.value = "buyer";
          updateActionButton("buyer");
          const mapDiv = document.getElementById("map");
                          const        objectListContainer        =
document.getElementById("object-list-container");
                         const        bookingListContainer        =
document.getElementById("booking-list-container");
          if (mapDiv && objectListContainer && bookingListContainer
&& mapDiv.style.display === "none") {
            objectListContainer.style.display = "block";
            bookingListContainer.style.display = "none";
          }

          roleIcon.addEventListener("click", function () {
            roleSwitcher.value = (roleSwitcher.value === "buyer") ?
"seller" : "buyer";
             // Викликаємо подію change вручну, щоб спрацювала вся
логіка
            roleSwitcher.dispatchEvent(new Event("change"));
          });
        }
      });
```

```
      function updateActionButton(role) {
         const actionButton = document.getElementById("action-
button");
        if (!actionButton) return;
        if (role === "buyer") {
              actionButton.innerHTML = '<i class="fas fa-
lock"></i><span> Збр. оголошення</span>';
          actionButton.href = "request_booking.php";
        } else if (role === "seller") {
            actionButton.innerHTML = '<i class="fas fa-plus-
circle"></i><span> Ств. оголошення</span>';
          actionButton.href = "create_listing.php";
        }
      }

       // --- МАРКЕРИ ОГОЛОШЕНЬ І БРОНЮВАНЬ ---
      let listingMarkers = [];
      let bookingMarkers = [];

      // Додає маркери оголошень МОМЕНТАЛЬНО по координатах
      async function addListingMarkers() {
        listingMarkers.forEach(m => map.removeLayer(m.marker));
        listingMarkers = [];
  const grouped = {};
        const nullCoords = [];
        for (const item of markersData) {
          if (item.lat && item.lon) {
            const key = item.lat + ',' + item.lon;
            if (!grouped[key]) {
              grouped[key] = {
                lat: item.lat,
                lon: item.lon,
                flats: [],
                    minPrice: parseFloat(item.price.replace(' $',
'').replace(',', '')),
                    maxPrice: parseFloat(item.price.replace(' $',
'').replace(',', '')),
                minRooms: parseInt(item.rooms, 10),
                maxRooms: parseInt(item.rooms, 10),
                street: item.street
              };
            }
            grouped[key].flats.push(item);
               const price = parseFloat(item.price.replace(' $',
'').replace(',', ''));
            const rooms = parseInt(item.rooms, 10);
            grouped[key].minPrice = Math.min(grouped[key].minPrice,
price);
            grouped[key].maxPrice = Math.max(grouped[key].maxPrice,
price);
            grouped[key].minRooms = Math.min(grouped[key].minRooms,
rooms);
```

```
            grouped[key].maxRooms = Math.max(grouped[key].maxRooms,
rooms);
          } else {
            nullCoords.push(item);
          }
        }
        Object.values(grouped).forEach(group => {
                  const     marker     =     L.marker([group.lat,
group.lon]).addTo(map);
                       marker.on('click',        function()        {
showObjectDetails(group.flats); });
          let priceText, roomsText;
          if (group.flats.length === 1) {
            priceText = group.flats[0].price + ' $';
            roomsText = group.flats[0].rooms + ' кімнати';
          } else {
            priceText = `${group.minPrice} $ - ${group.maxPrice} $`;
               roomsText  =  `${group.minRooms}  -  ${group.maxRooms}
кімнати`;
          }
          marker.bindTooltip(
            `${group.street}<br>${priceText}<br>${roomsText}`,
            { permanent: true, direction: "top", classNname: "marker-
label" }
          );
          listingMarkers.push({ marker });
        });
        for (const item of nullCoords) {
          if (!item.street || !item.city) continue;
          const query = `${item.street}, ${item.city}, Україна`;
                                  const             url             =
`https://nominatim.openstreetmap.org/search?format=json&limit=1&q=
${encodeURIComponent(query)}`;
          try {
            const response = await fetch(url, { headers: { 'Accept':
'application/json' } });
            const results = await response.json();
            if (results && results.length > 0) {
              const lat = parseFloat(results[0].lat);
              const lon = parseFloat(results[0].lon);
              // Оновити координати в БД через AJAX
              fetch('action/update_listing_coords.php', {
                method: 'POST',
                headers: { 'Content-Type': 'application/x-www-form-
urlencoded' },
                                                                  body:
`listing_id=${encodeURIComponent(item.listing_id)}&lat=${lat}&lon=
${lon}`
              });
              // Додати маркер на карту
              const marker = L.marker([lat, lon]).addTo(map);
                              marker.on('click',     function()     {
showObjectDetails([item]); });
```

```
            marker.bindTooltip(
              `${item.street}<br>${item.price} $<br>${item.rooms}
кімнати`,
                { permanent: true, direction: "top", className:
"marker-label" }
            );
            listingMarkers.push({ marker });
            item.lat = lat;
            item.lon = lon;
          }
        } catch (e) { /* ігнорувати */ }
      }
      hideAllTooltips();
    }

    async function addBookingMarkers() {
      bookingMarkers.forEach(m => map.removeLayer(m.marker));
      bookingMarkers = [];
      // Групуємо по координатах (lat/lon)
      const grouped = {};
      const nullCoords = [];
      for (const item of bookingsData) {
        if (item.lat && item.lon) {
          const key = item.lat + ',' + item.lon;
          if (!grouped[key]) {
            grouped[key] = {
              lat: item.lat,
              lon: item.lon,
              flats: [],
                  minPrice:  parseFloat(item.price.replace('  $',
'').replace(',', '')),
                  maxPrice:  parseFloat(item.price.replace('  $',
'').replace(',', '')),
              minRooms: parseInt(item.rooms, 10),
              maxRooms: parseInt(item.rooms, 10),
              street: item.street
            };
          }
          grouped[key].flats.push(item);
             const  price  =  parseFloat(item.price.replace('  $',
'').replace(',', ''));
          const rooms = parseInt(item.rooms, 10);
          grouped[key].minPrice = Math.min(grouped[key].minPrice,
price);
          grouped[key].maxPrice = Math.max(grouped[key].maxPrice,
price);
          grouped[key].minRooms = Math.min(grouped[key].minRooms,
rooms);
          grouped[key].maxRooms = Math.max(grouped[key].maxRooms,
rooms);
        } else {
          nullCoords.push(item);
        }
```

