**ДЕРЖАВНИЙ УНІВЕРСИТЕТ**
**ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ**
**НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ**
**ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ**
**КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ**

# КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Розробка веб-застосунку для купівлі/оренди нерухомості з використанням мов розмітки HTML/CSS, мови програмування Java Script»

на здобуття освітнього ступеня бакалавра
зі спеціальності 121 Інженерія програмного забезпечення
освітньо-професійної програми «Інженерія програмного забезпечення»

*Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело*

_____ Владислав ГРЕБЕЩЕНКО
*(підпис)*

Виконав: здобувач вищої освіти групи ПД-44

Владислав ГРЕБЕЩЕНКО

Керівник: Володимир САДОВЕНКО
*к. ф-м.н, доцент*

Рецензент: _____

**Київ 2024**

**ДЕРЖАВНИЙ УНІВЕРСИТЕТ**
**ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ**

**Навчально-науковий інститут інформаційних технологій**

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти Бакалавр

Спеціальність 121 Інженерія програмного забезпечення

Освітньо-професійна програма «Інженерія програмного забезпечення»

**ЗАТВЕРДЖУЮ**
Завідувач кафедри
Інженерії програмного забезпечення
_____ Ірина ЗАМРІЙ
« ____ » _____ 2024 р.

# ЗАВДАННЯ
# НА КВАЛІФІКАЦІЙНУ РОБОТУ

Гребещенку Владиславу Едуардовичу

1. Тема кваліфікаційної роботи: «Розробка веб застосунку для купівлі/оренди нерухомості з використанням мов розмітки HTML/CSS, мови програмування Java Script»
керівник кваліфікаційної роботи к.ф-м.н., доцент Володимир САДОВЕНКО
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «27» лютого 2024 р. № 36.

2. Строк подання кваліфікаційної роботи «28» травня 2024 р.

3. Вихідні дані до кваліфікаційної роботи:
3.1 Середовище розробки Visual Studio Code
3.2 HyperText Markup Language та каскадні системи стилей
3.3 Java Script та її бібліотеки (Node.js, React.js)

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
4.1. Огляд та аналіз існуючих веб-додатків.
4.2. Огляд та аналіз інструментів використаних для реалізації проекту.

4.3. Програмна реалізація веб-застосунку.
4.4. Висновки.

5. Перелік графічного матеріалу: *презентація*
1. Аналіз аналогів
2. Вимоги до застосунку
3. Програмні засоби реалізації
4. Діаграма активності
5. Схема бази даних
6. Екранні форми
7. Апробація результатів дослідження

6. Дата видачі завдання«28» лютого 2024 р.

## КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назва етапів кваліфікаційної роботи | Строк виконання етапів роботи | Примітка |
|---|---|---|---|
| 1 | Підбір та аналіз науково-технічної літератури | 28.02.-06.03.2024 | |
| 2 | Аналіз та дослідження існуючих аналогів | 07.03-13.03.2024 | |
| 3 | Огляд та аналіз інструментів використаних для реалізації проекту. | 14.03-20.03.2024 | |
| 4 | Визначення основних вимог до веб-застосунку. | 21.03-24.03.2024 | |
| 5 | Проектування візуальної частини та інтерфейсу. | 25.03-02.04.2024 | |
| 6 | Програмна реалізація веб-застосунку. | 03.04-25.04.2024 | |
| 7 | Тестування застосунку | 25.04-28.04.2024 | |
| 8 | Оформлення роботи: вступ, висновки, реферат | 29.04-05.05.2024 | |
| 9 | Розробка демонстраційних матеріалів | 06.05-12.05.2024 | |
| 10 | Попередній захист роботи | 13.05-31.05.2024 | |

Здобувач вищої освіти _____ Владислав ГРЕБЕЩЕНКО
*(підпис)*

Керівник
кваліфікаційної роботи _____ Володимир САДОВЕНКО

*(підпис)*

# РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 51 стор., 1 табл., 40 рис., 15 джерел.

*Мета роботи* – підтримка процесу пошуку нерухомості для покупки/оренди шляхом створення веб-застосунку за домогою мови Java Script та мов розмітки HTML/CSS

*Об'єкт дослідження* – процес пошуку нерухомості для покупки та/або оренди.

*Предмет дослідження* – пошук нерухомості/житла за допомогою веб-застосунків

*Короткий зміст роботи*: В роботі проаналізовано аналоги веб-застосунків, їх функціонал та можливості. Проаналізовано інструментальні засоби для створення програмного забезпечення, особливості їх використання. Розроблено алгоритм роботи застосунку та програмно реалізовані ключові функціональні можливості, зокрема: фільтр, пошук, можливість прямого контакту з орендодавцем. Проведено функціональне та модульне тестування додатку. В роботі використано мову розмітки HTML/CSS, мову програмування Java Script та її бібліотеки: React.js та Node.js.

Сферою використання застосунку є нерухомість: її пошук, підбір, огляд оренда або купівля.

КЛЮЧОВІ СЛОВА: HTML, CSS, Java Script, Visual Studio Code, Node.js, React.js.

ЗМІСТ

# ВСТУП

Пошук нерухомості для купівлі або оренди є однією з найважливіших задач у житті багатьох людей. Ця тема набуває особливої актуальності в сучасному світі через декілька ключових причин.

По-перше, урбанізація та міграційні процеси значно впливають на попит на житло. Все більше людей переїжджають до великих міст у пошуках кращих можливостей для роботи, освіти та покращення якості життя. Це створює високу конкуренцію на ринку нерухомості, що ускладнює процес пошуку відповідного житла.

По-друге, економічні зміни та коливання на ринку нерухомості додають невизначеності для потенційних покупців та орендарів. Інфляція, зростання процентних ставок на іпотечні кредити та зміни в законодавстві впливають на доступність та вартість житла. Людям необхідно бути добре поінформованими, щоб прийняти правильні рішення у складних економічних умовах.

По-третє, технологічний прогрес значно змінив підходи до пошуку нерухомості. Завдяки онлайн-платформам та мобільним додаткам, процес пошуку житла став більш зручним та швидким. Однак, це також призвело до нових викликів, таких як необхідність перевірки достовірності інформації та захисту від шахрайства.

Крім того, пошук нерухомості завжди пов'язаний з важливими життєвими подіями, такими як створення сім'ї, переїзд до іншого міста чи країни, або інвестиції у власне майбутнє. Від правильного вибору житла залежить якість життя людини, її фінансова стабільність та комфорт.

Таким чином, актуальність теми пошуку нерухомості для купівлі чи оренди обумовлена численними соціальними, економічними та технологічними факторами. Це робить її важливою для широкого кола людей, які прагнуть забезпечити себе та своїх близьких комфортним та безпечним житлом.

Пошук нерухомості може бути обумовлений багатьма факторами, що стосуються як особистих, так і професійних змін у житті людини. Однією з найпоширеніших причин є необхідність змінити житлові умови у зв'язку зі створенням або розширенням сім'ї. Люди часто шукають нове житло, коли одружуються, народжують дітей або коли їхні діти виростають і потребують окремого простору.

Іншою важливою причиною є зміна місця роботи або навчання. Переїзд до іншого міста чи країни для кращих кар'єрних можливостей або для здобуття освіти часто вимагає пошуку нової квартири або будинку. Також значним фактором може бути бажання покращити умови проживання. Люди прагнуть переїхати у більш зручні райони з кращою інфраструктурою, більшою кількістю зелених зон або ближче до центрів культурного та соціального життя.

Не менш важливою причиною є фінансові обставини. Поліпшення матеріального становища дозволяє придбати більш комфортне житло, тоді як складні економічні умови можуть змусити шукати дешевші варіанти оренди або продажу власної нерухомості. Інвестування в нерухомість також стає поширеним мотивом для пошуку житла. Люди розглядають придбання нерухомості як спосіб збереження і примноження капіталу, орендуючи його іншим або сподіваючись на зростання цін у майбутньому.

Крім того, певні життєві обставини, такі як розлучення, смерть близьких або необхідність догляду за старшими членами родини, можуть вимагати змін у житлових умовах. Інколи бажання змінити оточення і почати життя з чистого аркуша також стає рушійною силою для пошуку нового місця проживання.

Пошук нерухомості – це складний і багатоетапний процес, що супроводжується різноманітними викликами. Однією з основних складнощів є висока конкуренція на ринку. Особливо це стосується великих міст, де попит на житло значно перевищує пропозицію. Це може призвести до ситуації, коли бажане житло швидко знаходить іншого покупця або орендаря, змушуючи людей діяти дуже оперативно.

Юридичні аспекти також є важливим викликом. Оформлення угод з нерухомістю потребує знання законодавства та уваги до деталей, щоб уникнути шахрайства або помилок, які можуть призвести до втрати грошей або прав на майно. Необхідність перевірки чистоти прав на нерухомість, відсутності боргів і законності забудови додає складності процесу.

Крім того, пошук нерухомості потребує значних часових витрат. Перегляд великої кількості варіантів, зустрічі з агентами, відвідування об'єктів та ведення переговорів можуть затягнутися на довгі місяці. Водночас є ризик емоційного виснаження, особливо якщо процес затягується або виявляються непередбачувані проблеми.

Правильний вибір нерухомості має надзвичайно важливе значення, адже від цього залежить якість життя людини, її фінансова стабільність та безпека. Житло – це місце, де ми проводимо більшу частину нашого життя, відпочиваємо і відновлюємо сили. Тому воно повинно відповідати нашим потребам та бути комфортним. Також важливо враховувати інфраструктуру району, транспортну доступність, наявність шкіл, лікарень та інших необхідних закладів.

Агентства нерухомості залишаються надійним і популярним джерелом допомоги у пошуку житла. Професійні агенти мають доступ до баз даних, які можуть містити ексклюзивні пропозиції, недоступні для широкої публіки. Вони можуть надати консультації з юридичних і фінансових питань, організувати перегляди об'єктів, а також допомогти у веденні переговорів та оформленні угоди.

Соціальні мережі, такі як Facebook, Instagram, Telegram, також стали важливими інструментами для пошуку нерухомості. Багато людей та агентства публікують оголошення у спеціалізованих групах або на своїх сторінках. Цей метод дозволяє швидко зв'язатися з орендодавцем або продавцем і отримати актуальну інформацію про об'єкт.

Оголошення в місцевих ЗМІ - хоча цей метод стає менш популярним, багато людей все ще користуються оголошеннями в газетах та місцевих журналах. Це може бути корисним, особливо в менших містах або регіонах, де інтернет-охоплення може бути обмеженим.

Нерідко люди знаходять житло через особисті контакти та рекомендації знайомих, друзів або колег. Це один з найстаріших, але все ще ефективних методів, який дозволяє уникнути комісійних витрат агентствам і знайти надійні пропозиції.

Активне вивчення районів, де планується купівля чи оренда житла, також може бути корисним. Прогулянки по району, спілкування з місцевими жителями, вивчення інфраструктури та доступності транспорту дозволяють краще зрозуміти, чи підходить це місце для проживання.

Аукціони нерухомості

У деяких випадках, особливо для інвесторів, може бути цікавим варіант участі в аукціонах нерухомості. Це дозволяє придбати об'єкти за нижчими цінами, хоча такий метод потребує глибокого знання ринку та готовності ризикувати.

Кожен з цих методів має свої переваги та недоліки, і вибір найкращого способу пошуку залежить від конкретних обставин та потреб. Використання кількох різних інструментів одночасно може значно підвищити шанси знайти ідеальну нерухомість.

# 1. ОГЛЯД ТА АНАЛІЗ ІСНУЮЧИХ ВЕБ-ДОДАТКІВ, ВИЗНАЧЕННЯ ОСНОВНИХ ФУНКЦІЙ

## 1.1 Аналіз існуючих веб-додатків для купівлі/оренди нерухомості

На сьогодні існує багато веб-сайтів та веб-додатків, які надають допомогу щодо купівлі та оренди нерухомості в Україні. Проаналізувавши їх рішення дозволить нам виявити їх переваги та недоліки програмного забезпечення.

ованих рішень є веб-сайт: KyivInRealty.

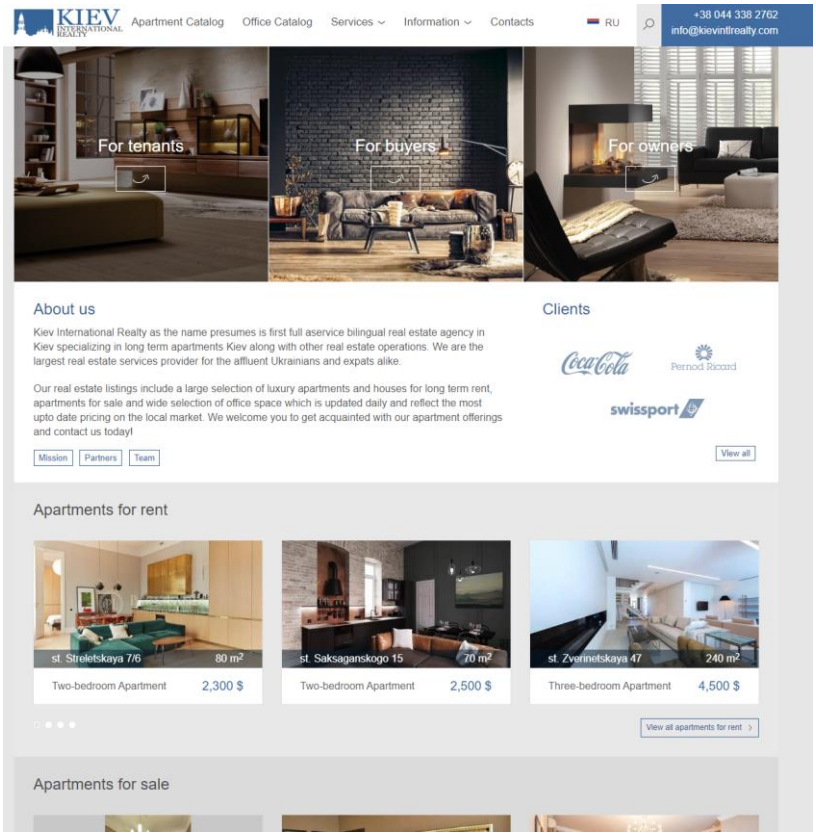



Рис. 1.1 Продукт-аналог KievInRealty

На перший погляд, сайт kievintlrealty.com виглядає досить професійно та має сучасний дизайн, який легко навігується. Ось деякі аспекти, які можна проаналізувати в контексті програмного забезпечення:

Каталог нерухомості: Сайт має розширений каталог доступної нерухомості в Києві, що дозволяє користувачам швидко знайти об'єкти за різними параметрами,

такими як тип нерухомості, ціновий діапазон, місцезнаходження тощо. Це свідчить про наявність системи управління контентом або спеціалізованого ПЗ для організації каталогу.

Функціональність пошуку і фільтрації: Сайт має ефективний механізм пошуку та фільтрації об'єктів нерухомості, що може бути реалізовано за допомогою складних алгоритмів пошуку та фільтрації, що відображається у програмному забезпеченні.

Інтерактивна карта: Наявність інтерактивної карти дозволяє користувачам зручно ознайомлюватися з місцем розташування об'єктів нерухомості. Це може вимагати використання спеціалізованих бібліотек або API для відображення карт.

Контактна інформація та форми зворотного зв'язку: Є зручні форми для зв'язку з агентством нерухомості, що може бути реалізовано за допомогою платформи управління контактами або CRM.

Швидкість і продуктивність: Ефективна робота сайту, його швидкість завантаження і відсутність помилок свідчать про високий рівень програмного забезпечення, яке використовується для розробки та підтримки.

Аналітика та відстеження користувацьких дій: Можливість аналізу поведінки користувачів на сайті, відстеження їхніх дій і взаємодії з контентом може бути реалізована за допомогою відповідних інструментів аналітики та вбудованих скриптів.

У цілому, цей сайт відображає високий рівень програмного забезпечення, що дозволяє забезпечити зручну та ефективну платформу для пошуку та оренди нерухомості, але хоч сайт і має багато позитивних аспектів, також існують деякі недоліки, які можуть бути важливими для користувачів:

Неінтуїтивний інтерфейс: Деякі елементи інтерфейсу можуть бути важкими для розуміння для новачків. Наприклад, меню або функціональність пошуку можуть бути неочевидними або несподіваними.

Обмежена інформація про об'єкти: Деякі об'єкти нерухомості можуть мати обмежену кількість фотографій або деталей, що може зменшити ефективність

пошуку для користувачів, які шукають більше інформації перед прийняттям рішення.

Відсутність розширених функцій фільтрації: Система фільтрації може бути обмеженою, не дозволяючи користувачам налаштовувати пошук за більш докладними критеріями, такими як особливості нерухомості, розмір кімнат, доступність поблизу сервісів тощо.

Неоптимізована мобільна версія: Мобільна версія сайту може бути менш зручною або неоптимізованою для використання на мобільних пристроях.

Брак персоналізації і рекомендацій: Сайт може не надавати персоналізованих рекомендацій або порад користувачам, що може зменшити залучення і взаємодію з ними.

Другим обраним веб-сайтом буде Real Estate Lviv, який має візуально привабливий і сучасний дизайн.

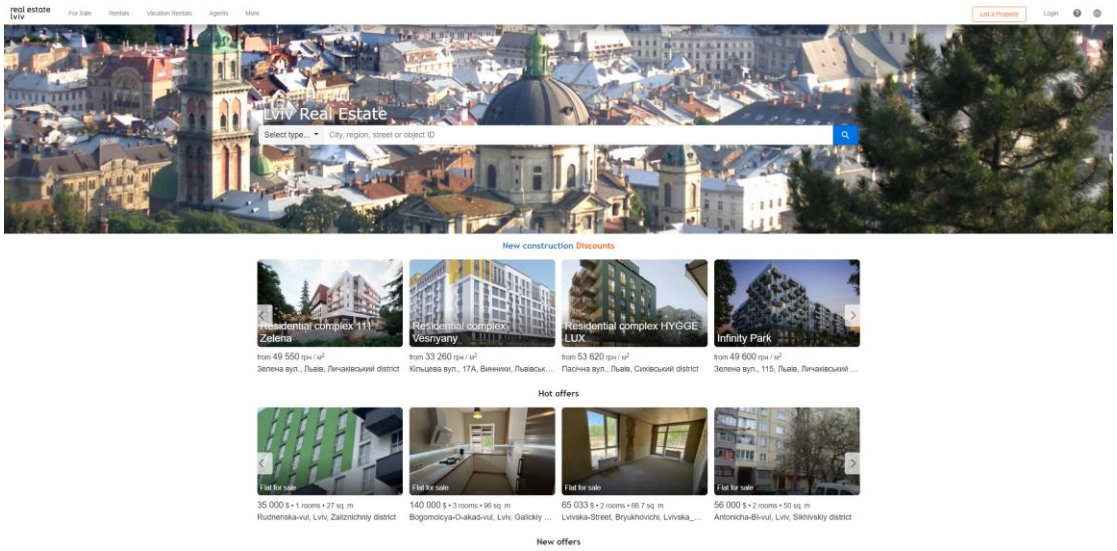



Рис.1.2 Продукт-аналог Real-Estate Lviv

Для аналізу програмного забезпечення слід розглянути наступні аспекти:

1. Каталог нерухомості: Сайт має розширений каталог доступної нерухомості в Львові та області, який легко фільтрується за різними критеріями, такими як тип, ціна, місцезнаходження і т.д. Це свідчить про використання програмного забезпечення для управління контентом або CMS.

2. Інтерактивність і мапи: Сайт має інтерактивні карти, які дозволяють користувачам оглядати нерухомість на карті, що може використовувати API картографічних сервісів.

3. Форми зворотного зв'язку і контактна інформація: Є різноманітні форми зворотного зв'язку та контактна інформація для отримання додаткової інформації або запису на перегляд, що може використовувати систему управління контактами або CRM.

4. Мультимовність: Наявність розділу з англомовним варіантом сайту підказує на використання мультиязикових функцій, що може бути реалізовано за допомогою спеціального програмного забезпечення або плагінів.

5. Оптимізація для мобільних пристроїв: Сайт виглядає добре на мобільних пристроях, що свідчить про його адаптивність та оптимізацію для різних типів пристроїв.

Однак, без прямого доступу до внутрішніх систем можна лише припускати про використання певного програмного забезпечення.

Хоча сайт має свої позитивні сторони, також є деякі недоліки, які можуть вплинути на його ефективність та зручність для користувачів:

1. Пошук та фільтрація: Система пошуку та фільтрації може бути обмеженою або не дуже ефективною. Деякі користувачі можуть зіткнутися з проблемами у виборі точних параметрів пошуку, або може бути недостатньо фільтрів для точного визначення їх потреб.

2. Інформація про нерухомість: Деякі об'єкти нерухомості можуть мати обмежену кількість фотографій або інформації, що може ускладнити прийняття рішення користувачем щодо зацікавленості у цих об'єктах.

3. Швидкість завантаження: Сайт може бути повільним у завантаженні, що може призвести до втрати зацікавленості користувачів та зменшення часу перебування на ньому.

4. Недостатня інтерактивність: Відсутність інтерактивних елементів або функціональності, такої як відеоогляди об'єктів нерухомості або віртуальні тури, може знизити залучення користувачів.

5. Неадаптивний дизайн: Сайт може не бути повністю адаптивним для різних типів пристроїв, що може призвести до нестабільності відображення на мобільних пристроях або планшетах.

Ці недоліки можуть вплинути на загальний досвід користувача та зробити пошук нерухомості менш зручним або ефективним.

Також пропоную оглянути зарубіжні сайти, та проаналізувати різницю між ними, їх можливостями та функціоналом.

