

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Інститут інформатики та радіoeлектроніки,
Факультет комп'ютерних наук і технологій

(повне найменування інституту, назва факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

магістра

(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ МЕТОДІВ
ПІДТРИМКИ ЗАМОВЛЕННЯ КОРОТКОСТРОКОВОЇ ОРЕНДИ ЖИТЛА
RESEARCH AND SOFTWARE IMPLEMENTATION OF METHODS FOR
SUPPORTING THE SHORT-TERM RENTAL ORDERS

Виконав: студент(ка) 2 курсу, групи КНТ-220м

Спеціальності 122 Комп'ютерні науки

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Системи штучного інтелекту

Гусєв А.О.

(прізвище та ініціали)

Керівник Скрупський С.Ю.

(прізвище та ініціали)

Рецензент Касьян М.М.

(прізвище та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування закладу вищої освіти)

Інститут, факультет ІРЕ, ФКНТ
Кафедра програмних засобів
Ступінь вищої освіти магістр
Спеціальність 122 Комп'ютерні науки
(код і найменування)

Освітня програма (спеціалізація) Системи штучного інтелекту
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
С.О. Субботін
“ ” 2021 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Гусєва Антона Олександровича
(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження та програмна реалізація методів підтримки замовлення короткострокової оренди житла
Research and Software Implementation of Methods for Supporting the Short-Term Rental Orders

керівник проєкту (роботи) Скрупський Степан Юрійович, к.т.н., доцент,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від “5” листопада 2021 року №435

2. Строк подання студентом проєкту (роботи) 13 грудня 2021 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз проблеми та постановка завдань дослідження. 2. Матеріали і методи. 3. Проєктування програмного забезпечення. 4. Основні рішення щодо реалізації компонентів системи. 5. Експлуатація, тестування та експериментальне дослідження програми.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-5 Основна частина	Скрупський С.Ю., доцент		
Нормоконтролер	Камінська Ж.К., асистент		

7. Дата видачі завдання “ 7 ” вересня 2021 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи	1 тиждень	Завдання, ТЗ
2	Аналіз та дослідження предметної області	2-5 тижні	Розділи 1, 2
3	Розробка архітектури програми	6 тиждень	Розділ 3
4	Розробка програми	7-9 тижні	Розділ 4
5	Тестування та експериментальне дослідження програми	10-11 тижні	Розділ 5
6	Оформлення пояснювальної записки та документів до неї. Нормоконтроль та рецензування	12 тиждень	Додатки
7	Захист роботи	13 тиждень	

Студент(ка)

(підпис) Гусєв А.О.
(прізвище та ініціали)

Керівник проєкту (роботи)

(підпис) Скрупський С.Ю.
(прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи магістра: 95 с., 24 рис., 4 табл., 3 дод., 21 джерело.

ОРЕНДА ЖИТЛА, КОРОТКОСТРОКОВА ОРЕНДА, ЦІНОВА ПРОПОЗИЦІЯ, ПРОГНОЗУВАННЯ ВАРТОСТІ, ДОВГА КОРОТКОСТРОКОВА ПАМ'ЯТЬ.

Об'єкт дослідження – процес замовлення короткострокової оренди житла. Предмет дослідження – методи підтримки замовлення короткострокової оренди житла.

Мета роботи – розробити програмне забезпечення замовлення короткострокової оренди житла для підтримки замовлень на основі результатів прогнозування вартості оренди житла в майбутньому періоді.

Матеріали, методи та технічні засоби: мови програмування C# і Python, система керування базами даних Microsoft SQL Server, штучні нейронні мережі, довга короткострокова пам'ять.

Результати. Розроблене програмне забезпечення дозволяє оформляти замовлення житла, прогнозувати вартість житла в майбутньому, враховувати користувачам при прийнятті власних рішень відгуки інших користувачів про їх минулий досвід оренди або господарів житла про їхніх гостей.

Висновки. Наукова новизна роботи полягає в запропонованому методі прогнозування вартості короткострокової оренди житла, який виконує прогноз на період з встановленим часовим інтервалом, використовуючи для цього штучну нейронну мережу на основі архітектури довгої короткострокової пам'яті, що дозволяє вибирати замовникам економічно обґрунтований момент для оформлення запланованого замовлення житла.