```
        }
        // Додаємо маркери для тих, у кого є координати
        Object.values(grouped).forEach(group => {
          const marker = L.marker([group.lat, group.lon], { icon:
L.icon({ iconUrl: 'images/booking.png', iconSize: [38, 38] })
}).addTo(map);
          marker.on('click', function() {
            // Якщо один об'єкт – показати деталі, якщо декілька –
список
            if (group.flats.length === 1) {
              showBookingDetails(group.flats[0]);
            } else {
              showBookingDetailsList(group.flats);
            }
          });
          let priceText, roomsText;
          if (group.flats.length === 1) {
            priceText = group.flats[0].price + ' $';
            roomsText = group.flats[0].rooms + ' кімнати';
          } else {
            priceText = `${group.minPrice} $ - ${group.maxPrice} $`;
              roomsText = `${group.minRooms} - ${group.maxRooms}
кімнати`;
          }
          marker.bindTooltip(
            `${group.street}<br>${priceText}<br>${roomsText}`,
            { permanent: true, direction: "top", className: "marker-
label" }
          );
          bookingMarkers.push({ marker });
        });
        for (const item of nullCoords) {
          if (!item.street || !item.city) continue;
          const query = `${item.street}, ${item.city}, Україна`;
                                const            url            =
`https://nominatim.openstreetmap.org/search?format=json&limit=1&q=
${encodeURIComponent(query)}`;
          try {
            const response = await fetch(url, { headers: { 'Accept':
'application/json' } });
            const results = await response.json();
            if (results && results.length > 0) {
              const lat = parseFloat(results[0].lat);
              const lon = parseFloat(results[0].lon);
              // Оновити координати в БД через AJAX
              fetch('action/update_booking_coords.php', {
                method: 'POST',
                headers: { 'Content-Type': 'application/x-www-form-
urlencoded' },
                                                                body:
`booking_id=${encodeURIComponent(item.booking_id)}&lat=${lat}&lon=
${lon}`
              });
```

```
            // Додати маркер на карту
             const marker = L.marker([lat, lon], { icon: L.icon({
iconUrl: 'images/booking.png', iconSize: [38, 38] }) }).addTo(map);
                          marker.on('click',    function()    {
showBookingDetails(item); });
            marker.bindTooltip(
              `${item.street}<br>${item.price} $<br>${item.rooms}
кімнати`,
                { permanent: true, direction: "top", className:
"marker-label" }
            );
            bookingMarkers.push({ marker });
            item.lat = lat;
            item.lon = lon;
          }
        } catch (e) { /* ігнорувати */ }
      }
      hideAllTooltips();
    }

    function hideAllMarkers() {
      listingMarkers.forEach(m => map.removeLayer(m.marker));
      bookingMarkers.forEach(m => map.removeLayer(m.marker));
      listingMarkers = [];
      bookingMarkers = [];
    }

    // --- ПЕРЕМИКАННЯ ЗА РОЛЛЮ ---
    document.addEventListener('DOMContentLoaded', function () {
       const  roleSwitcher  =  document.getElementById('user-role-
switcher');
      hideAllMarkers();
      if (roleSwitcher.value === 'buyer') {
        addListingMarkers();
      } else {
        addBookingMarkers();
      }
      roleSwitcher.addEventListener('change', function () {
        hideAllMarkers();
        if (this.value === 'buyer') {
          addListingMarkers();
        } else {
          addBookingMarkers();
        }
      });
    });

    // Перемикання видимості панелей сортування залежно від ролі
    (function() {
       const  roleSwitcher  =  document.getElementById('user-role-
switcher');
       const  sortListingsPanel  =  document.getElementById('sort-
listings-panel');
```

```
        const sortBookingsPanel = document.getElementById('sort-
bookings-panel');
        function updateSortPanelsByRole() {
              if (!roleSwitcher || !sortListingsPanel ||
!sortBookingsPanel) return;
          if (roleSwitcher.value === 'seller') {
            sortListingsPanel.style.display = 'none';
            sortBookingsPanel.style.display = '';
          } else {
            sortListingsPanel.style.display = '';
            sortBookingsPanel.style.display = 'none';
          }
        }
        if (roleSwitcher) {
                              roleSwitcher.addEventListener('change',
updateSortPanelsByRole);
                        document.addEventListener('DOMContentLoaded',
updateSortPanelsByRole);
        }
      })();

      // --- ВІКНО ДЕТАЛЕЙ БРОНЮВАННЯ ---
      function showBookingDetails(booking) {
          const container = document.getElementById('object-
container');
        const body = container.querySelector('.object-body');
         container.querySelector('.title').textContent = 'Список
бронювань';
                          const                  isFav                =
favBookings.includes(Number(booking.booking_id));
        body.innerHTML = `
          <div class="object-list-item">
            <div class="object-list-photo">
              <img src="images/booking.png" alt="Фото бронювання">
            </div>
            <div class="object-list-details">
              <h2 class="object-list-title">${booking.title}</h2>
                        <p><i class="fas fa-dollar-sign"></i>
<strong>Ціна:</strong> ${booking.price}</p>
                       <p><i class="fas fa-map-marker-alt"></i>
<strong>Місто:</strong> ${booking.city}</p>
                              <p><i class="fas fa-road"></i>
<strong>Вулиця:</strong> ${booking.street}</p>
                            <p><i class="fas fa-door-open"></i>
<strong>Кімнати:</strong> ${booking.rooms}</p>
                              <p><i class="fas fa-expand"></i>
<strong>Площа:</strong> ${booking.area} м²</p>
               <p><i class="fas fa-calendar-day"></i> <strong>Дата
публікації:</strong> ${booking.date}</p>
              <p>
                <span class='object-eye'><i class="fas fa-eye"></i>
${booking.views}</span> |
```

