ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра Інженерії програмного забезпечення

Пояснювальна записка

до бакалаврської роботи

на ступінь вищої освіти бакалавр

на тему: «**РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ РИНКУ НЕРУХОМОСТІ МОВОЮ C#**»

Виконав: студент 4 курсу, групи ПД-44

спеціальності 121 Інженерія програмного забезпечення
(шифр і назва спеціальності)

Сєрокуров А.І
(прізвище та ініціали)

Керівник Жебка В.В.
(прізвище та ініціали)

Рецензент _____
(прізвище та ініціали)

Київ – 2023

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

Навчально-науковий інститут Інформаційних технологій

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти - «Бакалавр»

Напрям підготовки -121 – "Інженерія програмного забезпечення"

ЗАТВЕРДЖУЮ

Завідувач кафедри

Інженерії програмного забезпечення

Негоденко В.В.

“ ___ ” _____ 2023 року

ЗАВДАННЯ

НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТУ

Сєрокурова Артема Ігоровича

(прізвище, ім‘я, по батькові)

1.Тема роботи: «Розробка програмного забезпечення для ринку нерухомості мовою C#

Керівник роботи: Жебка В.В., д.т.н., доц., зав. кафедри ТЦР ,

(прізвище, ім‘я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від «24» лютого 2023 року №26.

2. Строк подання студентом роботи «1» червня 2023 року

3. Вихідні дані до роботи

3.1 Науково-технічна література

3.2 Офіційна документація C#

3.3 Наукова-технічна література

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

4.1.Аналіз актуальності роботи та огляд існуючих додатків

4.2.Аналіз інструментів реалізації та розробка структури додатку

4.3.Реалізація програмної частини додатку.

4.4.Висновки

5. Перелік демонстраційних матеріалів

5.1. Тема дипломної роботи

5.2. Мета роботи. Об’єкт дослідження. Предмет дослідження

5.3. Задачі дипломної роботи

5.4. Аналіз аналогів

5.5 Результати дослідження та опис реалізації програми.

5.6 Екранні форми
5.7 Апробація результатів дослідження
5.8 Висновки
6. Дата видачі завдання 25.02.2023 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської роботи	Строк виконання етапів роботи	Примітка
1	Підбір науково-технічної літератури	27.02.2023-28.02.2023	виконано
2	Ознайомлення з літературою	01.03.2023-10.03.2023	виконано
3	Розробка загальної концепції ПЗ	11.03.2023-13.03.2023	виконано
4	Розробка основного алгоритму	14.03.2023-21.03.2023	виконано
5	Реалізація функціональної частини	22.03.2023-28.03.2023	виконано
6	Реалізація інтерфейсу	29.03.2023-01.04.2023	виконано
7	Реалізація допоміжного забезпечення	02.04.2023-10.04.2023	виконано
8	Тестування та налагодження програми	11.04.2023-24.05.2023	виконано
9	Оформлення програмної документації	25.04.2023-04.05.2023	виконано
10	Розробка обов‘язкових демонстраційних креслень	05.05.2023-15.05.2023	виконано
11	Попередній захист роботи	19.05.2023	виконано
12	Подання роботи в деканат	01.06.2023	виконано

Студент _____ _Сєрокуров А.І.___
(підпис) (прізвище та ініціали)

Керівник роботи _____ ____Жебка В.В.__
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Текстова частина бакалаврської роботи: 60 с., 6 табл., 34 рис., 9 дод., 21 джерела.

РИНОК НЕРУХОМОСТІ, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, БАЗА ДАНИХ, СУБД MYSQL, C#,

Мета роботи – підвищення ефективності пошуку нерухомості за допомогою розробленого програмного забезпечення мовою C#.

Об`єкт дослідження – процес пошуку нерухомості.

Предмет дослідження – програмне забезпечення пошуку нерухомості.

Методи дослідження – методи теорії інформації, методи багатокритеріальної оптимізації, методи оптимального управління, розробку програмного коду мовою C#, аналіз літературних джерел, обробка та аналіз отриманих результатів.

Розробка додатку для ринку нерухомості включає аналіз вимог і потреб користувачів, проектування системи, розробку та тестування додатку. Галузь застосування такого додатку охоплює агентства нерухомості, ріелторів, інвесторів та клієнтів. Агентства нерухомості можуть використовувати ПЗ для ефективного управління списками нерухомості та взаємодії з клієнтами. Ріелтори отримують інструменти для керування клієнтською базою даних та швидкого доступу до актуальної інформації про нерухомість. Інвестори можуть використовувати додаток для знаходження вигідних об'єктів нерухомості та оцінювання рентабельності проектів. Клієнти мають можливість шукати нерухомість, оглядати її та виставляти свою нерухомість на продаж.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ОС - Операційна система

DOM - Document Object Model

IDE – Integrated Development Environment

SQL - Structured Query Language

ВСТУП

Зараз стрімко розвиваються інформаційні технології, з’явилися електронні носії інформації та єдине середовище використання та зберігання інформації – Інтернет. У зв'язку з великим зростанням кількості користувачів в Інтернеті, збільшується кількість ресурсів, які надають різну інформацію, в тому числі рекламну.

Ще кілька років тому багато власників агентств не мали чіткого розуміння, навіщо їм потрібно програмне забезпечення, як воно повинно працювати і які функції повинно виконувати. Зараз картина змінилася, багато власників агентств розуміють, що власне програмне забезпечення стане хорошим майданчиком для залучення клієнтів. Звичайно, за умови, що воно добре побудовано, воно привабить цільових відвідувачів. Програмне забезпечення також свідчить про процвітання компанії, її солідність і сучасність.

Мета роботи – підвищення ефективності пошуку нерухомості за допомогою розробленого програмного забезпечення мовою C#.

Об`єкт дослідження – процес пошуку нерухомості.

Предмет дослідження – програмне забезпечення пошуку нерухомості.

Методи дослідження – методи теорії інформації, методи багатокритеріальної оптимізації, методи оптимального управління, розробку програмного коду мовою C#, аналіз літературних джерел, обробка та аналіз отриманих результатів.

Опираючись на поставлену мету, були поставлені наступні задачі:

- проаналізувати потреби на ринку нерухомості
- порівняти існуючі аналоги
- визначити технологію і архітектуру для розробки
- спроектувати додаток для ринку нерухомості
- реалізувати додаток
- протестувати роботу додатку

1. ПРЕДМЕТНА ГАЛУЗЬ ТА ПОСТАНОВКА ПРОБЛЕМИ

1.1 Область нерухомості

Агентство нерухомості здійснює професійний супровід усіх операцій, які можливі на ринку нерухомості. Перш за все, це купівля-продаж житлової та комерційної нерухомості, нерухомості в новобудовах, а також оренда квартир, кімнат, земельних ділянок. Перевагою агентства є об’ємна база варіантів нерухомості та земельних ділянок, що пропонуються до продажу чи оренди. Програмне забезпечення, що розробляється, дозволить розміщувати інформацію про агентство нерухомості та послуги, які воно пропонує, різноманітні оголошення про продаж чи оренду нерухомості, а також дозволить користувачеві залишати заявки та надавати можливість зв’язатися з агентством.

Створюючи програмне забезпечення, присвячений купівлі-продажу квартир і офісів, необхідно подбати про зручність навігації та пошуку необхідної інформації.

Потенційний клієнт, який заходить в додаток агентства нерухомості, повинен за лічені хвилини знайти те, що йому потрібно, інакше він може скористатися послугами конкурентів. Пошук об'єктів в додатку агентства нерухомості повинен здійснюватися одночасно за кількома параметрами. По-перше, вона повинна здійснюватися за вартістю нерухомості та її площі. Для будинків і квартир також запроваджується функція пошуку за кількістю кімнат. Ще один важливий момент, на який варто звернути увагу, це розташування об'єкта.

Потенційному покупцеві або орендарю необхідно вміти вибирати нерухомість в певних районах і вулицях.

Структура додатку агентства нерухомості складається з двох основних розділів - продаж і оренда. Кожну з них, для зручності пошуку, необхідно розбити на підрозділи, присвячені житловій, офісній та торговій нерухомості.

Після створення бази додатку агентства нерухомості необхідно забезпечити користувачам доступ до вичерпної інформації по конкретному об'єкту. Загальна

площа, місце розташування, опис інфраструктури та інтер'єру значно підвищать шанси агентства продати або здати об'єкт в оренду.

Не менш важливою є наявність фотографій, які дають уявлення про те, що представлено в додатку.

Гарним рішенням буде додати систему улюбленої нерухомості та систему відгуків про нерухомість, для того щоб користувач розумів чи є заявлені характеристики дійсними

Будь-яка особа, яка збирається придбати або орендувати нерухомість, завжди цікавиться юридичними аспектами угоди. Для підвищення престижу як додатку, так і самого агентства варто подумати про створення розділу, присвяченого законодавчим нормам і юридичним тонкощам операцій, пов'язаних з купівлею, продажем або орендою різних об'єктів.

1.2 Огляд аналогів на ринку

Flatfy - це відомий онлайн-сервіс, що спрощує процес пошуку нерухомості. З його допомогою можна швидко знайти квартири, будинки та комерційні приміщення за своїми вимогами. Flatfy пропонує широкий вибір доступних пропозицій та точний пошук, який дозволяє задавати фільтри за ціною, площею, розташуванням та іншими параметрами. Користувачі можуть переглядати детальну інформацію про об'єкти, включаючи фотографії, опис та контактні дані продавця. Flatfy забезпечує зручний і ефективний досвід пошуку нерухомості, що допомагає знайти ідеальне житло чи інвестиційний об'єкт.

Рисунок 1.1 – додаток «Flatfy»

Lun - це онлайн-сервіс, що допомагає знайти та орендувати житло в Україні. Завдяки зручному інтерфейсу та потужному пошуку, Lun.ua дозволяє швидко знайти підходящі пропозиції за критеріями, такими як ціна, розташування та тип нерухомості. Користувачі можуть переглядати детальну інформацію про об'єкти, включаючи фотографії, опис та контактні дані власників. Крім того, Lun.ua надає корисні додаткові функції, такі як фільтри, що допомагають точніше налаштувати пошук, а також оголошення про нерухомість для оренди від приватних осіб та агентств. Загалом, Lun.ua забезпечує зручний та ефективний досвід пошуку житла для користувачів в Україні.

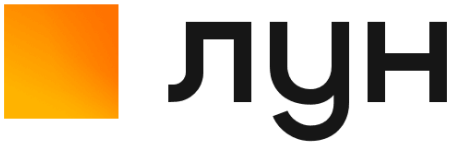


Рисунок 1.2 – додаток «LUN»

Українська онлайн система для агентств нерухомості Plektan. Забезпечує єдиний робочий простір для агентів. У ньому зберігається вся необхідна для роботи агента інформація про покупців і продавців нерухомості. Система дозволяє швидко підбирати пропозиції для клієнтів. Керівники відділів або компаній мають доступ до повної аналітики. Українська онлайн-система для агентств нерухомості Плектан - це набір інструментів, який дозволяє домогтися збільшення кількості клієнтів і підвищити конверсію при роботі з ними. Plektan забезпечує уніфікований робочий простір для агентів. У ньому зберігається вся необхідна для роботи агента інформація про покупців і продавців нерухомості. Система дозволяє швидко підбирати пропозиції для клієнтів. Керівники відділів або компаній мають доступ до повної аналітики про роботу співробітників і підприємства в цілому. Система Plektan дозволяє управляти конфліктами інтересів між агентами та компанією.

Рисунок 1.3 – програма «Plektan»

Безкоштовна ріелторська програма «MyHome» є основною частиною комплексу «Лідер Нерухомість» (програмне забезпечення для агентів та агентств) і є популярним інструментом у сфері нерухомості, що обумовлено певними особливостями розвитку останньої. Не забувайте, що кількість інформації про нерухомість зростає. Зростає не лише кількість інформації від кожного окремого агентства чи ріелтора, а й кількість об'єктів на ринку. Зростає кількість житлових масивів, населені пункти територіально збільшуються, зближуються одне з одним, об'єднують і примикають до великих міст. Інформаційні бази відчувають дедалі більший тиск, тому що обмін даними потребує все більше ресурсів. Телефон вже не може задовольнити потреби інформаційного потоку нового ринку нерухомості.

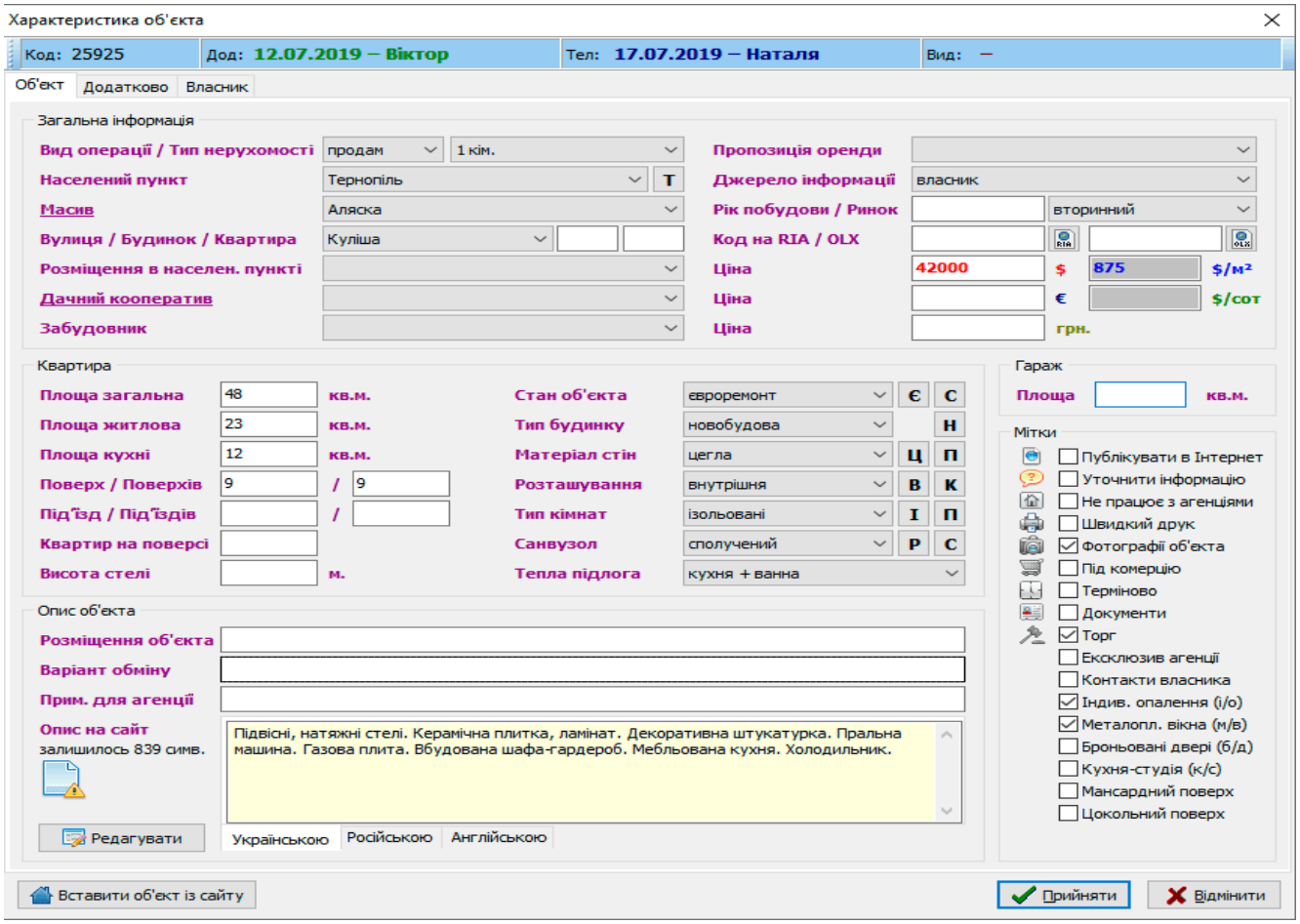


Рисунок 1.4 – програма «MyHome»

У таблиці 1 представлена порівняльна характеристика перерахованих вище продуктів.

Таблиця 1 - Порівняльна характеристика розглянутих аналогів

Назва	Flatfy	LUN	Plektan	MyHome	Розроблений Додаток
Кросплатформність	+	+	+	-	+
Реєстрація	+	+	+	+	+
Історія нерухомості користувача	+	+	+	-	+
Система улюбленої нерухомості	+	+	-	-	+
Система відгуків до нерухомості	-	-	-	-	+
Можливість використання API	+	+	+	-	+

1.3 Постановка задачі

Метою даної роботи є підвищення ефективності пошуку нерухомості за допомогою розробленого програмного забезпечення на мові C#. Додаток повинен надавати всю необхідну інформацію клієнтам про нерухомість та послуги, що надаються, а також мати дошку оголошень і можливість отримати додатковий канал для маркетингу та реклами.

Додаток що розробляється, має наступні розділи: орендувати нерухомість; купити нерухомість; подивитись історію купівлі; оглянути нерухомість; виставити на продаж свою нерухомість; переглянути виставлену нерухомість; додати та видалити відгук до нерухомості; переглянути відгуки; додати до улюбленого; переглянути улюблені;

У розділі «Купити» детально описується нерухомість згідно з оголошеннями про продаж; в розділі «Оренда» детально описано майно згідно з оголошеннями про оренду нерухомості; розділ «Оглянути» дозволяє користувачеві отримати інформацію про приміщення, його призначення та принципи роботи; розділ «Історія купівлі» містить інформацію про куплену або орендовану нерухомість користувача; розділ «Виставити на продаж» містить інформацію про нерухомість яку користувач може виставити на продаж; розділ «Переглянути нерухомість» містить інформацію про виставлену користувачем нерухомість; розділ «Додати відгук» містить інформацію про нерухомість до якої користувач може додати відгук; розділ «Видалити відгук» містить інформацію про нерухомість в якої користувач може видалити відгук; розділ «Переглянути відгуки» містить інформацію про всі відгуки до нерухомості; розділ «Додати до улюбленого» містить інформацію про нерухомість яку користувач може додати до улюбленого; розділ «Переглянути улюблені» містить інформацію про нерухомість яку користувач додав до улюблених.

На цьому додатку також є панель адміністратора, доступ до якої можливий після успішної авторизації в додатку за логіном і паролем, які є тільки у співробітників даного агентства. Ця панель містить такі розділи:

- додавання нерухомості;
- редагування нерухомості;
- видалення нерухомості.

Розділ «Редагування» дозволяє вносити зміни в до інформації стосовно нерухомості, «Додавання» записує отримані заявки від клієнтів. які додали свою нерухомість, а також розділ «Видалення» який дозволяє видаляти об’яви щодо нерухомості.

У роботі повинні бути реалізовані наступні завдання: опис предметної області, розробка форм документів для предметної області, дизайн додатку, розробка інформаційно-логічної моделі, оцінка трудомісткості проекту, розробка програмного забезпечення, розробка користувача. посібник та посібник адміністратора

2. ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ РИНКУ НЕРУХОМОСТІ

2.1 Вибір інструментальних засобів розробки

Для розробки програми була використана мова програмування C#, яка є однією з найпопулярніших та сучасних мов програмування. C# є об'єктно-орієнтованою та типобезпечною мовою, розробленою компанією Microsoft. Вона відповідає сучасним стандартам програмування і спеціально створена для розробки на платформі .NET Framework. Мова програмування була розроблена наприкінці 1990-х років групою інженерів на чолі зі Скоттом Вільтаумотом та Андерсом Хейлсбергом у компанії Microsoft як основна мова розробки додатків для платформи Microsoft .NET (програмної платформи від компанії Microsoft, призначеної для створення звичайних програм та веб-додатків) [16] , 17, 18].

Символ # (октоторп) у назві мови має кілька варіацій розшифровки: цей символ друкується на клавіатурі як Shift+3, що також може символізувати третю реалізацію C. Крім того, # можна інтерпретувати як дві пари плюсів ++ - що може означати новий крок у розвитку мови проти C++.

Інструментарій C# дозволяє вирішувати широке коло завдань, тому мова використовується для розробки:

- Веб-додатків;
- різних ігрових програм;
- мобільних додатків для Android та IOS;
- Програм для ОС Windows.

Мова C# має багато корисних функцій, таких як:

- поліморфізм;
- інкапсуляція;
- успадкування;
- події;

- атрибути;
- властивості;
- ітератори;
- навантаження операторів;
- узагальнені типи та методи;
- статична типізація;
- винятки;
- коментарі у форматі XML;
- складання сміття.

C# будучи об'єктно-орієнтованою мовою програмування підтримує всі три його основні складові: інкапсуляцію, поліморфізм та успадкування.

Інкапсуляція – механізм, який зв'язує разом код та дані, якими він маніпулює. Даний механізм дозволяє захищати цей код та дані від неправильного використання та зовнішнього доступу. Тобто це контроль доступу до полів і методів об'єкта за допомогою модифікаторів доступу: private, public і protected.

Поліморфізм - властивість системи, що дозволяє кільком об'єктам із загальними характеристиками мати доступ до одного інтерфейсу. Узагальнено концепцію поліморфізму слід висловити так: «один інтерфейс – безліч методів» [18].

Спадкування – така властивість, за допомогою якого один об'єкт може набувати властивості іншого. Тобто, з'являється можливість наслідувати

одними класами якості та поведінка інших класів.

Мова C# пов'язана з такими мовами програмування, як C, C++ та Java.

У C++ було визначено об'єктну модель, де тепер будується мову C#. Зв'язок з мовою C ґрунтується на успадкованому від нього синтаксисі, операторах та ключових словах. Між C# і Java прямого зв'язку немає, але обидві мови були розроблені, щоб створити спеціальний код, що переноситься.

Цей код базується на C та C++, використовуючи їх синтаксис та об'єктну модель. Володіння навичками тієї чи іншої мови полегшить перехід будь-якої з них.

Найважливіша риса мови програмування C# - здатність працювати у багатомовному оточенні. Під цим розуміється здатність коду, написаного різними мовами, працювати спільно, що є важливою здатністю при створенні великих програм.

У мови досить багато переваг:

– Підтримка більшості продуктів Microsoft;

– Використання об'єктно-орієнтованого підходу;

– наявність великої кількості бібліотек та шаблонів;

– типи даних мають фіксований розмір: 32-бітний int та 64-бітний long, що підвищує «мобільність» мови та спрощує програмування;

– схожість синтаксису мови з іншими мовами програмування, що полегшує перехід для програміста (наприклад, Java або C++);

– величезна кількість спеціальних готових конструкцій («синтаксичний цукор»), розроблених для розуміння та написання коду, які не мають значення при компіляції;

– автоматичне «складання сміття». Це полегшує роботу зі звільненням пам'яті, оскільки загальномовне середовище CLR сама викличе збирач сміття та очистить пам'ять;

– можливість створювати кросплатформні програми. За допомогою Xamarin на C# можна писати програми та програми для таких операційних систем, як iOS, Android, MacOS та Linux;

– Додано функціональне програмування (F#).

Однак є й недоліки:

– орієнтованість, переважно, лише з .NET (на Windows платформу);

– мова безкоштовна лише для невеликих компаній або груп учнів.

Також були використані ASP.NET для розробки веб-додатку та Entity Framework для зручного з’єднання з базою даніх. ASP.NET є фреймворком розробки веб-додатків, розробленим компанією Microsoft. Він забезпечує розширений набір інструментів та середовище для створення потужних та масштабованих веб-додатків. ASP.NET підтримує мови програмування, такі як C# та VB.NET, і

використовує модель програмування на основі подій та шаблонів для створення веб-сторінок, веб-служб та додатків.