Галузь використання – короткострокова оренда житла (квартир, будинків, готельних номерів).

ABSTRACT

Explanatory note to the qualifying work of the master: 95 pages, 24 figures, 4 tables, 3 appendixes, 21 sources.

HOUSING RENTAL, SHORT-TERM RENTAL, PRICE OFFER, PRICE FORECASTING, LONG SHORT-TERM MEMORY.

The object of research is the process of ordering short-term housing rental. The subject of research is methods of supporting the ordering of short-term housing rental.

The purpose of the work is to develop software for short-term housing rental ordering to support orders based on the results of forecasting the cost of housing in the future.

Materials, methods and technical means: C# and Python programming languages, Microsoft SQL Server database management system, artificial neural networks, long short-term memory.

Results. The developed software enables to place an order for housing, predict the cost of housing in the future, take into account user experience based on their own feedback about their past rental experience or experience of homeowners concerning their guests.

Conclusions. The scientific novelty of the work lies in the proposed method of forecasting the cost of short-term rental, which performs forecasting for a period with a fixed time interval, using an artificial neural network based on long short-term memory architecture, which allows customers to choose economically reasonable time to order planned rental.

Field of use is short-term housing rental, including apartments, houses, hotel rooms.

ЗМІСТ

	С.
Перелік скорочень та умовних познач.....	8
Вступ.....	9
1 Аналіз проблеми та постановка завдань дослідження	11
1.1 Огляд програмних засобів підтримки замовлення короткострокової оренди житла.....	11
1.2 Визначення функціональних вимог до програмного забезпечення	16
1.3 Висновки за розділом 1	18
2 Матеріали і методи.....	20
2.1 Задача прогнозування вартості короткострокової оренди житла	20
2.2 Нейромережна архітектура Long Short-Term Memory	22
2.3 Метод прогнозування вартості короткострокової оренди житла	24
2.4 Висновки за розділом 2	27
3 Проєктування програмного забезпечення	28
3.1 Вибір мови програмування	28
3.2 Проєктування бази даних	30
3.3 Структура програмного забезпечення	37
3.4 Висновки за розділом 3	38
4 Основні рішення щодо реалізації компонентів системи.....	39
4.1 Функціональна схема програмної системи	39
4.2 Опис модулів реалізованої програмної системи.....	40
4.3 Висновки за розділом 4	49
5 Експлуатація, тестування та експериментальне дослідження програми	50
5.1 Призначення програми	50
5.2 Умови виконання програми	50
5.3 Виконання програми.....	50
5.4 Експериментальне дослідження проблеми прогнозування вартості короткострокової оренди житла.....	56
5.5 Висновки за розділом 5	57

Висновки	58
Перелік джерел посилання	60
Додаток А Технічне завдання	63
Додаток Б Текст програми	67
Додаток В Слайди презентації.....	87

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

LINQ – Language-INtegrated Query;

LSTM – long short-term memory;

БД – база даних.

ВСТУП

Розвиток ринкової економіки призводить до того, що з часом сфера послуг оренди житла починає динамічно збільшуватись. Оренда житла може бути довгостроковою і короткостроковою. І для одного, і для іншого типу оренди, в залежності від певних факторів, вартість оренди може змінюватись. Зрозуміло, що знання цієї інформації, особливо, якщо такі зміни є не дуже великими, часто не є критичним фактором прийняття рішень орендаря. Орендарю важливіше знати загальні тенденції на ринку, щоб планувати власні фінанси. Хоча якщо вартість саме того житла, що він орендує, зміниться значно, то це звичайно важлива інформація для орендаря.

У свою чергу вартість короткострокової оренди є часто як раз критичним або важливим фактором прийняття рішень такого орендаря. Це пов'язано з тим, що вартість такого типу оренди значно більше піддається тимчасовим коливаннями. А оскільки оренда реалізується на короткий строк, то в залежності від положення конкретної вартості оренди в певний період загальне коливання вартості за весь період може бути значним, особливо у відносному вираженні. Короткостроковою буде вважатися оренда, яка здійснюється на строк від 1 дня і найчастіше не перевищує 1 місяця.