```
                        <span class='object-like' data-booking-
id="${booking.booking_id}"><i class="fas fa-heart"></i>
${booking.likes}</span>
            </p>
            <div class="object-buttons">
                 <a href="booking.php?id=${booking.booking_id}"
class="details-btn">Детальніше</a>
              <button class="object-list-favorite favorite${isFav
? ' favorited' : ''}" data-booking-id="${booking.booking_id}"
onclick="event.stopPropagation();
toggleFavoriteBooking(${booking.booking_id}, this)">
                <i class="${isFav ? 'fas' : 'far'} fa-heart"></i>
              </button>
            </div>
          </div>
        </div>
      `;
      container.style.display = 'block';
    }

    // --- ВІКНО ДЕТАЛЕЙ СПИСКУ БРОНЮВАНЬ ---
    function showBookingDetailsList(bookings) {
        const container = document.getElementById('object-
container');
      const body = container.querySelector('.object-body');
       container.querySelector('.title').textContent = 'Список
бронювань';
      body.innerHTML = '';
      bookings.forEach(function(booking) {
                                const isFav =
favBookings.includes(Number(booking.booking_id));
        body.innerHTML += `
          <div class="object-list-item">
            <div class="object-list-photo">
             <img src="images/booking.png" alt="Фото бронювання">
            </div>
            <div class="object-list-details">
              <h2 class="object-list-title">${booking.title}</h2>
                        <p><i class="fas fa-dollar-sign"></i>
<strong>Ціна:</strong> ${booking.price}</p>
                      <p><i class="fas fa-map-marker-alt"></i>
<strong>Місто:</strong> ${booking.city}</p>
                             <p><i class="fas fa-road"></i>
<strong>Вулиця:</strong> ${booking.street}</p>
                          <p><i class="fas fa-door-open"></i>
<strong>Кімнати:</strong> ${booking.rooms}</p>
                            <p><i class="fas fa-expand"></i>
<strong>Площа:</strong> ${booking.area} м²</p>
              <p><i class="fas fa-calendar-day"></i> <strong>Дата
публікації:</strong> ${booking.date}</p>
              <p>
               <span class='object-eye'><i class="fas fa-eye"></i>
${booking.views}</span> |
```

```
                        <span class='object-like' data-booking-
id="${booking.booking_id}"><i class="fas fa-heart"></i>
${booking.likes}</span>
                </p>
                <div class="object-buttons">
                    <a href="booking.php?id=${booking.booking_id}"
class="details-btn">Детальніше</a>
                 <button class="object-list-favorite favorite${isFav
? ' favorited' : ''}" data-booking-id="${booking.booking_id}"
onclick="event.stopPropagation();
toggleFavoriteBooking(${booking.booking_id}, this)">
                   <i class="${isFav ? 'fas' : 'far'} fa-heart"></i>
                  </button>
                </div>
              </div>
            </div>
          `;
        });
        container.style.display = 'block';
}
      function toggleFavoriteBooking(bookingId, btn) {
        const isFav = favBookings.includes(bookingId);
        const url = isFav ? 'action/remove_favorite_booking.php' :
'action/add_favorite_booking.php';
        fetch(url, {
          method: 'POST',
            headers: { 'Content-Type': 'application/x-www-form-
urlencoded' },
          body: 'booking_id=' + encodeURIComponent(bookingId)
        })
          .then(res => res.text())
          .then(data => {
            if (data.trim() === 'success') {
              // Оновити favBookings у JS
              if (isFav) {
                const idx = favBookings.indexOf(bookingId);
                if (idx !== -1) favBookings.splice(idx, 1);
              } else {
                favBookings.push(bookingId);
              }
                  document.querySelectorAll('button.favorite[data-
booking-id="' + bookingId + '"], button.object-list-favorite[data-
booking-id="' + bookingId + '"]').forEach(function(button) {
                const icon = button.querySelector('i');
                button.classList.toggle('favorited');
                if (button.classList.contains('favorited')) {
                  icon.classList.remove('far');
                  icon.classList.add('fas');
                } else {
                  icon.classList.remove('fas');
                  icon.classList.add('far');
                }
              });
```

```
            // Динамічно оновити кількість лайків
               fetch('action/get_booking_likes.php?booking_id=' +
encodeURIComponent(bookingId))
              .then(res => res.json())
              .then(obj => {
                if (obj && typeof obj.likes !== 'undefined') {
                    document.querySelectorAll('.object-like[data-
booking-id]')
                    .forEach(function(span) {
                      if (Number(span.getAttribute('data-booking-
id')) === bookingId) {
                                span.innerHTML = '<i class="fas fa-
heart"></i> ' + obj.likes;
                      }
                    });
                }
              });
          } else {
             alert(isFav ? 'Не вдалося видалити з обраних' : 'Не
вдалося додати в обране');
          }
        });
    }

    function toggleFavoriteListing(listingId, btn) {
      const isFav = favListings.includes(listingId);
      const url = isFav ? 'action/remove_favorite_listing.php' :
'action/add_favorite_listing.php';
      fetch(url, {
        method: 'POST',
          headers: { 'Content-Type': 'application/x-www-form-
urlencoded' },
        body: 'listing_id=' + encodeURIComponent(listingId)
      })
        .then(res => res.text())
        .then(data => {
          if (data.trim() === 'success') {
            // Оновити favListings у JS
            if (isFav) {
              const idx = favListings.indexOf(listingId);
              if (idx !== -1) favListings.splice(idx, 1);
            } else {
              favListings.push(listingId);
            }
                document.querySelectorAll('button.favorite[data-
listing-id="' + listingId + '"], button.object-list-favorite[data-
listing-id="' + listingId + '"]').forEach(function(button) {
              const icon = button.querySelector('i');
              button.classList.toggle('favorited');
              if (button.classList.contains('favorited')) {
                icon.classList.remove('far');
                icon.classList.add('fas');
              } else {
```

```
                icon.classList.remove('fas');
                icon.classList.add('far');
              }
            });