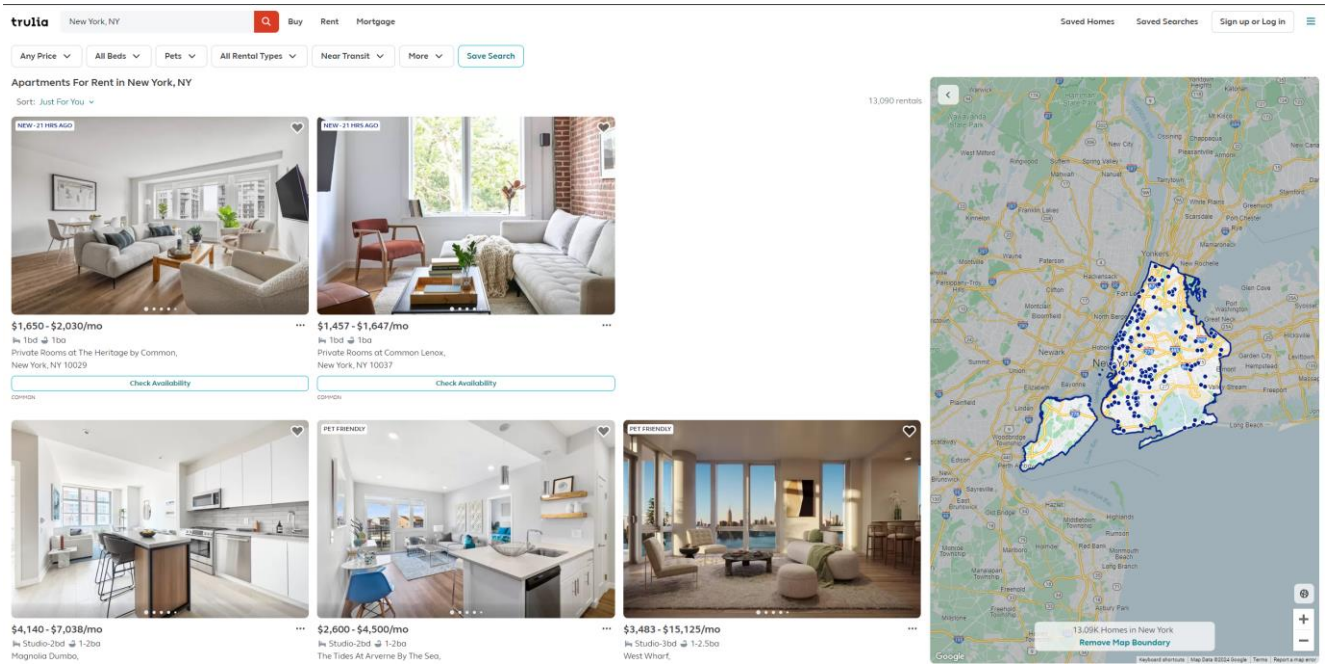



Рис 1.3 Зарубіжний продукт-аналог Trulia

Третій - Trulia - це велика онлайн-платформа для пошуку нерухомості, яка надає різноманітні можливості для користувачів, що шукають житло для оренди або продажу, зокрема в місті Нью-Йорк. Цей веб-сайт пропонує широкий вибір об'єктів нерухомості в різних цінових категоріях, розмірах та розташуваннях.

У своїй роботі Trulia використовує різні програмні засоби та технології для забезпечення зручного та ефективного пошуку нерухомості:

CMS (Система управління контентом): Використання CMS дозволяє команді Trulia легко керувати та оновлювати контент на веб-сайті, включаючи додавання нових оголошень про нерухомість, редагування існуючих списків та оновлення інформації про ціни та умови.

Система пошуку та фільтрації: Потужна система пошуку і фільтрації дозволяє користувачам швидко та ефективно знаходити нерухомість, яка відповідає їхнім потребам та вимогам. За допомогою різних параметрів, таких як ціна, кількість спалень і ванних кімнат, розмір та розташування, користувачі можуть точно налаштувати свій пошук.

Інтерактивна мапа: Наявність інтерактивної мапи дозволяє користувачам візуально оцінити розташування об'єктів нерухомості та їхнє оточення. Це дозволяє краще зрозуміти географічний контекст і вибрати найбільш підходяще розташування для своїх потреб.

Мобільна дружність: З урахуванням зростання використання мобільних пристроїв для доступу до Інтернету, Trulia забезпечує адаптивний дизайн, який забезпечує оптимальне відображення веб-сайту на різних розмірах екранів, зокрема на смартфонах та планшетах.

Аналітика та відстеження: Trulia використовує аналітичні інструменти для відстеження дій користувачів на сайті, збирання даних про їхній взаємодію з веб-сайтом та використання цих даних для вдосконалення функціональності та покращення досвіду користувача.

Сайт Trulia, подібно до багатьох інших, має свої недоліки, особливо у програмному забезпеченні. Один із найбільш очевидних недоліків полягає у швидкості завантаження сторінок. Іноді це може виявитися досить повільним, особливо при використанні мобільних пристроїв. Це може забагато займати часу користувачів та впливати на їхнє задоволення від користування сайтом.

Навігація та інтерфейс також можуть бути проблематичними. Деякі елементи інтерфейсу можуть бути заплутаними або не дуже інтуїтивними, що може викликати незручності для користувачів, які шукають інформацію.

Крім того, сайт може не бути повністю оптимізований для мобільних пристроїв, що може створювати проблеми з відображенням на різних типах пристроїв. Це може призвести до втрати користувачів, які відвідують сайт зі своїх смартфонів або планшетів.

Система пошуку та фільтрації також може бути недостатньою або недосконалою. Користувачам може бути важко точно налаштувати свої критерії пошуку, або система фільтрації може бути обмеженою у можливостях.

Також, сайт може не надавати достатньої взаємодії з користувачем. Наприклад, може відсутні персоналізовані рекомендації або інтерактивні можливості, які б додали значну цінність для користувачів.

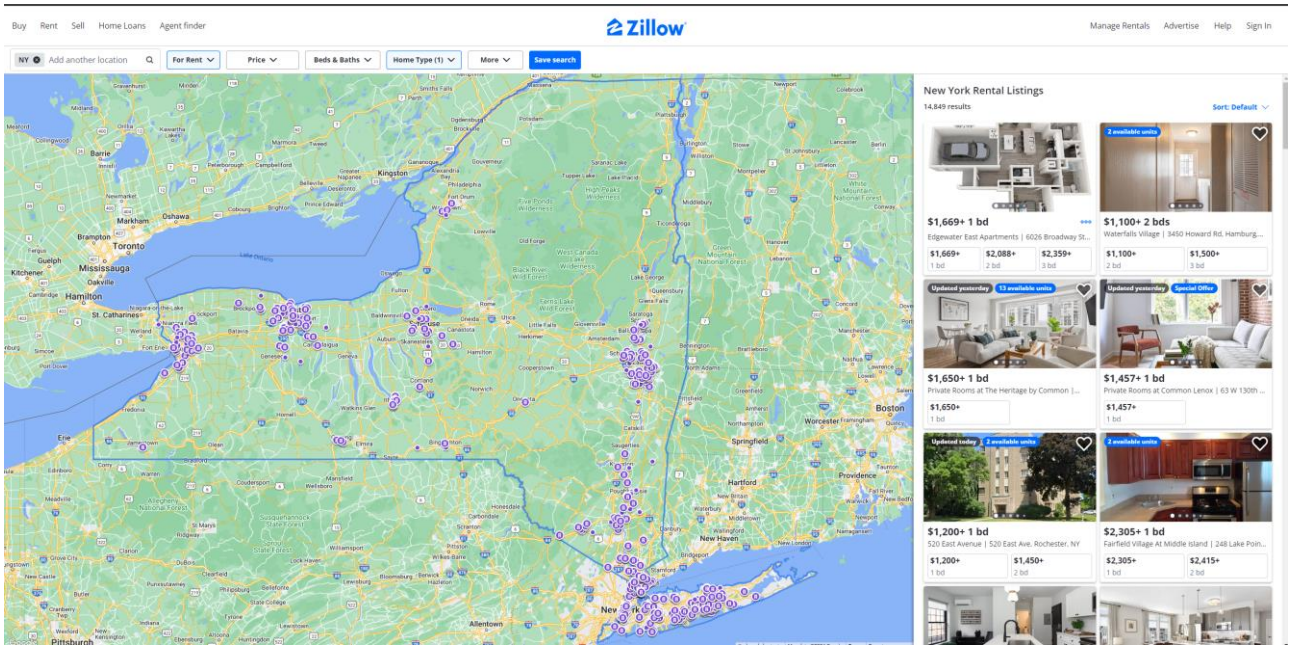



Рис 1.4 Зарубіжний продукт-аналог Zillow

Четвертий - Zillow.com базується на складній архітектурі програмного забезпечення, яка дозволяє обробляти та відображати величезну кількість даних у режимі реального часу. Сайт використовує сучасні технології, такі як мікросервісна архітектура, контейнеризація та оркестрація. Це забезпечує високу масштабованість, доступність та надійність системи. Backlend-частина написана на Java з використанням фреймворків Spring та Kafka, а фронтенд використовує React та Redux. Ефективне управління даними та швидкий пошук досягаються за допомогою технологій баз даних NoSQL, таких як Elasticsearch та Cassandra. Така архітектура дозволяє швидко реагувати на зміни попиту, додавати нові функції та масштабувати ресурси відповідно до навантаження.

Технології та інструменти, використані для розробки сайту:

Мови програмування

Для розробки Zillow.com використовуються такі мови програмування: Java для бекенду, JavaScript (з використанням React) для фронтенду, а також Python для обробки та аналізу даних.

Інфраструктура

Сайт працює на хмарній інфраструктурі AWS, використовуючи такі сервіси, як EC2, S3, RDS, Elasticsearch Service та інші. Це забезпечує масштабованість, доступність та надійність платформи.

Інструменти розробки

Для розробки, тестування та розгортання коду використовуються такі інструменти, як Git, Jenkins, Docker, Kubernetes, Ansible та Terraform.

Аналітика та моніторинг

Для аналізу користувацької поведінки, оптимізації продуктивності та виявлення проблем застосовуються інструменти, такі як Datadog, Splunk та Opsge. Розберемо по пунктам особливості дизайну та взаємодії користувача з сайтом:

Інтерфейс Zillow.com розроблений з урахуванням зручності користувача. Він має чітку структуру, інтуїтивно зрозумілі елементи керування та зручну навігацію. Це дозволяє користувачам легко знаходити необхідну інформацію та швидко виконувати пошук та перегляд варіантів оренди.

Візуальна привабливість

Дизайн сайту характеризується використанням великих, високоякісних фотографій, чистих ліній та приємних кольорових палітр. Це створює привабливий, сучасний вигляд, який допомагає користувачам легко сприймати інформацію та отримувати позитивні враження від взаємодії з платформою.

Сайт пропонує користувачам можливість налаштувати пошук за їхніми власними критеріями, такими як бюджет, кількість кімнат, особливості квартири тощо. Це дозволяє отримувати персоналізовані результати, що значно спрощує процес пошуку ідеального варіанта оренди.

Швидкість завантаження

Оптимізація коду, використання CDN, лаконічний дизайн та інші технічні рішення дозволяють забезпечити швидке завантаження сторінок сайту навіть при великих обсягах даних. Це покращує загальний користувацький досвід та підвищує конверсію.

Архітектура Zillow.com надає можливість легко масштабувати систему відповідно до зростаючого навантаження. Це дозволяє обслуговувати мільйони користувачів одночасно без втрати продуктивності чи доступності.

Таблиця 2.1

Порівняння існуючих застосунків-аналогів

| Характеристика | Zillow | Trulia | KyivInRealty | Real estate Lviv | Real State |
|---|---|---|---|---|---|
| Фільтрація нерухомості | + | + | + | + | + |
| Прямий контакт з орендодавцем | - | - | + | - | + |
| Функція «Обрана нерухомість» | - | + | - | - | + |
| Детальний опис нерухомості | - | + | + | + | + |
| Крос-браузерність | + | + | - | - | + |

## 1.2 Визначення основних функцій та вимог до додатку.

Основною метою створення мобільного додатку для пошуку та оренди нерухомості є надання зручного та ефективного інструменту для українців, які шукають орендувати чи придбати житло, офісні приміщення або комерційну нерухомість. Додаток дозволить користувачам швидко та легко знайти підходящі варіанти, отримати детальну інформацію про них та налагодити зв'язок з власниками або агентами нерухомості.

Ключовими завданнями додатку є: допомогти користувачам швидко знайти нерухомість, яка відповідає їхнім вимогам та побажанням; надати можливість отримати вичерпну інформацію про об'єкти, включаючи фотографії, характеристики, ціну та контактні дані; спростити процес взаємодії з власниками або агентами; дозволити зберігати та порівнювати обрані варіанти; а також дати змогу бронювати та оплачувати оренду або купівлю безпосередньо через мобільний додаток.

Основні функції додатку

Сайт для пошуку та оренди/купівлі нерухомості повинен забезпечити широкий спектр функцій, які спростять та оптимізують процес пошуку та взаємодії з об'єктами нерухомості. Серед ключових функцій додатку можна виділити наступні:

- Швидкий пошук за параметрами: Користувачі повинні мати змогу задавати фільтри для пошуку нерухомості за такими параметрами як локація, тип (квартира, будинок, офіс тощо), кількість кімнат, площа, ціна, наявність паркінгу, зручностей тощо. Це дозволить швидко знаходити варіанти, що відповідають їхнім вимогам.
- Детальний перегляд оголошень: Для кожного об'єкту нерухомості додаток має надавати вичерпну інформацію, включаючи фотографії, опис характеристик, план, ціну, контакти власника/агента, а також історію та рейтинг об'єкта.

- Зв'язок з власниками: Користувачі повинні мати можливість напряму зв'язуватись з власниками або агентами нерухомості через додаток, щоб задавати запитання, домовлятись про огляд або почати переговори.
- Порівняння варіантів: Функція порівняння дозволить користувачам зберігати обрані варіанти нерухомості, аналізувати їх характеристики та ціни, і легко вибирати найкращий варіант.

Перегляд детальної інформації про об'єкти

Одна з ключових функцій додатку для пошуку нерухомості - це надання користувачам вичерпної інформації про кожен доступний об'єкт. Коли користувач знаходить варіант, що йому цікавий, він повинен мати змогу детально вивчити всі його характеристики та особливості, щоб прийняти виважене рішення про оренду чи придбання.

Для кожного об'єкту нерухомості додаток має відображати детальний опис, включаючи кількість кімнат, площу, поверх, рік будівництва, наявність меблів та побутової техніки, а також особливості планування та ремонту. Також важливо відображати фотографії, щоб користувач міг на власні очі оцінити стан та привабливість приміщення. Крім того, додаток має містити контактну інформацію власника або агента, щоб користувач міг напряму з ними зв'язатись для уточнення деталей або узгодження показу об'єкта.

Для підвищення довіри та прозорості варто також відображати історію об'єкта, такі як дату останньої оренди чи продажу, динаміку ціни, а також рейтинг власника або агента за відгуками попередніх клієнтів. Це дасть користувачам розуміння реальної ситуації навколо кожного об'єкта і допоможе їм зробити найкращий вибір.

Легкий контакт з власниками/агентами

Один з ключових аспектів додатку для пошуку нерухомості - це можливість безпосереднього зв'язку між користувачами та власниками або агентами. Це дозволить швидко отримувати відповіді на запитання, домовлятися про перегляд об'єктів, вести переговори щодо оренди чи покупки. Інтегруючи функції телефонних дзвінків, відправки повідомлень, електронної пошти чи месенджерів

безпосередньо в додаток, користувачі зможуть легко налагоджувати контакт з продавцями нерухомості.

Обмін інформацією та документами

Крім простого зв'язку, додаток має надавати можливість обміну даними, документами та медіафайлами між користувачами та власниками/агентами. Це дозволить швидко узгоджувати деталі, надсилати фотографії, копії документів, або навіть укладати онлайн-договори прямо в додатку. Така інтеграція полегшить та пришвидшить процес взаємодії, зробивши його максимально зручним та ефективним для всіх сторін.

Відгуки та оцінки

Для підвищення прозорості та довіри, додаток також має містити систему рейтингів та відгуків про власників і агентів нерухомості. Користувачі, які вже працювали з певним агентом або власником, зможуть ділитися своїми враженнями та оцінками. Це дозволить майбутнім клієнтам ознайомитись з репутацією продавців, їхнім рівнем обслуговування та надійності, і обрати найкращого варіант для співпраці. Така система відгуків значно підвищить довіру до додатку та дозволить уникнути потенційних ризиків.

Порівняння варіантів нерухомості

Легкий огляд та порівняння

Функція порівняння варіантів є невід'ємною частиною ефективного додатку для пошуку нерухомості. Вона надасть користувачам змогу швидко оглядати та співставляти різні об'єкти, порівнюючи їх ключові характеристики та параметри. Це суттєво спростить процес прийняття рішення, дозволяючи легко виявити найкращу пропозицію, яка найбільше відповідає потребам та вимогам користувача.

Порівняння за критеріями

Користувачі повинні мати змогу обирати, за якими саме параметрами вони хочуть порівнювати варіанти нерухомості - це можуть бути площа, кількість кімнат, ціна, розташування, наявність паркінгу, додаткові зручності тощо. Додаток має надавати зручний інструмент для визначення пріоритетних критеріїв

та швидкого співставлення обраних об'єктів за ними. Це допоможе виявити найбільш оптимальний варіант, що максимально відповідає запитам користувача.

Збереження обраних об'єктів

Важливою функцією додатку має бути можливість зберігати обрані користувачем варіанти нерухомості для подальшого детального вивчення та порівняння. Користувачі зможуть формувати персональні списки, позначати цікаві об'єкти, що дозволить їм легко повернутися до них пізніше, переглянути всі характеристики та провести більш глибокий аналіз. Така функціональність забезпечить ефективне управління процесом пошуку та прийняття рішення.

Візуалізація порівнянь

Щоб зробити процес порівняння ще більш наочним та інтуїтивним, додаток може включати візуальні елементи, такі як графіки, діаграми або таблиці. Користувачі зможуть легко бачити ключові відмінності між об'єктами за різними параметрами, що значно спростить процес вибору найкращого варіанту. Така подача інформації підвищить ефективність прийняття рішень та забезпечить більшу впевненість користувачів у своєму виборі.

Структурування списків та нотатки

Для зручності користувачів, додаток має дозволяти структурувати збережені списки, наприклад, за категоріями (оренда квартир, купівля будинків, комерційна нерухомість тощо) або статусами (розглянуто, заброновний, відхилений). Також важливо надати можливість додавати особисті нотатки до кожного об'єкту, щоб фіксувати важливі деталі, враження або наступні кроки. Це значно полегшить пошук та аналіз варіантів на пізніших етапах.

Синхронізація зі всіма пристроями

Для забезпечення зручності та неперервності роботи, додаток має підтримувати синхронізацію збережених списків та нотаток між різними пристроями користувача - смартфоном, планшетом, комп'ютером. Це дозволить йому переглядати, редагувати та доповнювати інформацію про обрані об'єкти нерухомості в будь-який момент та з будь-якого свого пристрою. Така

функціональність зробить пошук та аналіз нерухомості ще більш ефективним та комфортним.

Інтеграція з картами та навігацією

Геолокація об'єктів

Для зручності користувачів, мобільний додаток для пошуку нерухомості має бути інтегрований з картографічними сервісами. Це дозволить відображати розташування доступних об'єктів на мапі, демонструючи їх точне місцезнаходження та наближення до важливих орієнтирів, таких як станції метро, торгові центри чи вузлові транспортні розв'язки. Користувачі зможуть легко оцінити зручність та доступність кожного варіанту.

Прокладання маршрутів

Інтеграція з навігаційними сервісами дозволить додатку надавати користувачам можливість швидко прокладати маршрути до обраних об'єктів нерухомості. Вони зможуть отримувати покрокові інструкції для пішого, автомобільного чи громадського транспорту, а також відстежувати час та відстань у дорозі. Така функціональність значно спростить процес огляду та відвідування потенційних варіантів, допомагаючи користувачам ефективно планувати свій час та маршрути.

Обмін місцезнаходженням

Додаток також має підтримувати можливість обміну місцезнаходженням між користувачем та власником чи агентом нерухомості. Це дозволить легко узгодити час та місце для проведення показу об'єкта, а також допоможе користувачам безпечно та комфортно дістатись до пункту призначення. Інтеграція з картами та службами геолокації забезпечить надійну синхронізацію місцезнаходження та пришвидшить процес взаємодії між сторонами угоди.

Висновки та наступні кроки

Розроблений мобільний додаток для пошуку та оренди/купівлі нерухомості забезпечує всі необхідні функції для ефективного та комфортного вирішення житлових та комерційних питань користувачів. Від швидкого пошуку за розширеними фільтрами до детального перегляду інформації про об'єкти, від

легкого зв'язку з власниками до можливості безпосередньої оплати – додаток пропонує цілісне та зручне рішення для процесу придбання чи оренди нерухомості.

Крім того, вбудовані можливості порівняння варіантів, збереження обраних об'єктів, а також інтеграція з мапами та навігацією значно спрощують та оптимізують пошук та прийняття рішень користувачами. Вони зможуть з упевненістю обирати найкращий варіант, що максимально відповідає їхнім вимогам та потребам. Зручний інтерфейс та інтуїтивно зрозумілі функції додатку забезпечать комфортне і продуктивне використання протягом всього процесу.

Наступні кроки розвитку

Для подальшого вдосконалення та розширення функціоналу додатку, планується реалізувати такі наступні кроки:

- Інтеграція з базами даних нерухомості для постійного оновлення списку доступних пропозицій
- Впровадження системи персоналізованих рекомендацій на основі вподобань та історії користувача
- Розробка мобільних додатків для iOS та Android платформ для охоплення ширшої аудиторії
- Впровадження голосового пошуку та управління для підвищення зручності використання
- Інтеграція з сервісами електронної комерції для змоги оплати та бронювання

Використовуючи інноваційні технології та постійно вдосконалюючи функціональність, ми прагнемо перетворити цей додаток на всеосяжне та незамінне рішення для пошуку та оренди/купівлі нерухомості в Україні.

# 2. ОГЛЯД ТА АНАЛІЗ ІНСТРУМЕНТІВ ВИКОРИСТАНИХ ДЛЯ РЕАЛІЗАЦІЇ ПРОЕКТУ

## 2.1 Мови створення скелету сайта

Мова гіпертекстової розмітки HTML та каскадна система стилей - HTML (HyperText Markup Language) та CSS (Cascading Style Sheets) - це два основних засоби для створення веб-сторінок. HTML відповідає за структуру та зміст веб-сторінки. Він використовує теги для розміщення елементів, таких як заголовки, параграфи, зображення, посилання тощо. HTML визначає, як вміст повинен бути організований та відображений на веб-сторінці.