Entity Framework (EF) є ORM (Object-Relational Mapping) для платформи .NET. Він надає зручний спосіб взаємодії з базами даних, де класи об'єктно-орієнтованої моделі відповідають таблицям бази даних. EF дозволяє розробникам працювати з базами даних за допомогою об'єктно-орієнтованого підходу, замість написання складних SQL-запитів. Він автоматично генерує SQL-запити за допомогою LINQ (Language Integrated Query) або методів розширення, спрощуючи розробку та підтримку бази даних.

З використанням ASP.NET і Entity Framework розробники можуть створювати веб-додатки, які мають потужний функціонал та зручно взаємодіють з базами даних.

2.2 Опис середовища та системи

Оскільки C# є однією з найпоширеніших мов програмування, існує велика кількість інтегрованих середовищ розробки з різним функціоналом. При виборі середовища розробки для програми, важливим критерієм є наявність профільника, оскільки висока продуктивність є однією з основних вимог. Для цього дослідження були обрані такі інтегровані середовища розробки, як MS Visual Studio, JetBrains Rider та Monodevelop, оскільки вони вважаються найбільш функціональними та поширеними.

Microsoft Visual Studio - це комплексне середовище розробки програмного забезпечення, яке пропонує розширені інструменти для різних типів програм. Використовуючи MS Visual Studio, розробники можуть створювати консольні програми, програми з графічним інтерфейсом, веб-сайти, веб-програми та веб-сервіси для різних платформ (рисунок 2.1).

Рисунок 2.1 – Microsoft Visual Studio

MS Visual Studio включає в себе вбудовані інструменти для редагування вихідного коду програм та здійснення рефакторингу. Вбудований відладчик має два рівні роботи: налагодження вихідного коду та налагодження машинного коду. Крім того, середовище надає додаткові інструменти, такі як редактори форм GUI та зручний редактор для створення програм з інтерфейсами, веб-редактор та редактор класів. MS Visual Studio підтримує створення та підключення плагінів для розширення функціональності. Воно також підтримує системи контролю версій вихідного коду, такі як Visual SourceSafe або Subversion, і надає безліч інструментів для редагування та проектування коду з використанням предметно-орієнтованих мов програмування.

JetBrains Rider - це потужне інтегроване середовище розробки (IDE) для мови C# та інших мов на платформі .NET. Забезпечуючи широкий набір функцій, Rider допомагає розробникам ефективно писати, аналізувати та налагоджувати код (рисунок 2.2).

Рисунок 2.2 – JetBrains Rider

Воно має потужний аналізатор коду, відладчик з широкими можливостями, інтелектуальне автодоповнення та підтримку рефакторингу. Rider надає розширену підтримку для систем контролю версій, інтегровані засоби для побудови та розгортання програм. Його продуктивність та налаштування роблять його відмінним вибором для розробників на платформі .NET, допомагаючи їм писати якісний код швидше та ефективніше.

MonoDevelop - це відкрите інтегроване середовище розробки (IDE), призначене для розробки програм на мовах програмування, що підтримуються платформою Mono, таких як C#, F# та інших (рисунок 2.3).

Рисунок 2.3 – MonoDevelop

Воно має основні функції, такі як редактор коду з відображенням синтаксису, відладчик, автодоповнення та підсвічування помилок. MonoDevelop підтримує інтеграцію з системами контролю версій, дозволяючи командам розробників спільно працювати над проектами. Воно також надає можливості для створення графічних інтерфейсів за допомогою розширення GTK#, що робить його популярним вибором для розробки програм на платформі Mono.

Для розробки цього веб-додатку була взята VS 2022 тому що Visual Studio є офіційним IDE для розробки програм на мові C# та вона надає розширені можливості для розробки програмних продуктів на платформі .NET.

Для роботи з серверною частиною було обрано локальний сервер Microsoft SQL Server. Microsoft SQL Server - це система управління базами даних (СУБД), розроблена корпорацією Microsoft. Вона надає потужні інструменти для зберігання, керування та обробки великого обсягу даних. SQL Server використовує мову запитів SQL для взаємодії з базами даних і підтримує широкий спектр функцій, таких як транзакції, реплікація даних, аналітика, інтеграція з іншими продуктами Microsoft. Вона надійна, масштабована і забезпечує високу продуктивність завдяки оптимізованій обробці запитів і індексуванню даних. SQL Server також підтримує розподілені бази даних та реплікацію для забезпечення доступності та надійності. Він використовується великими корпораціями, веб-додатками, електронною комерцією та багатьма іншими сферами для ефективного керування даними (рис. 2.4).

Рисунок 2.4 - Microsoft SQL Server

Для роботи з таблицями даних використовується SQL Server Management Studio. SQL Server Management Studio (SSMS) - це інструмент, розроблений для управління та адміністрування баз даних Microsoft SQL Server. Він надає зручний інтерфейс для розробки, виконання запитів, налаштування і моніторингу баз даних, а також для керування безпекою та резервним копіюванням. SSMS дозволяє розробникам і адміністраторам зручно працювати з SQL Server, забезпечуючи потужні можливості управління та аналізу даних (рисунок 2.5).

Рисунок 2.5 - SQL Server Management Studio

2.3 Основні вимоги до запуску та роботи з програмним забезпеченням

2.3.1. Функціональні вимоги

До програмного забезпечення висуваються наступні вимоги:

- можливість орендувати або купити нерухомість;
- реалізація додавання та продажу своєї нерухомості на веб-додатку
- реалізація пошуку та сортування нерухомості;
- реалізація методу підключення до бази даних;
- додавання або видалення нерухомості до улюблених;
- додавання або видалення коментарів до нерухомості;
- візуалізація результатів роботи програми у вигляді веб-додатку.

2.3.2. Вимоги до складу та параметрів технічних засобів

Мінімальні вимоги до апаратного забезпечення комп'ютера або мобільного пристрою для коректної роботи розробленого програмного продукту:

- Процесор – IntelCore I3 1.8 ГГц;
- обсяг оперативної пам'яті – 4ГБ;
- дискова підсистема – 128 ГБ;
- Мережевий адаптер - 100 Мбіт/с.

Розроблене програмне забезпечення може виконуватись на будь-яких операційних системах Windows, Linux, MacOS, Andoird, iOS за умови, що встановлен веб-браузер, необхідний для роботи продукту.

2.3.3. Вимоги до вхідних та вихідних даних

Вхідними даними для розробленого програмного продукту є база даних з інформацією про нерухомості та користувачів.

Вихідними даними є веб-додаток для ринку нерухомості з можливістю реєстрації, купівлі, продажі та оренди нерухомості.

2.3.4. Вимоги до інтерфейсу

Інтерфейс програмного продукту повинен бути декларативним. Користувач повинен мати можливість реєструватися в системі, переглядати свій профіль, додавати нерухомість до кошику або до обраного.

Взаємодія із застосунком здійснюється через додаток.

Користувач має можливість переглядати нерухомість та свій профіль.

Окрім цього, у користувача повинна бути можливість розширювати функціональність розробленого програмного продукту, маючи можливість здійснювати сортування товарів.

2.3.5. Вимоги до тестування програмного забезпечення

Для тестування програмного забезпечення необхідно виконати наступні дії:

1. Видалити нерухомість;

2. Зробити вхід в систему;

3. Редагувати нерухомість;

При виконанні вище перерахованих дій для тестування роботи додатку, користувач та адміністратор буде зареєстрований у системі, його данні з’являться у базі даних, та вони матимуть можливість обирати потрібну нерухомість та взаємодіяти з ними.

2.4 Проектування архітектури системи

Архітектура програмного забезпечення визначає, як системи функціонують на рівні їх структури. Вона описує набір компонентів, які були розроблені для виконання певних завдань або набору завдань. Архітектура програмного забезпечення створює основу, на якій можна змінювати, створювати або видаляти будь-яке програмне забезпечення, що використовується у компанії.

Архітектура програмного забезпечення має прямий вплив на якість, продуктивність, обслуговування та успіх системи з точки зору дизайну. Якщо компанія не звертає регулярну увагу на архітектуру програмного забезпечення, вона стикається з потенційними довгостроковими наслідками, такими як вразливість системи до збоїв, злому або низької продуктивності.

У сучасних системах програмного забезпечення поширені стандартні шаблони архітектури, відомі як архітектурні стилі. Зазвичай, для створення повноцінної системи використовується комбінація різних архітектурних стилів, особливо в тих випадках, коли системи були розроблені з плином часу або в них брали участь різні розробники.

В ході моделювання предметної області для «веб-додатку агентства нерухомості» за процесним підходом було виділено 3 рівня: Предметна область: Агенція нерухомості. Рівень 0: Користувачі (клієнти, адміністратори). Функціональність: реєстрація, авторизація, пошук нерухомості, кабінет. Рівень 1:

Додаткові функції для клієнтів (деталі нерухомості, обрані списки, додавання/редагування відгуків), адміністраторів (управління користувачами, додавання/редагування об'єктів). Рівень 2: Розширені функції (фільтрація, редагування улюблених, аналітика).

2.5 Проектування структури та організації компонентів

Use case діаграма - це графічне зображення, яке використовується для моделювання функціональності системи з точки зору її користувачів. Вона відображає різні взаємодії між користувачами (акторами) та системою (use case'ами), ілюструючи, які дії можуть бути виконані та які результати можуть бути отримані. На рисунку 2.4 відображена модель взаємодії клієнта та адміністратора.

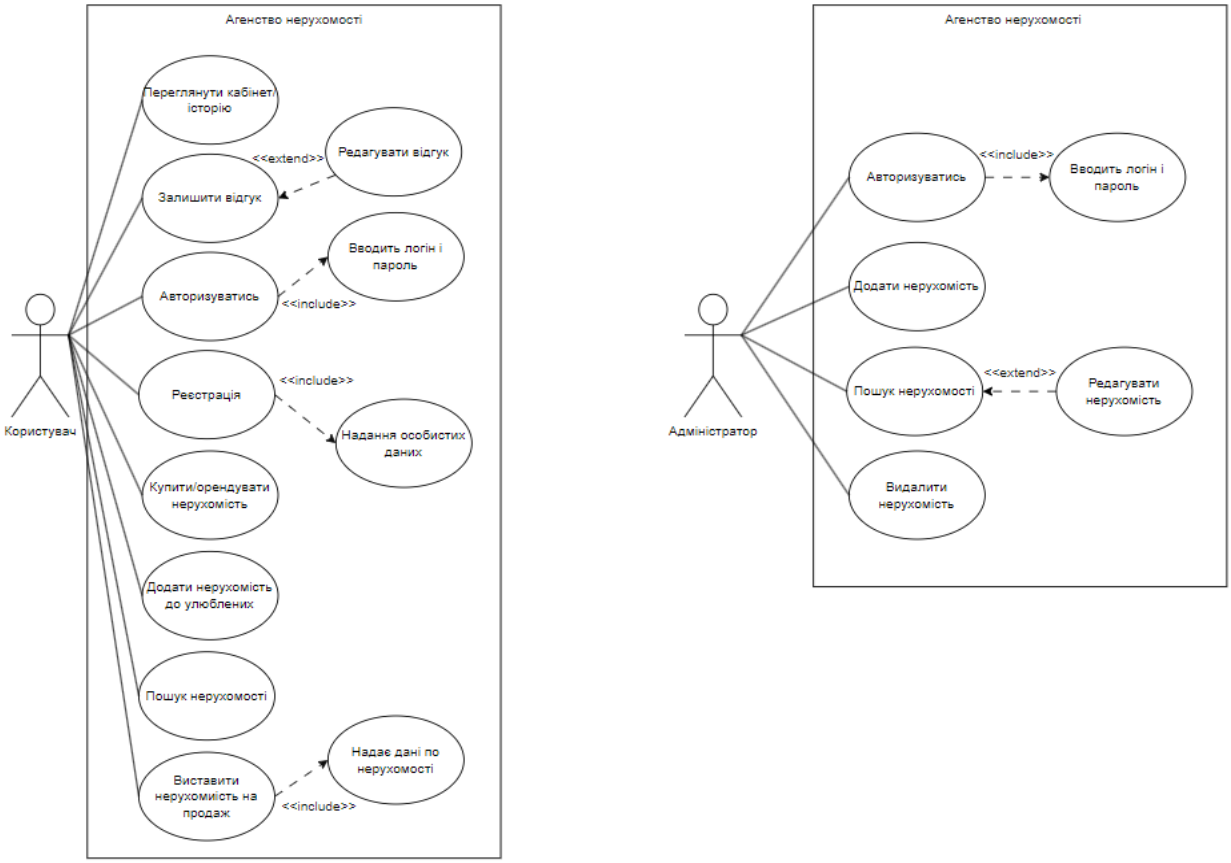


Рисунок 2.4 - Use Case діаграма

Для даної предметної області було виділено наступних акторів у таблиці 2.1:

Таблиця 2.1 – Актори у додатку та їх опис

Актор	**Короткий опис**
Адміністратор	Працівник, який додає, редагує та видаляє інформацію про нерухомість.
Клієнт	Звичайний користувач якому необхідно знайти інформацію по нерухомості, купити, орендувати, продати нерухомість або обрати до обраного чи додати відгук про нерухомість.

Можливості що надає система:

- Актор *Адміністратор* використовує систему для додавання, редагування, оновлення інформації по нерухомості, обробляє замовлення. Також використовує систему для редагування інформації по користувачам, може видаляти користувачів та змінювати права доступу для окремих користувачів.
- Актор *Клієнт* використовує систему для отримання послуг по купівлі, оренді чи продажі нерухомості, додавання до обраного нерухомості, сортування нерухомості, додавання відгуків.

Виділено наступні прецеденти у таблиці 2.2:

Таблиця 2.2 – Прецеденти у додатку та їх опис

Прецедент	**Короткий опис**
Пошук нерухомості	Запускається клієнтом. Дозволяє знаходити нерухомість за вигідною пропозицією та за вказаними характеристиками.
Купівля	Запускається клієнтом. Дозволяє купити нерухомість.
Оренда	Запускається клієнтом. Дозволяє орендувати нерухомість на певний час.
Редагування нерухомості	Запускається адміністратором. Дозволяє редагувати інформацію по нерухомості змінювати статус нерухомості.
Реєстрація	Запускається клієнтом. Дозволяє реєструвати профілі користувачів в системі.
Авторизація	Запускається всіма акторами. Дозволяє користувачам авторизувавшись отримати доступні їм дані.

Діаграма класів є статичною діаграмою, яка відображає структуру програми та її класів. Вона використовується для візуалізації, документування та моделювання різних аспектів системи. Крім того, діаграма класів може бути використана для генерації виконуваного коду у додатку. Діаграма класів описує атрибути, операції та обмеження класів системи. Вона є потужним інструментом для моделювання об'єктно-орієнтованих систем, оскільки може відображатися безпосередньо з об'єктно-орієнтованими мовами програмування. Крім того, діаграма класів іноді відома як структурна схема, оскільки вона представляє сукупність обмежень, асоціацій, співпраці та інших взаємозв'язків між класами системи. Діаграма класів показана на рис. 2.5.

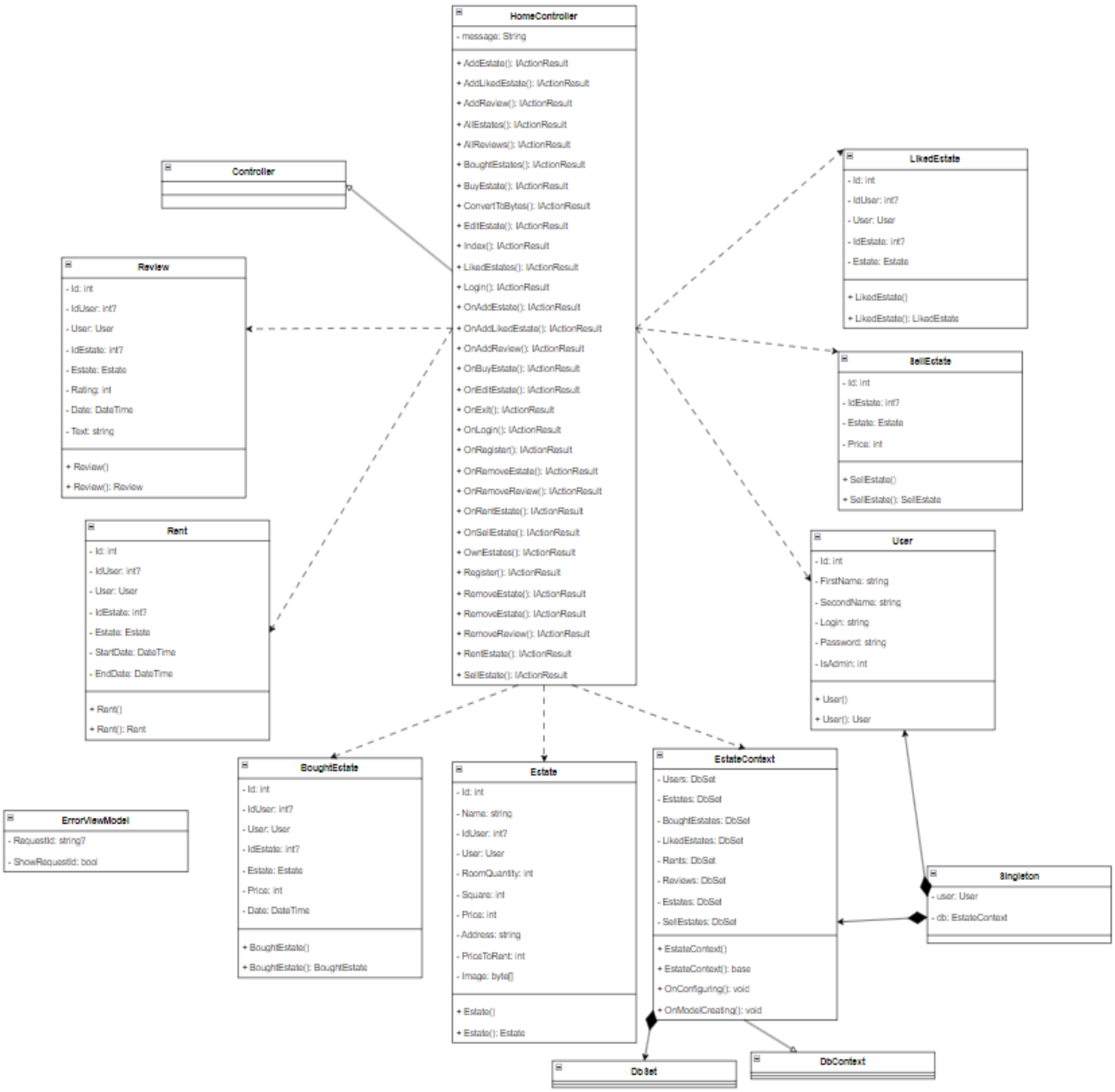


Рисунок 2.5 – Діаграма класів

Діаграма розгортання - це візуальне зображення фізичного розташування компонентів програмної системи та їх зв’язків з апаратними ресурсами в середовищі виконання. Представлений варіант діаграми розгортання для цього проекту представлений на рис. 2.6.

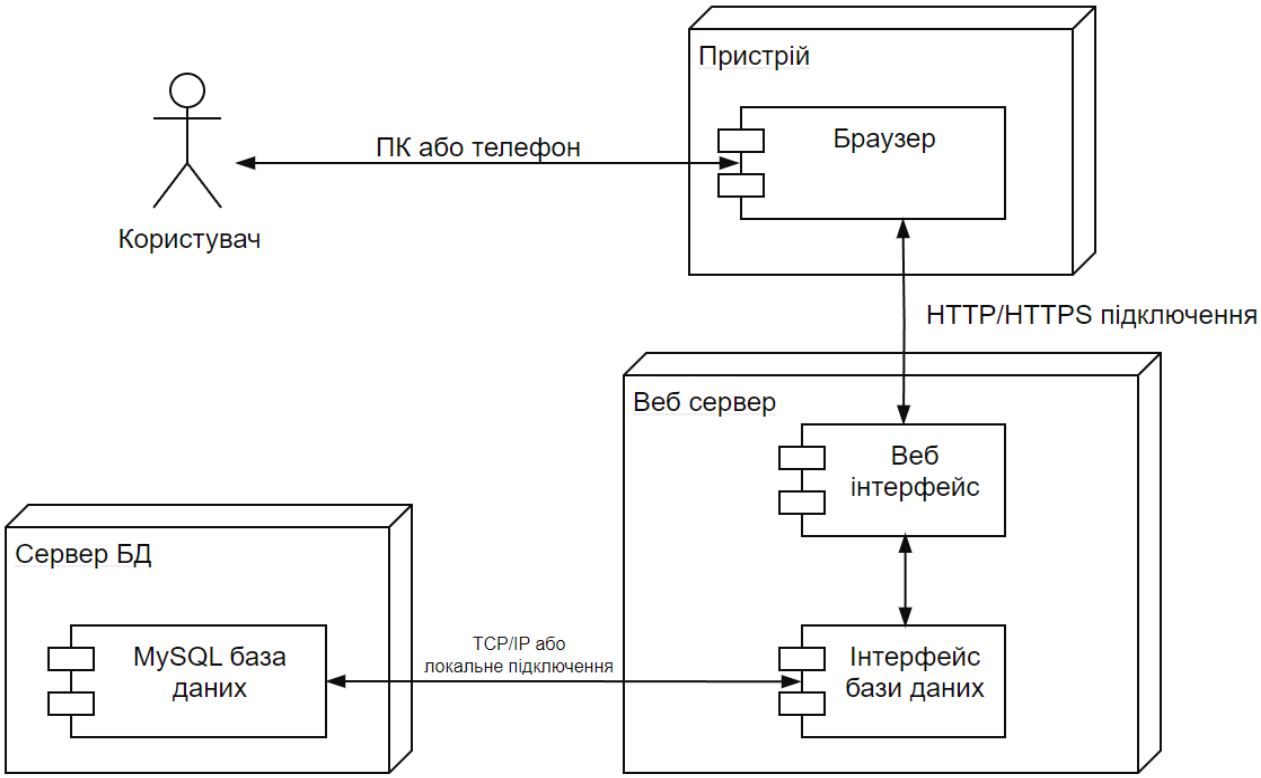


Рисунок 2.6 – Діаграма розгортання

2.6 Проектування інтерфейсу користувача

Логічні моделі для розробки програмного забезпечення для ринку нерухомості.

За допомогою діаграм розгортання, діаграм варіантів використання описуються основні користувачі системи та завдання, які має вирішувати система. За допомогою діаграм діяльності описується послідовність дій для кожного прецеденту, яка необхідна для досягнення поставленої мети.