            // Динамічно оновити кількість лайків
              fetch('action/get_listing_likes.php?listing_id=' +
encodeURIComponent(listingId))
              .then(res => res.json())
              .then(obj => {
                if (obj && typeof obj.likes !== 'undefined') {

                    document.querySelectorAll('.object-like[data-
listing-id]')
                    .forEach(function(span) {
                      if (Number(span.getAttribute('data-listing-
id')) === listingId) {
                                span.innerHTML = '<i class="fas fa-
heart"></i> ' + obj.likes;
                      }
                    });
                }
              });
          } else {
             alert(isFav ? 'Не вдалося видалити з обраних' : 'Не
вдалося додати в обране');
          }
        });
    }

    // --- Кастомне повідомлення про не знайдене місто та редирект
---
    function showCityNotFoundAlertAndRedirect() {
       var alertDiv = document.getElementById('city-not-found-
alert');
      if (alertDiv) {
        alertDiv.style.display = 'block';
        setTimeout(function() {
          window.location.href = 'search.php?city=Київ';
        }, 1000);
      } else {
        // fallback на alert, якщо div не знайдено
        alert('Місто не знайдено.');
        window.location.href = 'search.php?city=Київ';
      }
    }
      </script>
      <!-- Повідомлення про не знайдене місто -->
    <div                                id="city-not-found-alert"
style="display:none;position:fixed;top:30px;left:50%;transform:tra
nslateX(-50%);width:340px;z-
index:99999;background:#ffeb3b;color:#222;text-
```

```
align:center;padding:12px    0;font-size:18px;font-weight:bold;box-
shadow:0 2px 8px rgba(0,0,0,0.12);border-radius:8px;">
        Місто не знайдено, показуємо Київ
      </div>

      <script>
      function       addSorting(selectId,       containerSelector,
itemSelector, filterSelectId) {
        const select = document.getElementById(selectId);
        const container = document.querySelector(containerSelector);
            const     filterSelect     =     filterSelectId     ?
document.getElementById(filterSelectId) : null;
        if (!select || !container) return;
                      const                initial                =
Array.from(container.querySelectorAll(itemSelector));
        function applyFilterAndSort() {
          const sortVal = select.value;
           const  filterVal  =  filterSelect  ?  filterSelect.value  :
'all';
                                  const            items            =
Array.from(container.querySelectorAll(itemSelector));
          // Фільтрація
          items.forEach(i => {
             if  (filterVal  ===  'all'  ||  (i.dataset.offertype  ===
filterVal)) {
              i.style.display = '';
            } else {
              i.style.display = 'none';
            }
          });
           const  visible  =  items.filter(i  =>  i.style.display  !==
'none');
            const  hidden  =  items.filter(i  =>  i.style.display  ===
'none');
          if (sortVal === 'default') {
            items.forEach(i => container.removeChild(i));
            initial.forEach(i => container.appendChild(i));
            // Після reset – ще раз застосувати фільтр
            initial.forEach(i => {
               if (filterVal === 'all' || (i.dataset.offertype ===
filterVal)) {
                i.style.display = '';
              } else {
                i.style.display = 'none';
              }
            });
            return;
          }
          visible.sort((a, b) => {
           if (sortVal === 'price_asc') return (+a.dataset.price||0)
- (+b.dataset.price||0);
                    if    (sortVal    ===    'price_desc')    return
(+b.dataset.price||0) - (+a.dataset.price||0);
```

```
                    if    (sortVal    ===    'date_new')    return
(b.dataset.date||'').localeCompare(a.dataset.date||'');
                    if    (sortVal    ===    'date_old')    return
(a.dataset.date||'').localeCompare(b.dataset.date||'');
          if (sortVal === 'rooms_asc') return (+a.dataset.rooms||0)
- (+b.dataset.rooms||0);
                   if    (sortVal    ===    'rooms_desc')    return
(+b.dataset.rooms||0) - (+a.dataset.rooms||0);
           if (sortVal === 'area_asc') return (+a.dataset.area||0)
- (+b.dataset.area||0);
          if (sortVal === 'area_desc') return (+b.dataset.area||0)
- (+a.dataset.area||0);
                   if    (sortVal    ===    'views_desc')    return
(+b.dataset.views||0) - (+a.dataset.views||0);
           return 0;
         });
         items.forEach(i => container.removeChild(i));
         visible.forEach(i => container.appendChild(i));
         hidden.forEach(i => container.appendChild(i));
       }
       select.addEventListener('change', applyFilterAndSort);
        if  (filterSelect)  filterSelect.addEventListener('change',
applyFilterAndSort);
     }
     addSorting('sort-listings',  '#object-list-container  .object-
list-content', '.object-list-item', 'filter-offertype-listings');
     addSorting('sort-bookings', '#booking-list-container .object-
list-content', '.object-list-item', 'filter-offertype-bookings');
```

messenger.php

```
<script>
    document.addEventListener("DOMContentLoaded", function () {
      const roleSwitcher = document.getElementById("user-role-
switcher");
      const actionButton = document.getElementById("action-
button");
      const roleIcon = document.getElementById("role-icon");
      const chatList = document.getElementById('chat-list');
      const toggleButtons = document.querySelectorAll('.users-
toggle button[data-filter]');
      const mobileSwitch = document.getElementById('mobile-
switch');
      const mobileSwitchLabel = document.getElementById('mobile-
switch-label');
      let currentFilter = 'initiator';
      let selectedUserId = null;
      let selectedListingId = null;
      let selectedBookingId = null;
      let selectedUserName = '';
      let selectedUserPhoto = '';
      let selectedUserPhone = '';
      let currentPage = 0;
      let isLoading = false;
      let messagesPollingInterval = null;
      let lastMessagesHash = '';

      function startMessagesPolling() {
        stopMessagesPolling();
        if (!selectedUserId) return;
        messagesPollingInterval = setInterval(() => {
          if (selectedUserId) {
            fetch('action/get_messages.php?user_id=' +
encodeURIComponent(selectedUserId) + (selectedListingId ?
'&listing_id=' + encodeURIComponent(selectedListingId) : '') +
(selectedBookingId ? '&booking_id=' +
encodeURIComponent(selectedBookingId) : ''))
              .then(r => r.json())
              .then(data => {
                const newHash = JSON.stringify(data.messages);
                if (newHash !== lastMessagesHash) {
                  renderMessages(data.messages || []);
                  lastMessagesHash = newHash;
                }
              });
          }
        }, 2000);
      }
      function stopMessagesPolling() {
        if (messagesPollingInterval) {
          clearInterval(messagesPollingInterval);
          messagesPollingInterval = null;
```