CSS, з іншого боку, відповідає за стиль і вигляд веб-сторінки. Він дозволяє задавати розміри, кольори, шрифти, відступи та інші властивості елементів HTML. CSS використовується для того, щоб зробити веб-сторінки більш привабливими та функціональними для користувачів.

Разом HTML та CSS утворюють основу веб-дизайну та розробки. HTML відповідає за структуру сторінки, тоді як CSS відповідає за її вигляд. Вони доповнюють один одного, дозволяючи створювати зручні та естетично привабливі веб-сторінки.

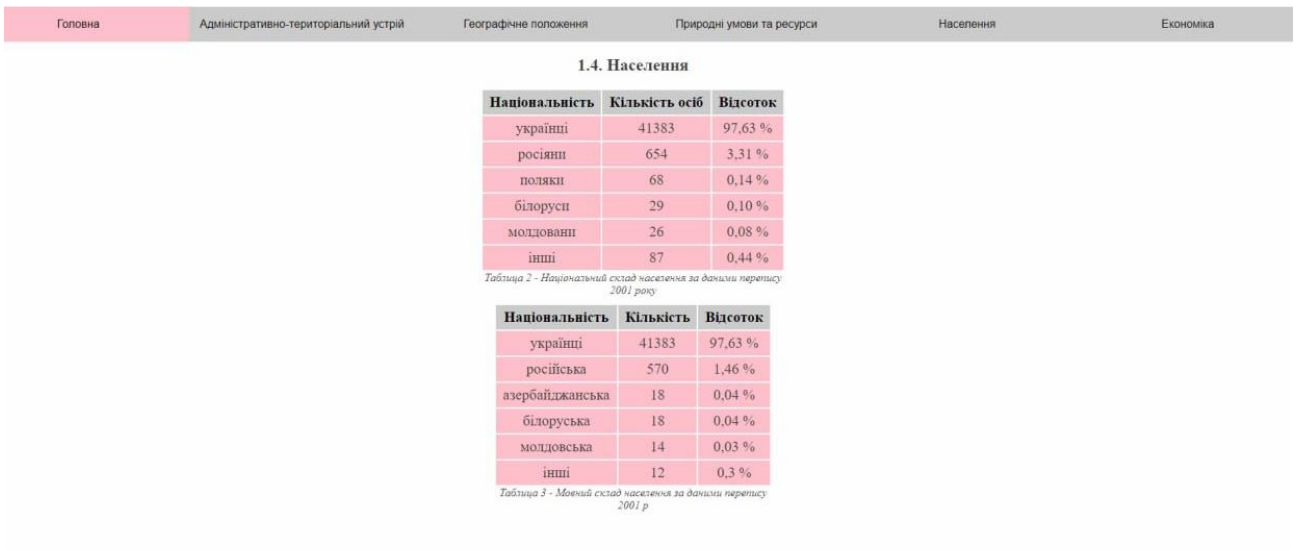
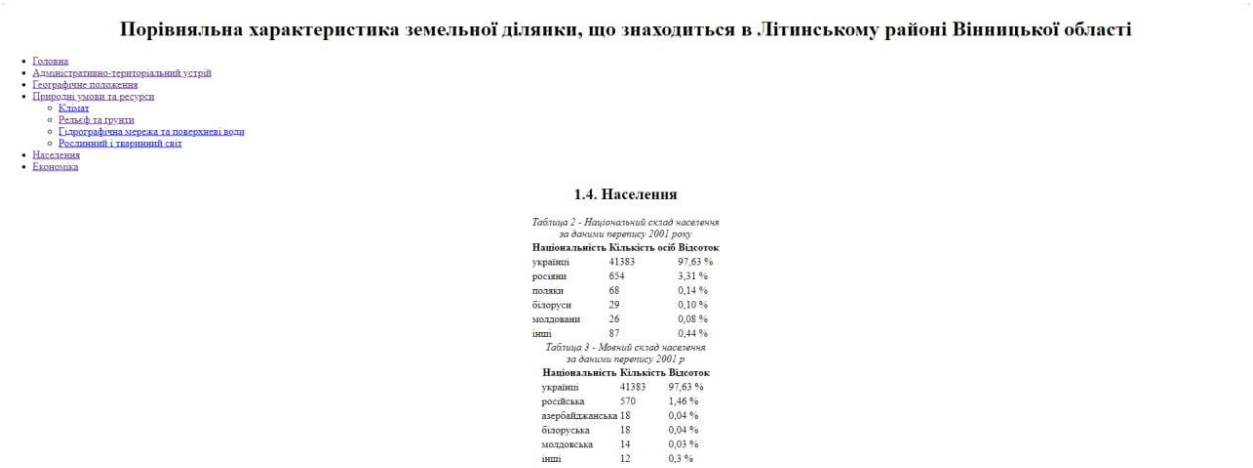



Рис 2.1 Вигляд HTML сторінки з CSS каскадом



**Порівняльна характеристика земельної ділянки, що знаходиться в Літинському районі Вінницької області**

- Головна
- Адміністративно-територіальний устрій
- Географічне положення
- Природні умови та ресурси
  - Клімат
  - Рельєф та ґрунти
  - Гідрографічна мережа та поверхневі води
  - Рослинний і тваринний світ
- Населення
- Економіка

**1.4. Населення**

*Таблиця 2 - Національний склад населення за даними перепису 2001 року*

| Національність | Кількість осіб | Відсоток |
|---|---|---|
| українці | 41383 | 97,63 % |
| росіяни | 654 | 3,31 % |
| поляки | 68 | 0,14 % |
| білоруси | 29 | 0,10 % |
| молдовани | 26 | 0,08 % |
| інші | 87 | 0,44 % |

*Таблиця 3 - Мовний склад населення за даними перепису 2001 р*

| Національність | Кількість | Відсоток |
|---|---|---|
| українці | 41383 | 97,63 % |
| російська | 570 | 1,46 % |
| азербайджанська | 18 | 0,04 % |
| білоруська | 18 | 0,04 % |
| молдовська | 14 | 0,03 % |
| інші | 12 | 0,3 % |

Рис 2.2  Вигляд HTML сторінки без CSS каскаду


Рис 2.3 Java Script

JavaScript (JS) - це високорівнева, інтерпретована мова програмування, яка використовується для створення динамічного контенту на веб-сторінках. JS використовується для реалізації різноманітних функцій, таких як взаємодія з користувачем, анімація, зміна вмісту сторінок, валідація даних та багато іншого.

JS працює на бокові клієнта (у веб-браузері), виконуючи скрипти, які вбудовані безпосередньо в HTML-код сторінки. Він може взаємодіяти з DOM (Document Object Model) - представленням структури HTML-сторінки в браузері,

змінюючи вміст, стилі та структуру сторінки під час її завантаження або після подій, спричинених діями користувача.

JavaScript також використовується для взаємодії з сервером за допомогою AJAX (Asynchronous JavaScript and XML), що дозволяє асинхронно відправляти та отримувати дані без перезавантаження сторінки. Це дозволяє створювати динамічні, інтерактивні веб-додатки, які реагують на дії користувачів без затримок.

JS є однією з основних мов програмування для веб-розробки, разом з HTML і CSS. Він дозволяє розробникам створювати різноманітні веб-додатки, від простих інтерактивних веб-сайтів до складних односторінкових додатків (SPA) та мобільних додатків за допомогою фреймворків, таких як React, Angular та Vue.js.


Рис 2.4 Node.js

Node.js - це виконавче середовище JavaScript, яке дозволяє виконувати JavaScript код на серверній стороні. Він побудований на базі двигуна V8 JavaScript, який розроблений Google для браузера Chrome. Одним з головних принципів Node.js є неблокуюча, подієва модель вводу/виводу, що дозволяє ефективно обробляти багато запитів одночасно.

Node.js працює на заснованій на подіях архітектурі, де сервер реагує на події (наприклад, отримання запиту від клієнта або завершення операції вводу/виводу) замість того, щоб очікувати блокуючі операції. Це дозволяє Node.js

ефективно використовувати один потік виконання для обробки багатьох запитів одночасно, що робить його ідеальним для високонавантажених застосунків.

Node.js використовується для створення різноманітних серверних додатків, веб-серверів, мережевих застосунків, API і багатьох інших. Він дозволяє розробникам використовувати JavaScript як мову програмування і на серверній стороні, і на клієнтській, що дозволяє створювати повністю інтегровані додатки зі спільним кодом.

Крім того, Node.js має велику екосистему зовнішніх модулів (написаних на JavaScript), доступ до яких можна отримати через менеджер пакетів npm. Це робить Node.js потужним і гнучким інструментом для розробки різноманітних додатків.


Рис 2.5 Express.js

Express.js - це мінімалістичний та гнучкий веб-фреймворк для Node.js, який дозволяє швидко створювати веб-додатки та API. Він надає простий, але потужний набір функцій для реалізації різних задач на стороні сервера.

Express.js працює на основі шляхів (routes) та обробників подій (event handlers), які виконуються при завантаженні відповідного URL. Зазвичай розробники визначають маршрути, які співпадають з конкретними URL-адресами, та пов'язують їх з функціями обробки, які виконуються при зверненні до цих маршрутів. Це дозволяє розробникам ефективно керувати різними запитами і відповідями на сервері.

Express.js надає ряд корисних вбудованих функцій, таких як обробка HTTP-запитів та відповідей, обробка маршрутів, робота з шаблонами HTML, реалізація middleware (проміжного програмного забезпечення) та багато іншого. Крім того, він має велику кількість сторонніх модулів, які можна використовувати для розширення його функціональності.

Express.js дозволяє розробникам швидко створювати масштабовані веб-додатки та API, зосереджуючись на бізнес-логіці та функціональності додатку, а не на низькорівневих деталях веб-сервера. Він є популярним вибором серед розробників Node.js завдяки своїй простоті, гнучкості та широким можливостям.

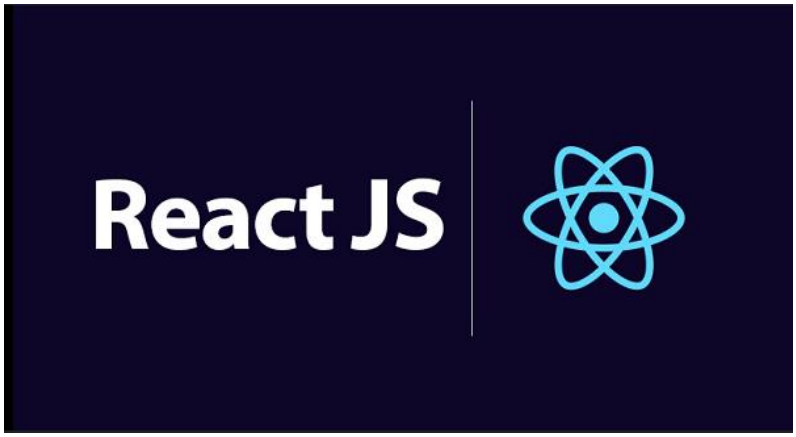



Рис 2.6 React.js

React.js - це відкрита JavaScript бібліотека для створення користувацьких інтерфейсів, що розробляється та підтримується Facebook. React дозволяє розробникам створювати веб-інтерфейси, які складаються з набору компонентів, які можуть бути повторно використані та ефективно оновлювати свій стан при зміні даних.

Робота React полягає в описі стану користувацького інтерфейсу та його оновлення на основі змін даних. React використовує віртуальний DOM (Document Object Model), який представляє собою віртуальне представлення реального DOM дерева веб-сторінки. Коли стан компоненту React змінюється, React перерисовує тільки необхідні частини віртуального DOM, а не всю сторінку. Після цього він порівнює віртуальний DOM з реальним та виконує тільки необхідні зміни, що робить процес оновлення швидшим та ефективнішим.

React також використовує JSX (JavaScript XML), що є розширенням JavaScript, що дозволяє вбудовувати HTML-подібний код безпосередньо в JavaScript. Це робить код React більш зрозумілим та легким для розробки.

React використовується для створення інтерактивних, динамічних веб-додатків та інтерфейсів, таких як соціальні мережі, ігри, сторінки з великим об'ємом даних, адмін-панелі та багато іншого. Він дозволяє розробникам швидко створювати потужні та ефективні веб-додатки, зосереджуючись на компонентах та їх взаємодії, а не на низькорівневих деталях маніпулювання DOM.

## 2.2 Інструменти для передачі даних між пристроями через мережу

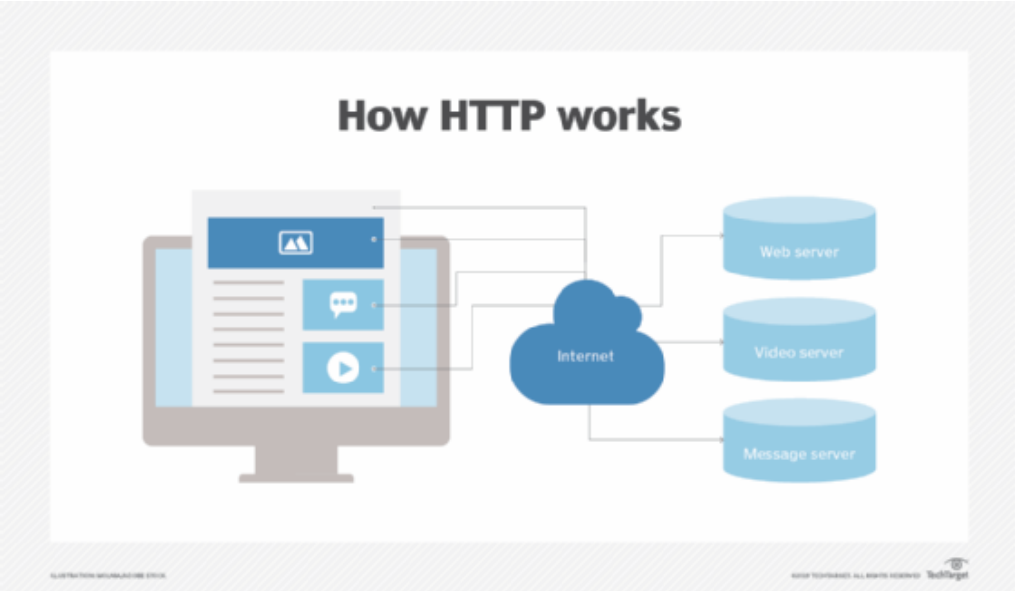



Рис 2.7 Схема роботи HTTP

HTTP (Hypertext Transfer Protocol) - це протокол передачі даних, який використовується для забезпечення комунікації між клієнтом та сервером в Інтернеті. HTTP визначає правила та формати, за якими дані передаються через мережу, забезпечуючи стандартизований спосіб обміну інформацією.

Робота HTTP базується на моделі клієнт-сервер, де клієнт (наприклад, веб-браузер) ініціює запит до сервера (наприклад, веб-сервера), щоб отримати або відправити дані. Клієнт формує запит у вигляді HTTP-запиту, який містить метод

запиту (наприклад, GET, POST, PUT, DELETE), адресу ресурсу та інші параметри.

Сервер отримує запит, обробляє його та повертає відповідь у вигляді HTTP-відповіді. Ця відповідь містить статус виконання запиту (наприклад, успішно чи невдача), заголовки, що містять метадані про відповідь, та, можливо, тіло, яке містить самі дані.

HTTP використовує TCP/IP для передачі даних через мережу. Він використовує порт 80 для незашифрованих з'єднань і порт 443 для з'єднань, зашифрованих за допомогою протоколу HTTPS.

HTTP використовується для реалізації різних сценаріїв в Інтернеті, таких як отримання веб-сторінок, відправка та отримання даних веб-службами, передача файлів, відправка електронної пошти та багато іншого. Він є основою для взаємодії між клієнтом та сервером в Інтернеті і грає ключову роль у веб-розробці та передачі даних в мережі.

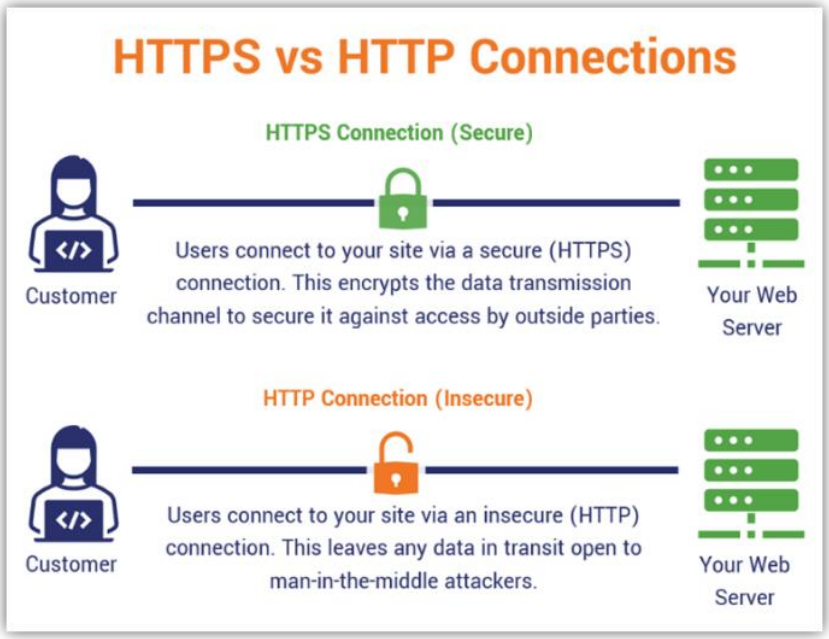



Рис 2.8 Схема роботи HTTPS

HTTPS (Hypertext Transfer Protocol Secure) - це захищена версія протоколу HTTP, яка використовує шифрування для захисту конфіденційності та цілісності передачі даних між клієнтом та сервером. HTTPS використовує шифрувальні

алгоритми для забезпечення безпеки передачі інформації через мережу Інтернету. HTTPS є стандартом безпеки для передачі конфіденційної інформації через Інтернет, такої як особисті дані, фінансові дані, паролі тощо. Він є важливим для захисту веб-додатків, електронної комерції, онлайн-платежів та будь-яких інших сервісів, які вимагають захисту конфіденційності користувачів і цілісності переданих даних.

Основні особливості та принципи роботи HTTPS:

1. Шифрування даних: Однією з основних особливостей HTTPS є шифрування даних, що передаються між клієнтом і сервером. Це забезпечує конфіденційність інформації, що передається, та захищає її від несанкціонованого доступу третіх осіб.
2. Аутентифікація сервера: HTTPS використовує цифрові сертифікати для аутентифікації серверів і підтвердження їхньої ідентичності. Клієнт може перевірити дійсність сертифіката сервера і переконатися, що він спілкується з правильним сервером, а не з підробленим.
3. Захист від перехоплення даних: Використання HTTPS мінімізує ризик перехоплення або зміни переданих даних під час їхньої передачі через мережу. Шифрування даних забезпечує їх цілісність та конфіденційність, що робить надійною взаємодію між клієнтом та сервером.
4. Використання TLS (Transport Layer Security): HTTPS базується на протоколі TLS, який забезпечує безпеку передачі даних на рівні транспортного рівня мережі. TLS включає в себе різні криптографічні протоколи та алгоритми для шифрування даних та забезпечення безпеки з'єднання.
5. Підтримка інших функцій безпеки: HTTPS дозволяє використовувати різні механізми безпеки, такі як сертифікати, цифрові підписи, аутентифікація клієнта, захист від перехоплення куки (cookie) тощо.

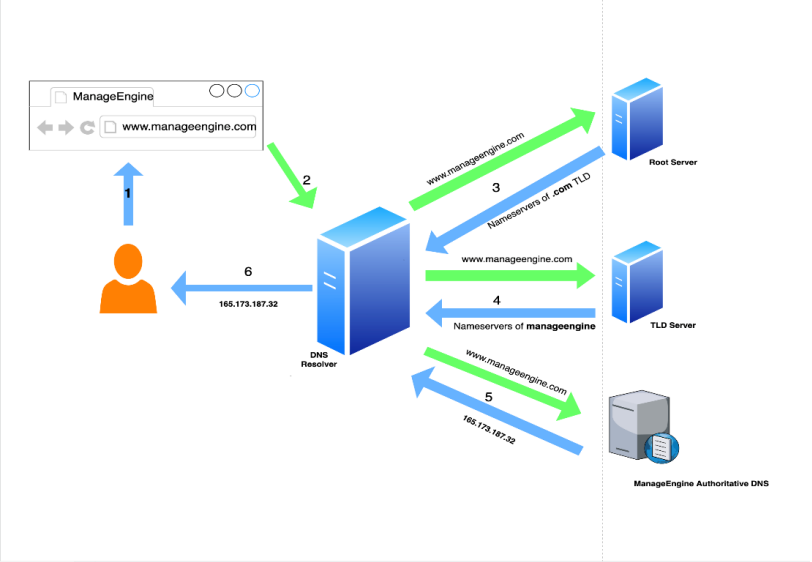



Рис 2.9 Схема роботи DNS

DNS (Domain Name System) - це система, що перетворює доменні імена в IP-адреси та навпаки. Вона дозволяє користувачам зручно використовувати зрозумілі доменні імена для доступу до ресурсів в Інтернеті, замість запам'ятовування числових IP-адрес.

DNS працює наступним чином: коли користувач вводить доменне ім'я веб-сайту, його браузер відправляє запит на DNS-сервер, який знаходить відповідну IP-адресу для цього доменного імені. Якщо DNS-сервер знає IP-адресу, він повертає її браузеру, і той може встановити з'єднання з веб-сервером. Якщо DNS-сервер не має інформації про це доменне ім'я, він направляє запит на розпізнавання до інших DNS-серверів в Інтернеті.

DNS також використовується для керування різними типами записів, такими як записи MX для пошти, CNAME для перенаправлення доменів, SPF для встановлення правил аутентифікації електронної пошти та іншими. В цілому, DNS є ключовою складовою Інтернету, оскільки він забезпечує зручний та ефективний доступ до ресурсів за допомогою доменних імен.