Діаграма діяльності - це візуальне зображення послідовності дій та процесів в системі, яке використовується для моделювання бізнес-процесів, алгоритмів та управління поведінкою програм. Діаграма класів показана на рис. 2.7, 2.8, 2.9, 2.10, 2.11, 2.12

Рисунок 2.7 – Діаграма діяльності реєстрації у системі

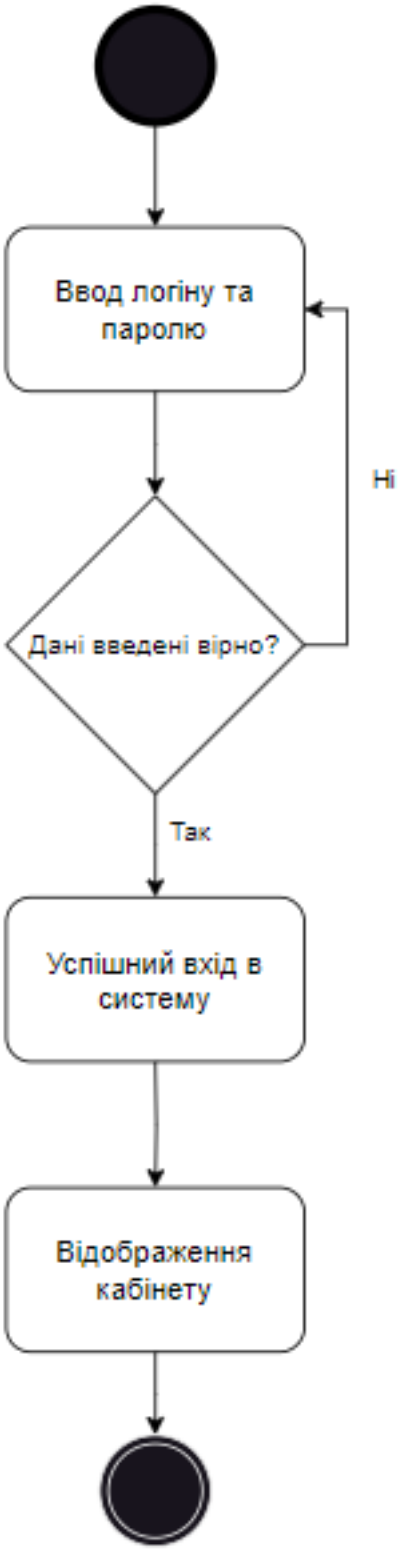


Рисунок 2.8 – Діаграма діяльності авторизації

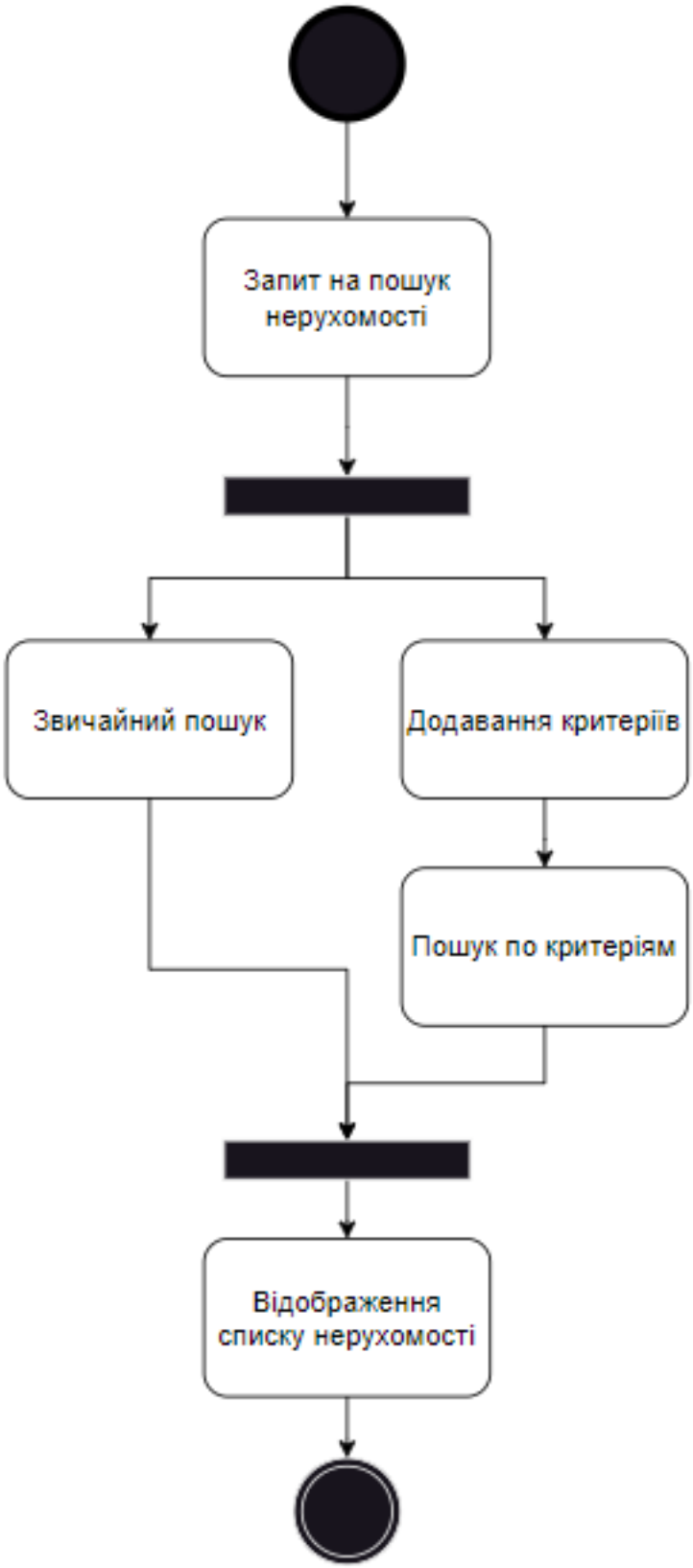


Рисунок 2.9 – Діаграма діяльності пошуку нерухомості

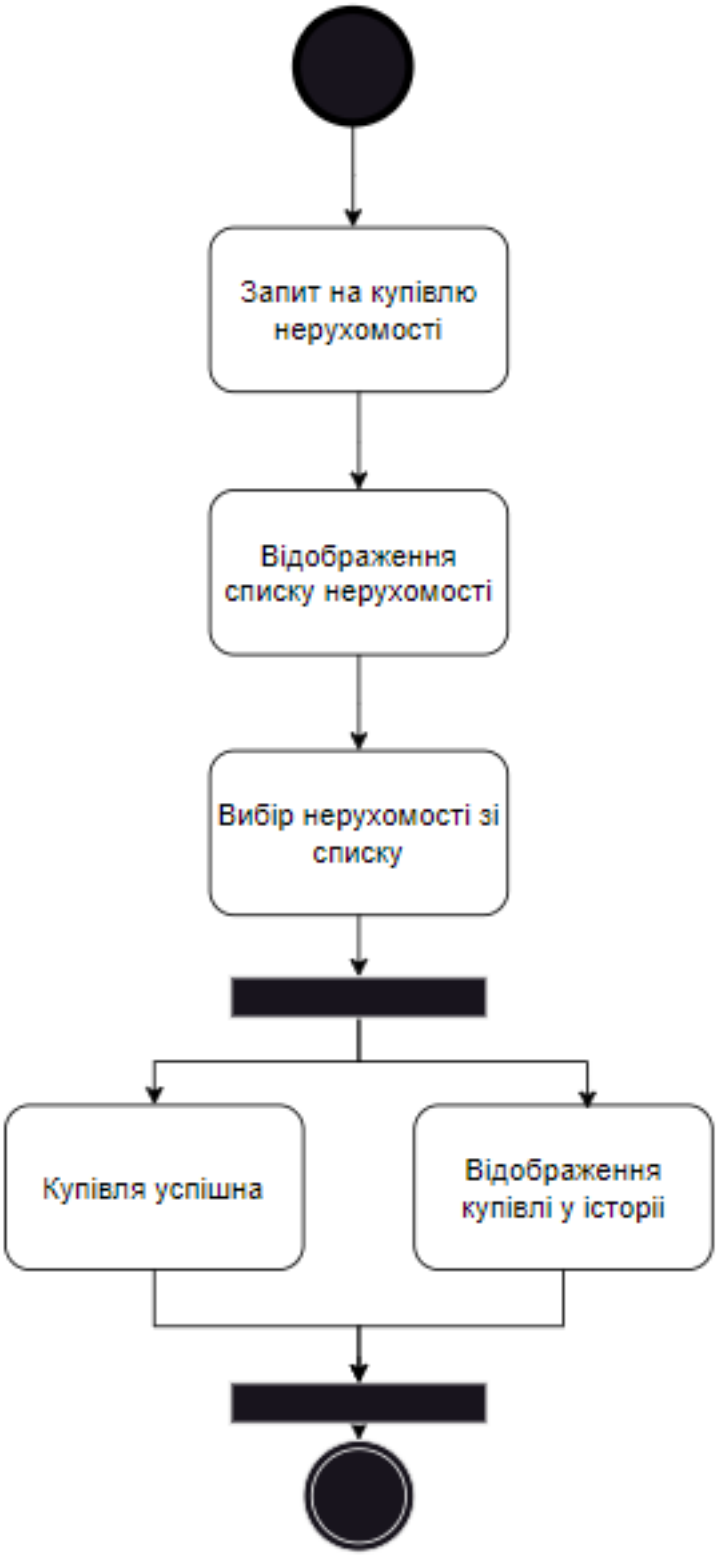


Рисунок 2.10 – Діаграма діяльності купівлі нерухомості

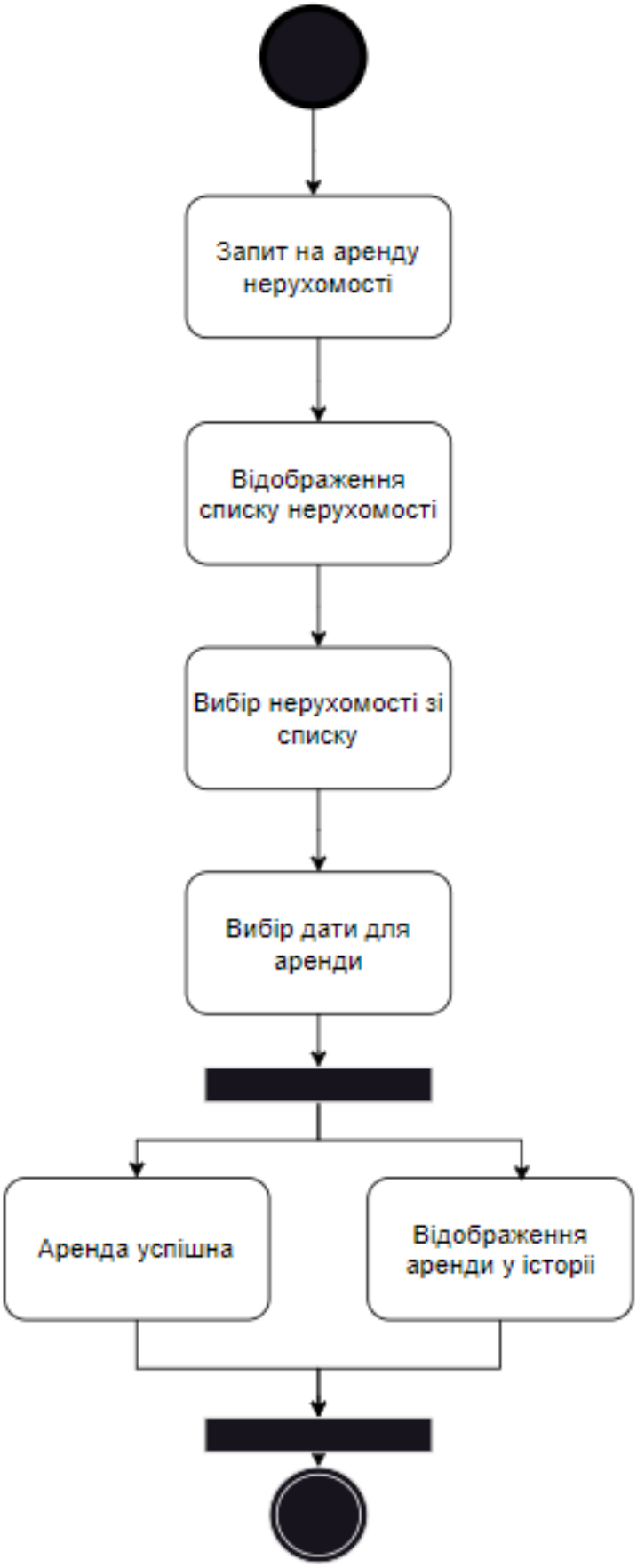


Рисунок 2.11 – Діаграма діяльності аренди нерухомості

Рисунок 2.12 – Діаграма діяльності продажу нерухомості

3. ЕТАПИ РОЗРОБКИ ТА СТВОРЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Структура програмного забезпечення

На етапі логічного проектування описується організація елементів, що становлять програмне рішення. Модель, отримана на стадії логічного проектування, повинна забезпечувати:

- незалежність від засобів розробки;
- простота моделі;
- відображення структури програмного продукту, що розробляється.

У логічній моделі створюються алгоритми чи інші логічні елементи, з яких здійснюється рішення прикладної задачі. На рис. 3.1. показана діаграма потоків даних, що показує взаємодію користувача та програми.

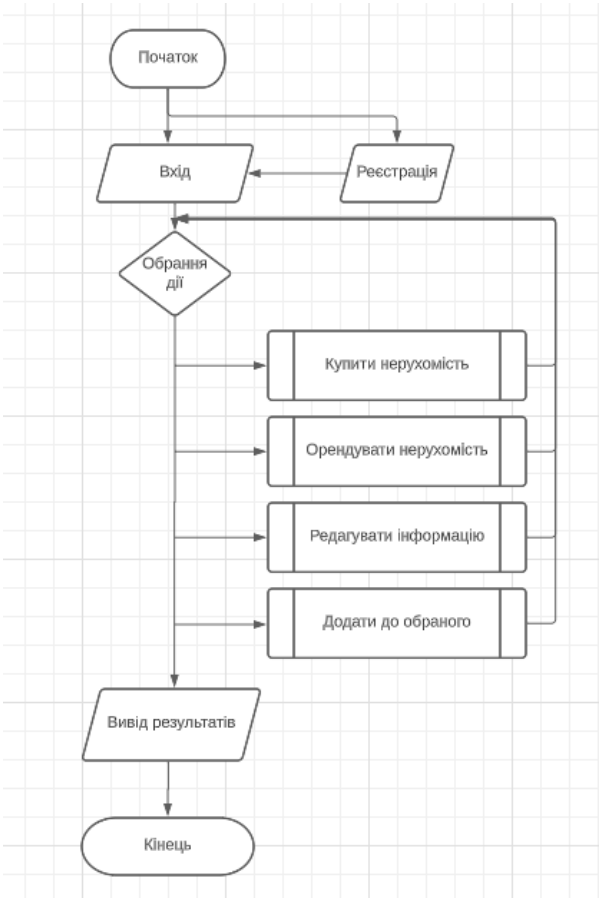


Рисунок 3.1 – Блок-схема роботи програми

3.2 Проектування та розробка БД

Для реалізації веб-додатка була використана стандартна клієнт-серверна архітектура з трьома рівнями: рівень інтерфейсу користувача, серверна частина для логічної обробки, та база даних для управління даними. Ця архітектура враховує вимоги до продуктивності та розподілу рівнів доступу до інформації. Використано клієнт-серверну архітектуру з товстим сервером і тонким клієнтом, щоб підтримувати доступ до додатка з будь-якого браузера та пристрою, мінімізуючи навантаження на пристрої користувача. Додатково, ця архітектура забезпечує віддалене зберігання даних, незалежно від сеансу користувача або роботи сервера. Схематично архітектура програми представлена рис. 3.2.

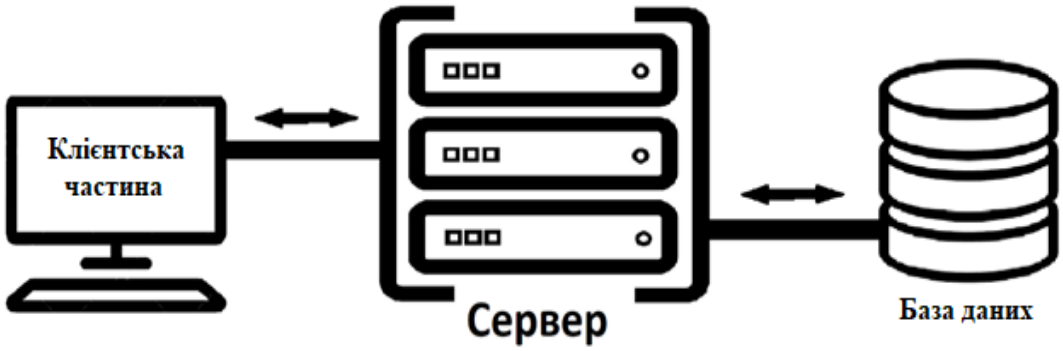


Рисунок 3.2 – Архітектура програми

У системі керування контентом застосовується СУБД MySQL. База даних складається із таблиці.

Усі дані зберігаються на локальному сервері в окремих базах даних

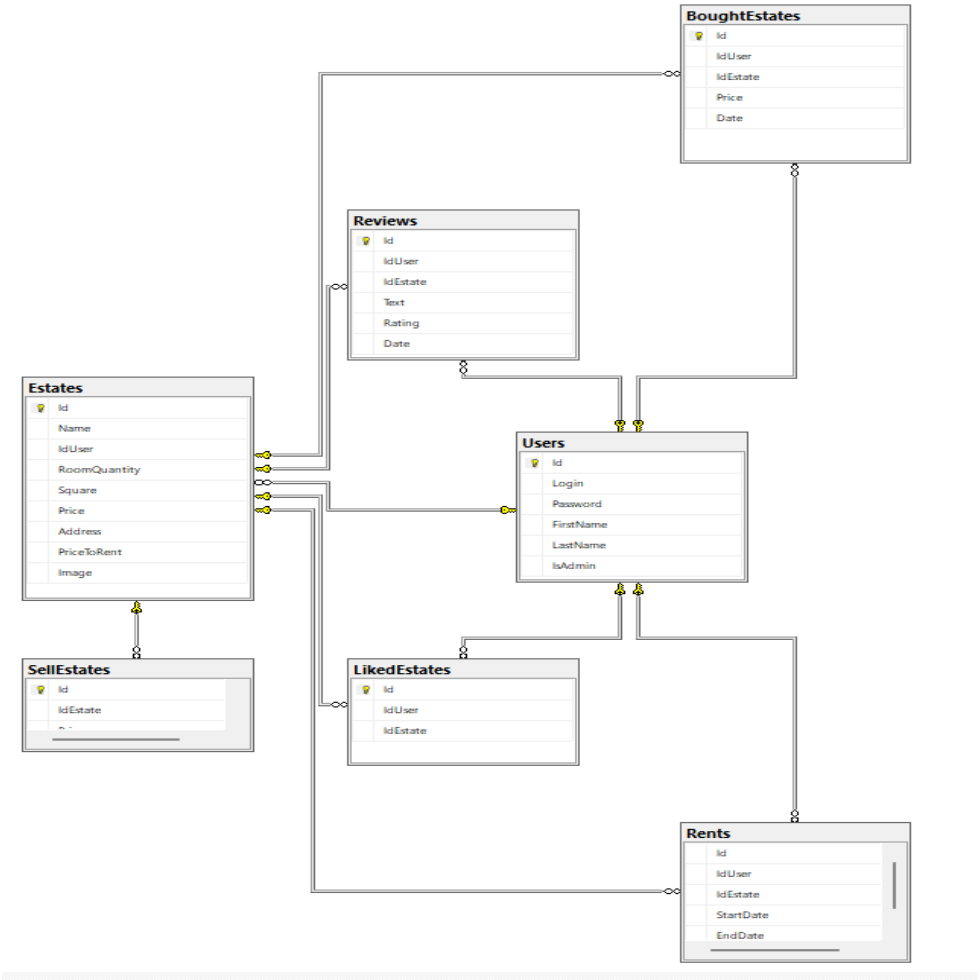


Рисунок 3.3 - Структура БД

Для підключення до бази даних та її використання наступний код програми.

```
namespace EstateManager.Models {
    public static class Singleton {
        public static EstateContext db { get; private set; } =
new EstateContext();
        public static User user { get; set; }
    }
}
```

```
 using Microsoft.EntityFrameworkCore;

namespace EstateManager.Models {
    public class EstateContext : DbContext {
        public DbSet<User> Users { get; set; }
        public DbSet<Estate> Estates { get; set; }
```

```
        public DbSet<BoughtEstate> BoughtEstates { get; set; }
        public DbSet<LikedEstate> LikedEstates { get; set; }
        public DbSet<Rent> Rents { get; set; }
        public DbSet<Review> Reviews { get; set; }
        public DbSet<SellEstate> SellEstates { get; set; }
        public EstateContext(DbContextOptions<EstateContext>
options) : base(options) { }
        public EstateContext() { }
        protected override void
OnConfiguring(DbContextOptionsBuilder optionsBuilder)
        => optionsBuilder.UseSqlServer("Data
Source=(localdb)\\MSSQLLocalDB;Initial Catalog=Estates;Integrated
Security=True;Connect
Timeout=30;Encrypt=False;TrustServerCertificate=False;ApplicationInt
ent=ReadWrite;MultiSubnetFailover=False");
        protected override void OnModelCreating(ModelBuilder
modelBuilder) {
            modelBuilder.Entity<Estate>().Navigation(p =>
p.User).AutoInclude();
            modelBuilder.Entity<BoughtEstate>().Navigation(p =>
p.User).AutoInclude();
            modelBuilder.Entity<BoughtEstate>().Navigation(p =>
p.Estate).AutoInclude();
            modelBuilder.Entity<LikedEstate>().Navigation(p =>
p.User).AutoInclude();
            modelBuilder.Entity<LikedEstate>().Navigation(p =>
p.Estate).AutoInclude();
            modelBuilder.Entity<Rent>().Navigation(p =>
p.User).AutoInclude();
            modelBuilder.Entity<Rent>().Navigation(p =>
p.Estate).AutoInclude();
            modelBuilder.Entity<Review>().Navigation(p =>
p.User).AutoInclude();
            modelBuilder.Entity<Review>().Navigation(p =>
p.Estate).AutoInclude();
```

```
                modelBuilder.Entity<SellEstate>().Navigation(p =>
p.Estate).AutoInclude();
            }

        }
    }
```

3.3 Опис функціоналу

Для входу в систему було створено форму із двох полів для логіна та паролю, та двох кнопок входу (рис. 3.6) та реєстрація(рис. 3.7)