```
        }
      }
      window.addEventListener('beforeunload',
stopMessagesPolling);

      function renderChats(chats) {
        const spinner = chatList.querySelector('.spinner');
        if (spinner) spinner.remove();
        chatList.innerHTML = '';
        const groupedChats = { initiator: [], receiver: [] };
        if (chats && Array.isArray(chats)) {
          chats.forEach(chat => {
            let tab = 'initiator';
            if (chat.first_real_sender_id &&
chat.first_real_sender_id != window.currentUserId) tab =
'receiver';
            groupedChats[tab].push(chat);
          });
        }
        const visibleChats = groupedChats[currentFilter] || [];
        if (!visibleChats.length) {
          chatList.innerHTML = '<li style="text-
align:center;color:#888;padding:20px;">Чатів не знайдено</li>';
          document.getElementById('messages').innerHTML = '';
          renderListingCard();
          return;
        }
        visibleChats.forEach(chat => {
          const li = document.createElement('li');
          li.dataset.userId = chat.other_id; // завжди id
співрозмовника
          li.dataset.listingId = chat.listing_id || '';
          li.dataset.bookingId = chat.booking_id || '';
          let photo = chat.profile_picture && chat.profile_picture
!== '' ? chat.profile_picture : 'images/photo.jpg';
          // if (chat.booking_id) {
          //   photo = 'images/booking.png';
          // }
          const name = chat.first_name + ' ' + chat.last_name;
          const date = chat.sent_at ? new
Date(chat.sent_at).toLocaleDateString('uk-UA', {day:'2-digit',
month:'2-digit', year:'numeric'}) : '';
          // --- booking title ---
          let titleHtml = '';
          if (chat.booking_id && chat['booking_title']) {
            titleHtml = `<div class='listing-title' style='font-
size:12px;color:#0057b7;font-weight:normal;line-height:1;margin-
top:2px;'>${chat['booking_title']}</div>`;
          } else if (chat.listing_title) {
            titleHtml = `<div class='listing-title' style='font-
size:12px;color:#0057b7;font-weight:normal;line-height:1;margin-
top:2px;'>${chat.listing_title}</div>`;
          }
```

```
          li.innerHTML = `
<img src="${photo}" alt="${name}">
<span class="user-name" style="display:flex;flex-
direction:column;gap:0;">
  <span>${name}</span>
  ${titleHtml}
</span>
<span class="last-message-date">${date}</span>
  `;
  li.querySelector('.user-name').dataset.name = name;
  li.querySelector('img').dataset.photo = photo;
  li.dataset.phone = chat.phone || '-';
  li.dataset.listingTitle = chat.listing_title || '';
  li.dataset.listingAddress = chat.address || '';
  li.dataset.bookingTitle = chat['booking_title'] || '';
  li.dataset.bookingAddress = chat['booking_address'] || '';
  let price = '';
  if (chat.booking_id && chat['booking_price']) {
    price = chat['booking_price'] + (chat['booking_currency'] ? '
' + chat['booking_currency'] : '');
  } else if (chat.offer_type === 'rent') {
    if (chat.price_rent && chat.currency_rent) price =
chat.price_rent + ' ' + chat.currency_rent + ' / міс';
  } else if (chat.offer_type === 'sale') {
    if (chat.price_sale && chat.currency_sale) price =
chat.price_sale + ' ' + chat.currency_sale;
  }
  li.dataset.listingPrice = price;
  let listingPhoto = 'images/object.jpg';
  if (chat.booking_id) {
    listingPhoto = 'images/booking.png';
  } else if (chat.listing_photos) {
    try {
      const arr = JSON.parse(chat.listing_photos);
      if (Array.isArray(arr) && arr.length > 0) listingPhoto =
arr[0];
    } catch(e) {}
  }
  li.dataset.listingPhoto = listingPhoto;
  if (chat.booking_id) {
    li.dataset.bookingOfferType = chat.booking_offer_type || '';
    li.dataset.bookingPriceRent = chat.booking_price_rent || '';
    li.dataset.bookingCurrencyRent = chat.booking_currency_rent ||
'';
    li.dataset.bookingPriceSale = chat.booking_price_sale || '';
    li.dataset.bookingCurrencySale = chat.booking_currency_sale ||
'';
    li.dataset.bookingTitle = chat.booking_title || '';
    li.dataset.bookingAddress = chat.booking_address || '';
  }
  chatList.appendChild(li);
});
      }
```

```
      function showChatListSpinner(show) {
        const overlay = document.getElementById('chat-list-
spinner-overlay');
        if (overlay) overlay.style.display = show ? 'flex' :
'none';
      }
      function loadChats(page = 0) {
        if (isLoading) return;
        isLoading = true;
        showChatListSpinner(true);
        let url =
`action/get_chats.php?filter=${currentFilter}&page=${page}`;
        fetch(url)
          .then(response => response.json())
          .then(data => {
            if (data.chats && data.chats.length > 0) {
              renderChats(data.chats);
              currentPage = page;
            } else if (page === 0) {
              chatList.innerHTML = '<li style="text-
align:center;color:#888;padding:20px;">Чатів не знайдено</li>';
              document.getElementById('messages').innerHTML = '';
              renderListingCard();
            }
          })
          .catch(err => {
            console.error('Error loading chats:', err);
            chatList.innerHTML = '<li style="text-
align:center;color:#888;padding:20px;">Чатів не знайдено</li>';
            document.getElementById('messages').innerHTML = '';
            renderListingCard();
          })
          .finally(() => {
            isLoading = false;
            showChatListSpinner(false);
          });
      }

      chatList.addEventListener('scroll', function () {
        if (chatList.scrollTop + chatList.clientHeight >=
chatList.scrollHeight - 10) {
          loadChats(currentPage + 1);
        }
      });

      function updateMobileSwitchUI(forceFilter) {
        if (forceFilter) currentFilter = forceFilter;
        // Оновлюємо підпис mobile-switch
        if (currentFilter === 'initiator') {
          mobileSwitchLabel.textContent = 'Шукаю житло';
        } else {
          mobileSwitchLabel.textContent = 'Здаю житло';
```