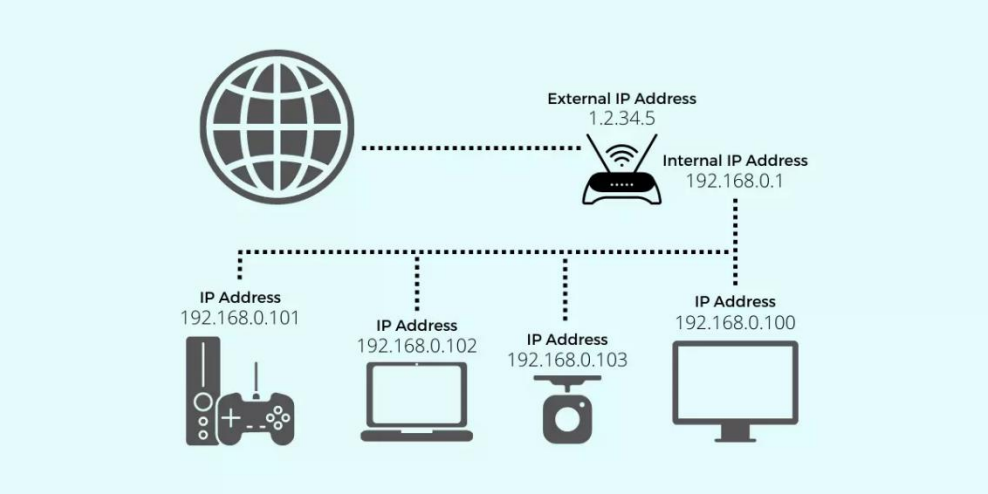



Рис 2.10 Схема роботи IP

IP (Internet Protocol) - це протокол, який визначає правила адресації та маршрутизації даних в мережі Інтернет. Кожен пристрій в мережі має свою унікальну IP-адресу, яка ідентифікує його в мережі.

IP працює наступним чином: кожен пакет даних, який надсилається через мережу, маркується IP-адресою відправника та IP-адресою отримувача. Коли пакет даних відправляється з одного пристрою до іншого, він проходить через маршрутизатори (роутери), які використовують IP-адреси для визначення шляху, по якому пакет повинен йти. Кожен маршрутизатор приймає пакет, перевіряє його адресу та вирішує, куди його відправити далі, доки пакет не дійде до свого пункту призначення.

IP не лише визначає адреси для маршрутизації даних, але також включає різні версії, такі як IPv4 та IPv6, а також низку протоколів, які працюють разом з IP для забезпечення функціональності Інтернету, таких як ICMP (Internet Control Message Protocol) для передачі повідомлень про помилки та стан мережі, інші протоколи для маршрутизації, а також протоколи верхніх рівнів, такі як TCP та UDP, які використовуються для передачі даних на вищих рівнях стеку протоколів Інтернету.

Узагальнено кажучи, IP є основою мережевої комунікації в Інтернеті, дозволяючи пристроям зв'язуватися та обмінюватися даними через глобальну мережу.

## 2.3 Ключові аспекти розробки веб-застосунку

Специфікація розробки дизайну веб-застосунку для пошуку нерухомості повинна охоплювати кілька ключових аспектів, щоб забезпечити ефективність, зручність і привабливість для користувачів. Основна мета полягає в створенні інтуїтивно зрозумілого і функціонального інтерфейсу, який полегшить процес пошуку, перегляду та вибору нерухомості.

Першочергово слід врахувати, що дизайн має бути максимально простим і зрозумілим. Інтерфейс повинен мати чітку і логічну структуру, що дозволить користувачам швидко знаходити необхідну інформацію та здійснювати необхідні дії. Це досягається через використання добре продуманих меню, зрозумілих іконок та легкої навігації.

Важливою частиною є розробка головної сторінки, яка має привертати увагу користувачів та забезпечувати швидкий доступ до основних функцій застосунку. На головній сторінці доцільно розмістити пошукову панель з фільтрами, яка дозволить користувачам вводити ключові параметри пошуку, такі як місце розташування, тип нерухомості, ціна тощо. Також важливо включити популярні категорії і нові пропозиції, що можуть зацікавити відвідувачів.

Дизайн карток оголошень повинен бути інформативним та привабливим. Кожна картка має містити ключову інформацію про об'єкт: фотографії, ціну, місце розташування, основні характеристики та короткий опис. Фотографії мають бути високої якості та відображати реальний стан об'єкту. Важливо забезпечити можливість швидкого перегляду деталей оголошення та контакту з продавцем або орендодавцем.

Особливу увагу слід приділити адаптивності дизайну. Веб-застосунок має коректно відображатися на різних пристроях і екранах, включаючи комп'ютери, планшети та смартфони. Це досягається через використання адаптивного дизайну, який автоматично підлаштовується під розмір екрану користувача, зберігаючи зручність і функціональність інтерфейсу.

Крім того, слід забезпечити високий рівень доступності для користувачів з обмеженими можливостями. Використання контрастних кольорів, зрозумілих шрифтів і можливість навігації за допомогою клавіатури – важливі елементи, що підвищують доступність веб-застосунку.

Загальна естетика дизайну повинна бути сучасною і привабливою, використовуючи приємну кольорову гамму, чіткі лінії та простоту у візуальних елементах. Важливо дотримуватися єдиного стилю у всіх елементах дизайну, що створює професійний вигляд і сприяє довірі користувачів.

Варто не забувати й про верифікацію. Верифікація на веб-сайтах необхідна для захисту облікових записів користувачів та боротьби зі спамом і шахрайством. Цей процес допомагає підтвердити ідентичність користувача та уникнути створення фальшивих або бот-акаунтів, що можуть використовуватися для небажаних дій, таких як розсилка спаму або зловмисне поведінка.

Верифікації в веб-застосунках можуть включати:


Рис. 2.11 Приклад листа верифікації

Електронна пошта або телефонна верифікація: Це один з найпоширеніших методів. Користувачі отримують спеціальне посилання або код підтвердження на свою електронну адресу або мобільний телефон, який потрібно ввести для підтвердження свого облікового запису.

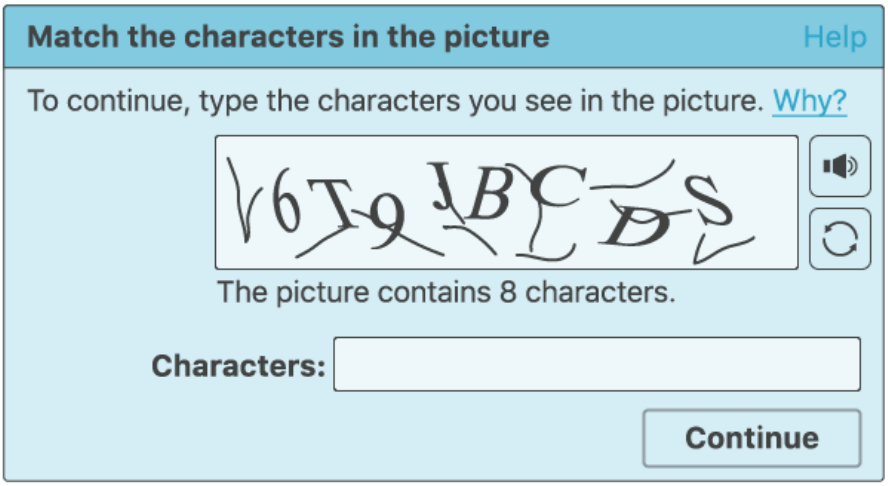



Рис. 2.12 Приклад капчі

Використання капчі: Деякі веб-застосунки використовують капчу для перевірки, що користувач - це людина, а не автоматизований бот. Капча може включати введення тексту, вирішення математичних завдань або візуальне визначення об'єктів на зображеннях.

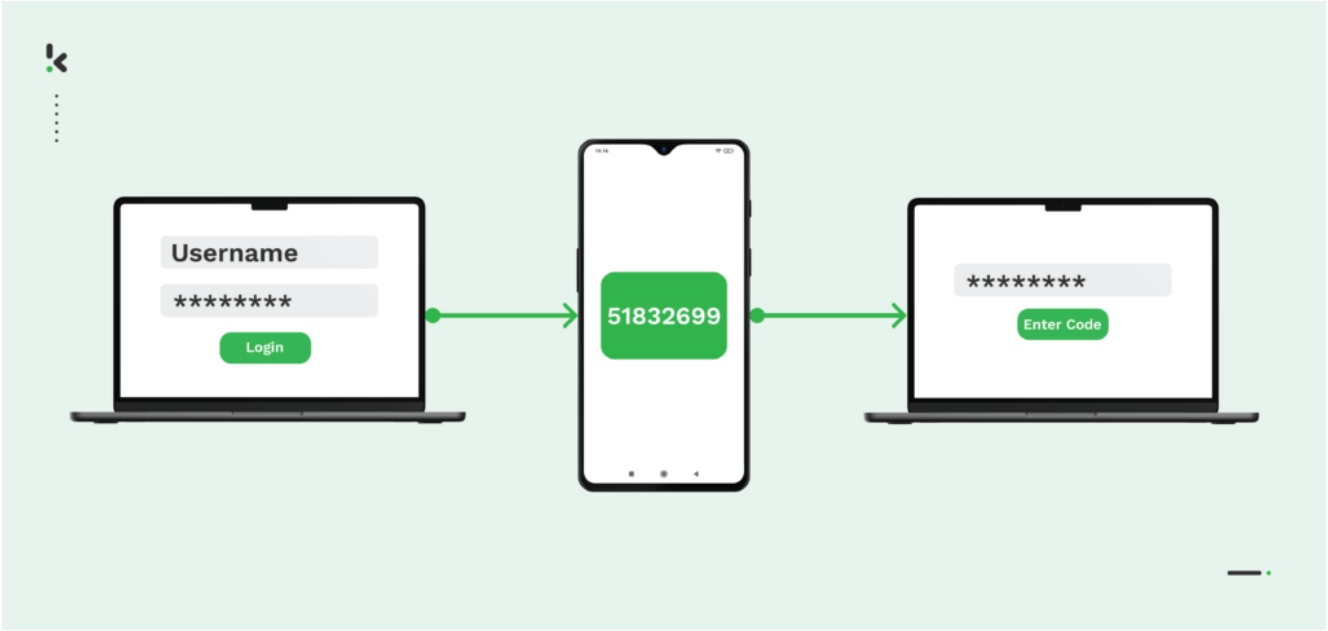



Рис 2.13 Приклад двофакторної аутентифікації

Системи двофакторної аутентифікації (2FA): Цей метод передбачає введення не тільки пароля, а й додаткового підтвердження, такого як одноразовий код, який надсилається на мобільний телефон або електронну пошту. Це забезпечує додатковий рівень безпеки.

Перевірка через соціальні мережі: Користувачі можуть мати можливість ввійти або зареєструватися за допомогою свого облікового запису в соціальній мережі. Це дає можливість швидко підтвердити особу користувача через вже підтверджений обліковий запис у Facebook, Google або інших соціальних мережах.

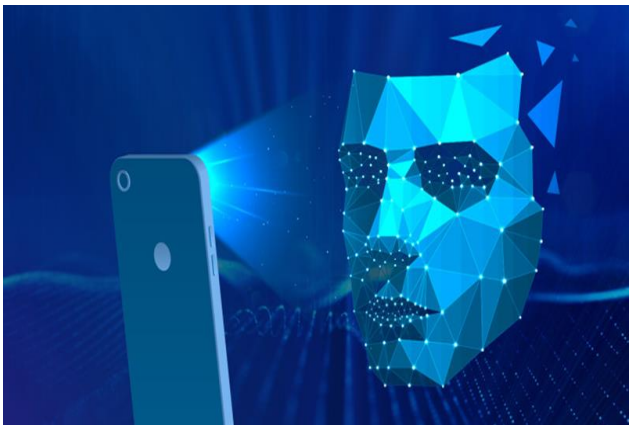


Рис 2.14 та 2.15 Приклади біометричної аутентифікації

Біометрична аутентифікація: Деякі веб-застосунки можуть підтримувати біометричну аутентифікацію, таку як відбитки пальців або сканування обличчя, як метод підтвердження ідентичності користувача.

Кожен з цих методів має свої переваги та недоліки, і вибір конкретного методу зазвичай залежить від потреб і вимог конкретної веб-платформи та її користувачів.

Валідація у веб-застосунках - це процес перевірки даних, які вводять користувачі, на відповідність певним критеріям або правилам. Вона забезпечує правильність та цілісність інформації, що надходить у систему. Валідація може включати перевірку формату електронної пошти, правильності введення номера телефону, обов'язковість заповнення певних полів у формах, перевірку наявності небажаних символів або відсутність вразливостей у введених даних. Вона допомагає уникнути помилок, запобігає атакам на систему та забезпечує зручність користування веб-застосунком.

Особливості валідації у веб-застосунках включають:

Перевірка формату даних: Наприклад, перевірка правильності введення електронної адреси, номера телефону, поштового індексу та інших форматів даних.

Перевірка на обов'язковість: Валідація може вимагати заповнення обов'язкових полів у формах, щоб уникнути відсутності важливої інформації.

Виявлення дублікатів: У деяких випадках важливо виявити та запобігти введенню дублікатів даних, щоб уникнути непередбачуваних помилок у системі.

Захист від небажаних символів: Валідація може включати перевірку наявності небажаних символів, таких як HTML-теги або скрипти, що можуть бути використані для атак на систему (Cross-Site Scripting).

Перевірка на валідність даних: Валідація може включати перевірку на відповідність даних певним правилам або діапазонам, наприклад, перевірка правильності введення дати, числа або іншої інформації.

Перевірка на відповідність зовнішнім стандартам: У деяких випадках може бути важливо перевірити, чи відповідають введені дані певним зовнішнім стандартам або вимогам, наприклад, перевірка введення адреси доставки за стандартами країни.

Ці особливості валідації допомагають забезпечити правильність та надійність введених даних у веб-застосунках і зменшити ризик виникнення помилок та безпекових проблем.

Рис. 2.16 Приклад фізичного сервера.

Сервер - це комп'ютер або програмне обладнання, що надає послуги і ресурси іншим комп'ютерам або програмам, відомим як клієнти. Основна функція сервера полягає у відповіді на запити клієнтів, забезпеченні доступу до різних ресурсів і даних, а також обробці і передачі інформації. Сервери можуть виконувати різноманітні завдання, такі як зберігання та обмін даними, надання веб-сторінок для перегляду в браузері, обробка електронних листів, керування базами даних, забезпечення доступу до спільних файлів у мережі та багато іншого. Усі ці функції дозволяють клієнтам отримувати доступ до необхідних ресурсів і послуг через мережу інтернет або локальну мережу.

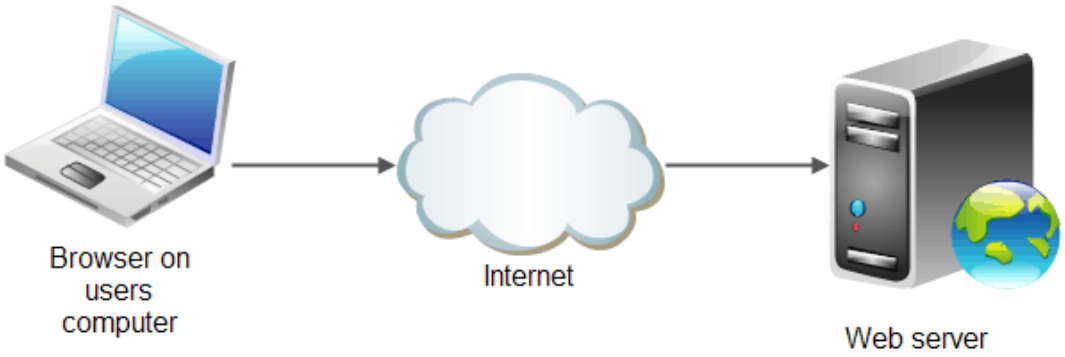



Рис. 2.17 Схема роботи сервера.

Існує кілька різних типів серверів, кожен з яких має свої особливості та використання. Один з найпоширеніших типів - це веб-сервери, які використовуються для надання веб-сторінок для перегляду в браузері. Поштові сервери відповідають за обмін електронними листами. Файлові сервери використовуються для зберігання та обміну файлами, дозволяючи користувачам спільно працювати над документами. Бази даних обслуговуються серверами баз даних, які забезпечують зберігання, доступ та керування даними. Також існують сервери додатків, які виконують програмні застосунки та послуги для користувачів, такі як чат-сервери, відео-конференційні сервери та інші. Кожен тип сервера має свої власні характеристики і призначення, що визначаються його функціональністю та вимогами до ресурсів.

Апаратне забезпечення серверів - це фізичне обладнання, яке використовується для підтримки роботи серверних систем. Сервери зазвичай мають значно більшу потужність та надійність, ніж персональні комп'ютери, оскільки вони призначені для обробки великих обсягів даних та обслуговування багатьох користувачів одночасно. Апаратне забезпечення серверів може включати в себе потужні процесори, великий обсяг оперативної пам'яті, швидкі та надійні накопичувачі даних, такі як твердотільні накопичувачі або RAID-масиви, а також спеціалізоване мережеве обладнання, що забезпечує швидкий та безперебійний доступ до мережі. Деякі сервери можуть мати спеціальні апаратні засоби для захисту даних та забезпечення безперебійної роботи, такі як дублювання живлення або системи резервного копіювання.

# 3. ПРОГРАМНА РЕАЛІЗАЦІЯ ВЕБ-ЗАСТОСУНКУ

## 3.1 Проектування веб-застосунку

Для розробки плану веб-застосунку спочатку потрібно провести збір вимог та визначити основні цілі проекту. Потім можна розділити проект на етапи розробки, визначити терміни виконання та встановити пріоритети для кожного етапу. Важливо також урахувати ресурси, необхідні для виконання кожного етапу, включаючи людські, фінансові та технічні ресурси. План повинен бути реалістичним, з урахуванням можливих ризиків та варіантів дій у разі виникнення непередбачених ситуацій. Крім того, важливо встановити механізми відстеження прогресу та звітування про виконання завдань на кожному етапі розробки.

План веб-додатку:

1. Створення основи веб-застосунку.
2. Створення будови сайту:
    1. Шапка веб-застосунку.
    2. Основний контент:
        1. Рекламний банер.
        2. Функції підбору нерухомості.
        3. Оголошення.
        4. Список обраного.
    3. Додаткова сторінка з детальною інформацію.
        1. Деталі оголошення.
        2. Форма контакту з орендодавцем.
    4. Нижній колонтитул веб-сторінки.

Під час проектування веб-застосунку можуть виникати різноманітні проблеми, які можуть ускладнити процес розробки та вплинути на якість кінцевого продукту. Одна з таких проблем - це неоднорідні вимоги користувачів. Різні групи користувачів можуть мати різні потреби та очікування щодо

функціональності та дизайну веб-застосунку. Наприклад, деякі користувачі можуть прагнути до максимальної простоти та інтуїтивності інтерфейсу, тоді як інші можуть бажати більш складних можливостей та розширених можливостей.

Суперечливі вимоги до продукту є ще однією проблемою, з якою можна зіткнутися. Замовник може мати неоднозначні або суперечливі вимоги до функціональності, дизайну чи інших аспектів веб-застосунку. Це може призвести до затримок у розробці та невизначеності щодо кінцевого результату.

Технічні обмеження також можуть ускладнити розробку веб-застосунку. Наприклад, обмеження щодо потужності сервера, підтримки браузерів, доступності API або інші технічні обмеження можуть обмежити можливості розробників та вплинути на швидкість, ефективність та стабільність веб-застосунку.

Проблеми безпеки є важливим аспектом розробки веб-застосунку. Недостатня увага до захисту від різних видів атак, таких як SQL-ін'єкції, Cross-Site Scripting (XSS) або витоки даних, може призвести до серйозних проблем із безпекою та вразливостями веб-застосунку.

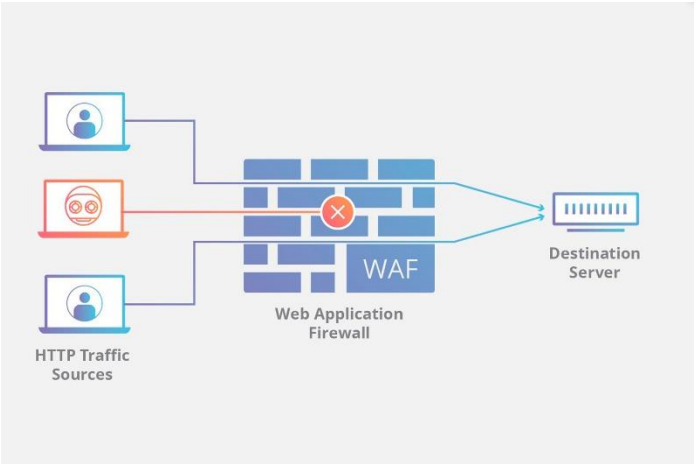



Рис. 3.1 Схема роботи захисту firewall.

Проблеми масштабованості також можуть виникнути, особливо якщо веб-застосунок стрімко зростає у популярності. Збільшення кількості користувачів може призвести до нестабільної роботи, затримок у завантаженні сторінок та інших проблем, які потребують уваги та виправлення.

Недостатнє уваження до дизайну та користувацького досвіду також може вплинути на успіх веб-застосунку. Поганий дизайн, незручний інтерфейс або недостатні функціональність можуть відлякувати користувачів та призвести до низької активності на сайті.

Уникнення та вирішення цих проблем може вимагати від розробників та команди проекту великих зусиль та уваги до деталей під час усього процесу розробки веб-застосунку.

Основа сайту на HTML, CSS та JavaScript складається з різних компонентів, які взаємодіють між собою для створення функціонального та привабливого веб-додатку. HTML відповідає за структуру та контент сторінок, визначаючи заголовки, параграфи, списки, посилання та інші елементи. CSS використовується для оформлення та стилізації веб-сторінок, включаючи кольори, шрифти, відступи, рамки та інші візуальні ефекти. JavaScript надає інтерактивність, динамічність та функціональність, дозволяючи взаємодіяти з користувачем, створювати анімацію, обробляти події та виконувати різноманітні завдання без перезавантаження сторінки. Взаємодія цих трьох мов дозволяє створити зручний, естетичний та функціональний веб-додаток для користувачів.