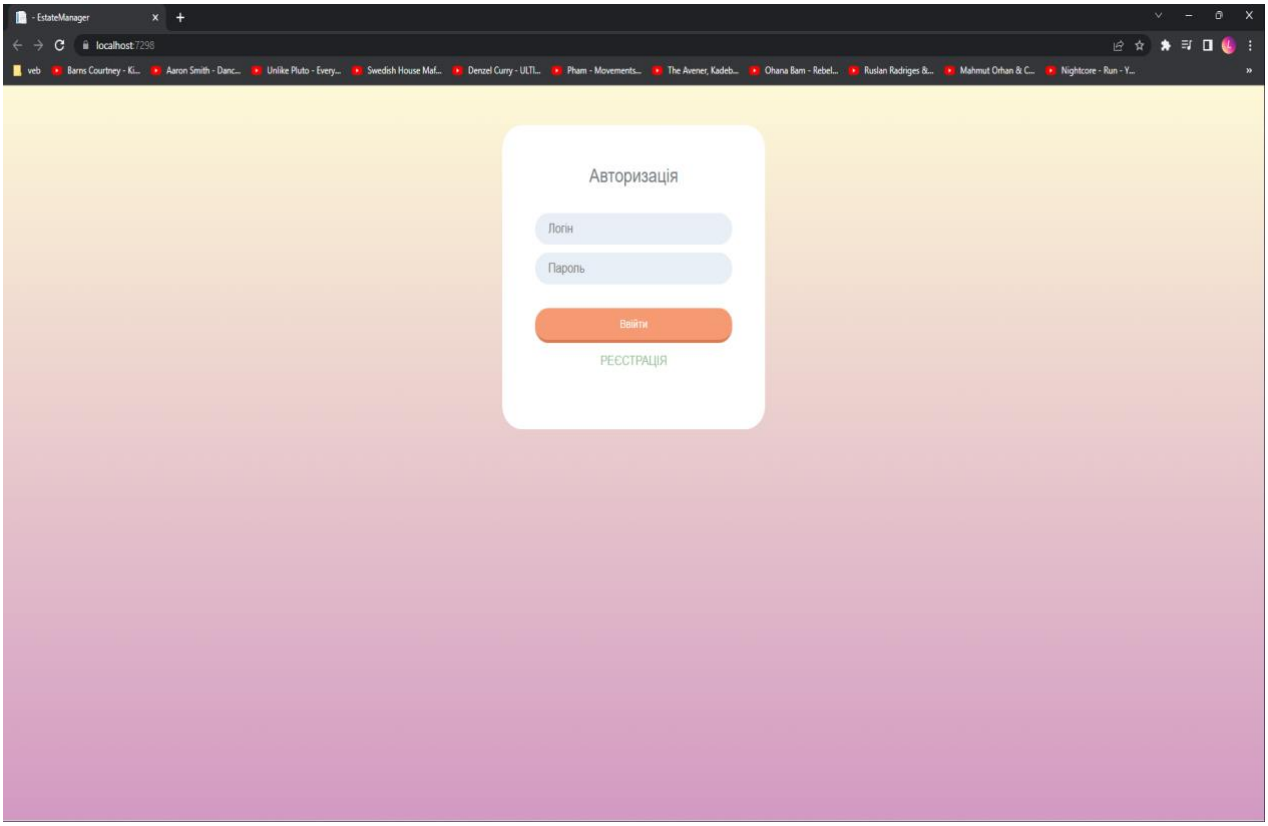


Рисунок 3.6 – Форма входу

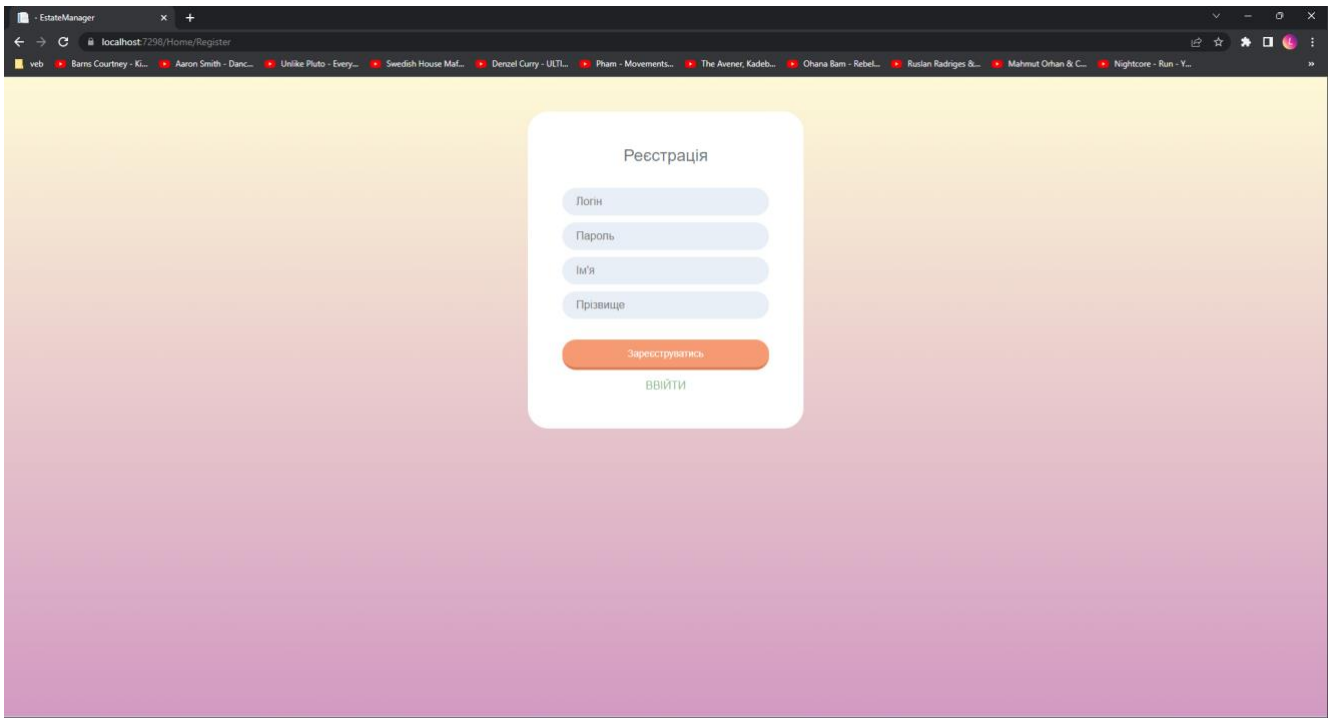


Рисунок 3.7 – Форма реєстрації

Після входу, перед користувачем з’являється наступна сторінка (рис. 3.8)

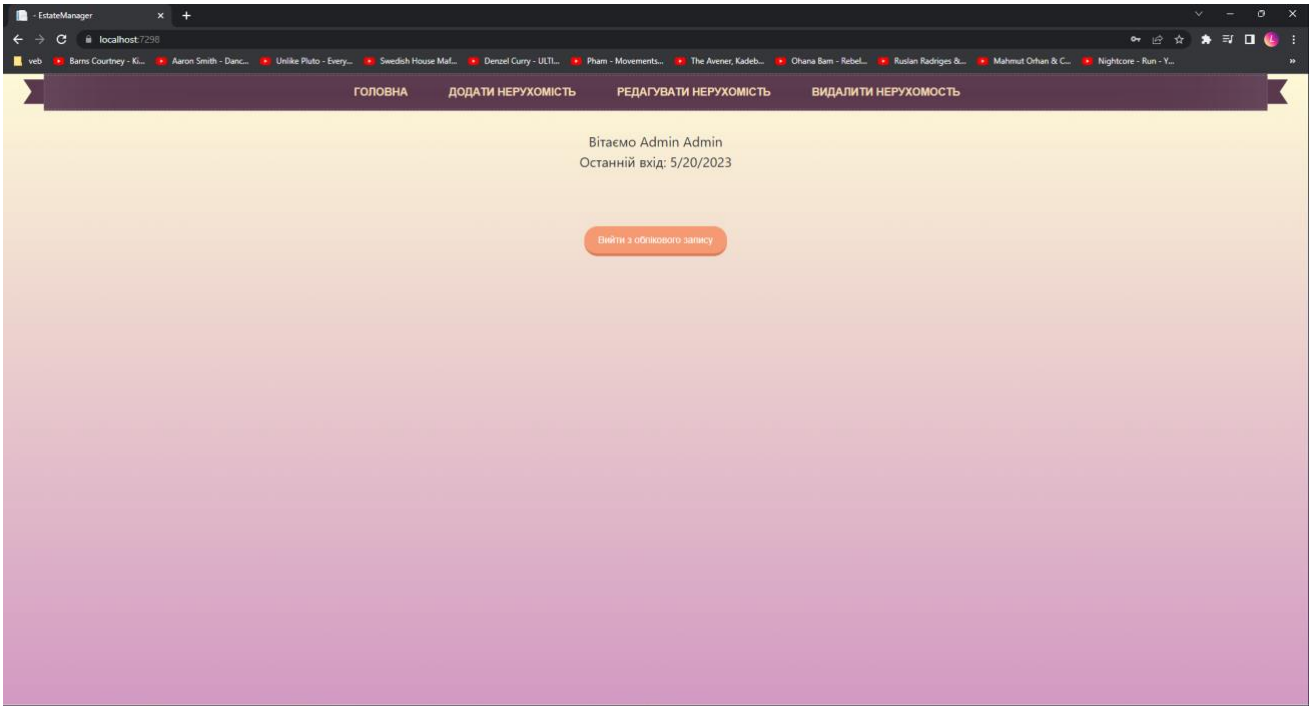


Рисунок 3.8 – вхід Admin

Ввійшовши до системи користувач може переглянути список нерухомості (рис. 3.9).

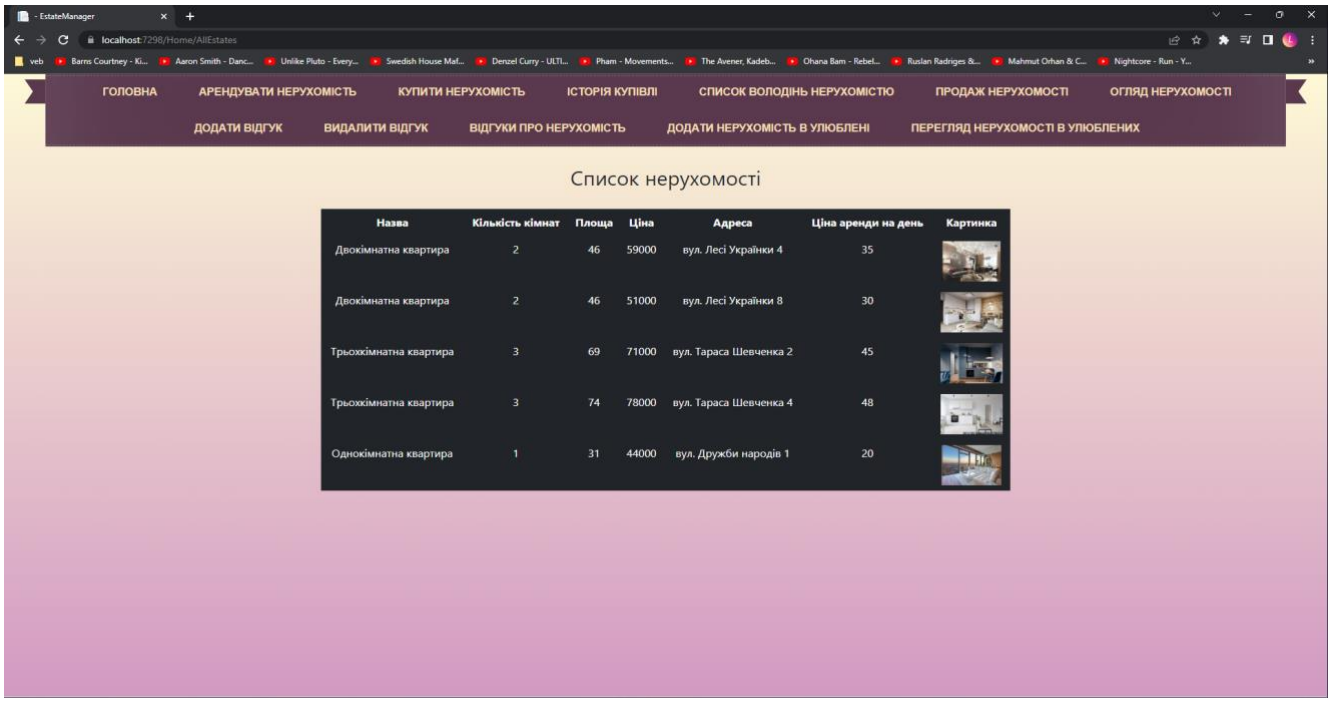


Рисунок 3.9 – Список нерухомості

Для оренди нерухомості необхідно натиснути на відповідну вкладку «Орендувати нерухомість» (рис. 3.10)

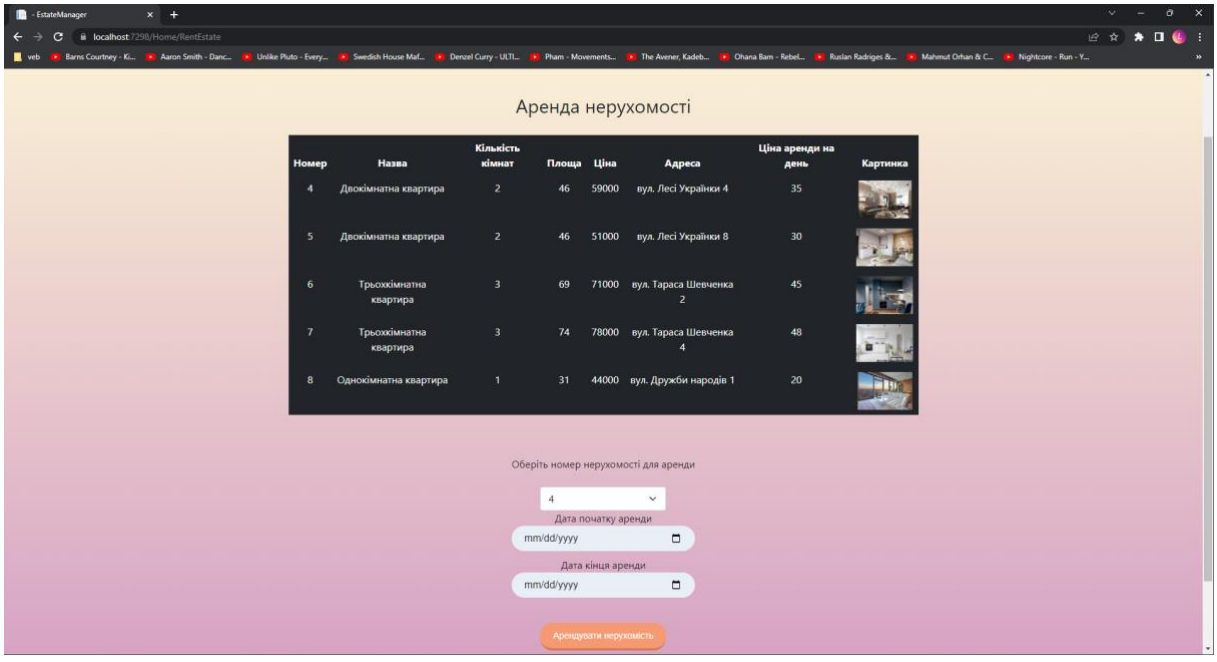


Рисунок 3.10 – Оренда нерухомості

Щоб купити нерухомість необхідно перейти н вкладку «Купити нерухомість» (рис. 3.11)

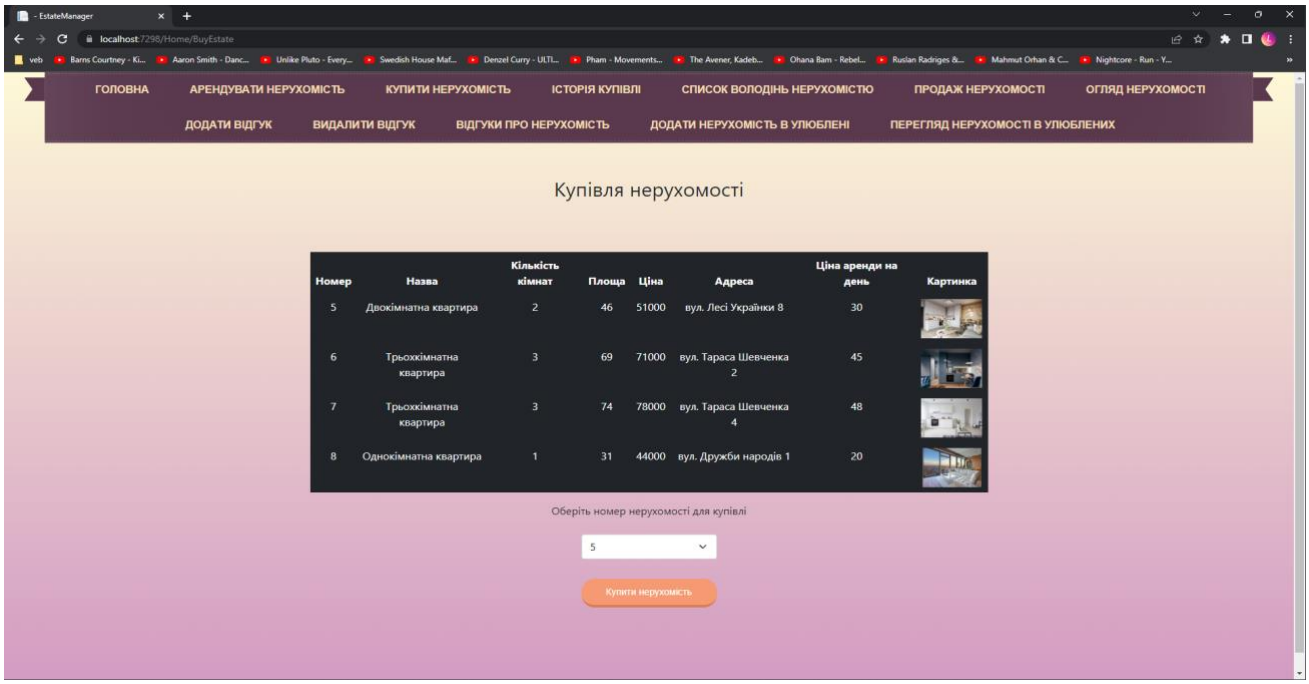


Рисунок 3.11 – Купівля нерухомості

Після купівлі нерухомості, вся інформація відображається на вкладці «Історія купленої нерухомості» (рис. 3.12)

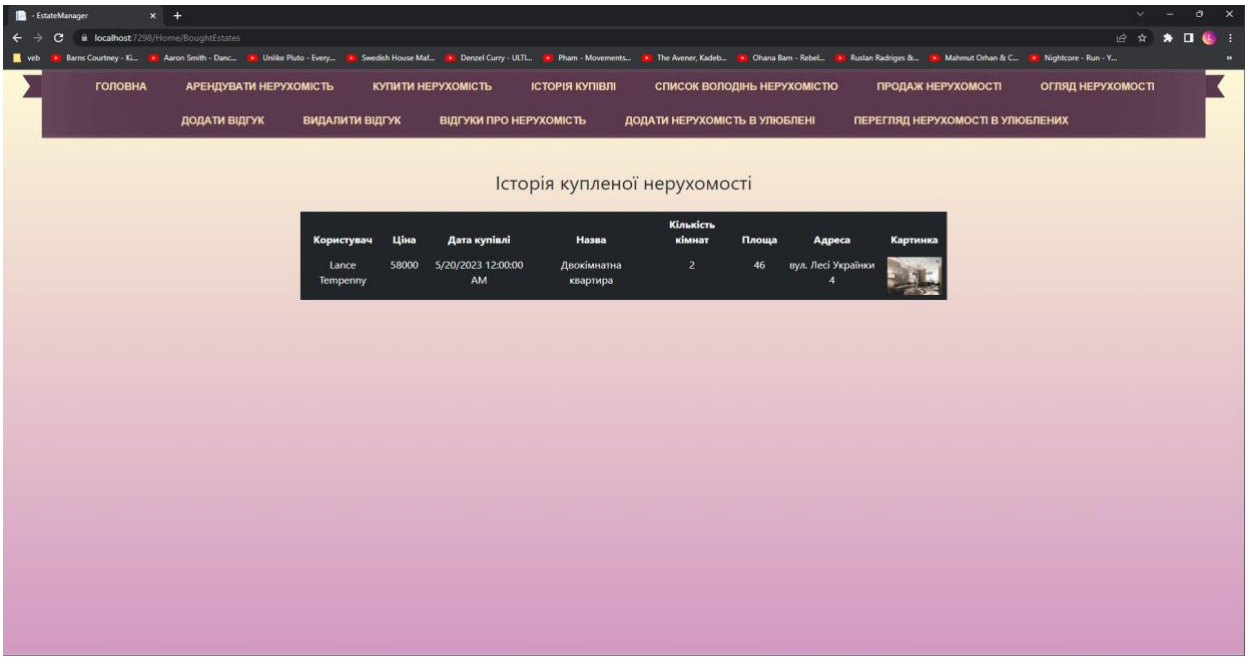


Рисунок 3.12 – Історія купленої нерухомості

Також можна переглянути список нерухомості якими володіє користувач (рис. 3.13)

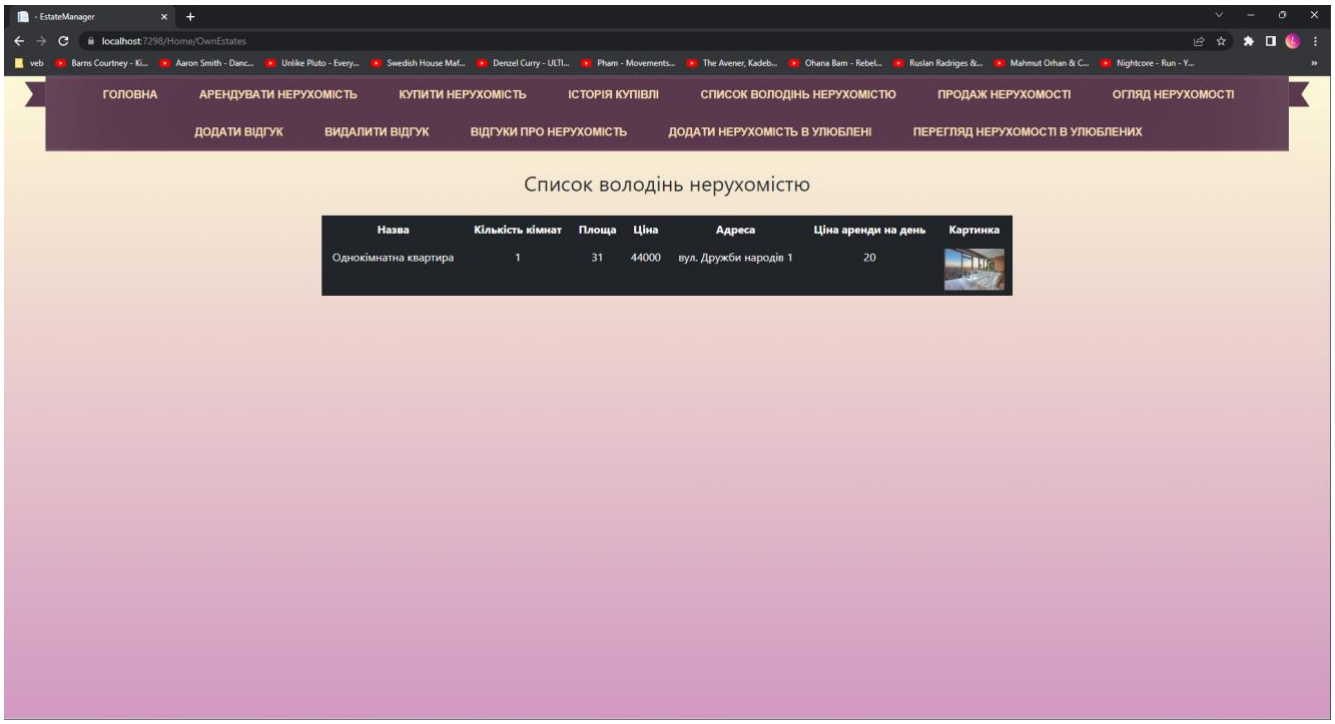


Рисунок 3.12 – Список володінь нерухомістю

Якщо користувач бажає продати нерухомість, він може перейти до вкладки «Продаж нерухомості» (рис. 3.13)