```
        }
        toggleButtons.forEach(b => b.classList.remove('active'));
        if (currentFilter === 'initiator') {
          toggleButtons[0].classList.add('active');
        } else {
          toggleButtons[1].classList.add('active');
        }
        if (mobileSwitch) {
          mobileSwitch.classList.toggle('active', currentFilter
=== 'receiver');
        }
        loadChats(0);
      }
      if (mobileSwitch) {
        mobileSwitch.addEventListener('click', function() {
          currentFilter = (currentFilter === 'initiator') ?
'receiver' : 'initiator';
          updateMobileSwitchUI();
        });
      }
      toggleButtons.forEach(btn => {
        btn.addEventListener('click', function() {
          currentFilter = this.dataset.filter;
          updateMobileSwitchUI();
        });
      });
      updateMobileSwitchUI(currentFilter);

      chatList.addEventListener('click', function(e) {
        let li = e.target.closest('li');
        // Ignore spinner clicks
        if (!li || li.classList.contains('spinner')) return;

        // Check if the selected chat is already active
        if (li.dataset.userId === selectedUserId &&
li.dataset.listingId === selectedListingId && li.dataset.bookingId
=== selectedBookingId) {
          return;
        }

        selectedUserId = li.dataset.userId;
        selectedListingId = li.dataset.listingId || '';
        selectedBookingId = li.dataset.bookingId || '';
        selectedUserName = li.querySelector('.user-name
span')?.textContent.trim() || '';
        selectedUserPhoto = li.querySelector('img')?.src ||
'images/photo.jpg';

        const chat = Array.from(chatList.children).find(item =>
item.dataset.userId == selectedUserId && item.dataset.listingId ==
selectedListingId && item.dataset.bookingId == selectedBookingId);
        selectedUserPhone = chat?.dataset.phone || '-';
```

```
        selectedUserPhone = selectedUserPhone.startsWith('+38') ?
selectedUserPhone : `+38${selectedUserPhone}`;

        renderChatHeader();

        let url = 'action/get_messages.php?user_id=' +
encodeURIComponent(selectedUserId);
        if (selectedListingId) url += '&listing_id=' +
encodeURIComponent(selectedListingId);
        if (selectedBookingId) url += '&booking_id=' +
encodeURIComponent(selectedBookingId);
        fetch(url)
          .then(r => r.json())
          .then(data => {
            if (!data.messages || data.messages.length === 0) {
              createEmptyChatIfNeeded(selectedUserId,
selectedListingId, selectedBookingId);
            } else {
              loadMessages();
            }
          });
      });
      function createEmptyChatIfNeeded(userId, listingId,
bookingId) {
        const formData = new FormData();
        formData.append('receiver_id', userId);
        formData.append('listing_id', listingId);
        if (bookingId) formData.append('booking_id', bookingId);
        formData.append('message', '');
        fetch('action/send_chat_message.php', {
          method: 'POST',
          body: formData
        })
        .then(r => r.json())
        .then(data => {
          loadChats(currentFilter);
          loadMessages();
        });
      }
      function showMessagesSpinner(show) {
        const overlay = document.getElementById('messages-spinner-
overlay');
        if (overlay) overlay.style.display = show ? 'flex' :
'none';
      }
      function loadMessages() {
        if (!selectedUserId) {
          renderChatHeader();
          renderListingCard();
          document.getElementById('messages').innerHTML = '';
          showMessagesSpinner(false);
          stopMessagesPolling();
          return;
```

```
        }
        const card = document.getElementById('listing-card');
        card.style.display = '';
        card.innerHTML = '';
        document.getElementById('messages').innerHTML = '';
        showMessagesSpinner(true);
        let url = 'action/get_messages.php?user_id=' +
encodeURIComponent(selectedUserId);
        if (selectedListingId) url += '&listing_id=' +
encodeURIComponent(selectedListingId);
        if (selectedBookingId) url += '&booking_id=' +
encodeURIComponent(selectedBookingId);
        fetch(url)
          .then(r => r.json())
          .then(data => {
            renderListingCard();
            renderMessages(data.messages || []);
            renderChatHeader();
            document.getElementById('chat-form').style.display =
'';
            document.getElementById('msg-input').disabled = false;
            document.getElementById('file-input').disabled =
false;
            document.getElementById('file-btn').disabled = false;
            lastMessagesHash = JSON.stringify(data.messages);
            startMessagesPolling();
          })
          .finally(() => {
            showMessagesSpinner(false);
          });
      }
      function renderChatHeader() {
        const el = document.getElementById('chat-header');
        el.style.display = '';
        // Якщо є дані - показати, якщо ні - плейсхолдери
        const photo = selectedUserPhoto || 'images/photo.jpg';
        const name = selectedUserName || 'Співрозмовник';
        const phone = selectedUserPhone || '-';
        // Додаємо data-user-id для фото
        el.innerHTML = `<div class="user-info" style="flex:1;">
          <img src="${photo}" alt="Фото" class="user-photo"
style="cursor:pointer;" data-user-id="${selectedUserId || ''}">
          <span class="user-name">${name}</span>
        </div>
        <span class='user-phone' style="margin-
left:auto;display:flex;align-items:center;gap:5px;"><i class='fas
fa-phone'></i> <span class='phone-number'>${phone}</span></span>`;
        const userPhoto = el.querySelector('.user-photo');
        if (userPhoto && selectedUserId) {
          userPhoto.addEventListener('click', function(e) {
            window.open('user.php?id=' +
encodeURIComponent(selectedUserId), '_blank');
          });
```