Для варіантів фільтрації та сортування в застосунку для пошуку нерухомості можна врахувати кілька можливостей. Перш за все, фільтрація може бути здійснена за різними параметрами, такими як місцезнаходження, тип нерухомості (наприклад, квартира, будинок, земельна ділянка), кількість кімнат, площа, ціновий діапазон тощо. Також можуть бути використані фільтри за характеристиками будівлі (наприклад, наявність балкона, гаража, ліфта) або обслуговуванням (наприклад, наявність парковки, безпеки, водопостачання). Щодо сортування, користувачі можуть мати можливість сортувати результати пошуку за ціною, датою розміщення оголошення, рейтингом об'єкту, розташуванням тощо. Комбінування різних фільтрів та варіантів сортування дозволить користувачам ефективно знаходити нерухомість, яка відповідає їхнім потребам і вимогам.

Діаграма варіантів використання (Use Case Diagram) є одним з основних інструментів, що використовується в об'єктно-орієнтованому аналізі та проектуванні програмного забезпечення. Вона допомагає візуалізувати функціональність системи з точки зору користувача. Основна мета цієї діаграми полягає в тому, щоб показати взаємодію між користувачами (акторами) та системою, а також описати різні сценарії використання системи.

Актори на діаграмі варіантів використання представляють зовнішні сутності, які взаємодіють із системою. Це можуть бути як реальні користувачі, так і інші системи або підсистеми. Вони позначаються у вигляді простих фігур (звичайно, це людські фігурки).

Варіанти використання (use cases) — це конкретні дії або послуги, які система надає акторам. Вони описуються овальними фігурами і містять назви, що відображають сутність дії (наприклад, "Зареєструватися", "Увійти в систему", "Оформити замовлення"). Кожен варіант використання пов’язаний з актором за допомогою ліній, що показують, хто ініціює або взаємодіє з певною функціональністю.

Важливим аспектом діаграми є можливість показувати відносини між різними варіантами використання, зокрема, включення (include) та розширення (extend). Включення означає, що один варіант використання обов'язково включає поведінку іншого, тоді як розширення показує, що варіант використання може розширювати поведінку іншого у певних умовах.

Діаграма варіантів використання корисна на початкових етапах розробки програмного забезпечення, оскільки вона допомагає розробникам і замовникам узгодити очікування щодо функціональності системи. Вона сприяє кращому розумінню вимог і забезпечує основу для детальнішого аналізу та проектування.

Цей тип діаграми є частиною мови моделювання UML (Unified Modeling Language) і широко використовується в інженерії програмного забезпечення для документування та комунікації вимог до системи.

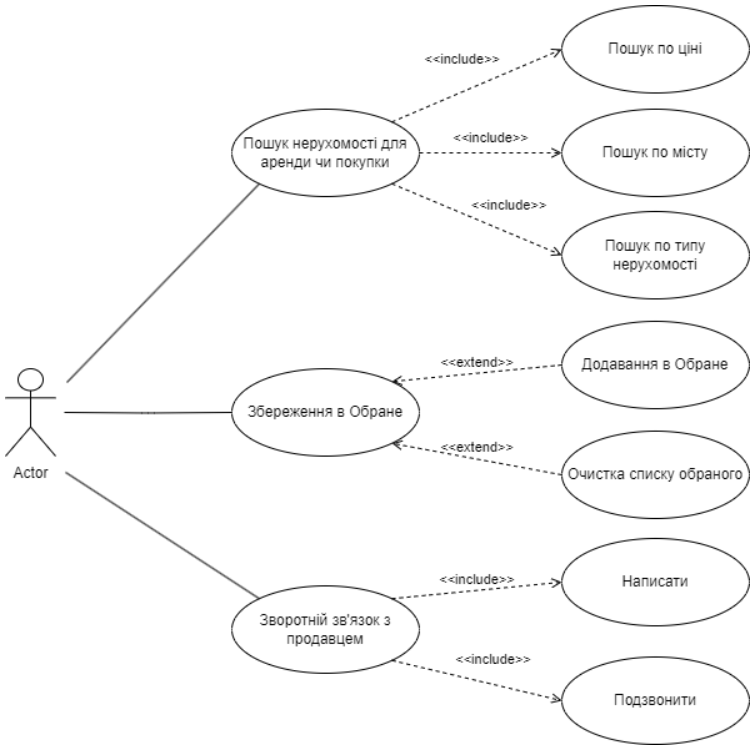



Рис. 3.2 Діаграма варіантів використання

Схема бази даних – це структурний опис бази даних, який визначає, як дані організовані і як вони взаємопов'язані. Вона містить всі логічні структури, які використовуються для зберігання та управління даними, включаючи таблиці, стовпці, індекси, ключі та відношення між таблицями. Схема бази даних описує типи даних, що зберігаються, а також їхні атрибути, обмеження та правила цілісності.

Ключовими елементами схеми бази даних є таблиці, які являють собою колекції даних, організованих у вигляді рядків і стовпців. Кожен стовпець має певний тип даних (наприклад, текстовий, числовий, дата) і може мати додаткові обмеження, такі як унікальність або обов’язковість заповнення. Рядки таблиць представляють індивідуальні записи даних.

Для забезпечення цілісності даних у схемі бази даних використовуються ключі. Первинний ключ (primary key) – це унікальний ідентифікатор кожного рядка в таблиці. Зовнішні ключі (foreign keys) встановлюють зв’язки між таблицями, дозволяючи зв'язувати дані в різних таблицях та підтримувати референціальну цілісність.

Схема бази даних також включає індекси, які підвищують продуктивність запитів шляхом швидкого пошуку та сортування даних. Індекси створюються на основі одного або кількох стовпців таблиці і допомагають оптимізувати виконання операцій вибірки.

Крім того, схема може містити уявлення (views), які є віртуальними таблицями, що представляють собою результат запиту. Уявлення спрощують доступ до даних і можуть використовуватися для обмеження доступу до певних частин даних.

Схема бази даних є важливою частиною проєктування бази даних, оскільки вона визначає логічну структуру даних і забезпечує основу для реалізації системи управління базами даних (СУБД). Вона допомагає розробникам зрозуміти, як дані будуть організовані, і забезпечує ефективний та надійний спосіб зберігання і доступу до даних.

Таким чином, схема бази даних є критично важливим документом у процесі розробки інформаційних систем, оскільки вона визначає, як дані будуть організовані, збережені та використовувані в додатках.

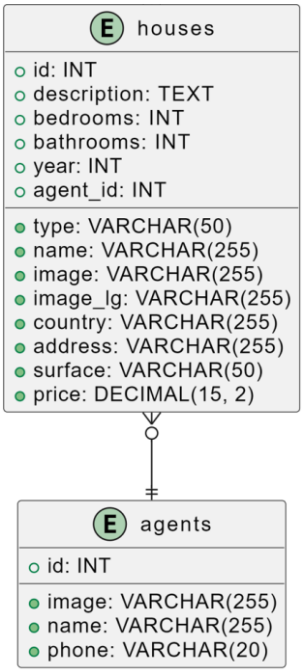



Рис. 3.3. База даних проекту.

Карта веб-застосунку — це візуальне представлення структури веб-сайту або веб-застосунку, яке показує всі сторінки та їхні взаємозв'язки. Вона слугує планом, що дозволяє розробникам, дизайнерам і зацікавленим сторонам зрозуміти, як організований вміст і як користувачі можуть взаємодіяти з різними частинами застосунку.

Ця карта зазвичай складається з блоків або вузлів, що представляють окремі сторінки або компоненти веб-застосунку. Лінії або стрілки між цими блоками показують, як сторінки пов'язані між собою і якими шляхами користувачі можуть переходити з однієї сторінки на іншу. Це може включати основні навігаційні шляхи, посилання на додаткові ресурси або взаємодії, що виникають при виконанні певних дій.

Карта веб-застосунку є корисним інструментом на етапах планування та проектування, оскільки вона допомагає:

- Визначити логічну структуру сайту або застосунку.
- Забезпечити, щоб всі необхідні сторінки і функції були враховані.
- Показати, як користувачі можуть виконувати основні завдання і досягати своїх цілей.
- Виявити можливі проблеми з навігацією або організацією вмісту.

Завдяки карті веб-застосунку команда розробників може ефективніше співпрацювати і координувати свої дії, а також легко вносити зміни до структури перед початком розробки. Це також спрощує комунікацію з замовниками або іншими зацікавленими сторонами, які можуть не мати технічних знань, але повинні розуміти загальну логіку роботи веб-застосунку.

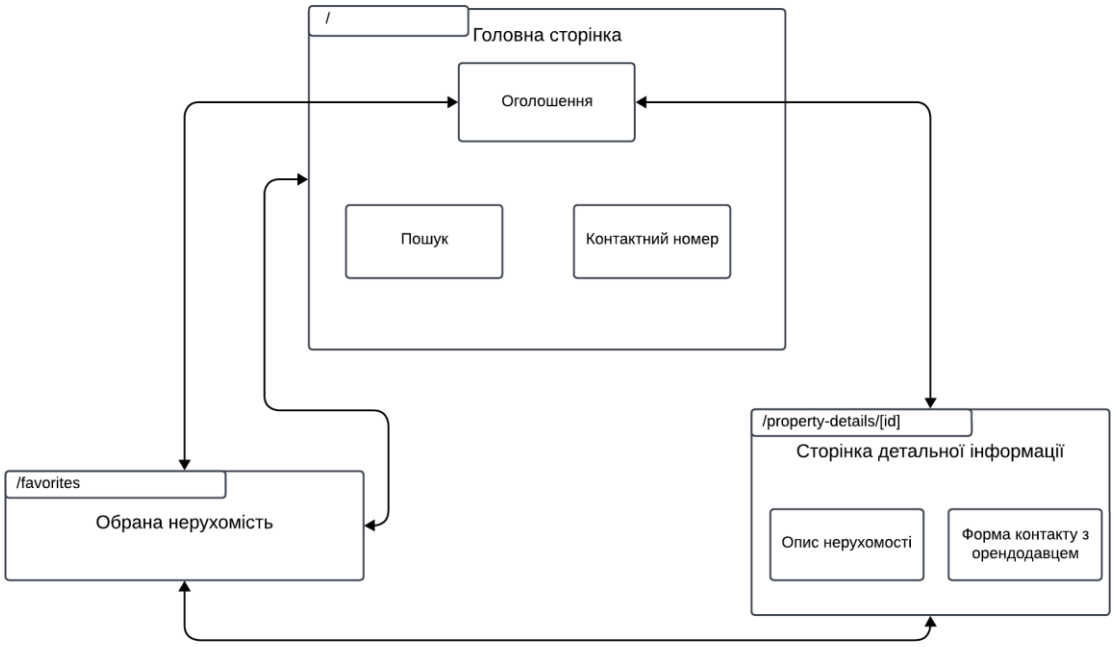



Рис. 3.4. Карта веб-застосунку

## 3.2 Реалізація проекту

Розглянемо основні проблеми з якими зіштовхуються при реалізації програмного забезпечення, а саме:

Проблеми з якістю - недоліки в якості коду, несправності програмного забезпечення або недостатній рівень тестування можуть призвести до проблем з якістю продукту.

Технічні виклики - розробка складних алгоритмів, інтеграція зі сторонніми системами або вирішення технічних проблем можуть вимагати значних зусиль та часу.

Невідповідність вимогам - проекти можуть стикатися з проблемами невідповідності вимогам замовника або клієнта. Це може бути через неправильне розуміння вимог, недостатню комунікацію або зміни вимог під час розробки.

Отже, після підключення всіх необхідних бібліотек, створюємо основу веб-додатку. Для початку будемо створювати шапку веб-застосунку та почнемо з Логотипу сайта, який  матиме назву “Real State” та одразу оформимо його.

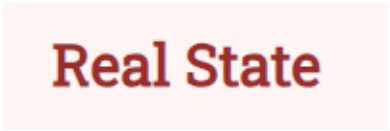



Рис. 3.5 Логотип веб-додатку

На правій частині веб-сторінки реалізуємо одну з вимог до сайту, а саме можливість збереження обраних варіантів, тому створюємо простір для їх збереження та далі розташовуємо контактний номер.

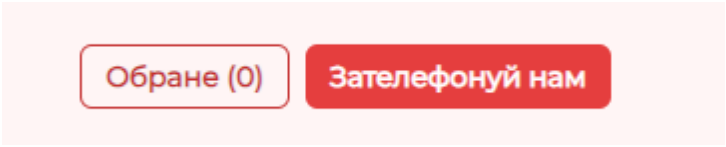



Рис. 3.6 Список обраних та контактні данні

Наступний крок – рекламний банер, створюємо його та наповнюємо його інформацією. Також додаємо декілька світлин.


Рис. 3.7 Рекламний банер

Розроблюємо функції пошуку за фільтром, розділяємо його на три категорії: місцезнаходження, тип нерухомості та вартість та надаємо фільтру візуальне оформлення відповідно до теми сторінки.

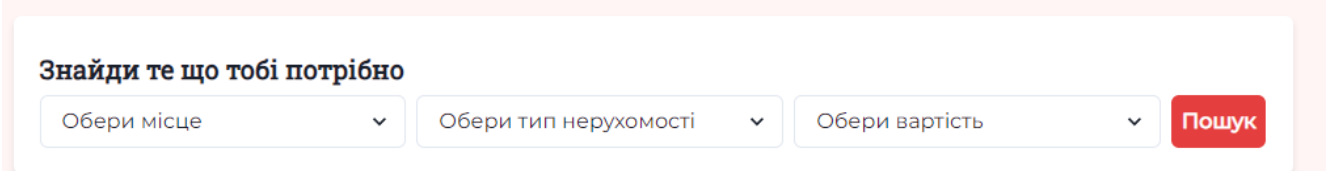



Рис. 3.8 Фільтр

Далі створюємо «бокси» оголошенн та розміщуємо основу інформацію про нерухомість: її зображення, вартість, назву оголошення, місцезнаходження та коротко про наявність спальних кімнат, ванних та площу. Отримуємо такий результат:

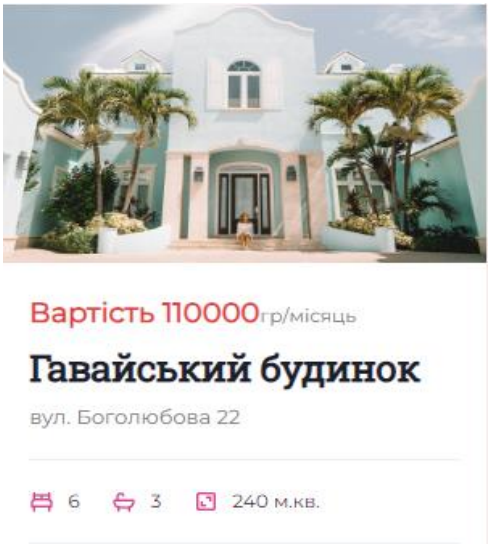



Рис. 3.9 Оголошення

Після цього реалізуємо сторінку детального опису нерухомості та розмістимо колонку з контактними данними орендодавця, надаємо можливість зв’язку з ним.

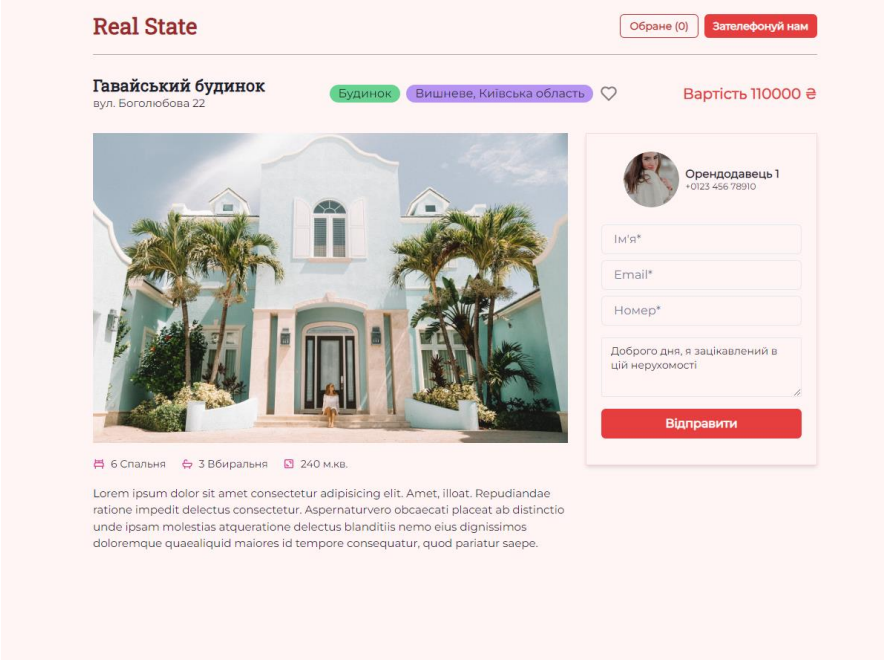



Рис. 3.10 Сторінка детального опису

Далі створюємо сторінку з обраною нерухомістю, розміщуємо під кожним оголошенням кнопку з можливістю додати його до списку. Створюємо можливість перейти на сторінку з детальним описом, видалити варіант окремо та повністю очистити список. Також на кожне оголошення додаємо короткий опис.

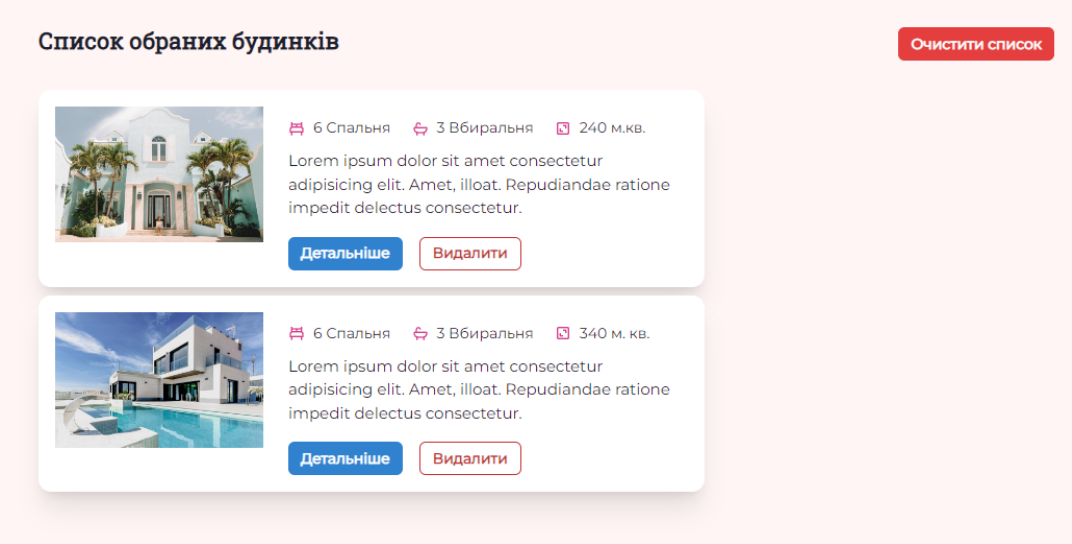



Рис. 3.11 Список обраних

Створючи нижній колонтитул, отримуємо фінальний вигляд проекту:

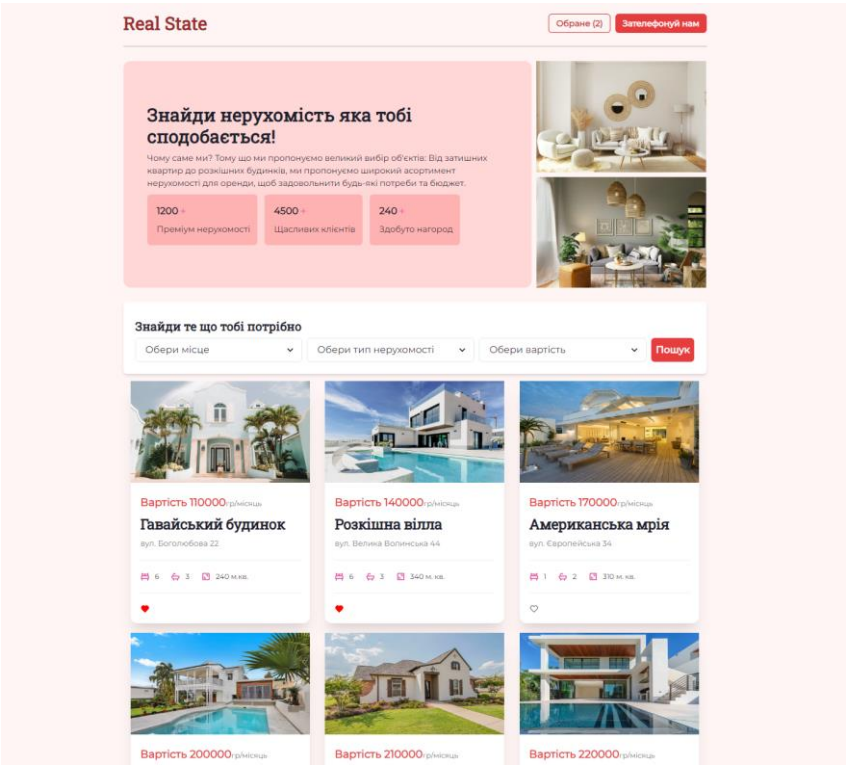



Рис. 3.12 Головна сторінка

Переходимо до тестування веб-застосунку, почнемо з фільтра.

Обираємо в фільтрі місце – Львів, тип – будинок, та ціновий діапазон 110.000 – 140.000 грн, отримуємо бажаний результат.