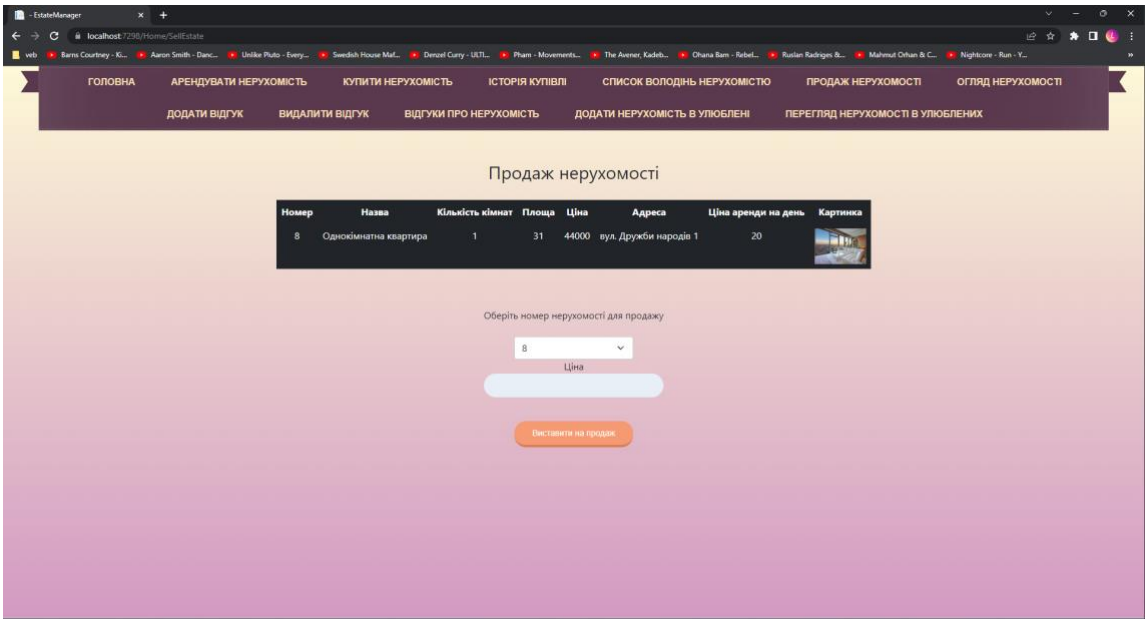


Рисунок 3.13 – Продаж нерухомості

Є також можливість переглянути відгуки (рис. 3.14) додати або видалити відгук (рис. 3.15)

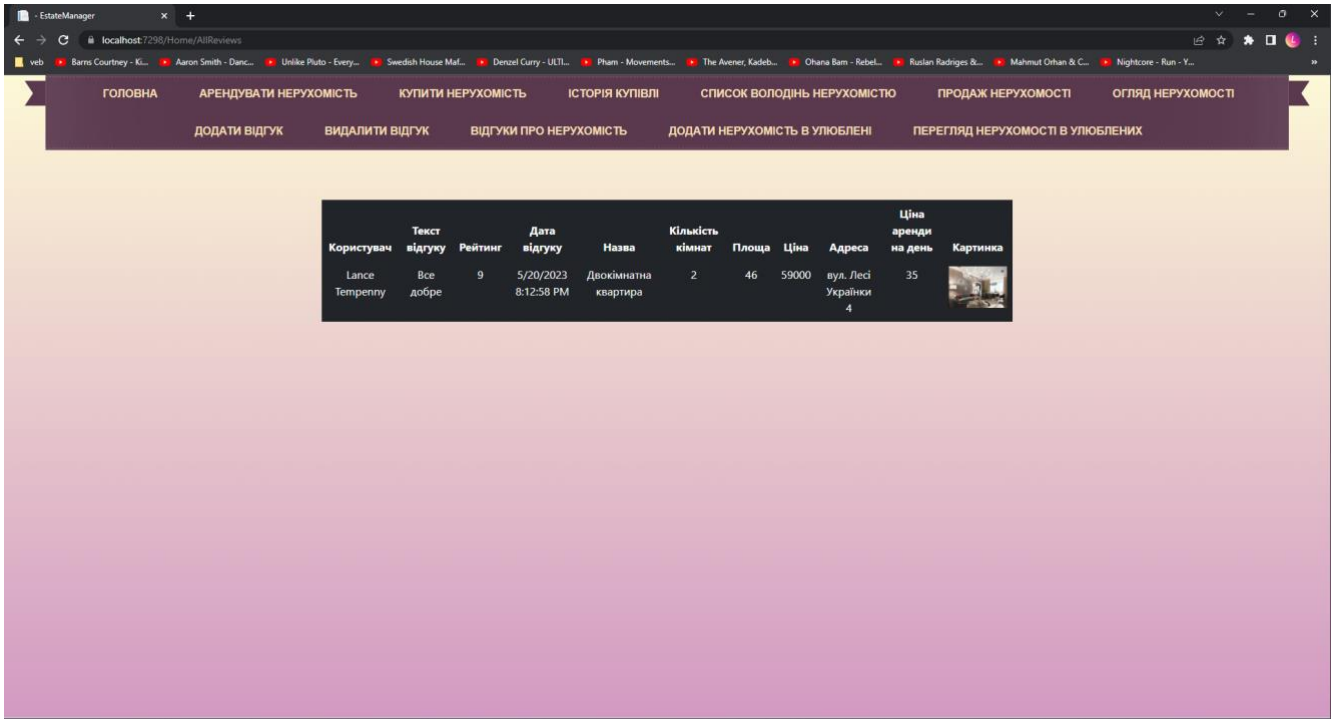


Рисунок 3.14 – Перелік відгуків

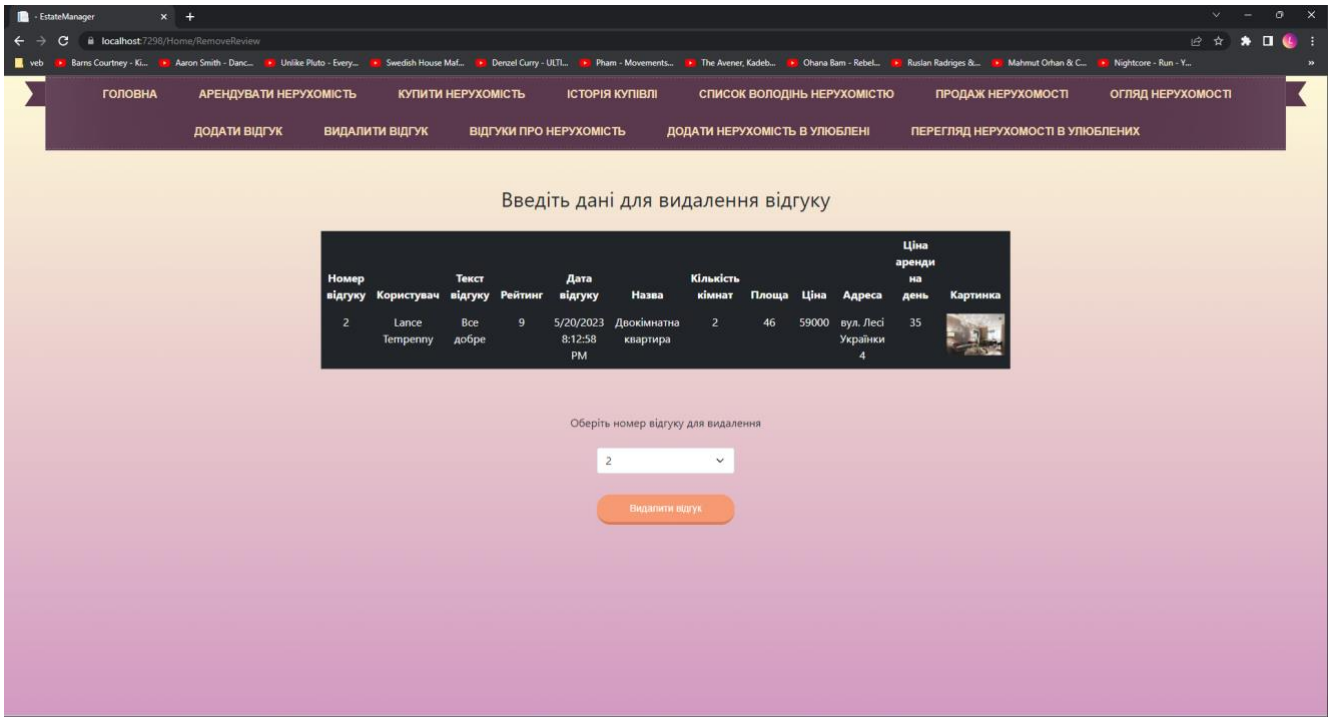


Рисунок 3.15 – Видалення відгуку

Якщо користувачу сподобалась якась нерухомість (рис. 3.16), він може її додати до улюблених, а потім може повернутися до них (рис. 3.17).

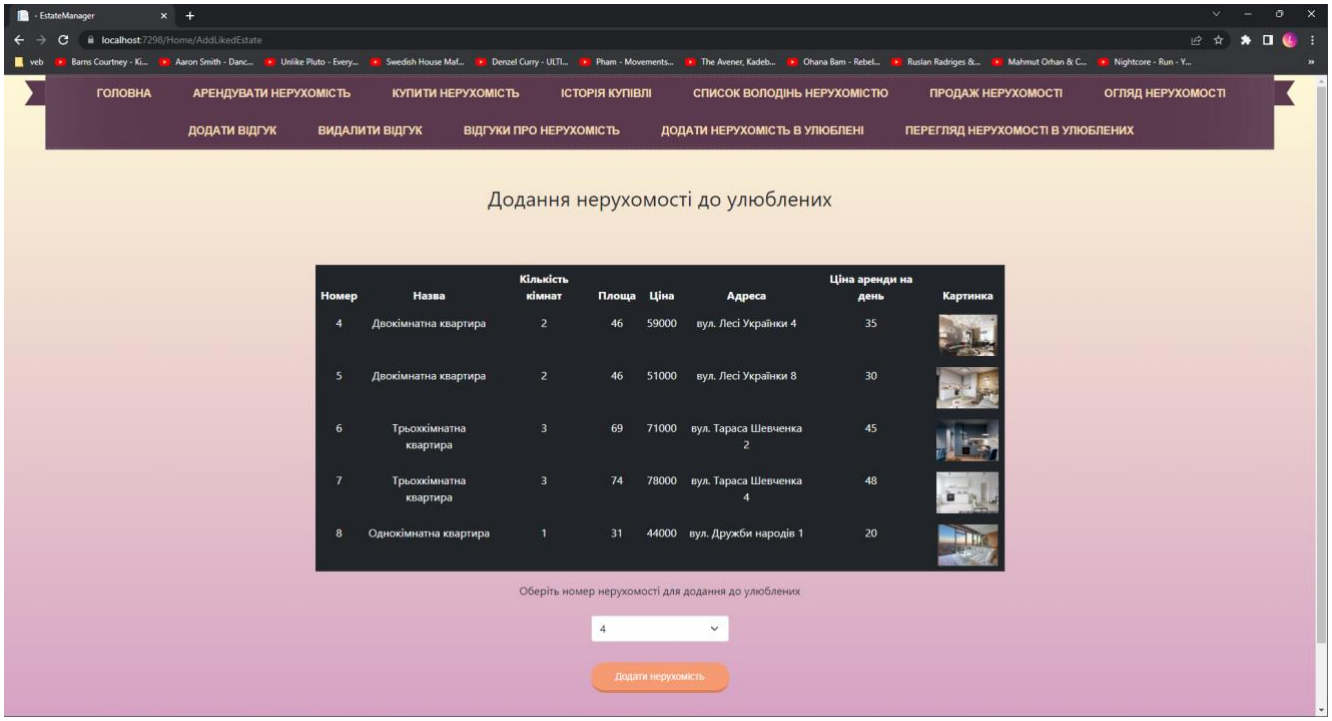


Рисунок 3.16 – Перелік нерухомостей

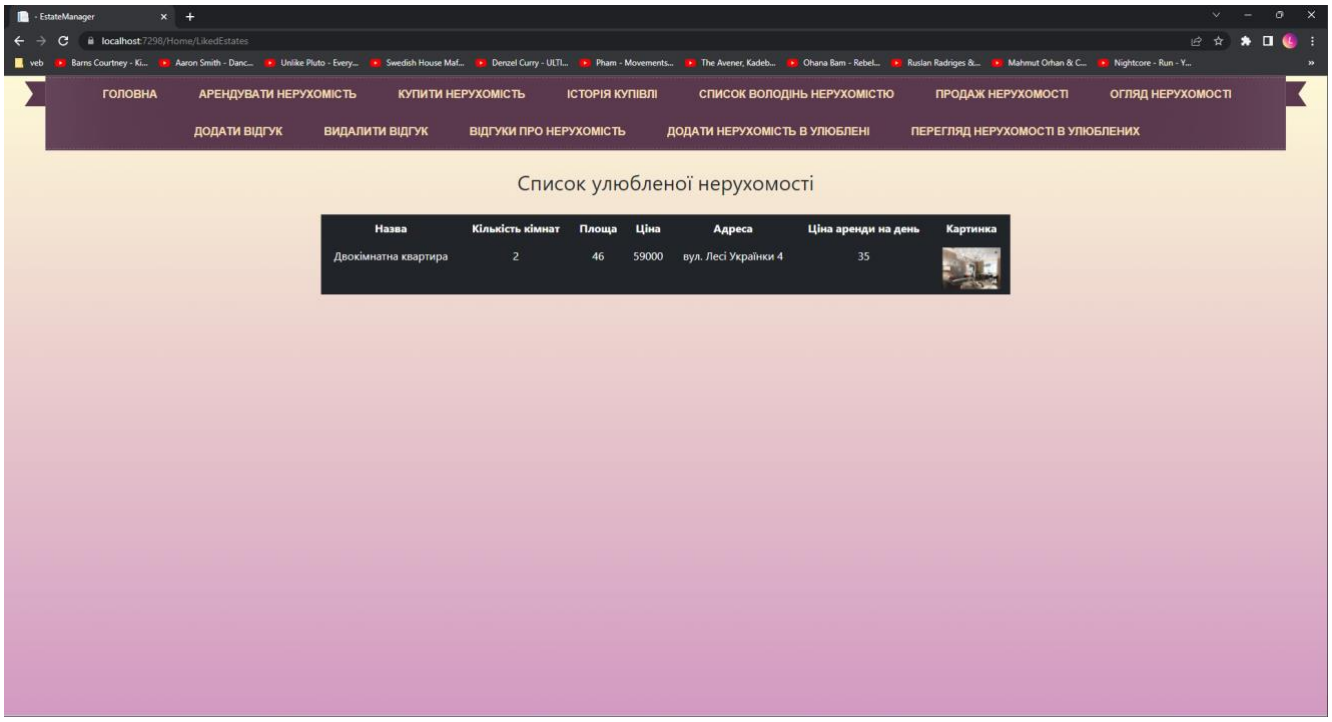


Рисунок 3.17 – Перелік улюблених нерухомостей

Якщо у користувача є нерухомість, то він може її додати до бази даних. Для цього необхідно ввести дання про нерухомості (рис. 3.18).

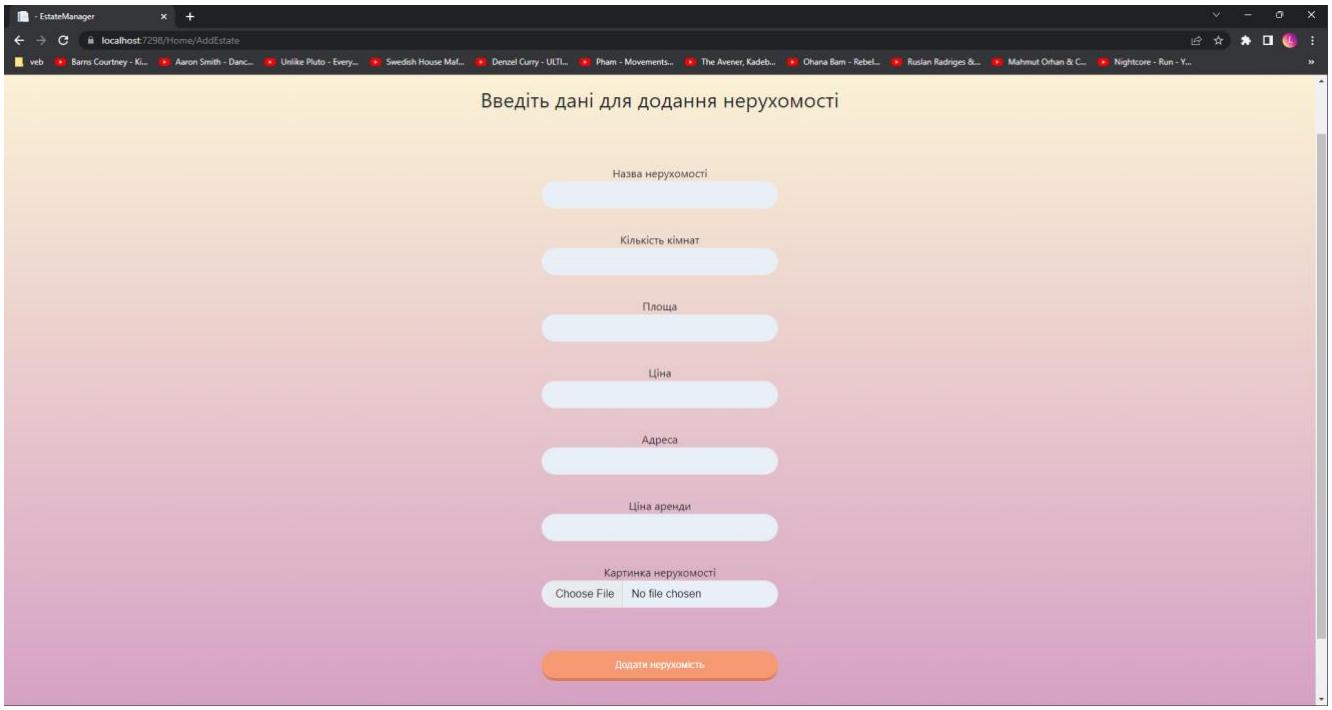


Рисунок 3.18 – Додавання нерухомості

3.4 Тестування

Під час тестування, усі виникаючі помилки одразу ж виправлялися. Нижче наведено декілька прикладів тестування роботи програми.

Тест на видалення нерухомості

Ідентифікатор тест-варіанту	Адміністратор видаляє обрану нерухомість
Набір вхідних даних	Номер для видалення
Очікувані результати	Адміністратор успішно видаляє обрану нерухомість
Виконувані дії	Адміністратор обирає номер для видалення, та натискає на кнопку.

Результат тесту наведено на рис. 3.19

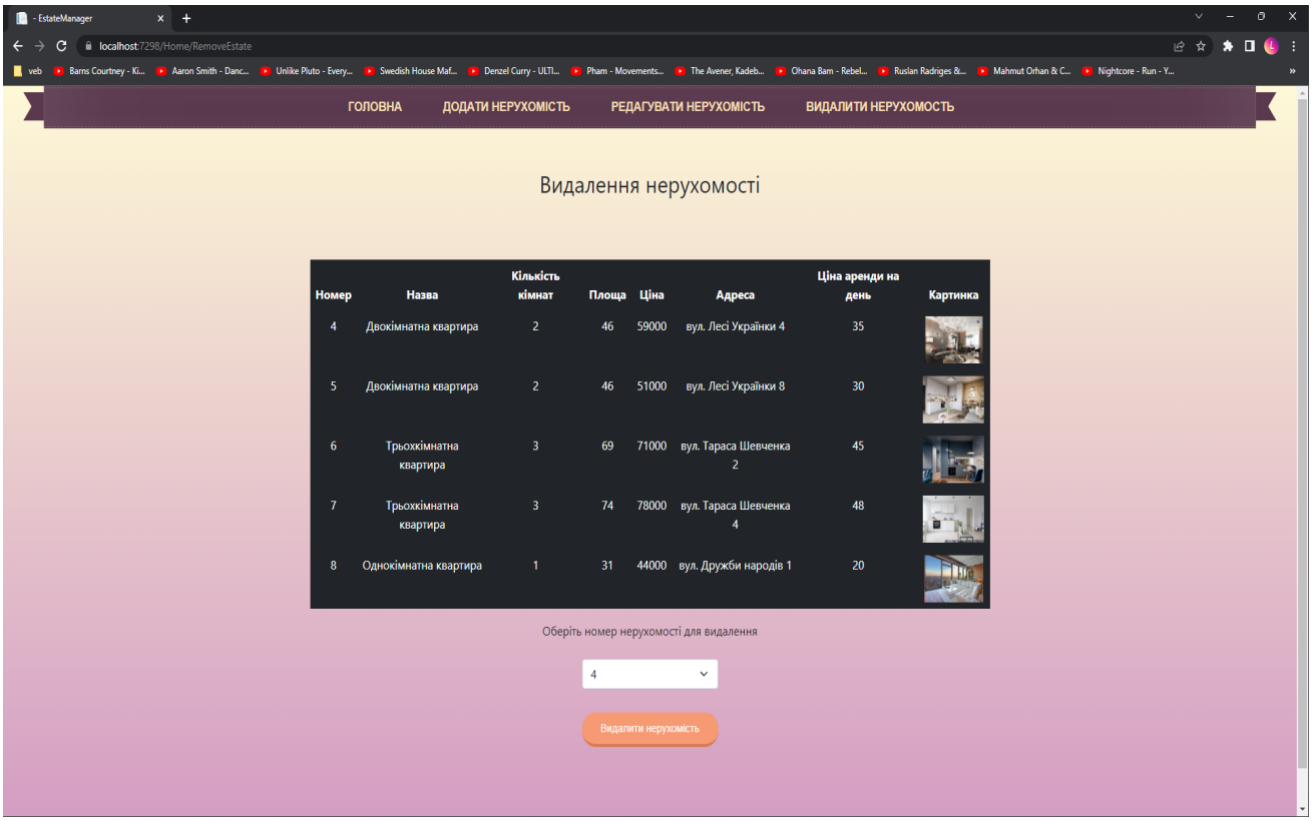


Рисунок 3.19 – результати тестування

Тест пройдено успішно.

Тест на вхід у систему

Ідентифікатор тест-варіанту	Вхід в систему
Набір вхідних даних	Логін та пароль
Очікувані результати	Користувач входить у систему
Виконувані дії	Користувач вводить логін та пароль

Результат тесту наведено на рис. 3.20.

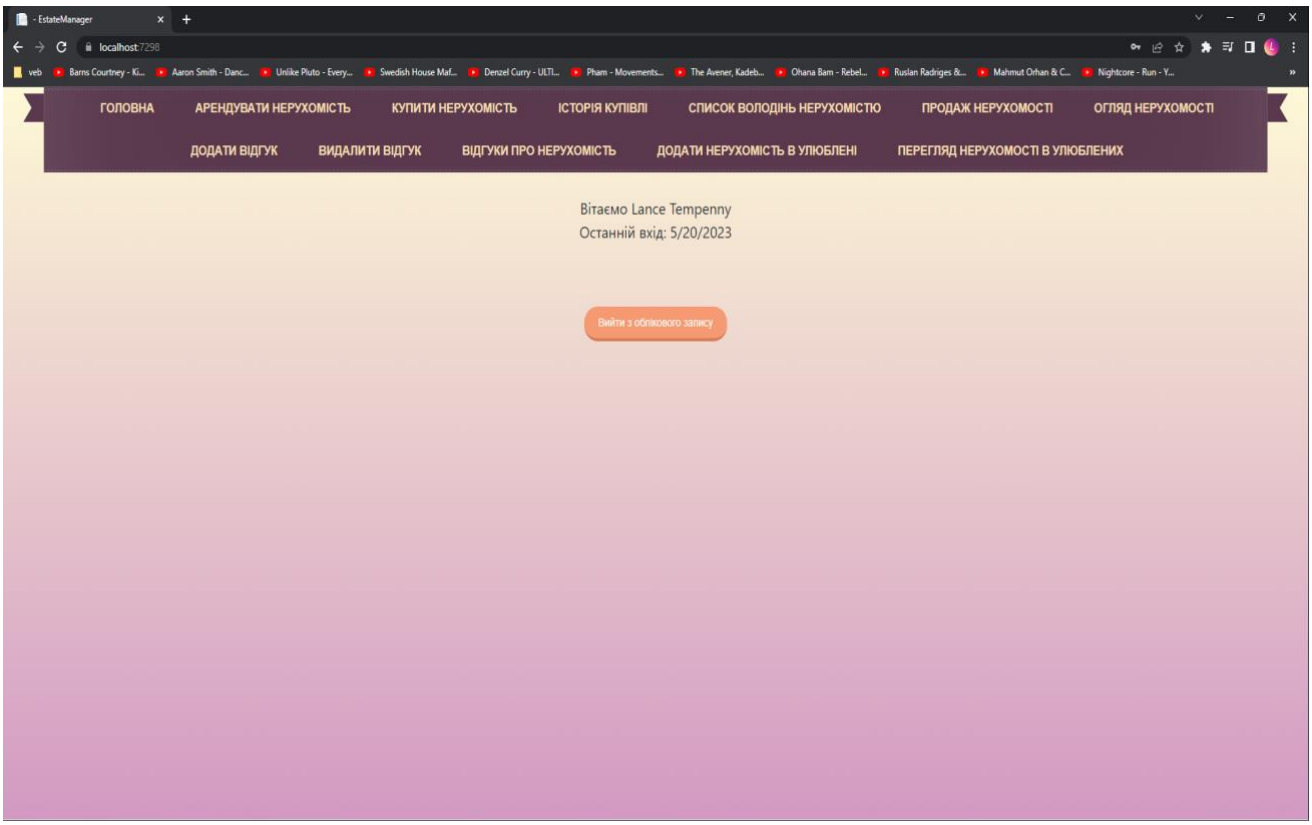


Рисунок 3.20 – вхід в систему

Тест пройдено успішно.