Тому, зважаючи на можливість значного коливання, точна оцінка вартості житла на певний період є важливою оцінкою результатів прогнозування вартості оренди житла, а відповідно якісне прогнозування є дуже важливим фактором прийняття рішення про оренду. Відповідно важливо мати інструмент такого якісного прогнозування. Цей інструмент повинен включати:

- з одного боку програмне забезпечення, яке дозволить інтегрувати прогнозування безпосередньо в замовлення оренди житла, адже саме там здійснює свою діяльність особа, яка приймає рішення стосовно проблеми, що розглядається;

- з іншого боку адекватну методологічну основу прогнозування: відповідно виникає потреба у створенні методу прогнозування вартості

короткострокової оренди житла, який зможе забезпечити достатню точність отриманих результатів прогнозування.

Відповідно метою роботи було визначено розроблення програмного забезпечення замовлення короткострокової оренди житла для підтримки замовлень на основі результатів прогнозування вартості житла в майбутньому періоді.

1 АНАЛІЗ ПРОБЛЕМИ ТА ПОСТАНОВКА ЗАВДАНЬ ДОСЛІДЖЕННЯ

1.1 Огляд програмних засобів підтримки замовлення короткострокової оренди житла

На даний момент існує декілька відомих великих платформ короткострокової оренди житла, а також цілий ряд невеликих відповідних програмних засобів.

Найвідомішою платформою у даній сфері є Airbnb (рис. 1.1) [1]. Вона направлена на оренду не готельних номерів, а саме різного роду житла, куди зокрема можуть належати і готельні номери, але швидше це квартири, окремі кімнати, спільні кімнати, також це можуть бути окремі будинки, проживання в хостелі.