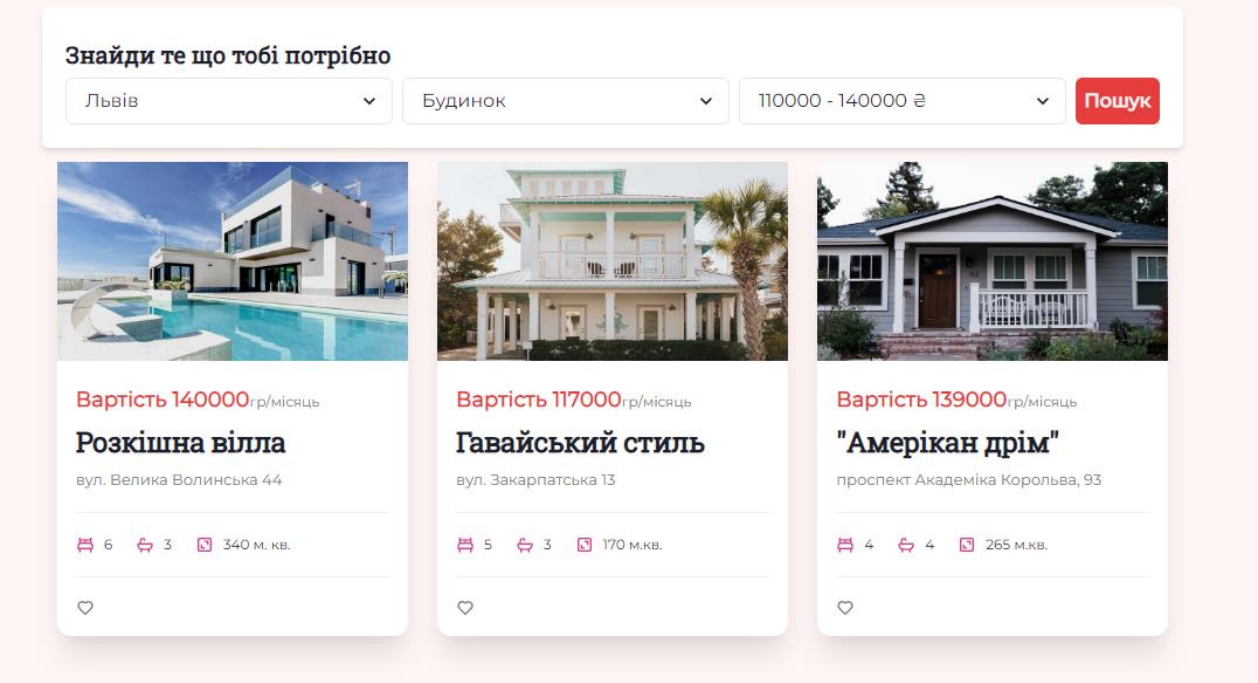



Рис. 3.13 Тестування фільтрів

Також, варто згадати про можливість помилки, або відсутність нерухомості яка підходить користувачу за фільтром, отримуємо повідомлення:

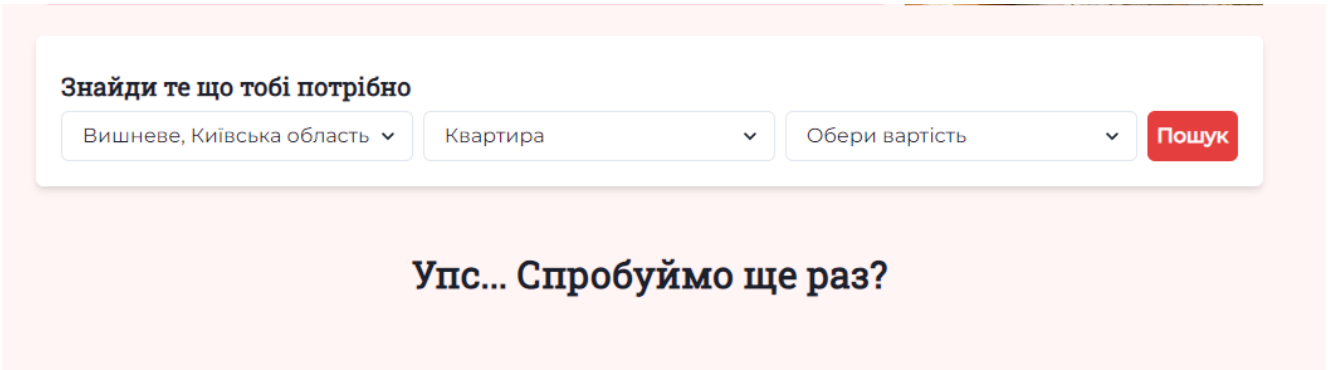



Рис. 3.14 Екран помилки

Далі тестуємо можливість списку обраної нерухомості, для початку тестуємо можливість додавання та прибирання нерухомості зі списку з головної сторінки застосунку.

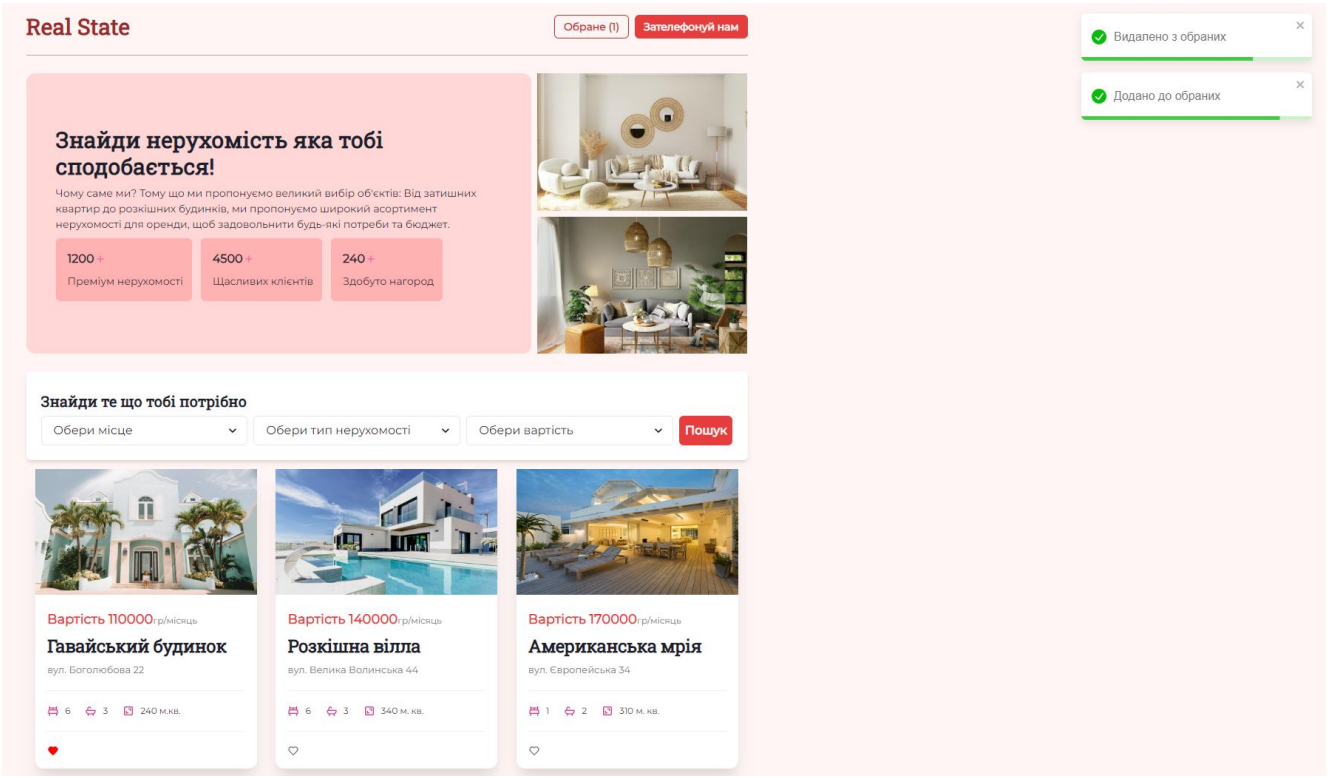



Рис. 3.15 Тестування функції обраної нерухомості

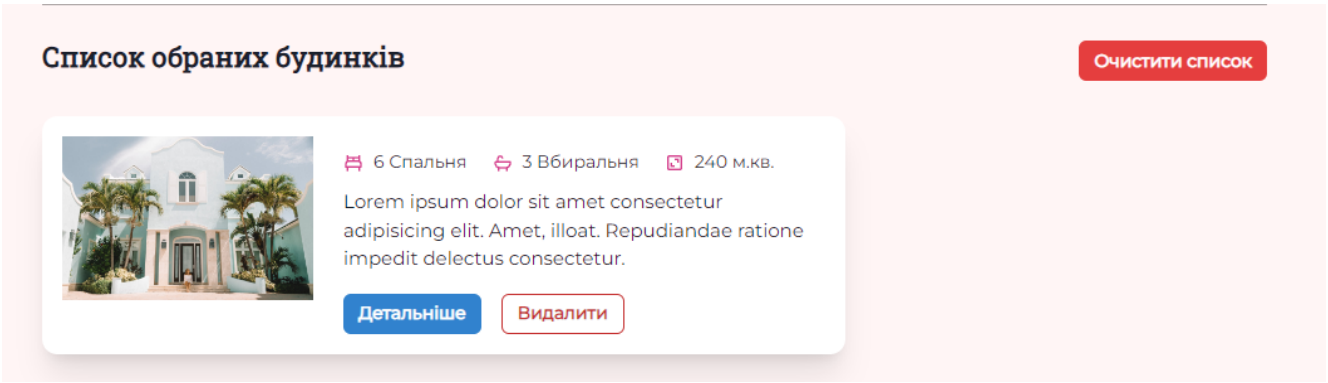



Рис. 3.16 Тестування функції обраної нерухомості

Далі тестуємо можливість окремого видалення збереженого оголошення та можливість повністю очистити список.

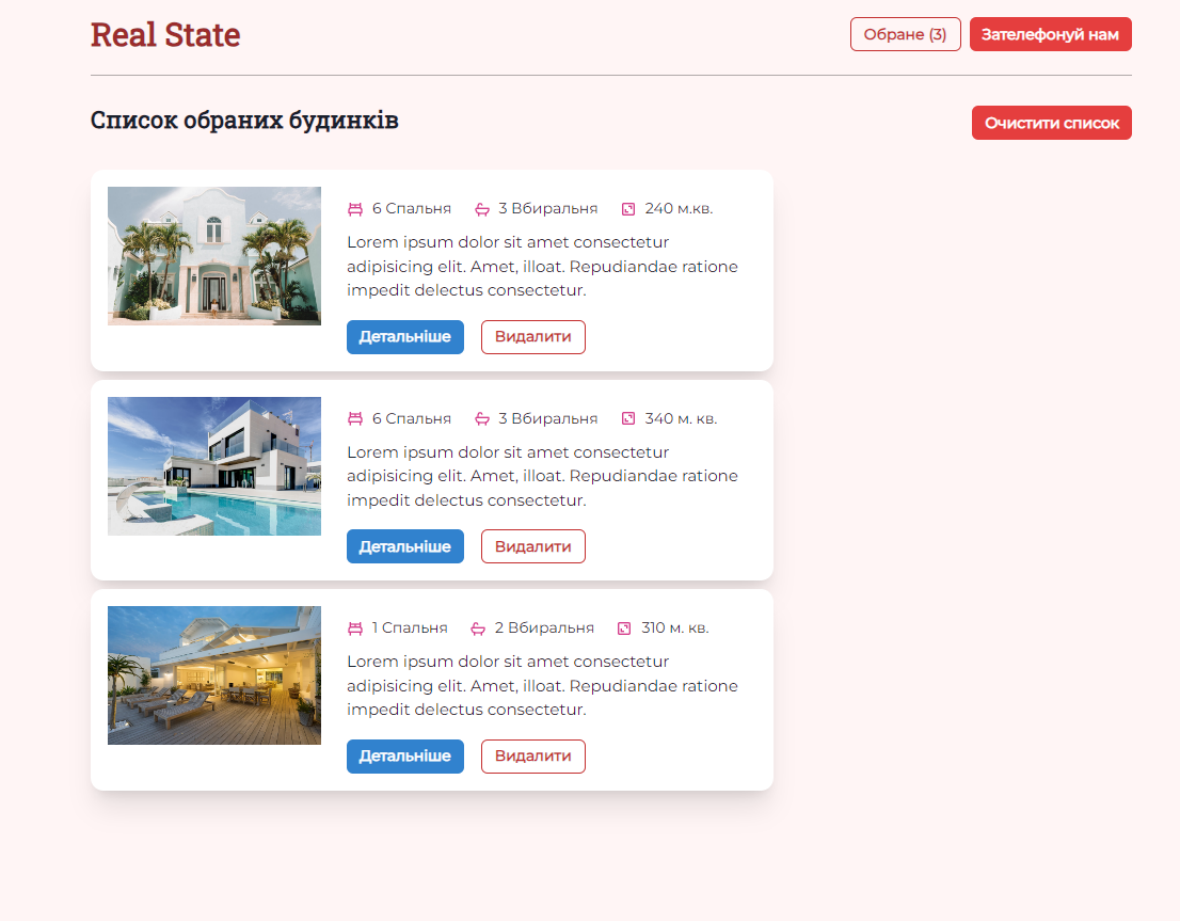



Рис. 3.17 Тестування списку обраної нерухомості

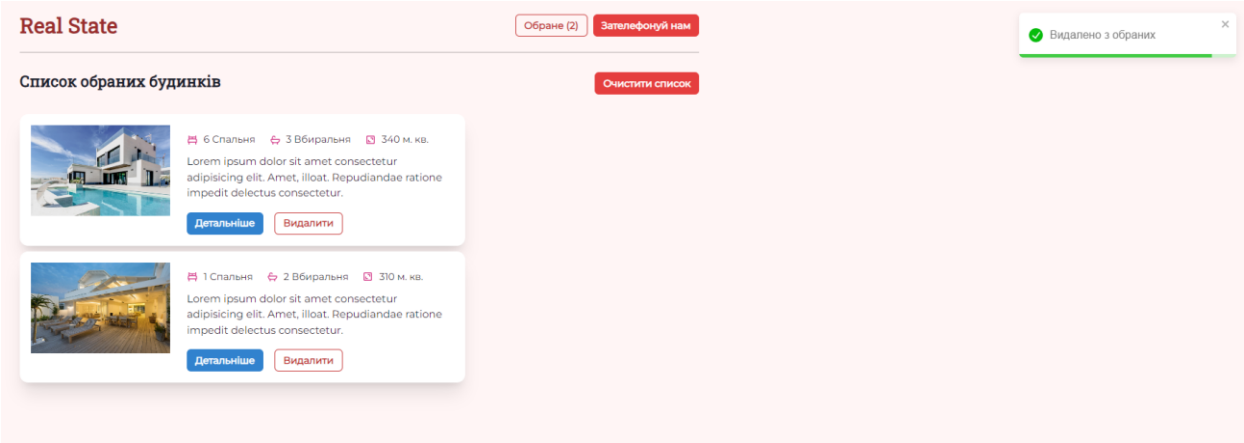



Рис. 3.18 Тестування списку обраної нерухомості

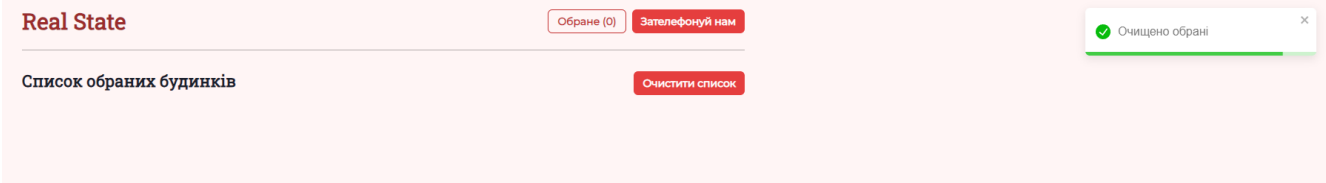



Рис. 3.19 Тестування списку обраної нерухомості

### 3.3 Висновки до розділу.

Веб-застосунок для пошуку купівлі або оренди нерухомості відіграє важливу роль у сучасному ринку нерухомості, надаючи користувачам зручний та ефективний інструмент для знаходження відповідного житла. У цьому розділі було детально розглянуто процес програмної реалізації такого застосунку, що включає декілька важливих аспектів та етапів.

Однією з головних задач було створення функціональної пошукової системи яка є одним з ключових елементів. Використання сучасних алгоритмів пошуку та фільтрації дозволяє користувачам швидко отримувати найбільш релевантні результати, враховуючи їхні індивідуальні вимоги та побажання. Гнучка система фільтрів за ціною, місцезнаходженням, типом нерухомості та іншими параметрами підвищує точність та ефективність пошуку.

Важливою частиною розробки також є інтеграція з базами даних та забезпечення надійного зберігання інформації. Використання сучасних технологій зберігання даних забезпечує надійність та безпеку користувацької інформації.

Загалом, програмна реалізація веб-застосунку для пошуку купівлі або оренди нерухомості передбачає врахування багатьох технічних та функціональних аспектів, спрямованих на забезпечення зручності, ефективності та безпеки для користувачів.

# ВИСНОВКИ

Дипломна робота присвячена розробці веб-застосунку на тему «Купівля/оренда нерухомості з використанням мов розмітки HTML/CSS та мови програмування Java Script».

Була проведена робота по аналізу проблем з якими зіштовхуються орендарі, були оглянуті аналоги існуючих програмних забезпечень в галузі покупки/оренди нерухомості. Були визначені основні функції та вимоги до веб-застосунку.

За результатами проведених робіт було обрано середовище розробки Visual Studio Code, в якому застосовувались мова розмітки HTML та CSS, мова програмування Java Script та її бібліотеки, зокрема Node.js та React.js.

У процесі створення веб-сайту була створена база даних, яка використовується для збереження необхідної інформації. Було реалізовано декілька функцій, такі як пошук нерухомості за критеріями, детальний огляд обраних варіантів та можливість зв’язку з орендарями.

В дипломній роботі була врахована та реалізована вимога щодо зрозумілого та адаптивного інтерфейсу, що дозволяє користувачам використовувати веб-сторінку з різних пристроїв, таких як персональний комп’ютер та смартфон.

Гребещенко В.Е., Садовенко В.С. Розробка веб-застосунку для покупки/оренди нерухомості за допомогою мов розмітки HTML/CSS та Java Script. Всеукраїнська науково-практична конфереція «Сучасні інтелектуальні інформаційні технології в науці та освіті» 15 травня 2024., м. Київ. Державний університет інформаційно-комунікаційних технологій. Збірник тез. К.: ДУІКТ, 2024. С. 46-49.

Гребещенко В.Е., Садовенко В.С. Сучасні технології підбору нерухомості. Всеукраїнська науково-практична конференція «Сучасні інтелектульні інформаційні технології в науці та освіті» 15 травня 2024.,Київ. Державний університет інформаційно-комунікаційних технологій. Збірник тез. К.: ДУІКТ, 2024. С. 209-211.

# ПЕРЕЛІК ПОСИЛАНЬ

1. Гречко В. В. Інвестиції в нерухомість: аналіз та стратегія / В. В. Гречко. – Київ: Кондор, 2015. – 312 с.

2. Коваленко І. Ю. Ринок нерухомості: тенденції та перспективи / І. Ю. Коваленко. – Харків: Право, 2015. – 278 с.

3. Сидоренко А. М. Управління об'єктами нерухомості / А. М. Сидоренко. – Львів: Новий Світ-2000, 2015. – 345 с.

4. Ткаченко П. П. Оцінка нерухомості: методи та практики / П. П. Ткаченко. – Дніпро: Ліра, 2015. – 256 с.

5. Шевченко О. О. Пошук нерухомості для інвестування: алгоритми та технології / О. О. Шевченко. – Одеса: Астропринт, 2015. – 290 с.

6. Василенко С. В. Управління комерційною нерухомістю / С. В. Василенко. – Київ: Знання, 2015. – 320 с.

7. Іванова Н. П. Ринок житлової нерухомості: сучасний стан та перспективи розвитку / Н. П. Іванова. – Харків: Видавництво ХНЕУ, 2015. – 310 с

8.HyperText Markup Language for beginners [Електронний ресурс] - https://www.w3schools.com

9. Java Script basics [Електронний ресурс] - https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/JavaScript_basics

10. Java Script with Code Academy [Електронний ресурс] - https://www.codecademy.com/learn/introduction-to-javascript

11. Introduction to Node.js [Електронний ресурс] - https://nodejs.org/en/learn/getting-started/introduction-to-nodejs

12. Kiev International Realty [Електронний ресурс] - https://kievintlrealty.com/.

13. Real Estate Lviv [Електронний ресурс] - https://www.real-estate.lviv.ua/en/

14. Trulia [Електронний ресурс] - https://www.trulia.com/

15. Zillow: Real Estate, Apartments, Mortgages & Home Values [Електронний ресурс] - https://www.zillow.com/

# ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ


ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ


## Розробка веб-застосунку для купівлі/оренди нерухомості з використанням мов розмітки HTML/CSS, мови програмування Java Script

Виконав студент 4 курсу

групи ПД-44
Гребещенко Владислав Едуардович
Керівник роботи

К.ф-м.н., доц., професор кафедри ІПЗ Садовенко Володимир Сергійович

Київ – 2024

## Мета, об’єкт та предмет дослідження

- **Мета роботи** - підтримка процесу пошуку нерухомості для купівлі/оренди шляхом створення веб-застосунку за допомогою мови Java Script та мов розмітки HTML/CSS.

- **Об’єкт дослідження –** процес пошуку нерухомості для купівлі/оренди.

- **Предмет дослідження –** веб-застосунок для пошуку нерухомості для купівлі/оренди.

# Задачі дипломної роботи

1. Провести аналіз існуючих веб-застосунків з пошуку нерухомості, визначити їх переваги та недоліки.

2. Визначити функціональні та нефункціональні вимоги до веб-застосунку для пошуку купівлі/оренди нерухомості.

3. Проектування візуальної частини та інтерфейсу користувача веб-застосунку для пошуку купівлі/оренди нерухомості.

4. Програмна реалізація веб-застосунку для пошуку купівлі/оренди нерухомості.

5. Тестування веб-застосунку з пошуку нерухомості.