```
        }
      }
      document.getElementById('chat-form').style.display = '';
      function renderListingCard() {
        const card = document.getElementById('listing-card');
        if (selectedBookingId) {
          const chat = Array.from(chatList.children).find(item =>
item.dataset.bookingId == selectedBookingId);
          const title = chat?.dataset.bookingTitle || 'Без назви';
          const address = chat?.dataset.bookingAddress || 'Адреса
не вказана';
          let price = '-';
          const offerType = chat?.dataset.bookingOfferType ||
chat?.dataset.booking_offertype || '';
          const priceRent = chat?.dataset.bookingPriceRent ||
chat?.dataset.booking_pricerent || '';
          const priceSale = chat?.dataset.bookingPriceSale ||
chat?.dataset.booking_pricesale || '';
          const currencyRent = chat?.dataset.bookingCurrencyRent
|| chat?.dataset.booking_currencyrent || '';
          const currencySale = chat?.dataset.bookingCurrencySale
|| chat?.dataset.booking_currencysale || '';
          if (offerType === 'rent' && priceRent && currencyRent) {
            price = `${priceRent} ${currencyRent} <span
style=\"font-size:13px;color:#888;\">/ міс</span>`;
          } else if (offerType === 'sale' && priceSale &&
currencySale) {
            price = `${priceSale} ${currencySale}`;
          }
          card.innerHTML = `
            <div style="display:flex;align-
items:center;gap:12px;padding:5px 15px;text-align:left;border-
top:1px solid #ddd;border-left:1px solid #ddd;border-right:1px
solid #ddd;">
              <img src="images/booking.png" alt="Фото бронювання"
style="width:50px;height:50px;object-fit:cover;border-
radius:8px;">
              <div style="display:flex;flex-
direction:column;gap:2px;">
                <div style="font-weight:bold;font-
size:14px;">${title}</div>
                <div style="font-
size:12px;color:#666;">${address}</div>
                <div style="font-size:10px;color:#0057b7;font-
weight:bold;">${price}</div>
              </div>
            </div>
          `;
          card.style.display = '';
          return;
        }
        if (!selectedListingId) {
          card.style.display = 'none';
```

```
          card.innerHTML = '';
          return;
        }
        const chat = Array.from(chatList.children).find(item =>
item.dataset.listingId == selectedListingId);
        if (!chat) {
          card.style.display = 'none';
          card.innerHTML = '';
          return;
        }
        const title = chat.dataset.listingTitle || 'Без назви';
        const address = chat.dataset.listingAddress || 'Адреса не
вказана';
        let price = chat.dataset.listingPrice || 'Ціна не
вказана';
        if (price !== 'Ціна не вказана' && price.includes('|')) {
          const [amount, currency, offerType] = price.split('|');
          price = `${amount} ${currency}`;
          if (offerType === 'rent') {
            price += ' <span style="font-size:13px;color:#888;">/
міс</span>';
          }
        }
        const photo = chat.dataset.listingPhoto ||
'images/object.jpg';
        card.innerHTML = `
          <a href="object.php?id=${selectedListingId}"
target="_blank" style="text-decoration: none; color: inherit;
display: flex; align-items: center; gap: 12px; padding: 5px 15px;
text-align: left; border-top: 1px solid #ddd; border-left: 1px
solid #ddd; border-right: 1px solid #ddd;">
            <img src="${photo}" alt="Фото"
style="width:50px;height:50px;object-fit:cover;border-
radius:8px;">
            <div style="display:flex;flex-
direction:column;gap:2px;">
              <div style="font-weight:bold;font-
size:14px;">${title}</div>
              <div style="font-
size:12px;color:#666;">${address}</div>
              <div style="font-size:10px;color:#0057b7;font-
weight:bold;">${price}</div>
            </div>
          </a>
        `;
        card.style.display = '';
      }
      function renderMessages(messages) {
        const box = document.getElementById('messages');
        if (!messages.length) {
          box.innerHTML = '<div style="text-
align:center;color:#888;padding:20px;    text-align: left; ">Немає
повідомлень</div>';
```

```
          return;
        }
        box.innerHTML = '';
        let firstMsg = null;
        for (let i = 0; i < messages.length; i++) {
          if ((messages[i].message && messages[i].message.trim()
!== '') || messages[i].file_path) {
            firstMsg = messages[i];
            break;
          }
        }
        let chatTab = null;
        if (firstMsg) {
          chatTab = (firstMsg.sender_id == window.currentUserId) ?
'initiator' : 'receiver';
        }
        let lastDate = null;
        const now = new Date();
        const todayStr = now.toLocaleDateString('uk-UA');
        const yesterday = new Date(now.getFullYear(),
now.getMonth(), now.getDate() - 1);
        const yesterdayStr = yesterday.toLocaleDateString('uk-
UA');
        messages.forEach(msg => {
          if (!(msg.message && msg.message.trim() !== '') &&
!msg.file_path) return;
          const msgDateObj = new Date(msg.sent_at);
          const msgDateStr = msgDateObj.toLocaleDateString('uk-
UA');
          let dateLabel = '';
          if (msgDateStr !== lastDate) {
            if (msgDateStr === todayStr) {
              dateLabel = 'Сьогодні';
            } else if (msgDateStr === yesterdayStr) {
              dateLabel = 'Вчора';
            } else {
              dateLabel = msgDateStr;
            }
            const dateDiv = document.createElement('div');
            dateDiv.style.cssText = 'text-
align:center;color:#888;font-size:13px;margin:18px 0 8px 0;';
            dateDiv.textContent = dateLabel;
            box.appendChild(dateDiv);
            lastDate = msgDateStr;
          }
          const div = document.createElement('div');
          div.className = 'message ' + (msg.sender_id ==
window.currentUserId ? 'sent' : 'received');
          let fileHtml = '';
          if (msg.file_path) {
            const ext =
msg.file_path.split('.').pop().toLowerCase();
```