Тест на редагування нерухомості

Ідентифікатор тест-варіанту	Користувач змінює інформацію по нерухомості
Набір вхідних даних	нова інформація по нерухомості
Очікувані результати	користувач успішно змінює інформацію по нерухомості
Виконувані дії	Користувач обирає номер нерухомості для зміни, та вводить у відповідні поля значення.

Результат тесту наведено на рис. 3.21

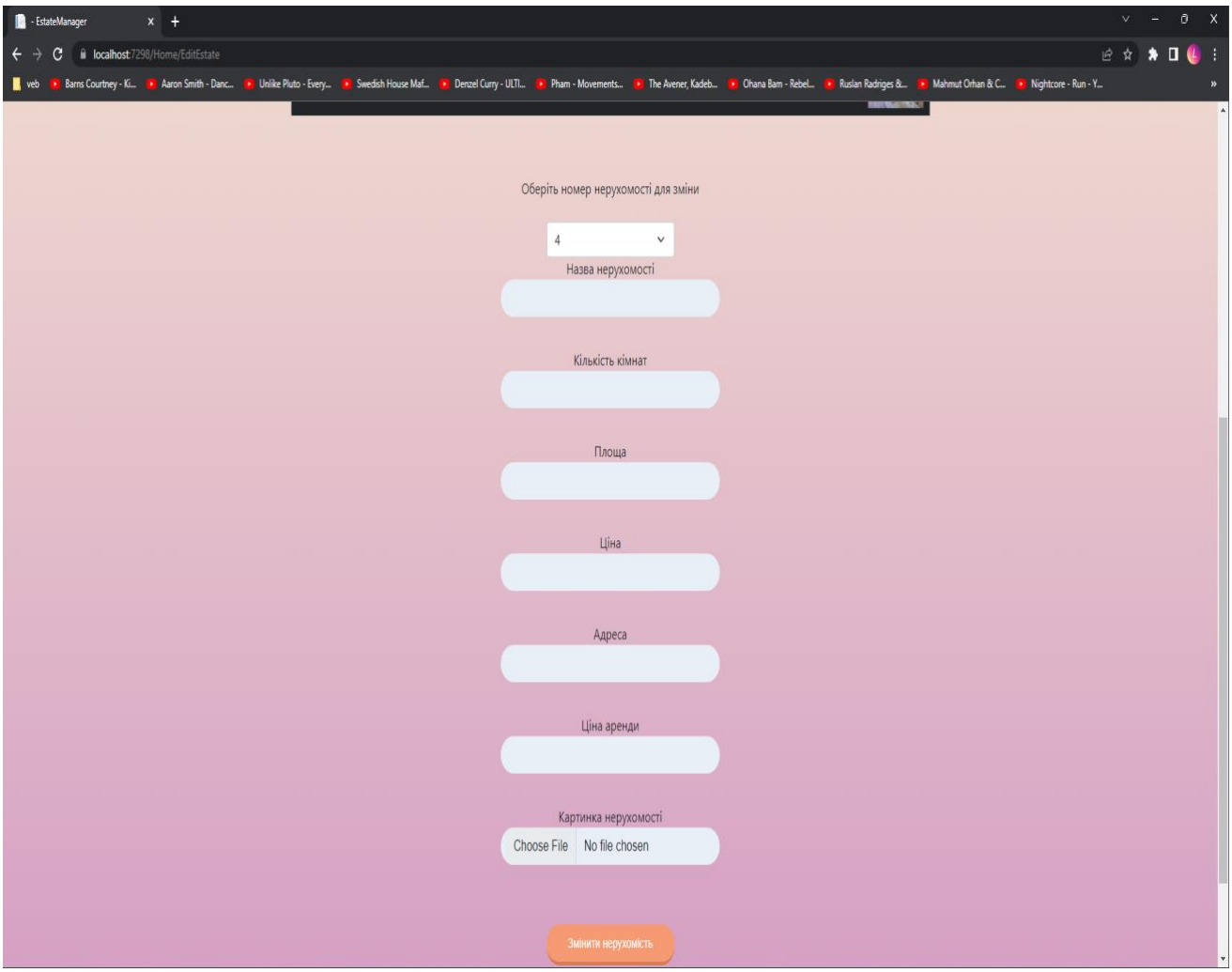


Рисунок 3.21 – результати тестування

Тест пройдено успішно.

Таким чином, нами було проведене тестування роботи розробленого додатка. За результатами випробувань програмне забезпечення показало свою працездатність. Розроблений веб-сайт пройшов всі тестування та показав відмінний результат.

ВИСНОВКИ

В ході цієї роботи було розроблено програмне забезпечення для ринку нерухомості.

Програма надає всю необхідну інформацію клієнтам про послуги, що надаються, а також має дошку оголошень і дає можливість отримати додатковий канал для маркетингу та реклами.

У процесі впровадження було проведено аналітичний розгляд проблеми розробки програмного забезпечення, проаналізовано існуючі аналоги агентств нерухомості та розглянуто їх функціональність.

Також проаналізовано тематику, сформульовано вимоги до програми.

Під час проектування додатку були розроблені діаграми: діаграма прецедентів та діаграма класів; для цієї схеми були визначені та описані основні елементи та функціональні блоки; діаграма діяльності, для цієї діаграми були описані розділи авторизації, типовий перебіг подій і винятки; діаграма розгортання.

Також проведено: розробку інформаційного забезпечення та визначено основні інформаційні об’єкти, а також описано таблиці реляційної бази даних.

Було представлено комп’ютерну реалізацію програмного забезпечення на мові C#, включаючи повний опис роботи програми та скріншоти, складено методичні рекомендації для адміністратора та користувача агентства нерухомості.

При тестуванні роботи, програма показала відмінній результат. В подальшому може бути доповнена додатковими методами та функціоналом.

ПЕРЕЛІК ПОСИЛАНЬ

1. Бабій М. С. Теорія програмування. Суми: Вид-во СумДУ, 2009. 181 с.
2. Вінник В. Ю. Алгоритмічні мови та основи програмування: мова С. Житомир: ЖДТУ, 2007. 328 с.
3. Ковалюк Т. В. Алгоритмізація та програмування: підручник з грифом МОН України. Львів: Магнолія-2016, 2013. 400 с.
4. Мартынов Н. Н. Программирование для Windows на C/C++ М.: Бином-Пресс, 2014. 528 с.
5. Нікітченко М. С. Теоретичні основи програмування. Київ: КНУ ім. Т. Г. Шевченка, 2019. 200 с.
6. Трофименко О. Г. С++. Теорія та практика. 587 с.
7. Документація Visual Studio [електронний ресурс]. Режим доступу: https://docs.microsoft.com/ru-ru/visualstudio/ide/?view=vs-2017
8. Введення мови програмування С і С++ [електронний ресурс]/режим доступу: http://www.intuit.ru/studies/courses/1039/231/info
9. Верифікація програмного забезпечення [електронний ресурс]/режим доступу: http://www.intuit.ru/studies/courses/1040/209/info
10. Інструменти, алгоритми та структури даних [електронний ресурс] / режим доступу: http://www.intuit.ru/studies/courses/683/539/info
11. MySQL. [Електронний ресурс] – Режим доступу: https://www.mysql.com/ (дата звернення 12.05.23р)
12. Г. Шілдт «Самовчитель С#» - Київ.: Освіта. 2017. – 670с.
13. А. Мєшков, Ю. Тихомиров "Visual C++ і MFC" - Харків.: Місто. 2018 – 1017с.
14. Культін Н. «C# в задачах та прикладах» - Одеса.: МКТ, 2012. - 288 с.
15. Петров Б., Алексєєв Т. "C#" - Запоріжжя, 2021. - 370 с.: іл. Брагін І. «Побудова мереж» – Київ.:Київ, 2020. – 480 с.: іл.
16. MySQL. [Електронний ресурс] – Режим доступу: https://www.mysql.com/ (дата звернення 10.05.23р)

17. Г. Шілдт «Самовчитель C#» - Київ.: Освіта. 2017. – 670с.

18. А. Мєшков, Ю. Тихомиров "Visual C++ і MFC" - Харків.: Місто. 2018 – 1017с.

19. Культін Н. «C# в задачах та прикладах» - Одеса.: МКТ, 2012. - 288 с.

20. Петров Б., Алексєєв Т. "C#" - Запоріжжя, 2021. - 370 с.: іл.

21. Брагін І. «Побудова мереж» – Київ.:Київ, 2020. – 480 с.: іл.

ДОДАТКИ

Додаток А

User.cs

```
namespace EstateManager.Models {
    public class User {
        public int Id { get; set; }
        public string Login { get; set; }
        public string Password { get; set; }
        public string FirstName { get; set; }
        public string LastName { get; set; }
        public int IsAdmin { get; set; }
        public User() { }
        public User(int id, string login, string password,
string firstName, string lastName, int isAdmin) {
            Id = id;
            Login = login;
            Password = password;
            FirstName = firstName;
            LastName = lastName;
            IsAdmin = isAdmin;
        }
    }
}
```

Singleton.cs

```
namespace EstateManager.Models {
    public static class Singleton {
        public static EstateContext db { get; private set; } =
new EstateContext();
        public static User user { get; set; }
    }
}
```

SellEstate.cs

```
using System.ComponentModel.DataAnnotations.Schema;

namespace EstateManager.Models {
    public class SellEstate {
        public int Id { get; set; }
        public int? IdEstate { get; set; }
        [ForeignKey("IdEstate")]
        public Estate Estate { get; set; }
        public int Price { get; set; }
        public SellEstate() { }
        public SellEstate(int id, Estate estate, int price) {
            Id = id;
            Estate = estate;
            Price = price;
        }
```

```
        }
    }
```

Rewiew.cs

```
using System.ComponentModel.DataAnnotations.Schema;

namespace EstateManager.Models {
    public class Review {
        public int Id { get; set; }
        public int? IdUser { get; set; }
        [ForeignKey("IdUser")]
        public User User { get; set; }
        public int? IdEstate { get; set; }
        [ForeignKey("IdEstate")]
        public Estate Estate {  get; set; }
        public string Text { get; set; }
        public int Rating { get; set; }
        public DateTime Date { get; set; }
        public Review() { }
        public Review(int id, User user, Estate estate, string
text, int rating, DateTime date) {
            Id = id;
            User = user;
            Estate = estate;
            Text = text;
            Rating = rating;
            Date = date;
        }
    }
}
```

Rent.cs

```
using System.ComponentModel.DataAnnotations.Schema;

namespace EstateManager.Models {
    public class Rent {
        public int Id { get; set; }
        public int? IdUser { get; set; }
        [ForeignKey("IdUser")]
        public User User { get; set; }
        public int? IdEstate { get; set; }
        [ForeignKey("IdEstate")]
        public Estate Estate { get; set; }
        public DateTime StartDate {  get; set; }
        public DateTime EndDate { get; set; }
        public Rent() { }
        public Rent(int id, User user, Estate estate, DateTime
startDate, DateTime endDate) {
            Id = id;
            User = user;
```

```
            Estate = estate;
            StartDate = startDate;
            EndDate = endDate;
        }
    }
}
```

LikedEstate.cs

```
using System.ComponentModel.DataAnnotations.Schema;

namespace EstateManager.Models {
    public class LikedEstate {
        public int Id { get; set; }
        public int? IdUser { get; set; }
        [ForeignKey("IdUser")]
        public User User { get; set; }
        public int? IdEstate { get; set; }
        [ForeignKey("IdEstate")]
        public Estate Estate { get; set; }
        public LikedEstate() { }
        public LikedEstate(int id, User user, Estate estate) {
            Id = id;
            User = user;
            Estate = estate;
        }
    }
}
```

EstateContext.cs

```
using Microsoft.EntityFrameworkCore;

namespace EstateManager.Models {
    public class EstateContext : DbContext {
        public DbSet<User> Users { get; set; }
        public DbSet<Estate> Estates { get; set; }
        public DbSet<BoughtEstate> BoughtEstates { get; set; }
        public DbSet<LikedEstate> LikedEstates { get; set; }
        public DbSet<Rent> Rents { get; set; }
        public DbSet<Review> Reviews { get; set; }
        public DbSet<SellEstate> SellEstates { get; set; }
        public      EstateContext(DbContextOptions<EstateContext>
options) : base(options) { }
        public EstateContext() { }
        protected                  override                  void
OnConfiguring(DbContextOptionsBuilder optionsBuilder)
        =>                      optionsBuilder.UseSqlServer("Data
Source=(localdb)\\MSSQLLocalDB;Initial     Catalog=Estates;Integrated
Security=True;Connect
Timeout=30;Encrypt=False;TrustServerCertificate=False;ApplicationInt
ent=ReadWrite;MultiSubnetFailover=False");
```

```
            protected override void OnModelCreating(ModelBuilder
modelBuilder) {
                modelBuilder.Entity<Estate>().Navigation(p         =>
p.User).AutoInclude();
                modelBuilder.Entity<BoughtEstate>().Navigation(p   =>
p.User).AutoInclude();
                modelBuilder.Entity<BoughtEstate>().Navigation(p   =>
p.Estate).AutoInclude();
                modelBuilder.Entity<LikedEstate>().Navigation(p    =>
p.User).AutoInclude();
                modelBuilder.Entity<LikedEstate>().Navigation(p    =>
p.Estate).AutoInclude();
                modelBuilder.Entity<Rent>().Navigation(p           =>
p.User).AutoInclude();
                modelBuilder.Entity<Rent>().Navigation(p           =>
p.Estate).AutoInclude();
                modelBuilder.Entity<Review>().Navigation(p         =>
p.User).AutoInclude();
                modelBuilder.Entity<Review>().Navigation(p         =>
p.Estate).AutoInclude();
                modelBuilder.Entity<SellEstate>().Navigation(p     =>
p.Estate).AutoInclude();
            }

        }
    }
```

Estate.cs

```
    using System.ComponentModel.DataAnnotations.Schema;

    namespace EstateManager.Models {
        public class Estate {
            public int Id { get; set; }
            public string Name { get; set; }
            public int? IdUser { get; set; }
            [ForeignKey("IdUser")]
            public User User { get; set; }
            public int RoomQuantity { get; set; }
            public int Square { get; set; }
            public int Price { get; set; }
            public string Address { get; set; }
            public int PriceToRent { get; set; }
            public byte[] Image { get; set; }
            public  Estate(int  id,  string  name,  User  user,  int
roomQuantity,  int  square,  int  price,  string  address,  int
priceToRent, byte[] image) {
                Id = id;
                Name = name;
                User = user;
                RoomQuantity = roomQuantity;
                Square = square;
```

```
                Price = price;
                Address = address;
                PriceToRent = priceToRent;
                Image = image;
            }
            public Estate() { }
        }
    }
```

ErrorViewModel.cs

```
    namespace EstateManager.Models {
        public class ErrorViewModel {
            public string? RequestId { get; set; }

            public          bool          ShowRequestId          =>
!string.IsNullOrEmpty(RequestId);
        }
    }
```

BoughtEstate.cs

```
    using System.ComponentModel.DataAnnotations.Schema;

    namespace EstateManager.Models {
        public class BoughtEstate {
            public int Id { get; set; }
            public int? IdUser { get; set; }
            [ForeignKey("IdUser")]
            public User User { get; set; }
            public int? IdEstate { get; set; }
            [ForeignKey("IdEstate")]
            public Estate Estate { get; set; }
            public int Price { get; set; }
            public DateTime Date {  get; set; }
            public BoughtEstate() { }
            public  BoughtEstate(int  id,  User  user,  Estate  estate,
int price, DateTime date) {
                Id = id;
                User = user;
                Estate = estate;
                Price = price;
                Date = date;
            }
        }
    }
```

Program.cs

```
    var builder = WebApplication.CreateBuilder(args);

    // Add services to the container.
    builder.Services.AddControllersWithViews();
```

```
    var app = builder.Build();

    // Configure the HTTP request pipeline.
    if (!app.Environment.IsDevelopment()) {
        app.UseExceptionHandler("/Home/Error");
        // The default HSTS value is 30 days. You may want to change
this for production scenarios, see https://aka.ms/aspnetcore-hsts.
        app.UseHsts();
    }

    app.UseHttpsRedirection();
    app.UseStaticFiles();

    app.UseRouting();

    app.UseAuthorization();

    app.MapControllerRoute(
        name: "default",
        pattern: "{controller=Home}/{action=Index}/{id?}");

    app.Run();
```

HomeController.cs

```
    using EstateManager.Models;
    using Microsoft.AspNetCore.Mvc;
    using System.Diagnostics;

    namespace EstateManager.Controllers {
        public class HomeController : Controller {
            static string message = "";

            public IActionResult Index() {
                if (Singleton.user == null) {
                    return View("Login");
                }
                return View("Index");
            }
            public IActionResult Login() {
                ViewData["Message"] = message;
                message = "";
                return View("Login");
            }
            public byte[] ConvertToBytes(IFormFile image) {
                byte[] imageBytes = null;
                BinaryReader          reader          =          new
BinaryReader(image.OpenReadStream());
                imageBytes = reader.ReadBytes((int)image.Length);
                return imageBytes;
            }
```

```
            public IActionResult OnLogin(string Login, string
Password) {
                try {
                    Singleton.user = Singleton.db.Users.Where(u =>
u.Login == Login && u.Password == Password).ToList()[0];
                }
                catch {
                    message = "Неправильний логін або пароль";
                    return RedirectToAction("Login");
                }
                return RedirectToAction("Index");
            }
            public IActionResult Register() {
                return View("Register");
            }
            public IActionResult OnRegister(string Login, string
Password, string Name, string Surname) {
                User u = new User(0, Login, Password, Name, Surname,
0);
                Singleton.db.Add(u);
                Singleton.db.SaveChanges();
                message = "Успішно доданого нового користувача";
                return RedirectToAction("Login");
            }
            public IActionResult AddEstate() {
                ViewData["Message"] = message;
                message = "";
                return View("AddEstate");
            }
            [HttpPost]
            public IActionResult OnAddEstate(string Name, string
RoomQuantity, string Square, string Price, string Address, string
PriceToRent, IFormFile Image) {
                try {
                    Estate e = new Estate(0, Name, Singleton.user,
Convert.ToInt32(RoomQuantity), Convert.ToInt32(Square),
                        Convert.ToInt32(Price),             Address,
Convert.ToInt32(PriceToRent), ConvertToBytes(Image));
                    Singleton.db.Add(e);
                    Singleton.db.SaveChanges();
                    List<Estate> l = Singleton.db.Estates.ToList();
                    e = l[l.Count - 1];
                    SellEstate   se   =   new   SellEstate(0,   e,
Convert.ToInt32(Price));
                    message = "Успішно додано!";
                }
                catch {
                    message = "Введено неправильні дані";
                }
                return RedirectToAction("AddEstate");
            }
            public IActionResult RemoveEstate() {
                ViewData["Message"] = message;
```

```
            message = "";
            ViewBag.Estates = Singleton.db.Estates.ToList();
            return View("RemoveEstate");
        }
        public IActionResult OnRemoveEstate(string combobox) {
            try {
                int id = Convert.ToInt32(combobox);
                Estate e =
Singleton.db.Estates.SingleOrDefault(e => e.Id == id);
                Singleton.db.Estates.Remove(e);
                Singleton.db.SaveChanges();
                message = "Успішно видалено!";
            }
            catch {
                message = "Введено неправильні дані";
            }
            return RedirectToAction("RemoveEstate");
        }
        public IActionResult EditEstate() {
            ViewData["Message"] = message;
            message = "";
            ViewBag.Estates = Singleton.db.Estates.ToList();
            return View("EditEstate");
        }
        public IActionResult OnEditEstate(string combobox,
string Name, string RoomQuantity, string Square, string Price,
string Address, string PriceToRent, IFormFile Image) {
            try {
                int id = Convert.ToInt32(combobox);
                Estate e =
Singleton.db.Estates.SingleOrDefault(e => e.Id == id);
                e.Name = Name;
                e.RoomQuantity = Convert.ToInt32(RoomQuantity);
                e.Square = Convert.ToInt32(Square);
                e.Price = Convert.ToInt32(Price);
                e.Address = Address;
                e.PriceToRent = Convert.ToInt32(PriceToRent);
                e.Image = ConvertToBytes(Image);
                Singleton.db.SaveChanges();
                message = "Успішно змінено!";
            }
            catch {
                message = "Введено неправильні дані";
            }
            return RedirectToAction("EditEstate");
        }
        public IActionResult OnExit() {
            Singleton.user = null;
            return RedirectToAction("Login");
        }
        public IActionResult OwnEstates() {
            ViewData["Message"] = message;
            message = "";
```

```
            ViewBag.Estates = Singleton.db.Estates.Where(e =>
e.IdUser == Singleton.user.Id).ToList();
            return View("OwnEstates");
        }
        public IActionResult RentEstate() {
            ViewData["Message"] = message;
            message = "";
            ViewBag.Estates = Singleton.db.Estates.ToList();
            return View("RentEstate");
        }
        public IActionResult OnRentEstate(string combobox,
DateTime StartDate, DateTime EndDate) {
            try {
                int id = Convert.ToInt32(combobox);
                Estate e =
Singleton.db.Estates.SingleOrDefault(e => e.Id == id);
                Rent r = new Rent(0, Singleton.user, e,
StartDate, EndDate);
                Singleton.db.Rents.Add(r);
                Singleton.db.SaveChanges();
                message = "Успішно арендовано!";
            }
            catch {
                message = "Введіть правильні дані";
            }
            return RedirectToAction("RentEstate");
        }
        public IActionResult AllEstates() {
            ViewData["Message"] = message;
            message = "";
            ViewBag.Estates = Singleton.db.Estates.ToList();
            return View("AllEstates");
        }
        public IActionResult AddReview() {
            ViewData["Message"] = message;
            message = "";
            ViewBag.Estates = Singleton.db.Estates.ToList();
            return View("AddReview");
        }
        public IActionResult OnAddReview(string combobox, string
Text, int Rating) {
            try {
                int id = Convert.ToInt32(combobox);
                Estate e =
Singleton.db.Estates.SingleOrDefault(e => e.Id == id);
                Review r = new Review(0, Singleton.user, e,
Text, Rating, DateTime.Now);
                Singleton.db.Reviews.Add(r);
                Singleton.db.SaveChanges();
                message = "Успішно додано!";
            }
            catch {
                message = "Введіть правильні дані";
```

```
            }
            return RedirectToAction("AddReview");
        }
        public IActionResult AllReviews() {
            ViewBag.Estates = Singleton.db.Estates.ToList();
            ViewBag.Users = Singleton.db.Users.ToList();
            ViewBag.Reviews = Singleton.db.Reviews.ToList();
            return View("AllReviews");
        }
        public IActionResult RemoveReview() {
            ViewData["Message"] = message;
            message = "";
            ViewBag.Estates = Singleton.db.Estates.ToList();
            ViewBag.Users = Singleton.db.Users.ToList();
            ViewBag.Reviews = Singleton.db.Reviews.ToList();
            return View("RemoveReview");
        }
        public IActionResult OnRemoveReview(string combobox) {
            try {
                int id = Convert.ToInt32(combobox);
                Review                    r                    =
Singleton.db.Reviews.SingleOrDefault(r => r.Id == id);
                Singleton.db.Reviews.Remove(r);
                Singleton.db.SaveChanges();
                message = "Успішно видалено!";
            }
            catch {
                message = "Введіть правильні дані";
            }
            return RedirectToAction("RemoveReview");
        }
        public IActionResult AddLikedEstate() {
            ViewData["Message"] = message;
            message = "";
            ViewBag.Estates = Singleton.db.Estates.ToList();
            return View("AddLikedEstate");
        }
        public IActionResult OnAddLikedEstate(string combobox) {
            try {
                int id = Convert.ToInt32(combobox);
                Estate                    e                    =
Singleton.db.Estates.SingleOrDefault(e => e.Id == id);
                LikedEstate    le    =    new    LikedEstate(0,
Singleton.user, e);
                Singleton.db.LikedEstates.Add(le);
                Singleton.db.SaveChanges();
                message = "Успішно додано!";
            }
            catch {
                message = "Введено неправильні дані";
            }
            return RedirectToAction("AddLikedEstate");
        }
```

```
            public IActionResult LikedEstates() {
                List<int?> l = Singleton.db.LikedEstates.Where(e =>
e.IdUser == Singleton.user.Id).Select(e => e.IdEstate).ToList();
                ViewBag.Estates = Singleton.db.Estates.Where(e =>
l.Contains(e.Id)).ToList();
                return View("LikedEstates");
            }
            public IActionResult BuyEstate() {
                ViewData["Message"] = message;
                message = "";
                List<Estate> el = new List<Estate>();
                foreach(SellEstate se in
Singleton.db.SellEstates.ToList()) {
                    el.Add(se.Estate);
                }
                ViewBag.Estates = el;
                return View("BuyEstate");
            }
            public IActionResult OnBuyEstate(string combobox) {
                try {
                    int id = Convert.ToInt32(combobox);
                    Estate e =
Singleton.db.Estates.SingleOrDefault(e => e.Id == id);
                    e.User = Singleton.user;
                    Singleton.db.SaveChanges();
                    SellEstate se =
Singleton.db.SellEstates.SingleOrDefault(s => s.IdEstate == e.Id);
                    Singleton.db.SellEstates.Remove(se);
                    Singleton.db.SaveChanges();
                    BoughtEstate be = new BoughtEstate(0,
Singleton.user, e, e.Price, DateTime.Now);
                    Singleton.db.BoughtEstates.Add(be);
                    Singleton.db.SaveChanges();
                    message = "Успішно куплено!";
                }
                catch {
                    message = "Щось пішло не так";
                }
                return RedirectToAction("BuyEstate");
            }
            public IActionResult BoughtEstates() {
                ViewBag.Estates = Singleton.db.BoughtEstates.Where(e
=> e.IdUser == Singleton.user.Id).ToList();
                return View("BoughtEstates");
            }
            public IActionResult SellEstate() {
                ViewData["Message"] = message;
                message = "";
                ViewBag.Estates = Singleton.db.Estates.Where(e =>
e.IdUser == Singleton.user.Id).ToList();
                return View("SellEstate");
            }
```

```
            public IActionResult OnSellEstate(string combobox,
string Price) {
                try {
                    int id = Convert.ToInt32(combobox);
                    Estate e =
Singleton.db.Estates.SingleOrDefault(e => e.Id == id);
                    e.Price = Convert.ToInt32(Price);
                    Singleton.db.SaveChanges();
                    SellEstate se = new SellEstate(0, e, e.Price);
                    Singleton.db.SellEstates.Add(se);
                    Singleton.db.SaveChanges();
                    message = "Успішно додано на продаж!";
                }
                catch {
                    message = "Введіть правильні дані";
                }
                return RedirectToAction("SellEstate");
            }
        }
    }
```