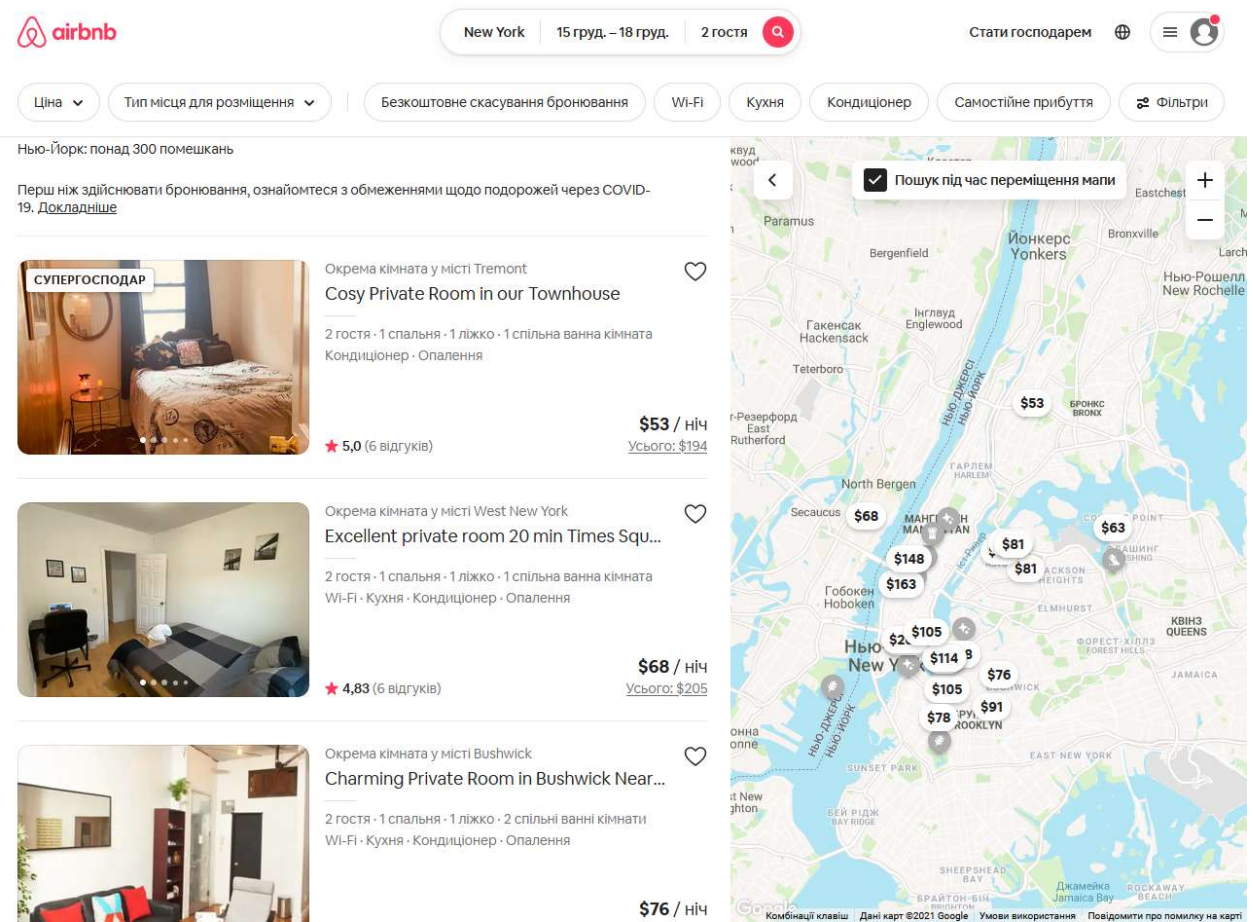


Рисунок 1.1 – Інтерфейс вебзастосунку Airbnb

Вебзастосунок Airbnb надає можливість будь-якому користувачу виконати пошук потрібного йому місця проживання, підтримуючи достатньо велику кількість критеріїв пошуку окрім безпосередньо дат поїздки, міста, куди відбувається поїздка, кількості гостей. До таких розширених параметрів належать наявність wi-fi, кондиціонера, парковки, можливість самостійного прибуття тощо.

Кожному користувачу надається можливість виставити власне житло на оренду. Тобто розділення користувачів на гостей і власників у застосунку немає. Задля виконання бронювання житла або виставлення власного житла в оренду потрібно звичайно зареєструватися і авторизуватися в програмі.

Авторизованому користувачу дозволяється писати повідомлення власнику житла, запитувати про бронювання, бронювати житло, керувати власними бронюваннями.

За кожним оголошенням можна визначити дати бронювання, кількість гостей, отримуючи інформацію при цьому про доступність житла в заданий період часу, а в результаті надається інформація про вартість бронювання житла в заданий період часу. При цьому вартість обраховується достатньо гнучко, тобто господар може визначити знижки за тривале або відносно тривале перебування.

Важливо відзначити, що окрім бронювання підтримується дуже важлива в даному процесі система відгуків [2]-[3]. Власники житла залишають відгуки про гостей, а гості в свою чергу про житло і його власників. Це дозволяє створювати оцінки для кожного оголошення, окрім того надавати перелік відгуків про житло або про гостя, що в свою чергу може використовуватися іншими господарями або гостями при створенні нових замовлень оренди житла. Загалом таким чином організована система довіри щодо даних, представлених на вебсайті. При цьому залишити відгук можна лише після того, як оренда завершилась, тобто зробити це раніше або зробити це, не проживаючи фактично, неможливо. Можливість залишення відгуку доступна протягом певного періоду часу, при цьому відгук другої

сторони стане відомим першій стороні тільки після того, як вона залишить власний відгук. Таким чином система відгуків достатньо зручно організована для отримання достовірної інформації.

Загалом платформа цілком і сам вебзастосунок зокрема є достатньо потужними засобами в даній галузі. Але при цьому в програмі відсутні засоби прогнозування. Тобто користувач може отримати інформацію тільки про те, скільки фактично буде коштувати його потенційне бронювання, але передбачити зміни його вартості не може. Єдина можливість у даному напрямку – інформування про те, коли зустрічається оголошення, житло за яким зазвичай повністю заброньоване, тобто це житло, яке не залишається в основному без оренди: така інформація з'являється у вигляді відповідного повідомлення.