```
            if
(["jpg","jpeg","png","gif","bmp","webp"].includes(ext)) {
              fileHtml = `<div class='file'><a
href='${msg.file_path}' target='_blank'><img
src='${msg.file_path}' style='width:320px;height:240px;aspect-
ratio:4/3;object-fit:cover;border-radius:12px;'></a></div>`;
            } else {
              const fname = msg.file_path.split('/').pop();
              let label = 'Документ';
              if (ext === 'pdf') label = 'PDF-документ';
              else if (ext === 'doc' || ext === 'docx') label =
'Word-документ';
              else if (ext === 'xls' || ext === 'xlsx') label =
'Excel-документ';
              else if (ext === 'txt') label = 'Текстовий файл';
              else if (ext) label = 'Документ (' + ext + ')';
              fileHtml = `<div class='file'><a
href='${msg.file_path}' target='_blank'><i class='fas fa-
paperclip'></i> ${label}</a></div>`;
            }
          }
          let safeMsg = msg.message ?
msg.message.replace(/[&<>"']/g, function(c) {
            return
{'&':'&amp;','<':'&lt;','>':'&gt;','\"':'&quot;','\'':'&#39;'}[c];
          }) : '';
          const time = msgDateObj.toLocaleTimeString('uk-UA',
{hour: '2-digit', minute: '2-digit'});
          let showTime = false;
          if (msg.sender_id == window.currentUserId) {
            showTime = true;
          } else if (chatTab === currentFilter) {
            if ((chatTab === 'receiver' && msg.sender_id !=
window.currentUserId)) {
              showTime = true;
            }
          }
          div.innerHTML = `<div class="text" style="text-
align:${msg.sender_id == window.currentUserId ? 'right' :
'left'};">${fileHtml}${safeMsg}${showTime ? `<div style=\"font-
size:10px;color:#888;margin-top:5px; text-align: ${(msg.sender_id
== window.currentUserId) ? 'right' : 'left'};\">${time}</div>` :
''}</div>`;
          box.appendChild(div);
        });
        box.scrollTop = box.scrollHeight;
      }

      document.getElementById('file-
btn').addEventListener('click', function() {
        document.getElementById('file-input').click();
      });
```

```
      document.getElementById('file-
input').addEventListener('change', function() {
        const file = this.files[0];
        if (file) {
          document.getElementById('msg-input').focus();
        }
      });

      document.getElementById('chat-
form').addEventListener('submit', function(e) {
        e.preventDefault();
        const messageInput = document.getElementById('msg-input');
        const fileInput = document.getElementById('file-input');
        const messageText = messageInput.value.trim();
        const file = fileInput.files[0];
        if (!messageText && !file) return;

        const formData = new FormData();
        formData.append('receiver_id', selectedUserId);
        formData.append('listing_id', selectedListingId);
        if (selectedBookingId) formData.append('booking_id',
selectedBookingId);
        formData.append('message', messageText);
        if (file) formData.append('file', file);

        fetch('action/send_chat_message.php', {
          method: 'POST',
          body: formData
        })
        .then(response => response.json())
        .then(data => {
          if (data.success) {
            // Додаємо повідомлення одразу в DOM
            const box = document.getElementById('messages');
            const div = document.createElement('div');
            div.className = 'message sent';
            let fileHtml = '';
            if (file) {
              const ext =
file.name.split('.').pop().toLowerCase();
              const tempUrl = URL.createObjectURL(file);
              if
(["jpg","jpeg","png","gif","bmp","webp"].includes(ext)) {
                fileHtml = `<div class='file'><img
src='${tempUrl}' style='width:320px;height:240px;aspect-
ratio:4/3;object-fit:cover;border-radius:12px;'></div>`;
              } else {
                fileHtml = `<div class='file'><i class='fas fa-
paperclip'></i> ${file.name}</div>`;
              }
            }
            let safeMsg = messageText.replace(/[&<>"]'/g,
function(c) {
```

```
              return
{'&':'&amp;','<':'&lt;','>':'&gt;','\"':'&quot;','\'':'&#39;'}[c];
            });
            const time = new Date().toLocaleTimeString('uk-UA',
{hour: '2-digit', minute: '2-digit'});
            div.innerHTML = `<div
class="text">${fileHtml}${safeMsg}<div style=\"font-
size:10px;color:#888;margin-top:5px; text-align:
right;\">${time}</div></div>`;
            box.appendChild(div);
            box.scrollTop = box.scrollHeight;
            messageInput.value = '';
            fileInput.value = '';
          } else {
            console.error('Failed to send message:', data.error);
          }
        })
        .catch(err => console.error('Error sending message:',
err));
    });
    if ((window.initialUserId && window.initialUserId !== 0) ||
(window.initialBookingId && window.initialBookingId !== 0)) {
      // Якщо booking_id присутній, не створюємо чат із
listing_id=0, booking_id=null
      if (window.initialBookingId && window.initialBookingId !==
0) {
        selectedBookingId = window.initialBookingId;
        selectedListingId = window.initialListingId;
        selectedUserId = window.initialUserId;
        // Створити чат одразу, якщо його ще немає
        const formData = new FormData();
        formData.append('receiver_id', window.initialUserId);
        formData.append('listing_id', window.initialListingId ?
window.initialListingId : ''); // якщо 0, то ''
        formData.append('booking_id', window.initialBookingId);
        formData.append('message', '');
        fetch('action/send_chat_message.php', {
          method: 'POST',
          body: formData
        })
        .then(r => r.json())
        .then(data => {
          loadChats(currentFilter);
        });
      } else if (window.initialUserId && window.initialUserId !==
0) {
        const formData = new FormData();
        formData.append('receiver_id', window.initialUserId);
        formData.append('listing_id', window.initialListingId);
        formData.append('message', '');
        fetch('action/send_chat_message.php', {
          method: 'POST',
          body: formData
```

```
        })
        .then(r => r.json())
        .then(data => {
          loadChats(currentFilter);
        });
      }
      function trySelectInitialChat() {
        const li = Array.from(document.querySelectorAll('#chat-
list li')).find(li =>
          li.dataset.userId == window.initialUserId &&
          (window.initialListingId ? li.dataset.listingId ==
window.initialListingId : true) &&
          (window.initialBookingId ? li.dataset.bookingId ==
window.initialBookingId : true)
        );
        if (li) {
          li.click();
        } else {
          if (window._chatTries === undefined) window._chatTries =
0;
          if (window._chatTries++ < 7)
setTimeout(trySelectInitialChat, 300);
        }
      }
      trySelectInitialChat();
    }
  });
  </script>
```