3

# Аналіз аналогів

| Характеристика | Zillow | Trulia | KyivInRealty | Real estate Lviv | Real State |
|---|---|---|---|---|---|
| Фільтрація нерухомості | + | + | + | + | + |
| Прямий контакт з орендодавцем | - | - | + | - | + |
| Функція «Обрана нерухомість» | - | + | - | - | + |
| Детальний опис нерухомості | - | + | + | + | + |
| Крос-браузерність | + | + | - | - | + |

4

## Вимоги до застосунку

**Функціональні вимоги:**

1. Пошук нерухомості за фільтром.
2. Можливість детального огляду та опису нерухомості.
3. Можливість прямого контакту з орендодавцем.
4. Можливість додати нерухомість до списку обраного.

**Нефункціональні вимоги:**

1. Категорії фільтру: тип нерухомості, місцезнаходження та вартість.
2. Крос-браузерність (Google version 125<, Edge version 124<, Firefox ver. 124<, Opera version 110).

5

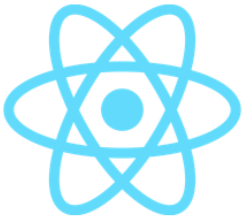
Слайд 6

## Програмні засоби реалізації

Java Script

Visual Studio Code

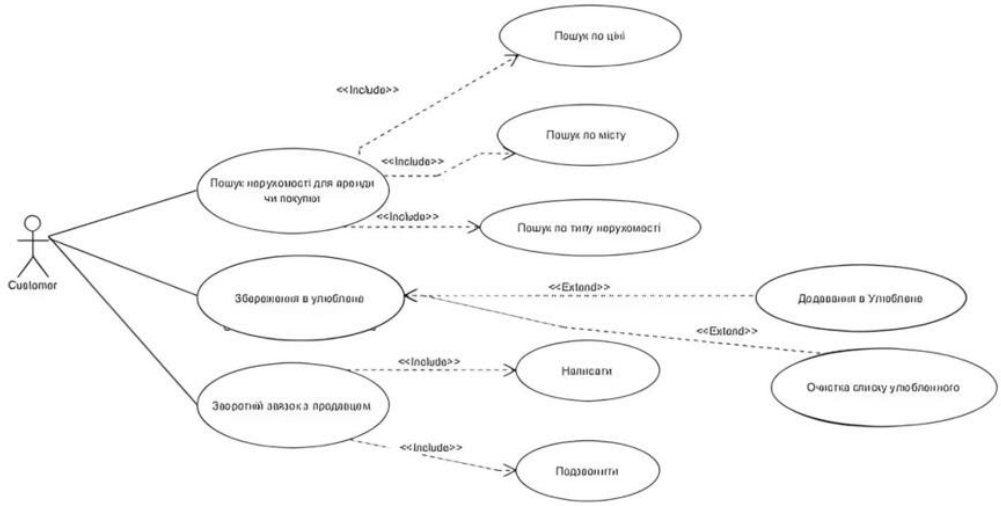
Node.js



React.js

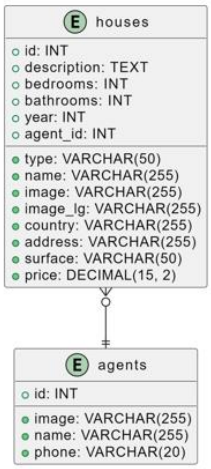
6

# Діаграма варіантів використання



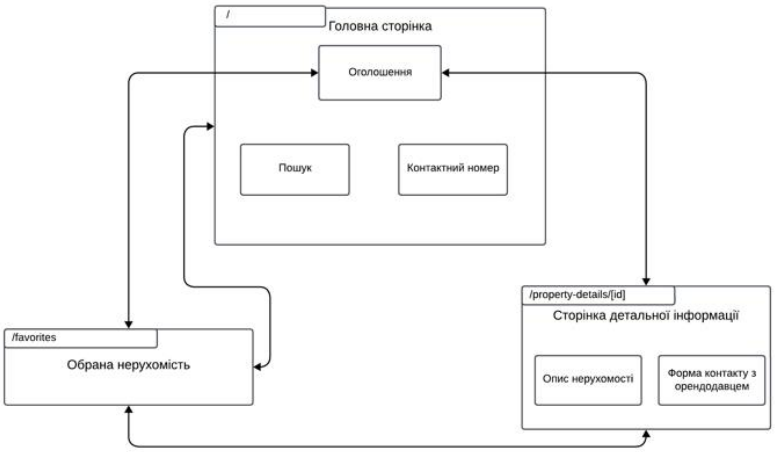

7

# Схема бази даних



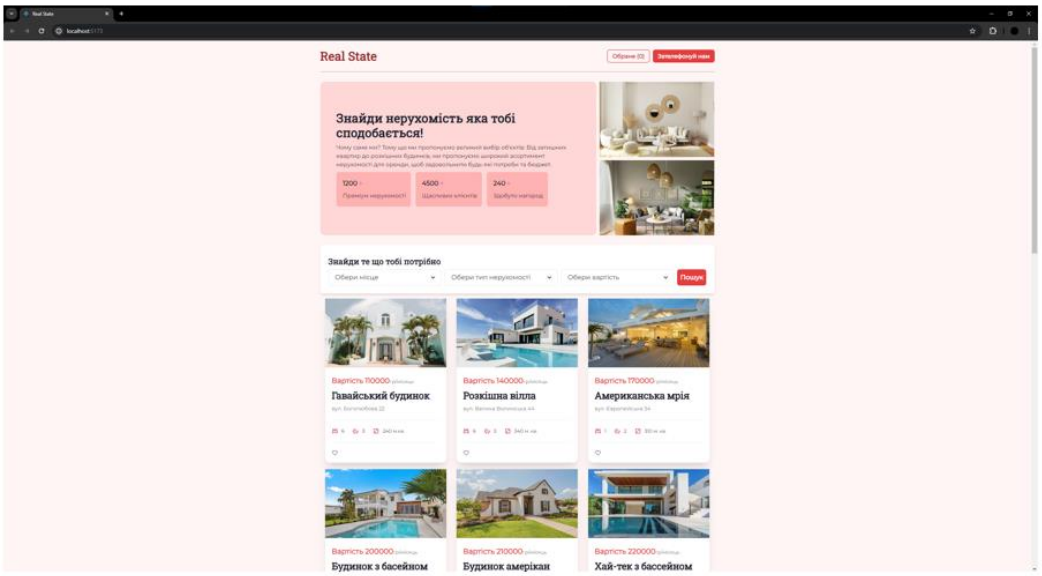

8

# Карта веб-застосунку




9

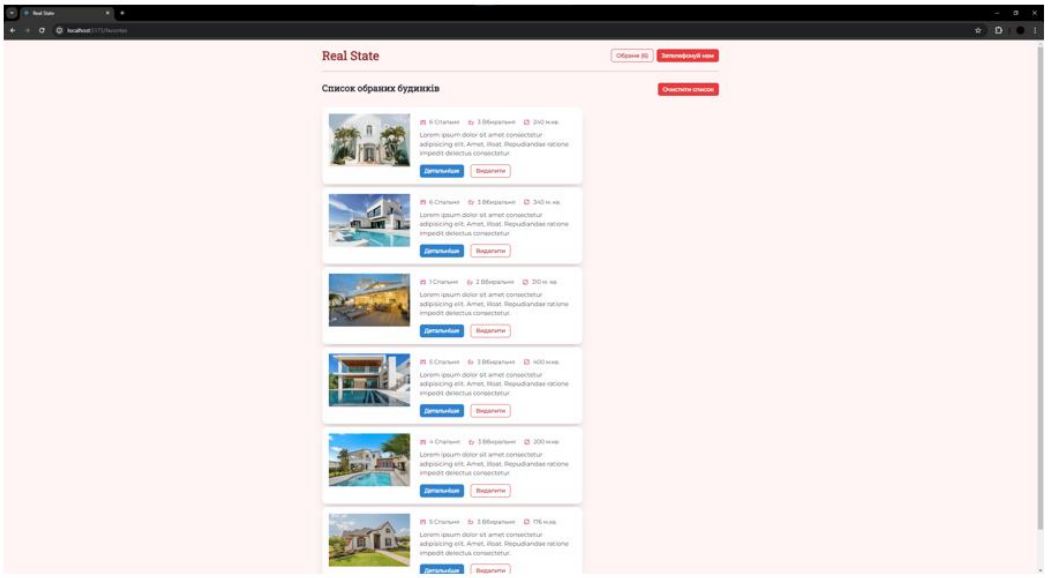
# Екранні форми




Головна сторінка

10

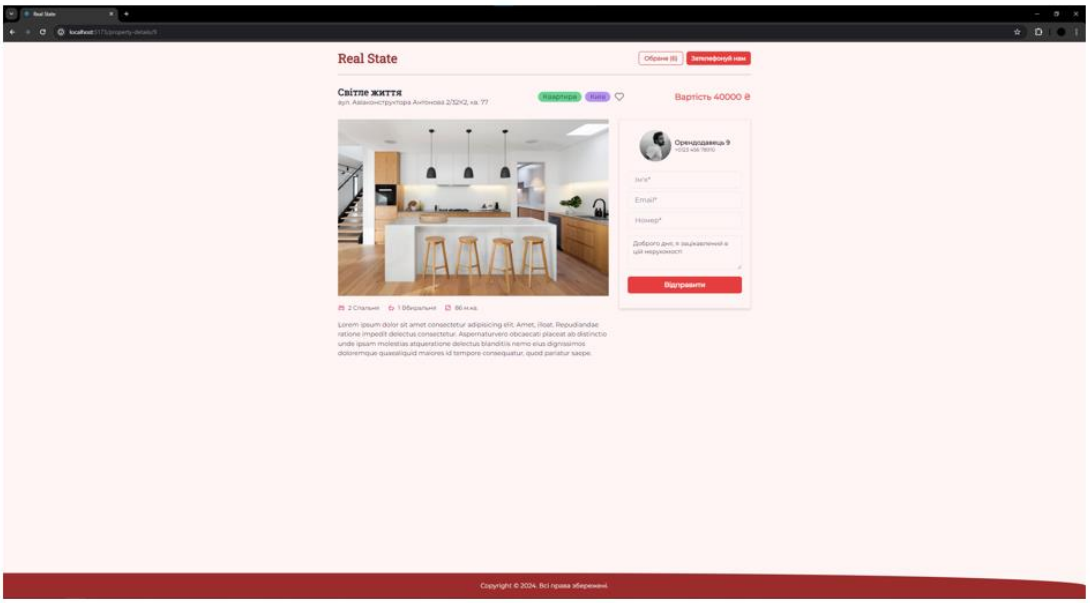
# Екранні форми




Список обраних будинків

11

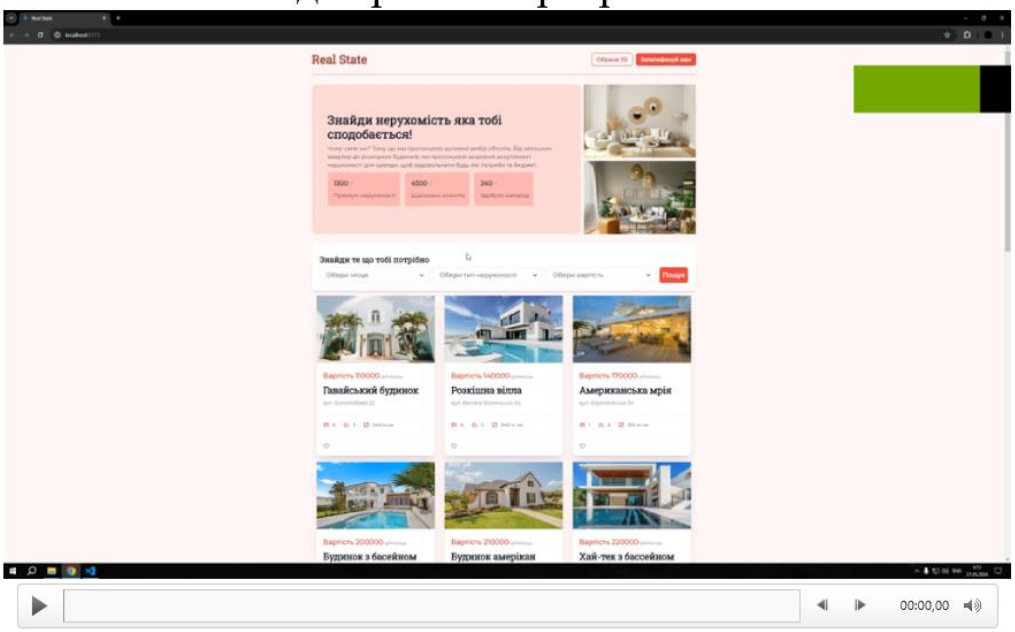
# Екранні форми




Сторінка детального опису

12

## Відео роботи програми




13

## АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

1. Гребещенко В.Е., Садовенко В.С. Сучасні технології підбору нерухомості. Сучасні інтелектуальні інформаційні технології в науці та освіті. IV Всеукраїнська Науково-практична конференція. Збірник тез. 10.05.2024, ДУІКТ, м. Київ. К.: ДУІКТ, 2024, С.
2. Гребещенко В.Е., Садовенко В.С. Розробка веб-застосунку для покупки/оренди нерухомості за допомогою мов розмітки HTML/CSS та Java Script. Інтелектуальний аналіз даних. IV Всеукраїнська Науково-практична конфереція. Збірник тез. 17.05.2024, ДУІКТ, м. Київ. К.:ДУІКТ, 2024, С.

# ВИСНОВКИ

1. Проведений аналіз існуючих веб-застосунків з пошуку нерухомості, визначено їх переваги та недоліки.
2. Визначено функціональні та нефункціональні вимоги до програмного забезпечення, ключовими з яких були список обраних та крос-браузерність.
3. Проведено огляд технологій, які були використані при розробці веб-застосунку.
4. Спроектовано та розроблено візуальну частину.
5. Проведено тестування веб-застосунку для виявлення та виправлення помилок.