Booking.com (рис. 1.2) є альтернативним застосунком, який може використовуватися в цій галузі [4]. Він визначає відповідну платформу, на якій взаємодіють власники житла з гостями. При цьому на даному вебсайті представлено більше готельних номерів ніж звичайних квартир, апартаментів, хоча вони також присутні. Доступна також можливість виставлення власного житла для оренди. Це відбувається через окремий партнерський вебсайт.

Застосунок надає можливість пошуку житла, при цьому задається в основних параметрах також кількість дітей. Також наявна широка система фільтрації результатів пошуку. Зокрема, враховуючи направленість на готелі, тут можна визначати кількість зірок готелю, можна також обмежувати значення оцінки, виставленої користувачами. Такі засоби відсутні в Airbnb. Також додатково у фільтрах присутні конкретні особливості для людей з фізичними вадами, для осіб з інвалідністю. При чому набір конкретних характеристик достатньо великий, тобто в цьому сенсі пошук також достатньо гнучкий. В цілому слід відзначити, що кількість критеріїв фільтрації результатів пошуку набагато більша ніж в Airbnb.

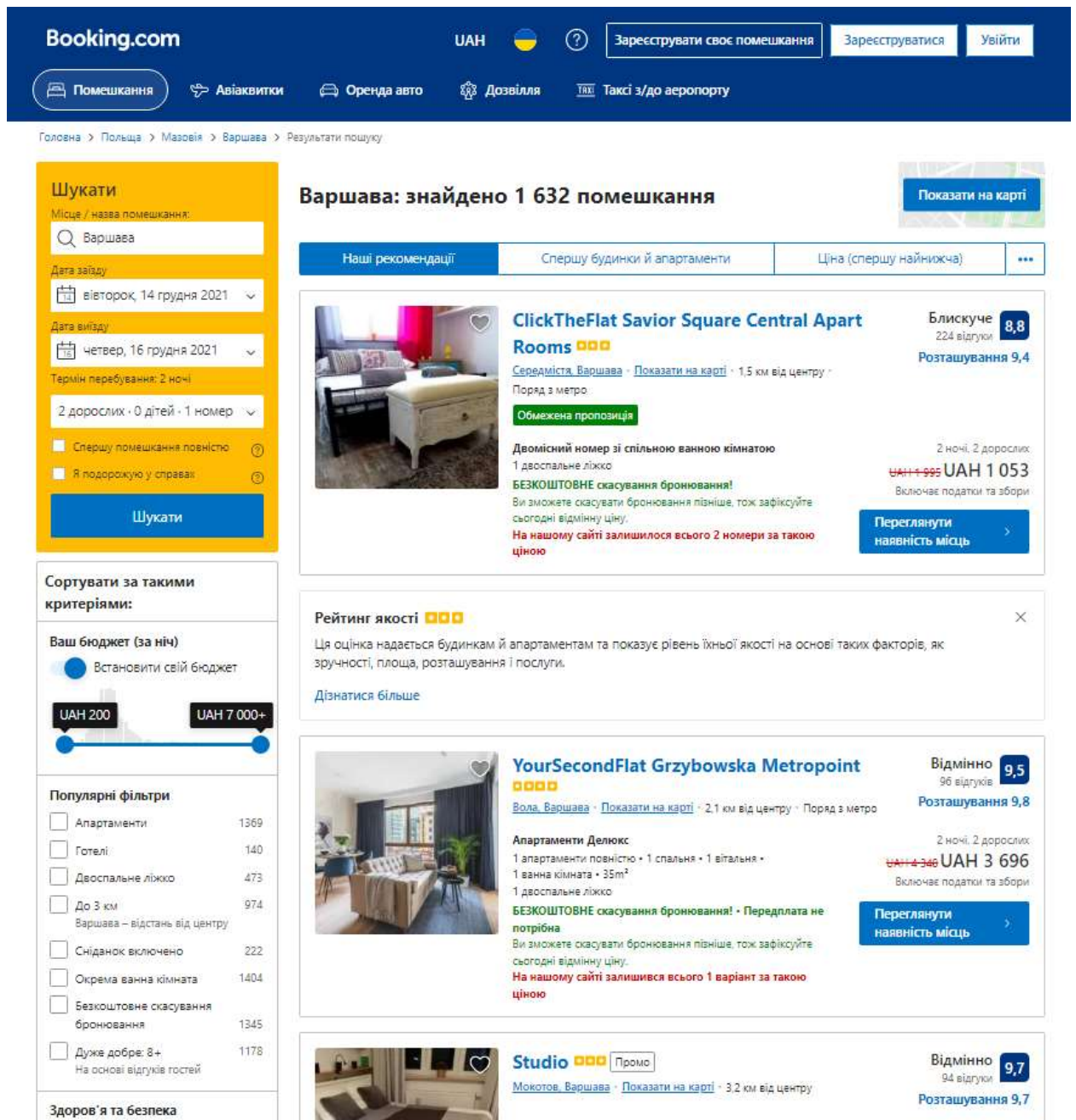


Рисунок 1.2 – Інтерфейс вебзастосунку Booking.com

При бронюванні користувач може залишити повідомлення, але це повідомлення є по суті коментарем щодо особливостей розміщення від гостя господарю. Якщо ж користувач має інші запитання, то він може переглянути перелік запитань і відповідей, щоб знайти на нього відповідь. Якщо відповідь не знайдено, то він може залишити таке запитання для господаря. Тобто система відправлення повідомлень і спілкування між користувачами неможлива на відміну від Airbnb.

Окрім того Booking надає можливість не тільки оренди житла, але і оренди автомобілів, покупки квитків на літаки, пошуку варіантів проведення дозвілля, замовлення таксі з аеропорту і до аеропорту, тобто це платформи, яка є ширшою вже на даний момент відносно платформи оренди житла. Airbnb також надає на даний момент засоби пошуку варіантів проведення дозвілля в містах.