Site.scss

```
    body {
        background-image: linear-gradient(to top, #d299c2 0%,
#fef9d7 100%);
        margin-bottom: 60px;
    }

    .buttonwidth {
        max-width: 200px;
    }

    html {
        font-size: 14px;
    }

    @media (min-width: 768px) {
        html {
            font-size: 16px;
        }
    }

    .btn:focus, .btn:active:focus, .btn-link.nav-link:focus, .form-
control:focus, .form-check-input:focus {
        box-shadow: 0 0 0 0.1rem white, 0 0 0 0.25rem #258cfb;
    }

    html {
        position: relative;
        min-height: 100%;
    }

    * {
```

```
        box-sizing: border-box;
    }

    .decor {
        position: relative;
        max-width: 400px;
        margin: 50px auto 0;
        background: white;
        border-radius: 30px;
    }

    .form-left-decoration, .form-right-decoration {
        content: "";
        position: absolute;
        width: 50px;
        height: 20px;
        background: rgba(224,184,201);
        border-radius: 20px;
    }

    .form-left-decoration {
        bottom: 60px;
        left: -30px;
    }

    .form-right-decoration {
        top: 60px;
        right: -30px;
    }

    .form-left-decoration:before,        .form-left-decoration:after,
.form-right-decoration:before, .form-right-decoration:after {
        content: "";
        position: absolute;
        width: 50px;
        height: 20px;
        border-radius: 30px;
        background: white;
    }

    .form-left-decoration:before {
        top: -20px;
    }

    .form-left-decoration:after {
        top: 20px;
        left: 10px;
    }

    .form-right-decoration:before {
        top: -20px;
        right: 0;
    }
```

```
.form-right-decoration:after {
    top: 20px;
    right: 10px;
}

.circle {
    position: absolute;
    bottom: 80px;
    left: -55px;
    width: 20px;
    height: 20px;
    border-radius: 50%;
    background: white;
}

.form-inner {
    padding: 50px;
}

.form-inner input, .form-inner textarea {
    display: block;
    width: 100%;
    padding: 0 20px;
    margin-bottom: 10px;
    background: #E9EFF6;
    line-height: 40px;
    border-width: 0;
    border-radius: 20px;
    font-family: 'Roboto', sans-serif;
}

.form-inner input[type="submit"] {
    margin-top: 30px;
    background: #f69a73;
    border-bottom: 4px solid #d87d56;
    color: white;
    font-size: 14px;
}

.form-inner textarea {
    resize: none;
}

.form-inner h3 {
    margin-top: 0;
    font-family: 'Roboto', sans-serif;
    font-weight: 500;
    font-size: 24px;
    color: #707981;
}

.centertext {
```

```
        text-align: center;
    }

    .center {
        display: flex;
        flex-direction: column;
        justify-content: center;
        align-items: center;
    }

    a {
        color: darkseagreen;
        font-family: helvetica;
        text-decoration: none;
        text-transform: uppercase;
    }

    a:hover {
        text-decoration: underline;
        color: gray;
    }

    a:active {
        color: gray;
    }

    .top-menu {
        color: wheat;
        margin: 0 60px;
        position: relative;
        background: #5A394E;
        box-shadow: inset 1px 0 0 rgba(255,255,255,.1), inset -1px 0
0    rgba(255,255,255,.1),    inset    150px    0    150px    -150px
rgba(255,255,255,.12),    inset    -150px    0    150px    -150px
rgba(255,255,255,.12);
    }

    .top-menu:before,
    .top-menu:after {
        content: "";
        position: absolute;
        z-index: 2;
        left: 0;
        width: 100%;
        height: 3px;
    }

    .top-menu:before {
        top: 0;
        border-bottom: 1px dashed rgba(255,255,255,.2);
    }

    .top-menu:after {
```

```
    bottom: 0;
    border-top: 1px dashed rgba(255,255,255,.2);
}

.menu-main {
    list-style: none;
    padding: 0;
    margin: 0;
    text-align: center;
}

.menu-main:before,
.menu-main:after {
    content: "";
    position: absolute;
    width: 50px;
    height: 0;
    top: 8px;
    border-top: 18px solid #5A394E;
    border-bottom: 18px solid #5A394E;
    transform: rotate(360deg);
    z-index: -1;
}

.menu-main:before {
    left: -30px;
    border-left: 12px solid rgba(255, 255, 255, 0);
}

.menu-main:after {
    right: -30px;
    border-right: 12px solid rgba(255, 255, 255, 0);
}

.menu-main li {
    display: inline-block;
    margin-right: -4px;
}

.menu-main a {
    text-decoration: none;
    display: inline-block;
    padding: 15px 30px;
    font-family: 'PT Sans Caption', sans-serif;
    color: wheat;
    transition: .3s linear;
}

.menu-main a.current,
.menu-main a:hover {
    background: rgba(0,0,0,.2);
}
```

```
    @media (max-width: 680px) {
        .top-menu {
            margin: 0;
        }

        .menu-main li {
            display: block;
            margin-right: 0;
        }

        .menu-main:before,
        .menu-main:after {
            content: none;
        }

        .menu-main a {
            display: block;
        }
    }

    .text20 {
        font-size: 20px;
    }

    .text30 {
        font-size: 30px;
    }

    .text40 {
        font-size: 40px;
    }

    .text50 {
        font-size: 50px;
    }

    .width1000 {
        width: 1000px;
    }

    .width750 {
        width: 750px;
    }
```

_Layout.cshtml

```
    <!DOCTYPE html>
    <html lang="en">
    <head>
        <meta charset="utf-8" />
        <meta name="viewport" content="width=device-width, initial-
scale=1.0" />
        <title>@ViewData["Title"] - EstateManager</title>
```

```
        <link                                          rel="stylesheet"
href="~/lib/bootstrap/dist/css/bootstrap.min.css" />
        <link   rel="stylesheet"   href="~/css/site.css"   asp-append-
version="true" />
        <link    rel="stylesheet"    href="~/EstateManager.styles.css"
asp-append-version="true" />
    </head>
    <body>
        <header>
            @if(Singleton.user != null){
            @if (Singleton.user.IsAdmin == 0) {
                <nav class="top-menu">
                    <ul class="menu-main">
                        <li><a         asp-action="Index"         asp-
controller="Home"><b>Головна</b></a></li>
                        <li><a      asp-action="RentEstate"      asp-
controller="Home"><b>Арендувати нерухомість</b></a></li>
                        <li><a       asp-action="BuyEstate"       asp-
controller="Home"><b>Купити нерухомість</b></a></li>
                        <li><a    asp-action="BoughtEstates"    asp-
controller="Home"><b>Історія купівлі</b></a></li>
                        <li><a      asp-action="OwnEstates"      asp-
controller="Home"><b>Список володінь нерухомістю</b></a></li>
                        <li><a      asp-action="SellEstate"      asp-
controller="Home"><b>Продаж нерухомості</b></a></li>
                        <li><a      asp-action="AllEstates"      asp-
controller="Home"><b>Огляд нерухомості</b></a></li>
                        <li><a       asp-action="AddReview"       asp-
controller="Home"><b>Додати відгук</b></a></li>
                        <li><a     asp-action="RemoveReview"     asp-
controller="Home"><b>Видалити відгук</b></a></li>
                        <li><a      asp-action="AllReviews"      asp-
controller="Home"><b>Відгуки про нерухомість</b></a></li>
                        <li><a    asp-action="AddLikedEstate"    asp-
controller="Home"><b>Додати нерухомість в улюблені</b></a></li>
                        <li><a     asp-action="LikedEstates"     asp-
controller="Home"><b>Перегляд нерухомості в улюблених</b></a></li>
                    </ul>
                </nav>
            }
            else {
                <nav class="top-menu">
                    <ul class="menu-main">
                        <li><a         asp-action="Index"         asp-
controller="Home"><b>Головна</b></a></li>
                        <li><a       asp-action="AddEstate"       asp-
controller="Home"><b>Додати нерухомість</b></a></li>
                        <li><a      asp-action="EditEstate"      asp-
controller="Home"><b>Редагувати нерухомість</b></a></li>
                        <li><a     asp-action="RemoveEstate"     asp-
controller="Home"><b>Видалити нерухомость</b></a></li>
                    </ul>
                </nav>
```

```
            }
            }
        </header>
        <div class="container">
            <main role="main" class="pb-3">
                @RenderBody()
            </main>
        </div>

        <footer class="border-top footer text-muted">

        </footer>
        <script src="~/lib/jquery/dist/jquery.min.js"></script>
        <script
src="~/lib/bootstrap/dist/js/bootstrap.bundle.min.js"></script>
        <script            src="~/js/site.js"            asp-append-
version="true"></script>
        @await RenderSectionAsync("Scripts", required: false)
    </body>
    </html>
```

Login.html

```
    <form   class="decor   centertext"   asp-action="OnLogin"   asp-
controller="Home">
        <div class="form-inner">
            <h3>Авторизація</h3>
            <br />
            <input type="text" name="Login" placeholder="Логін">
            <input          type="password"          name="Password"
placeholder="Пароль">
            <input type="submit" value="Ввійти">
            <a               asp-action="Register"               asp-
controller="Home">Реєстрація</a>
        </div>
        <label class="text20">@ViewData["Message"]</label>
    </form>
```

AllEstates.cshtml

```
    <div class="center">
        <br />
        <label class="text30">Список нерухомості</label>
        <br />
        <table  class="table  table-dark  table-borderless  table-hover
centertext width1000">
            <thead class="thead">
                <tr>
                    <th>Назва</th>
                    <th>Кількість кімнат</th>
                    <th>Площа</th>
                    <th>Ціна</th>
                    <th>Адреса</th>
                    <th>Ціна аренди на день</th>
                    <th>Картинка</th>
```

```
                </tr>
            </thead>
            @{
                foreach (Estate e in ViewBag.Estates) {
                    string source = "data:image;base64," +
Convert.ToBase64String(e.Image);
                    <tr>
                        <td>@e.Name</td>
                        <td>@e.RoomQuantity</td>
                        <td>@e.Square</td>
                        <td>@e.Price</td>
                        <td>@e.Address</td>
                        <td>@e.PriceToRent</td>
                        <td><img height="60" src="@source" /></td>
                    </tr>
                }
            }
        </table>
        <label class="text20">@ViewData["Message"]</label>
    </div>
```

BuyEstate.cshtml

```
    <div class="center">
        <br />
        <label class="text20">@ViewData["Message"]</label>
        <br />
        <label class="text30">Купівля нерухомості</label>
        <br />
        <form class="form-inner centertext center" asp-
action="OnBuyEstate" asp-controller="Home" enctype="multipart/form-
data">
            <table class="table table-dark table-borderless table-
hover centertext width1000">
                <thead class="thead">
                    <tr>
                        <th>Номер</th>
                        <th>Назва</th>
                        <th>Кількість кімнат</th>
                        <th>Площа</th>
                        <th>Ціна</th>
                        <th>Адреса</th>
                        <th>Ціна аренди на день</th>
                        <th>Картинка</th>
                    </tr>
                </thead>
                @{
                    foreach (Estate e in ViewBag.Estates) {
                        string source = "data:image;base64," +
Convert.ToBase64String(e.Image);
                        <tr>
                            <td>@e.Id</td>
                            <td>@e.Name</td>
                            <td>@e.RoomQuantity</td>
```

```
                            <td>@e.Square</td>
                            <td>@e.Price</td>
                            <td>@e.Address</td>
                            <td>@e.PriceToRent</td>
                            <td><img    height="60"    src="@source"
/></td>
                        </tr>
                    }
                }
            </table>
            <label>Оберіть номер нерухомості для купівлі</label>
            <br />
            <select name="combobox" class="form-select buttonwidth">
                @foreach (Estate e in ViewBag.Estates) {
                    <option>@e.Id</option>
                }
            </select>
            <input    type="submit"    value="Купити    нерухомість"
class="buttonwidth" />
        </form>
    </div>
```

AddReview.cshtml

```
    <div class="center">
        <br />
        <label class="text20">@ViewData["Message"]</label>
        <br />
        <label    class="text30">Введіть    дані    для    додання
відгуку</label>
        <br />
        <table class="table table-dark table-borderless table-hover
centertext width1000">
            <thead class="thead">
                <tr>
                    <th>Номер</th>
                    <th>Назва</th>
                    <th>Кількість кімнат</th>
                    <th>Площа</th>
                    <th>Ціна</th>
                    <th>Адреса</th>
                    <th>Ціна аренди на день</th>
                    <th>Картинка</th>
                </tr>
            </thead>
            @{
                foreach (Estate e in ViewBag.Estates) {
                    string    source    =    "data:image;base64,"    +
Convert.ToBase64String(e.Image);
                    <tr>
                        <td>@e.Id</td>
                        <td>@e.Name</td>
                        <td>@e.RoomQuantity</td>
                        <td>@e.Square</td>
```

```
                        <td>@e.Price</td>
                        <td>@e.Address</td>
                        <td>@e.PriceToRent</td>
                        <td><img height="60" src="@source" /></td>
                    </tr>
                }
            }
        </table>
        <form    class="form-inner    centertext    center"    asp-
action="OnAddReview"  asp-controller="Home"  enctype="multipart/form-
data">
            <label>Оберіть номер нерухомості для відгуку</label>
            <br />
            <select name="combobox" class="form-select buttonwidth">
                @foreach (Estate e in ViewBag.Estates) {
                    <option>@e.Id</option>
                }
            </select>
            <label>Текст відгуку</label>
            <input type="text" name="Text" />
            <br />
            <label>Оцінка</label>
            <input  type="range"  class="form-range"  min="0"  max="10"
step="1" name="Rating"/>
            <br/>
            <input      type="submit"      value="Додати      відгук"
class="buttonwidth" />
        </form>
    </div>
```

AddEstate.cshtml

```
    <div class="center">
        <br />
        <label class="text20">@ViewData["Message"]</label>
        <br />
        <label    class="text30">Введіть    дані    для    додання
нерухомості</label>
        <br />
        <form class="form-inner centertext" asp-action="OnAddEstate"
asp-controller="Home" enctype="multipart/form-data">
            <label>Назва нерухомості</label>
            <input type="text" name="Name"/>
            <br/>
            <label>Кількість кімнат</label>
            <input type="text" name="RoomQuantity" />
            <br />
            <label>Площа</label>
            <input type="text" name="Square" />
            <br />
            <label>Ціна</label>
            <input type="text" name="Price" />
            <br />
            <label>Адреса</label>
```

```
            <input type="text" name="Address" />
            <br />
            <label>Ціна аренди</label>
            <input type="text" name="PriceToRent" />
            <br />
            <label>Картинка нерухомості</label>
            <input type="file" class="form-control" name="Image" />
            <br />
            <input type="submit" value="Додати нерухомість" />
        </form>
    </div>
```

AddLikedEstate.cshtml

```
    <div class="center">
        <br />
        <label class="text20">@ViewData["Message"]</label>
        <br />
        <label      class="text30">Додання      нерухомості      до
улюблених</label>
        <br />
        <form    class="form-inner    centertext    center"    asp-
action="OnAddLikedEstate"                        asp-controller="Home"
enctype="multipart/form-data">
            <table  class="table  table-dark  table-borderless  table-
hover centertext width1000">
                <thead class="thead">
                    <tr>
                        <th>Номер</th>
                        <th>Назва</th>
                        <th>Кількість кімнат</th>
                        <th>Площа</th>
                        <th>Ціна</th>
                        <th>Адреса</th>
                        <th>Ціна аренди на день</th>
                        <th>Картинка</th>
                    </tr>
                </thead>
                @{
                    foreach (Estate e in ViewBag.Estates) {
                        string   source   =   "data:image;base64,"   +
Convert.ToBase64String(e.Image);
                        <tr>
                            <td>@e.Id</td>
                            <td>@e.Name</td>
                            <td>@e.RoomQuantity</td>
                            <td>@e.Square</td>
                            <td>@e.Price</td>
                            <td>@e.Address</td>
                            <td>@e.PriceToRent</td>
                            <td><img    height="60"    src="@source"
/></td>
                        </tr>
                    }
                }
```

```
            </table>
            <label>Оберіть номер нерухомості для додання до
улюблених</label>
            <br />
            <select name="combobox" class="form-select buttonwidth">
                @foreach (Estate e in ViewBag.Estates) {
                    <option>@e.Id</option>
                }
            </select>
            <input    type="submit"    value="Додати    нерухомість"
class="buttonwidth" />
        </form>
    </div>
```

AllReviews.cshtml

```
    <div class="center">
        <br />
        <label class="text20">@ViewData["Message"]</label>
        <br />
        <br />
        <table class="table table-dark table-borderless table-hover
centertext width1000">
            <thead class="thead">
                <tr>
                    <th>Користувач</th>
                    <th>Текст відгуку</th>
                    <th>Рейтинг</th>
                    <th>Дата відгуку</th>
                    <th>Назва</th>
                    <th>Кількість кімнат</th>
                    <th>Площа</th>
                    <th>Ціна</th>
                    <th>Адреса</th>
                    <th>Ціна аренди на день</th>
                    <th>Картинка</th>
                </tr>
            </thead>
            @{
                foreach (Review r in ViewBag.Reviews) {
                    List<Estate> el = ViewBag.Estates;
                    List<User> ul = ViewBag.Users;
                    Estate  e  =  el.SingleOrDefault(e  =>  e.Id  ==
r.IdEstate);
                    User  u  =  ul.SingleOrDefault(e  =>  e.Id  ==
r.IdUser);
                    string   source   =   "data:image;base64,"   +
Convert.ToBase64String(e.Image);
                    <tr>
                        <td>@u.FirstName @u.LastName</td>
                        <td>@r.Text</td>
                        <td>@r.Rating</td>
                        <td>@r.Date</td>
                        <td>@e.Name</td>
```

```
                        <td>@e.RoomQuantity</td>
                        <td>@e.Square</td>
                        <td>@e.Price</td>
                        <td>@e.Address</td>
                        <td>@e.PriceToRent</td>
                        <td><img height="60" src="@source" /></td>
                    </tr>
                }
            }
        </table>
    </div>
```

BoughtEstates.cshtml

```
    <div class="center">
        <br />
        <label class="text20">@ViewData["Message"]</label>
        <br />
        <br />
        <table class="table table-dark table-borderless table-hover
centertext width1000">
            <thead class="thead">
                <tr>
                    <th>Користувач</th>
                    <th>Ціна</th>
                    <th>Дата купівлі</th>
                    <th>Назва</th>
                    <th>Кількість кімнат</th>
                    <th>Площа</th>
                    <th>Адреса</th>
                    <th>Картинка</th>
                </tr>
            </thead>
            @{
                foreach (BoughtEstate b in ViewBag.Estates) {
                    string source = "data:image;base64," +
Convert.ToBase64String(b.Estate.Image);
                    <tr>
                        <td>@b.User.FirstName @b.User.LastName</td>
                        <td>@b.Price</td>
                        <td>@b.Date</td>
                        <td>@b.Estate.Name</td>
                        <td>@b.Estate.RoomQuantity</td>
                        <td>@b.Estate.Square</td>
                        <td>@b.Estate.Address</td>
                        <td><img height="60" src="@source" /></td>
                    </tr>
                }
            }
        </table>
    </div>
```