# ДОДАТОК Б. ЛІСТИНГ ПРОГРАМНИХ МОДУЛІВ

```
import { Stack, VStack, Heading, Text, Box, HStack, Image, Input, Textarea, Button } from "@chakra-ui/react"
import { BiBed, BiBath, BiArea } from "react-icons/bi";

import { useContext } from "react";
import { Link, useParams } from "react-router-dom";

import { HouseContext } from "../../context/HouseContext";
import { useFavorite } from "../../context/FavoriteContext";

const HouseDetails = () => {

  const {favorites, removeFavorite, clearFavorites} = useFavorite();
  const { houses } = useContext(HouseContext);


  const favoriteHouses = favorites.map(favorite => {
    return houses.find(house => house.id == favorite.houseId);
  });

  console.log(favoriteHouses, houses);

  return (
    <>
      <Stack direction={{base: 'column', md: 'row'}} justify='space-between' align={{md: 'center'}}  my='28px'>
        <Box className="flex justify-between w-full">
          <Heading fontSize='22px'>
            Список обраних будинків
          </Heading>

          <Button onClick={clearFavorites} colorScheme='red' size='sm'>Очистити список</Button>
        </Box>
      </Stack>

      <Stack direction={{base:'column', lg: 'row'}} gap='6' align='flex-start'>
        <VStack align='left' maxW='640px'>
          {favoriteHouses.map((favorite) => (
            <Box key={favorite.id}
            className="w-full bg-white shadow-xl rounded-xl p-4 flex gap-6 items-start"
            >
              <Image
              className="w-50"
               src={favorite.imageLg} alt='house' />

              <div>
              <Stack py='10px' spacing={{sm: '3', md: '5'}} direction={{base: 'column', md: 'row'}}>
                <HStack>
                    <BiBed style={{ color: "#D53F8C" }} />
                    <Text fontSize="14px">{favorite.bedrooms} Спальня</Text>
                </HStack>
                <HStack>
                    <BiBath style={{ color: "#D53F8C" }} />
                    <Text fontSize="14px">{favorite.bathrooms} Вбиральня</Text>
                </HStack>

                <HStack>
                    <BiArea style={{ color: "#D53F8C" }} />
                    <Text fontSize="14px">{favorite.surface}</Text>
                </HStack>
              </Stack>

              <Text fontSize='15px'>{favorite.description.split(' ').slice(0, 15).join(' ')}</Text>

              <Link to={`/property-details/${favorite.id}`}>
                <Button
                className="mt-4 mr-4"
                 colorScheme='blue' size='sm'>Детальніше</Button>
              </Link>

              <Button
              className="mt-4"
              variant={'outline'}
               onClick={() => removeFavorite(favorite.id)} colorScheme='red' size='sm'>Видалити</Button>



              </div>
            </Box>
          ))}
        </VStack>
      </Stack>
    </>
  )
}

export default HouseDetails;
```

```
import { Flex, Heading, Button,  HStack, chakra, ButtonGroup, useBreakpointValue, Divider } from '@chakra-ui/react';
import { Link } from 'react-router-dom';
import NavMobile from './NavMobile';
import { useFavorite } from '../../context/FavoriteContext';
import { useState } from 'react';



const Header = () => {
  const isDesktop = useBreakpointValue({ base: false, lg: true })

  const [showPhone, setShowPhone] = useState(false);

  const {
    favorites,
    clearFavorites,
  } = useFavorite();

  return (
    <chakra.header id="header" borderBottom='1px solid rgb(0,0,0,0.3)'>
      <Flex w='100%' py='5' align='center' justify='space-between'>
        <Link to='/'>
          <Heading fontSize='3xl' color='red.700'>Real State</Heading>
        </Link>
        {
          isDesktop ? (
          <>
            <HStack>
            <Link to='/favorites'>

              <Button size='sm' variant='outline'>
                Обране

                ({favorites.length})
              </Button>
        </Link>

              <Button size='sm' variant='solid' onClick={() => setShowPhone(!showPhone)}>
                Зателефонуй нам</Button>
            </HStack>
          </>
          ) : (
            <NavMobile />
          )
        }
      </Flex>
```

```
        {showPhone && (
          <chakra.div w='100%' bg='red.700' p='3' color='white' textAlign='center'>
            <p>Телефон: +0123 456 7890</p>
            <ButtonGroup>
              <Button onClick={() => setShowPhone(false)}>Затвори</Button>
            </ButtonGroup>
          </chakra.div>
        )}

        {/* <Divider color='pink.800' w={}='20px' />  */}
      </chakra.header>
    )
}

export default Header
```

```jsx
import { useRef } from 'react';

import { ButtonGroup, VStack, Input, Drawer, DrawerBody, DrawerCloseButton, DrawerContent, DrawerFooter, DrawerHeader, Drawe
import { FiMenu } from 'react-icons/fi';

const NavMobile = () => {
  const { isOpen, onOpen, onClose } = useDisclosure();
  const btnRef = useRef();

  return (
    <>
        <IconButton variant='ghost'
            icon={<FiMenu fontSize='1.35rem' />}
            aria-label='Open Menu'
            onClick={onOpen} ref={btnRef}
        />
        <Drawer isOpen={isOpen} placement='right' onClose={onClose} finalFocusRef={btnRef}>
            <DrawerOverlay />
            <DrawerContent>
                <DrawerCloseButton />
                <Center>
                <DrawerHeader>Menu</DrawerHeader>
                </Center>
                <DrawerBody px='14' mt='4'>
                    <VStack as='nav' spacing='8' alignItems='left'>
                        {
                            ['Головна', 'Про нас'].map((item)=>(
                                <Button variant='link' key={item}>{item}</Button>
                            ))
                        }
                        <Button size='sm' variant='solid'>Контакт</Button>
                        <Button size='sm' variant='outline'>Ввійти</Button>
                    </VStack>
                </DrawerBody>
            </DrawerContent>
        </Drawer>
    </>
  )
}

export default NavMobile
```

```jsx
import {
  VStack,
  Divider,
  Heading,
  HStack,
  Image,
  Stack,
  Text,
  Flex,
} from "@chakra-ui/react";
import { BiBed, BiBath, BiArea, BiHeart, BiSolidHeart } from "react-icons/bi";
import { useFavorite } from "../../context/FavoriteContext";

const HouseItem = ({ house }) => {
  const { favorites, toggleFavorite, isFavorite } = useFavorite();

  const onClickFavorite = (e) => {
    e.preventDefault();
    e.stopPropagation();

    toggleFavorite(house.id);
  };

  return (
    <Flex justify='center' align='center'>
        <Stack justify='center' width="300px" bg="white" boxShadow="xl" borderRadius="xl">
          <Image src={house.imageLg} h='170' alt='houses' />

          <VStack p='4' align='left'>
            <Text mt="-1" fontWeight="extrabold" fontSize="18px" color="red.500">
            Вартість {house.price}
            <span style={{ fontSize: 12, color: "grey", fontWeight: "normal" }}>
                грн/місяць
            </span>
            </Text>

            <Heading fontSize="24px" letterSpacing="tight">
            {house.name}
            </Heading>

            <Text fontSize="13px" color="grey">
            {house.address}
            </Text>

            <Divider my="2.5" />

            <HStack spacing="5">
            <HStack>
                <BiBed style={{ color: "#D53F8C" }} />
```

```
            {house.name}
            </Heading>

            <Text fontSize="13px" color="grey">
            {house.address}
            </Text>

            <Divider my="2.5" />

            <HStack spacing="5">
            <HStack>
                <BiBed style={{ color: "#D53F8C" }} />
                <Text fontSize="12px">{house.bedrooms}</Text>
            </HStack>

            <HStack>
                <BiBath style={{ color: "#D53F8C" }} />
                <Text fontSize="12px">{house.bathrooms}</Text>
            </HStack>

            <HStack>
                <BiArea style={{ color: "#D53F8C" }} />
                <Text fontSize="12px">{house.surface}</Text>
            </HStack>
            </HStack>

            <Divider my="2.5" />

            {isFavorite(house.id) ? (
              <BiSolidHeart style={{ color: 'red' }} onClick={onClickFavorite} />
            ) : (
              <BiHeart style={{ color: 'grey' }} onClick={onClickFavorite} />
            )}
          </VStack>
        </Stack>
    </Flex>
  );
};

export default HouseItem;
```

```
import { Center, Grid, Heading, Spinner, Stack } from '@chakra-ui/react';
import { useContext } from "react";
import { Link } from 'react-router-dom';

import { HouseContext } from "../../context/HouseContext";
import HouseItem from './HouseItem';

const HouseList = () => {
  const { houses, isLoading } = useContext(HouseContext);

  if(isLoading){
    return (
      <Center>
        <Spinner align='center' color='red.500' />
      </Center>
    )
  }

  if (houses.length === 0) {
    return (
      <Stack maxH='400px'>
        <Heading size="lg" p={{base: '6', md: '10'}} align="center">
          Упс... Спробуймо ще раз?
        </Heading>
      </Stack>
    );
  }

  return (
    <Grid my='3' rowGap='4' gridTemplateColumns='repeat(auto-fit, minmax(300px, 1fr))'
    >
      {
        houses.map(item=>
          <Link to={`/property-details/${item.id}`} key={item.id}>
            <HouseItem key={item.id} house={item} />
          </Link>
        )
      }
    </Grid>
  )
}

export default HouseList
```

```jsx
import { Textarea, Image, VStack, HStack, Box, Text, Input, Button } from '@chakra-ui/react'
import { useState } from 'react';
import { toast } from 'react-toastify';

const Form = ({searchedHouse}) => {
  const [formData, setFormData] = useState({
    name: '',
    email: '',
    phone: '',
    message: 'Доброго дня, я зацікавлений в цій нерухомості'
  });

  const handleChange = (e) => {
    // check if all fields are filled
    if (!formData.name || !formData.email || !formData.phone || !formData.message) {
      toast.error('Будь ласка, заповніть всі поля');
      return;
    }

    toast.success('Ваше повідомлення відправлено');
  }

  return (
    <VStack border='1px' borderColor='red.100' boxShadow='md' px='5' py='6'>
      <HStack>
        <Image borderRadius='full' boxSize='75px' src={searchedHouse.agent.image} />
        <Box>
          <Text mb='-3px' fontWeight='extrabold' fontSize='15px'>{searchedHouse.agent.name}</Text>
          <Text style={{fontSize: '12px'}}>+{searchedHouse.agent.phone}</Text>
        </Box>
      </HStack>

      <VStack my='3px' spacing='2px'>
        <form>
          <Input mt='3' mb='2' placeholder="Ім'я*" value={formData.name} onChange={(e) => setFormData({...formData, name: e
          <Input placeholder="Email*" value={formData.email} onChange={(e) => setFormData({...formData, email: e.target.val
          <Input my='2' placeholder="Номер*" value={formData.phone} onChange={(e) => setFormData({...formData, phone: e.tar
          <Textarea my='2' placeholder='Message*' size='sm'
          value={formData.message}
          onChange={(e) => setFormData({...formData, message: e.target.value})} />
          <HStack my='2'>
            <Button w='full' px='4' colorScheme='red' onClick={handleChange}>
              Відправити</Button>
            {/* <Button w={{base: 'full', md: '50%'}} variant='outline'>Номер</Button> */}
          </HStack>
        </form>
      </VStack>
    </VStack>
```

```jsx
import { Stack, VStack, Heading, Text, Box, HStack, Image, Input, Textarea, Button } from "@chakra-ui/react"
import { BiBed, BiBath, BiArea, BiSolidHeart, BiHeart } from "react-icons/bi";

import { useContext } from "react";
import { useParams } from "react-router-dom";

import { HouseContext } from "../../context/HouseContext";
import Form from "./Form";
import { useFavorite } from "../../context/FavoriteContext";

const HouseDetails = () => {

  const {propertyId} = useParams();
  const { houses } = useContext(HouseContext);

  const searchedHouse = houses.find(house=> house.id== propertyId)
  const { favorites, toggleFavorite, isFavorite } = useFavorite();

  const onClickFavorite = (e) => {
    e.preventDefault();
    e.stopPropagation();

    toggleFavorite(searchedHouse.id);
  }

  return (
    <>
      <Stack direction={{base: 'column', md: 'row'}} justify='space-between' align={{md: 'center'}}  my='28px'>
        <Box>
          <Heading fontSize='22px'>{searchedHouse.name}</Heading>
          <Text fontSize='15px'>{searchedHouse.address}</Text>
        </Box>

        <HStack>
          <Text px='3' borderRadius='full' bg='green.300'>{searchedHouse.type}</Text>
          <Text px='3' borderRadius='full' bg='purple.300'>{searchedHouse.country}</Text>

          {isFavorite(searchedHouse.id) ? (
            <BiSolidHeart
            size={26}
            style={{ color: 'red' }} onClick={onClickFavorite} />
          ) : (
            <BiHeart
            size={26}
            style={{ color: 'grey' }} onClick={onClickFavorite} />
          )}
        </HStack>
```

```
        <Text fontWeight="extrabold" fontSize="20px" color="red.500">Вартість {searchedHouse.price}  ₴</Text>
      </Stack>

      <Stack direction={{base:'column', lg: 'row'}} gap='6' align='flex-start'>
        <VStack align='left' maxW='640px'>
          <Image src={searchedHouse.imageLg} alt='house' />

          <Stack py='10px' spacing={{sm: '3', md: '5'}} direction={{base: 'column', md: 'row'}}>
            <HStack>
                <BiBed style={{ color: "#D53F8C" }} />
                <Text fontSize="14px">{searchedHouse.bedrooms} Спальня</Text>
            </HStack>

            <HStack>
                <BiBath style={{ color: "#D53F8C" }} />
                <Text fontSize="14px">{searchedHouse.bathrooms} Вбиральня</Text>
            </HStack>

            <HStack>
                <BiArea style={{ color: "#D53F8C" }} />
                <Text fontSize="14px">{searchedHouse.surface}</Text>
            </HStack>
          </Stack>

          <Text fontSize='15px'>{searchedHouse.description}</Text>

        </VStack>

        <Form searchedHouse={searchedHouse} />
      </Stack>
    </>
  )
}

export default HouseDetails;
```

```
import { Select } from '@chakra-ui/react'
import { useContext } from 'react';
import { HouseContext } from '../../context/HouseContext';

const LocationFilter = () => {

  const {setCountry, countries} = useContext(HouseContext);

  const locationHandler = (event)=> {
    setCountry(event.target.value);
  }

  return (
    <Select placeholder='Обери місце' onChange={locationHandler}>
      {
        countries.map((country, index)=>
          <option key={index}>{country}</option>
        )
      }
    </Select>
  );
};

export default LocationFilter;
```

```jsx
import { FormControl, FormLabel, Select } from "@chakra-ui/react";
import { useContext } from "react";
import { HouseContext } from "../../context/HouseContext";

const PriceFilter = () => {
  const { setPrice } = useContext(HouseContext);

  const prices = [
    { value: "20000 - 30000 ₴" },
    { value: "30000 - 110000 ₴" },
    { value: "110000 - 140000 ₴" },
    { value: "140000 - 170000 ₴" },
    { value: "170000 - 200000 ₴" },
    { value: "200000 - 230000 ₴" },
  ];

  const priceHandler = (event) => {
    setPrice(event.target.value);
  };

  return (
    <Select placeholder="Обери вартість" onChange={priceHandler}>
      {prices.map((price, index) =>
          <option key={index}>{price.value}</option>
        )
      }
    </Select>
  );
};

export default PriceFilter;
```

```jsx
import { Select } from '@chakra-ui/react'
import { useContext } from 'react';
import { HouseContext } from '../../context/HouseContext';

const PropertyTypeFilter = () => {

  const {setProperty, properties} = useContext(HouseContext);

  const propertyTypeHandler = (event)=> {
    setProperty(event.target.value);
  }

  return (
    <Select placeholder='Обери тип нерухомості' onChange={propertyTypeHandler}>
      {
        properties.map((type, index)=>
          <option key={index}>{type}</option>
        )
      }
    </Select>
  );
};

export default PropertyTypeFilter;
```

```jsx
import { Button, Flex, Heading } from '@chakra-ui/react'
import { useContext} from "react";
import { HouseContext } from '../../context/HouseContext';

import LocationFilter from "./LocationFilter";
import PriceFilter from "./PriceFilter";
import PropertyTypeFilter from "./PropertyTypeFilter";

const Search = () => {

  const { searchHandler } = useContext(HouseContext);

  return (
    <Flex my='3' direction='column' borderRadius='md' bg='#fff' boxShadow='md' p='5'>

      <Heading py='2' size={{base: 'sm', md: 'md'}}> Знайди те що тобі потрібно</Heading>

      <Flex gap={{base: 3, md: 2}} direction={{base: 'column', md:'row'}} borderRadius='30'>
        <LocationFilter />
        <PropertyTypeFilter />
        <PriceFilter />
        <Button onClick={searchHandler} p={{base: 3, md: 2}} size="100%">Пошук</Button>
      </Flex>
    </Flex>
  )
}

export default Search
```

```jsx
import {
  HStack,
  VStack,
  Button,
  Text,
  Heading,
  Stack,
  Box,
  Image,
} from "@chakra-ui/react";
import { BiPlus } from "react-icons/bi";

import { bannerData } from "../data";
import Apartment1Lg from "../assets/images/apartments/a1lg.png";
import Apartment6Lg from "../assets/images/apartments/a6lg.png";

const Banner = () => {
  return (
    <>
      <Stack direction="row" my='6' overflow='hidden'>
        <VStack
          flexGrow='1'
          px={{ sm: "6", md: "10" }}
          py={{ sm: '8',  md: "16" }}
          bg="red.100"
          justify="center"
          align="left"
          borderRadius="xl"
        >
          <Heading fontSize={{ base: "xl", sm: "2xl", md: "3xl" }}>
            Знайди нерухомість яка тобі сподобається!
          </Heading>
          <Text fontSize="sm">
            Чому саме ми?
            Тому що ми пропонуємо великий вибір об'єктів: Від затишних квартир до розкішних будинків, ми пропонуємо широки
          </Text>


          <HStack spacing="3">
            {bannerData.map((item, index) => (
              <VStack
                key={index}
                bg="red.200"
                p="4"
                borderRadius="md"
                align="left"
                pr="3"
              >
```

```
            <HStack>
              <Text fontSize={{sm: '14px', md: 'md'}} fontWeight="extrabold" mr="-2">
                {Object.keys(item)}
              </Text>{" "}
              <BiPlus style={{ color: "#ED64A6" }} />
            </HStack>
            <Text fontSize={{sm: '12px', md: 'sm'}}>{Object.values(item)}</Text>
          </VStack>
        ))}
      </HStack>
    </VStack>

    <VStack justify='center'>
      <Box h='100%' display={{ base: "none", lg: "block", xl:'none' }} >
        <Image
          src={Apartment1Lg}
          alt="house"
          h='100%'
          objectFit='cover'
        />
      </Box>
      <Box h='50%' display={{ base: "none", xl: "block" }}>
        <Image
          src={Apartment1Lg}
          alt="house"
          style={{height: '100%', width: '100%', objectFit: 'contain'}}
        />
      </Box>
      <Box h='50%' display={{ base: "none", xl: "block" }}>
        <Image
          src={Apartment6Lg}
          alt="house"
          style={{height: '100%', width: '100%', objectFit: 'contain'}}
        />
      </Box>
    </VStack>
  </Stack>
 </>
 );
};

export default Banner;
```

```
import { Text, Center} from '@chakra-ui/react';

const Footer = () => {
  return (
    <>
      <Center borderTopEndRadius='50%' py='20px' bg='red.700' color='white'
      className='h-[60px] mt-20px'
      >
        <Text fontSize='15px'>Copyright &copy; 2024. Всі права збережені.</Text>
      </Center>
    </>
  )
}

export default Footer
```

```tsx
import { useLocalStorage } from "@uidotdev/usehooks";
import { toast } from "react-toastify";


interface Favorite {
  timestamp: number;
  houseId: string;
}

export const useFavorite = () => {
  const [
    favorites,
    setFavorites,
  ] = useLocalStorage<Favorite[]>("favorites", []);


  const addFavorite = (houseId: string) => {
    const newFavorite = {
      timestamp: Date.now(),
      houseId,
    };

    setFavorites([...favorites, newFavorite]);

    toast.success("Додано до обраних");
  };

  const removeFavorite = (houseId: string) => {
    setFavorites(favorites.filter((favorite) => favorite.houseId !== houseId));

    toast.success("Видалено з обраних");
  };

  const toggleFavorite = (houseId: string) => {
    if (isFavorite(houseId)) {
      removeFavorite(houseId);
    } else {
      addFavorite(houseId);
    }
  }

  const clearFavorites = () => {
    setFavorites([]);

    toast.success("Очищено обрані");
  }
```

```tsx
  const isFavorite = (houseId: string) => {
    return favorites.some((favorite) => favorite.houseId === houseId);
  }

  return {
    favorites,
    addFavorite,
    removeFavorite,
    toggleFavorite,
    isFavorite,
    clearFavorites,
  }
}
```

```
import { createContext, useState, useEffect } from 'react';
import { housesData } from '../data';

export const HouseContext = createContext('');

const HouseProvider = ({children}) =>{

    const [houses, setHouses] = useState(housesData);
    const [country, setCountry] = useState('Обери місце');
    const [countries, setCountries] = useState([]);
    const [price, setPrice] = useState('Обери вартість');
    const [property, setProperty] = useState('Обери тип нерухомості');
    const [properties, setProperties] = useState([]);
    const [isLoading, setIsLoading] = useState(false);

    useEffect(() => {
        const allCountries = houses.map(house=>{
            return house.country;
        })
        const uniqueCountries = [...new Set(allCountries)];
        setCountries(uniqueCountries);
    }, []);

    useEffect(() => {
        const allPropertyTypes = houses.map(house=>{
            return house.type;
        })
        const uniquePropertyTypes = [...new Set(allPropertyTypes)];
        setProperties(uniquePropertyTypes);
    }, []);

    const searchHandler=()=>{
        setIsLoading(true);

        // checking selection
        const isDefault = (str)=> {
            return str.split(' ').includes('Обери') || str.includes('no selection')
        }
        const minPrice = parseInt(price.split(' ')[0]);
        const maxPrice = parseInt(price.split('- ')[1]);

        const filteredHouses = housesData.filter(house=> {
            const housePrice = parseInt(house.price);
            // no selection
            if((isDefault(country) || !country) && (isDefault(price) || !price) && (isDefault(property) || !property) ){
              console.log('no selection', country, price, property, isDefault(country) && isDefault(price) && isDefault(pro
                return house;
            }
```

```
        // country is selected
        if(!isDefault(country) && isDefault(price) && isDefault(property)){
            return house.country === country;
        }

        // price is selected
        if(isDefault(country) && !isDefault(price) && isDefault(property)){
            return (housePrice >= minPrice) && (housePrice <= maxPrice);
        }

        // property is selected
        if(isDefault(country) && isDefault(price) && !isDefault(property)){
            return house.type === property;
        }

        // country & price is selected
        if(!isDefault(country) && !isDefault(price) && isDefault(property)){
            return house.country === country && (housePrice >= minPrice) && (housePrice <= maxPrice);
        }

        // country & property is selected
        if(!isDefault(country) && isDefault(price) && !isDefault(property)){
            return house.country === country && house.type === property;
        }

        // price & property is selected
        if(isDefault(country) && !isDefault(price) && !isDefault(property)){
            return (housePrice >= minPrice) && (housePrice <= maxPrice) && house.type === property;
        }

        // all are selected
        if(house.country === country && housePrice >= minPrice && housePrice <= maxPrice && house.type === property){
            return house;
        }
    })

    // setHouses(filteredHouses)
    setTimeout(() => {
        filteredHouses.length>0 ? setHouses(filteredHouses) : setHouses([]);
        setIsLoading(false);
    }, 1000);
}
```

```
    return(
        <HouseContext.Provider value={{
            houses,
            country,
            setCountry,
            countries,
            price,
            setPrice,
            property,
            setProperty,
            properties,
            searchHandler,
            isLoading
        }}>
            {children}
        </HouseContext.Provider>
    )
}

export default HouseProvider;
```

```
import Banner from '../components/Banner'
import Search from '../components/Search/Search'
import HouseList from '../components/Houses/HouseList';

const Home = () => {
  return (
    <>
      <Banner />
      <Search />
      <HouseList />
    </>
  )
}

export default Home;
```

```
import HouseDetails from "../components/PropertyDetails/HouseDetails";

const PropertyDetails = () => {
  return (
    <>
      <HouseDetails />
    </>
  )
}

export default PropertyDetails
```

```jsx
import { Routes, Route } from 'react-router-dom';
import { Container } from '@chakra-ui/react'

import Header from './components/Header/Header';
import Home from './routes/Home';
import PropertyDetails from './routes/PropertyDetails';
import Footer from './components/Footer'
import HouseProvider from './context/HouseContext';
import HouseDetails from './components/PropertyDetails/HouseDetails';
import FavoritesPage from './components/FavoritesPage/FavoritesPage';

import React from 'react';

import { ToastContainer } from 'react-toastify';
import 'react-toastify/dist/ReactToastify.css';

const App = () => {
  return (
    <HouseProvider>
      <Container maxW='container.lg' px='6' className='min-h-[calc(100vh-80px)]'>
        <Header />
        <Routes>
          <Route path='/' element={<Home />} />
          <Route path='favorites' element={ <FavoritesPage /> } />
          <Route path='property-details' element={ <PropertyDetails /> } >
            <Route path=":propertyId" element={<HouseDetails />} />
          </Route>
          <Route path="*"
                element={ <main style={{ padding: "1rem" }}>
                            <p>There's nothing here!</p>
                          </main>
                        }
          />
        </Routes>
      </Container>
      <Footer />

      <ToastContainer />
    </HouseProvider>
  )
}

export default App
```

```jsx
export const housesData = [
  {
    id: 1,
    type: 'Будинок',
    name: 'Гавайський будинок ',
    description:
      'Lorem ipsum dolor sit amet consectetur adipisicing elit. Amet, illoat. Repudiandae ratione impedit delectus consectet
    image: House1,
    imageLg: House1Lg,
    country: 'Вишневе, Київська область',
    address: 'вул. Боголюбова 22',
    bedrooms: '6',
    bathrooms: '3',
    surface: '240 м.кв.',
    year: '2016',
    price: '110000',
    agent: {
      image: Agent1,
      name: 'Орендодавець 1',
      phone: '0123 456 78910',
    },
  },
```

```
export const bannerData = [
    {1200: 'Преміум нерухомості'},
    {4500: 'Щасливих клієнтів'},
    {'240': 'Здобуто нагород'}
]

// import house images
import House1 from './assets/images/houses/house1.png';
import House2 from './assets/images/houses/house2.png';
import House3 from './assets/images/houses/house3.png';
import House4 from './assets/images/houses/house4.png';
import House5 from './assets/images/houses/house5.png';
import House6 from './assets/images/houses/house6.png';
import House7 from './assets/images/houses/house7.png';
import House8 from './assets/images/houses/house8.png';
import House9 from './assets/images/houses/house9.png';
import House10 from './assets/images/houses/house10.png';
import House11 from './assets/images/houses/house11.png';
import House12 from './assets/images/houses/house12.png';
// import house large images
import House1Lg from './assets/images/houses/house1lg.png';
import House2Lg from './assets/images/houses/house2lg.png';
import House3Lg from './assets/images/houses/house3lg.png';
import House4Lg from './assets/images/houses/house4lg.png';
import House5Lg from './assets/images/houses/house5lg.png';
import House6Lg from './assets/images/houses/house6lg.png';
import House7Lg from './assets/images/houses/house7lg.png';
import House8Lg from './assets/images/houses/house8lg.png';
import House9Lg from './assets/images/houses/house9lg.png';
import House10Lg from './assets/images/houses/house10lg.png';
import House11Lg from './assets/images/houses/house11lg.png';
import House12Lg from './assets/images/houses/house12lg.png';

// import apartments images
import Apartment1 from './assets/images/apartments/a1.png';
import Apartment2 from './assets/images/apartments/a2.png';
import Apartment3 from './assets/images/apartments/a3.png';
import Apartment4 from './assets/images/apartments/a4.png';
import Apartment5 from './assets/images/apartments/a5.png';
import Apartment6 from './assets/images/apartments/a6.png';
// import apartments large images
import Apartment1Lg from './assets/images/apartments/a1lg.png';
import Apartment2Lg from './assets/images/apartments/a2lg.png';
import Apartment3Lg from './assets/images/apartments/a3lg.png';
import Apartment4Lg from './assets/images/apartments/a4lg.png';
import Apartment5Lg from './assets/images/apartments/a5lg.png';
import Apartment6Lg from './assets/images/apartments/a6lg.png';
```

```
@import url('https://fonts.googleapis.com/css2?family=Montserrat&family=Roboto+Slab&display=swap');

body{
    font-family: 'Montserrat', sans-serif;
    font-family: 'Roboto Slab', serif;
}
```

```jsx
import React from 'react'
import ReactDOM from 'react-dom/client'
import { BrowserRouter } from 'react-router-dom'
import { ChakraProvider } from '@chakra-ui/react'

import App from './App'
import 'virtual:uno.css'
import './index.css'

import { theme } from './Theme';

ReactDOM.createRoot(document.getElementById('root')).render(
  <React.StrictMode>
    <ChakraProvider theme={theme}>
      <BrowserRouter>
        <App />
      </BrowserRouter>
    </ChakraProvider>
  </React.StrictMode>
)
```

```jsx
import { extendTheme, theme as base, withDefaultVariant } from "@chakra-ui/react"

const breakpoints = {
  sm: '320px',
  md: '500px',
  lg: '720px',
  xl: '960px',
  '2xl': '1200px',
}

export const theme = extendTheme({
  breakpoints,
  fonts: {
    heading: `'Roboto Slab', ${base.fonts.heading}`,
    body: `'Montserrat', sans-serif`,
  },
  styles: {
    global: {
      body: {
        bg: 'red.50'
      }
    }
  },
  components: {
    Button: {
      defaultProps: {
        colorScheme: 'red', // default is gray
      }
    },
    Input:{
      defaultProps: {
        focusBorderColor: 'red.500'
      }
    },
    Select:{
      baseStyle: {
        _focus: {
          borderColor: 'red.500'
        }
      }
    },
    Textarea:{
      defaultProps: {
        focusBorderColor: 'red.500'
      }
    }
  }
});
```