EditEstate.cshtml

```
    <div class="center">
```

```
        <br />
        <label class="text20">@ViewData["Message"]</label>
        <br />
        <label     class="text30">Введіть     дані     для     зміни
нерухомості</label>
        <br />
        <table class="table table-dark table-borderless table-hover
centertext width1000">
            <thead class="thead">
                <tr>
                    <th>Номер</th>
                    <th>Назва</th>
                    <th>Кількість кімнат</th>
                    <th>Площа</th>
                    <th>Ціна</th>
                    <th>Адреса</th>
                    <th>Ціна аренди на день</th>
                    <th>Картинка</th>
                </tr>
            </thead>
            @{
                foreach (Estate e in ViewBag.Estates) {
                    string    source    =    "data:image;base64,"    +
Convert.ToBase64String(e.Image);
                    <tr>
                        <td>@e.Id</td>
                        <td>@e.Name</td>
                        <td>@e.RoomQuantity</td>
                        <td>@e.Square</td>
                        <td>@e.Price</td>
                        <td>@e.Address</td>
                        <td>@e.PriceToRent</td>
                        <td><img height="60" src="@source" /></td>
                    </tr>
                }
            }
        </table>
        <form    class="form-inner    centertext    center"    asp-
action="OnEditEstate" asp-controller="Home" enctype="multipart/form-
data">
            <label>Оберіть номер нерухомості для зміни</label>
            <br />
            <select name="combobox" class="form-select buttonwidth">
                @foreach (Estate e in ViewBag.Estates) {
                    <option>@e.Id</option>
                }
            </select>
            <label>Назва нерухомості</label>
            <input type="text" name="Name" />
            <br />
            <label>Кількість кімнат</label>
            <input type="text" name="RoomQuantity" />
            <br />
```

```
            <label>Площа</label>
            <input type="text" name="Square" />
            <br />
            <label>Ціна</label>
            <input type="text" name="Price" />
            <br />
            <label>Адреса</label>
            <input type="text" name="Address" />
            <br />
            <label>Ціна аренди</label>
            <input type="text" name="PriceToRent" />
            <br />
            <label>Картинка нерухомості</label>
            <input type="file" class="form-control" name="Image" />
            <br />
            <input    type="submit"    value="Змінити    нерухомість"
class="buttonwidth" />
        </form>
    </div>
```

Index.cshtml

```
    <div class="center centertext text20">
        <br/>
        <label>Вітаємо                        @Singleton.user.FirstName
@Singleton.user.LastName</label>
        <label>Останній                                          вхід:
@DateTime.Now.ToShortDateString()</label>
        <form class="form-inner centertext" asp-action="OnExit" asp-
controller="Home" enctype="multipart/form-data">
            <input  type="submit"  value="Вийти  з  облікового  запису"
/>
        </form>
    </div>
```

LikedEstates.cshtml

```
    <div class="center">
        <br />
        <label class="text30">Список улюбленої нерухомості</label>
        <br />
        <table  class="table  table-dark  table-borderless  table-hover
centertext width1000">
            <thead class="thead">
                <tr>
                    <th>Назва</th>
                    <th>Кількість кімнат</th>
                    <th>Площа</th>
                    <th>Ціна</th>
                    <th>Адреса</th>
                    <th>Ціна аренди на день</th>
                    <th>Картинка</th>
                </tr>
            </thead>
```

```
            @{
                foreach (Estate e in ViewBag.Estates) {
                    string source = "data:image;base64," +
Convert.ToBase64String(e.Image);
                    <tr>
                        <td>@e.Name</td>
                        <td>@e.RoomQuantity</td>
                        <td>@e.Square</td>
                        <td>@e.Price</td>
                        <td>@e.Address</td>
                        <td>@e.PriceToRent</td>
                        <td><img height="60" src="@source" /></td>
                    </tr>
                }
            }
        </table>
        <label class="text20">@ViewData["Message"]</label>
    </div>
```

Login.cshtml

```
    <form class="decor centertext" asp-action="OnLogin" asp-
controller="Home">
        <div class="form-inner">
            <h3>Авторизація</h3>
            <br />
            <input type="text" name="Login" placeholder="Логін">
            <input type="password" name="Password"
placeholder="Пароль">
            <input type="submit" value="Ввійти">
            <a asp-action="Register" asp-
controller="Home">Реєстрація</a>
        </div>
        <label class="text20">@ViewData["Message"]</label>
    </form>
```

OwnEstates.cshtml

```
    <div class="center">
        <br />
        <label class="text30">Список володінь нерухомістю</label>
        <br />
            <table class="table table-dark table-borderless table-
hover centertext width1000">
                <thead class="thead">
                    <tr>
                        <th>Назва</th>
                        <th>Кількість кімнат</th>
                        <th>Площа</th>
                        <th>Ціна</th>
                        <th>Адреса</th>
                        <th>Ціна аренди на день</th>
                        <th>Картинка</th>
                    </tr>
```

```
                </thead>
                @{
                    foreach (Estate e in ViewBag.Estates) {
                        string source = "data:image;base64," +
Convert.ToBase64String(e.Image);
                        <tr>
                            <td>@e.Name</td>
                            <td>@e.RoomQuantity</td>
                            <td>@e.Square</td>
                            <td>@e.Price</td>
                            <td>@e.Address</td>
                            <td>@e.PriceToRent</td>
                            <td><img height="60" src="@source"
/></td>
                        </tr>
                    }
                }
            </table>
        <label class="text20">@ViewData["Message"]</label>
    </div>
```

Register.cshtml

```
    <form class="decor centertext" asp-action="OnRegister" asp-
controller="Home">
        <div class="form-inner">
            <h3>Реєстрація</h3>
            <br />
            <input type="text" name="Login" placeholder="Логін">
            <input type="password" name="Password"
placeholder="Пароль">
            <input type="text" name="Name" placeholder="Ім'я">
            <input type="text" name="Surname"
placeholder="Прізвище">
            <input type="submit" value="Зареєструватись">
            <a asp-action="Index" asp-controller="Home">Ввійти</a>
        </div>
    </form>
```

RemoveEstate.cshtml

```
    <div class="center">
        <br />
        <label class="text20">@ViewData["Message"]</label>
        <br />
        <label class="text30">Видалення нерухомості</label>
        <br />
        <form class="form-inner centertext center" asp-
action="OnRemoveEstate" asp-controller="Home"
enctype="multipart/form-data">
            <table class="table table-dark table-borderless table-
hover centertext width1000">
                <thead class="thead">
                    <tr>
```

```
                        <th>Номер</th>
                        <th>Назва</th>
                        <th>Кількість кімнат</th>
                        <th>Площа</th>
                        <th>Ціна</th>
                        <th>Адреса</th>
                        <th>Ціна аренди на день</th>
                        <th>Картинка</th>
                    </tr>
                </thead>
                @{
                    foreach(Estate e in ViewBag.Estates) {
                        string source = "data:image;base64," +
Convert.ToBase64String(e.Image);
                        <tr>
                            <td>@e.Id</td>
                            <td>@e.Name</td>
                            <td>@e.RoomQuantity</td>
                            <td>@e.Square</td>
                            <td>@e.Price</td>
                            <td>@e.Address</td>
                            <td>@e.PriceToRent</td>
                            <td><img height="60" src="@source"
/></td>
                        </tr>
                    }
                }
            </table>
            <label>Оберіть номер нерухомості для видалення</label>
            <br/>
            <select name="combobox" class="form-select buttonwidth">
                @foreach(Estate e in ViewBag.Estates){
                    <option>@e.Id</option>
                }
            </select>
            <input type="submit" value="Видалити нерухомість"
class="buttonwidth" />
        </form>
    </div>
```

RentEstate.cshtml

```
    <div class="center">
        <br />
        <label class="text20">@ViewData["Message"]</label>
        <br />
        <label class="text30">Аренда нерухомості</label>
        <br />
        <table class="table table-dark table-borderless table-hover
centertext width1000">
            <thead class="thead">
                <tr>
                    <th>Номер</th>
                    <th>Назва</th>
```

```
                    <th>Кількість кімнат</th>
                    <th>Площа</th>
                    <th>Ціна</th>
                    <th>Адреса</th>
                    <th>Ціна аренди на день</th>
                    <th>Картинка</th>
                </tr>
            </thead>
            @{
                foreach (Estate e in ViewBag.Estates) {
                    string source = "data:image;base64," +
Convert.ToBase64String(e.Image);
                    <tr>
                        <td>@e.Id</td>
                        <td>@e.Name</td>
                        <td>@e.RoomQuantity</td>
                        <td>@e.Square</td>
                        <td>@e.Price</td>
                        <td>@e.Address</td>
                        <td>@e.PriceToRent</td>
                        <td><img height="60" src="@source" /></td>
                    </tr>
                }
            }
        </table>
        <form class="form-inner centertext center" asp-
action="OnRentEstate" asp-controller="Home" enctype="multipart/form-
data">

            <label>Оберіть номер нерухомості для аренди</label>
            <br />
            <select name="combobox" class="form-select buttonwidth">
                @foreach (Estate e in ViewBag.Estates) {
                    <option>@e.Id</option>
                }
            </select>
            <label>Дата початку аренди</label>
            <input type="date" class="form-control" name="StartDate"
/>
            <label>Дата кінця аренди</label>
            <input type="date" class="form-control" name="EndDate"
/>
            <input type="submit" value="Арендувати нерухомість"
class="buttonwidth" />
        </form>
    </div>
```

SellEstate.cshtml

```
    <div class="center">
        <br />
        <label class="text20">@ViewData["Message"]</label>
        <br />
        <label class="text30">Продаж нерухомості</label>
```

```
        <br />
        <table class="table table-dark table-borderless table-hover centertext width1000">
            <thead class="thead">
                <tr>
                    <th>Номер</th>
                    <th>Назва</th>
                    <th>Кількість кімнат</th>
                    <th>Площа</th>
                    <th>Ціна</th>
                    <th>Адреса</th>
                    <th>Ціна аренди на день</th>
                    <th>Картинка</th>
                </tr>
            </thead>
            @{
                foreach (Estate e in ViewBag.Estates) {
                    string source = "data:image;base64," + Convert.ToBase64String(e.Image);
                    <tr>
                        <td>@e.Id</td>
                        <td>@e.Name</td>
                        <td>@e.RoomQuantity</td>
                        <td>@e.Square</td>
                        <td>@e.Price</td>
                        <td>@e.Address</td>
                        <td>@e.PriceToRent</td>
                        <td><img height="60" src="@source" /></td>
                    </tr>
                }
            }
        </table>
        <form class="form-inner centertext center" asp-action="OnSellEstate" asp-controller="Home" enctype="multipart/form-data">

            <label>Оберіть номер нерухомості для продажу</label>
            <br />
            <select name="combobox" class="form-select buttonwidth">
                @foreach (Estate e in ViewBag.Estates) {
                    <option>@e.Id</option>
                }
            </select>
            <label>Ціна</label>
            <input type="text" name="Price"/>
            <input type="submit" value="Виставити на продаж" class="buttonwidth" />
        </form>
    </div>
```

Додаток Б

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Розробка програмного забезпечення для ринку нерухомості мовою C#

Виконав студент 4 курсу

Групи ПД-44
Сєрокуров Артем Ігорович
Керівник роботи

д.т.н., доц., зав. кафедри ТЦР Жебка Вікторія Вікторівна

Київ – 2023

МЕТА, ОБ’ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

- **Мета роботи -** підвищення ефективності пошуку нерухомості за допомогою розробленого програмного забезпечення мовою C#.
- **Об’єкт дослідження –** процес пошуку нерухомості.
- **Предмет дослідження –** програмне забезпечення пошуку нерухомості.

ЗАДАЧІ ДИПЛОМНОЇ РОБОТИ

1. Дослідити і проаналізувати існуюче програмне забезпечення у сфері нерухомості.
2. Розглянути актуальність програмного забезпечення для ринку нерухомості.
3. Розглянути ресурси для програмного забезпечення.
4. Розробити вимоги до програмного забезпечення на основі аналізу аналогів.
5. Розробити і створити програмне забезпечення для ринку нерухомості за допомогою мови програмування C#.
6. Провести тестування програмного забезпечення

3

АНАЛІЗ АНАЛОГІВ

Назва	Flatfy	LUN	Plektan	MyHome	Розроблений Додаток
Кросплатформність	+	+	+	-	+
Реєстрація	+	+	+	+	+
Історія нерухомості користувача	+	+	+	-	+
Система улюбленої нерухомості	+	+	-	-	+
Система відгуків до нерухомості	-	-	-	-	+
Можливість використання API	+	+	+	-	+

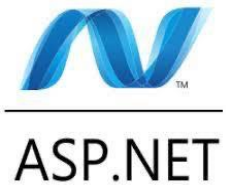
4

ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1. Вхід та головне меню користувача та адміністратора.
2. Можливість орендувати на деякий час або купити нерухомість.
3. Можливість виставляти нерухомість на продаж.
4. Видалення нерухомості з додатку.
5. Реалізація пошуку та сортування нерухомості.
6. Перегляд історії оренди, продажу, купівлі нерухомості у користувача.
7. Додавання та редагування відгуків до нерухомості.
8. Додавання та видалення улюбленої нерухомості у списку.

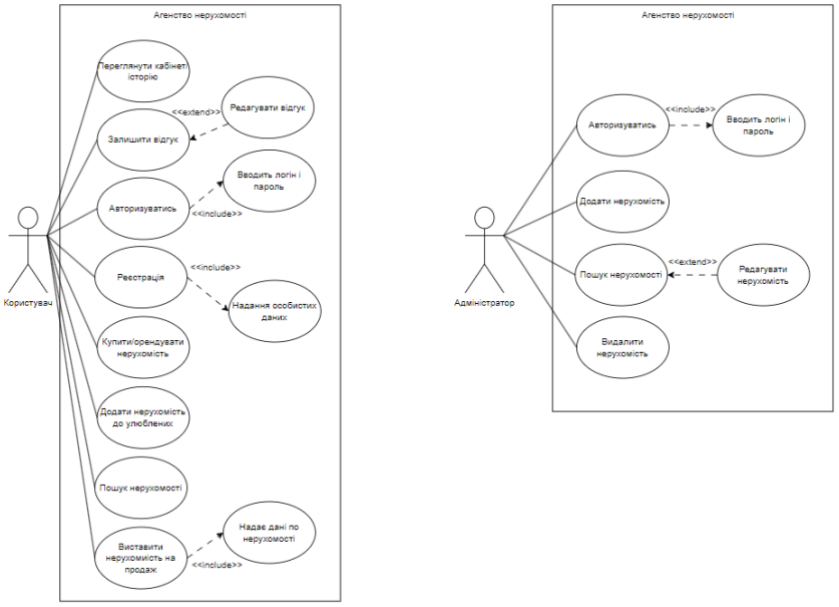
5

ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ



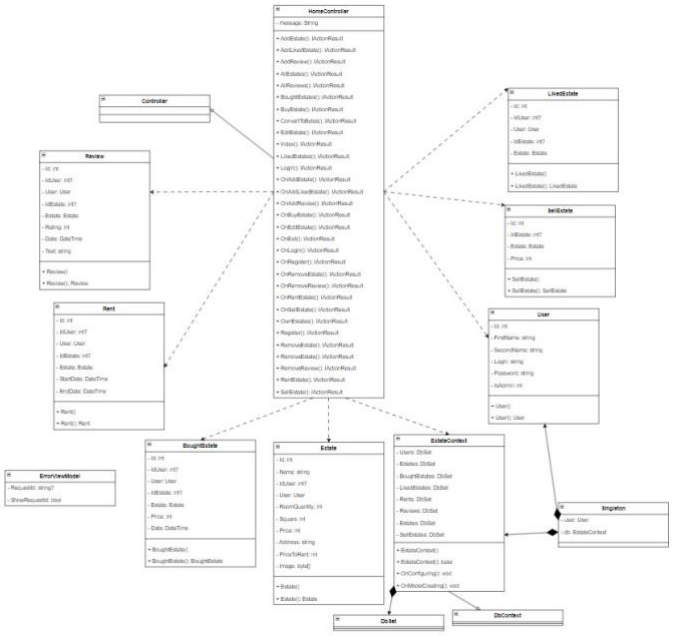
6

ДІАГРАМИ ВАРІАНТІВ ВИКОРИСТАННЯ



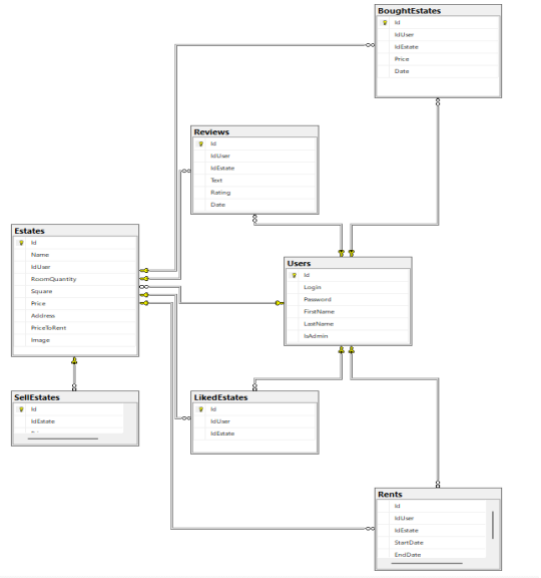
7

ДІАГРАМА КЛАСІВ



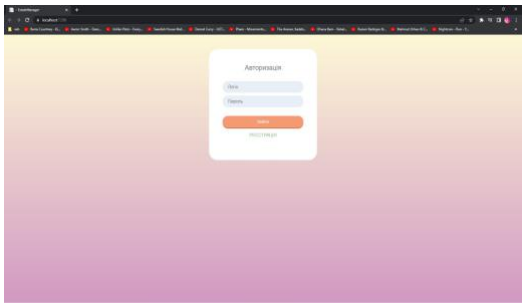
8

БАЗА ДАНИХ

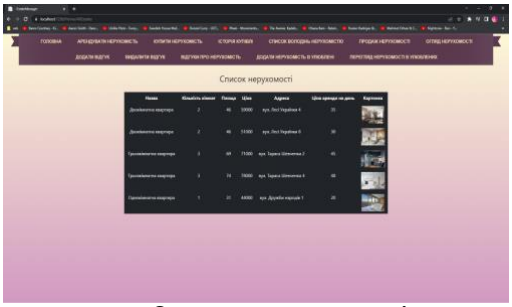


9

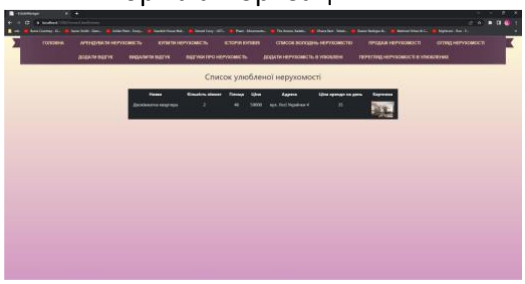
ЕКРАННІ ФОРМИ



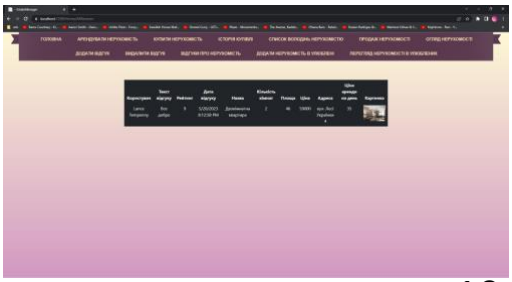
Форма авторизації



Список нерухомості

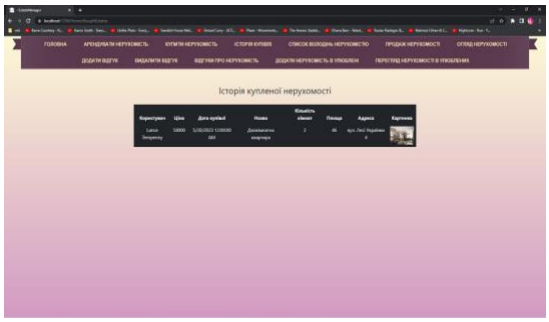


Список улюбленої нерухомості

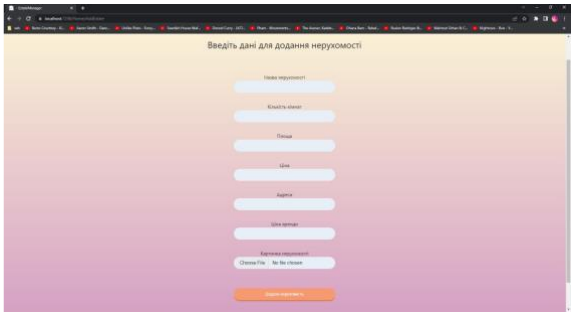


Відгуки

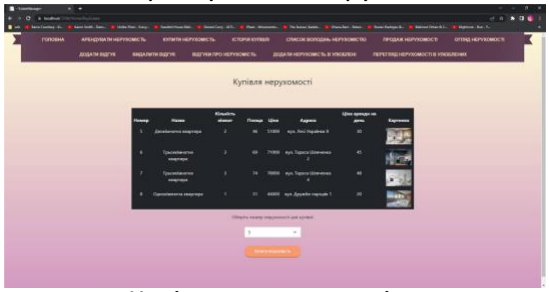
ЕКРАННІ ФОРМИ



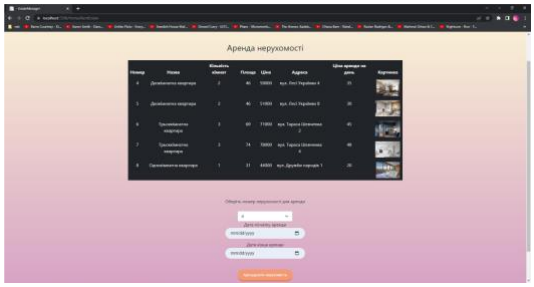
Історія купленої нерухомості



Додавання нерухомості



Купівля нерухомості



Аренда нерухомості

11

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

1. Сєрокуров А.І Розробка програмного забезпечення для ринку нерухомості мовою C# / Жебка В.В, Сєрокуров А.І // Сучасні аспекти діджиталізації та інформатизації в програмній та комп'ютерній інженерії, 01- 03 червня 2023р., ДУТ, м. Київ - К: ДУТ, 2023. Подано до друку.
2. Сєрокуров А.І Опис технології ASP.NET/ Жебка В.В, Сєрокуров А.І // Сучасні аспекти діджиталізації та інформатизації в програмній та комп'ютерній інженерії, 01- 03 червня 2023р., ДУТ, м. Київ - К: ДУТ, 2023. Подано до друку.

ВИСНОВКИ

1. Досліджено і проаналізовано існуючі додатки до ринку нерухомості, їх переваги та недоліки.
2. Проведено аналіз засобів розробки програмного забезпечення. В основі була взята мова C# та фреймворк ASP.NET.
3. Спроектовано архітектуру програмного забезпечення за допомогою патерну MVC.
4. Розроблено реляційну базу даних для програмного забезпечення за допомогою SQL.
5. Розроблено програмне забезпечення для продажу, оренди, купівлі нерухомості.
6. Реалізовано керування над нерухомістю, додавання відгуків та систему улюбленої нерухомості.
7. Проведено тестування програмного забезпечення.

ДЯКУЮ ЗА УВАГУ!