Щодо даного застосунку також слід відзначити, що в ньому відсутні засоби прогнозування вартості житла, тобто можна дізнатися цю інформацію по факту вартості на конкретні вільні на даний момент дати. Ніяких історичних даних застосунок також не надає.

Результати порівняння проаналізованих вебзастосунків зведено до табл. 1.1.

Таблиця 1.1 – Порівняння програмних аналогів

Критерій	Airbnb	Booking.com	Розробка
Короткострокове бронювання	+	+	+
Відправлення повідомлень господарю	+	—	+
Відгуки про гостей	+	—	+
Прогнозування вартості оренди житла	—	—	+

Основною відмінністю розробки, яка виконується в роботі, є наявність засобів прогнозування вартості оренди житла, що не надається Airbnb. Така функція відсутня в Booking.com, як і функції відправлення повідомлень господарю, створення відгуків про гостей (доступні тільки відгуки про житло).

1.2 Визначення функціональних вимог до програмного забезпечення

Функціями програми (рис. 1.3) для забезпечення підтримки взаємодії між власниками житла та їхніми гостями є наступні:

- подання оголошення про оренду житла;
- редагування оголошення;
- визначення цінової пропозиції для оголошення на задані дати;
- перегляд оголошення про оренду житла;
- бронювання житла на задані дати;
- створення відгуку про житло;
- створення відгуку про гостя;
- перегляд відгуків про себе;
- перегляд власних відгуків;
- перегляд повідомлень з власником житла;
- відправлення повідомлення власнику житла;
- пошук житла в заданому місті на задані дати з заданими додатковими параметрами;
- обчислення вартості бронювання житла;
- перегляд власних оголошень;
- деактивація оголошення;
- перегляд бронювань житла власником;
- перегляд власних бронювань;
- скасування бронювання житла;
- обчислення сумарного рейтинга житла;
- обчислення сумарного рейтинга користувача;
- прогнозування вартості оренди житла на наступні місяці;
- обчислення середньої вартості оренди житла за місяць;
- змінювання профілю користувача;
- авторизація і реєстрація користувачів.

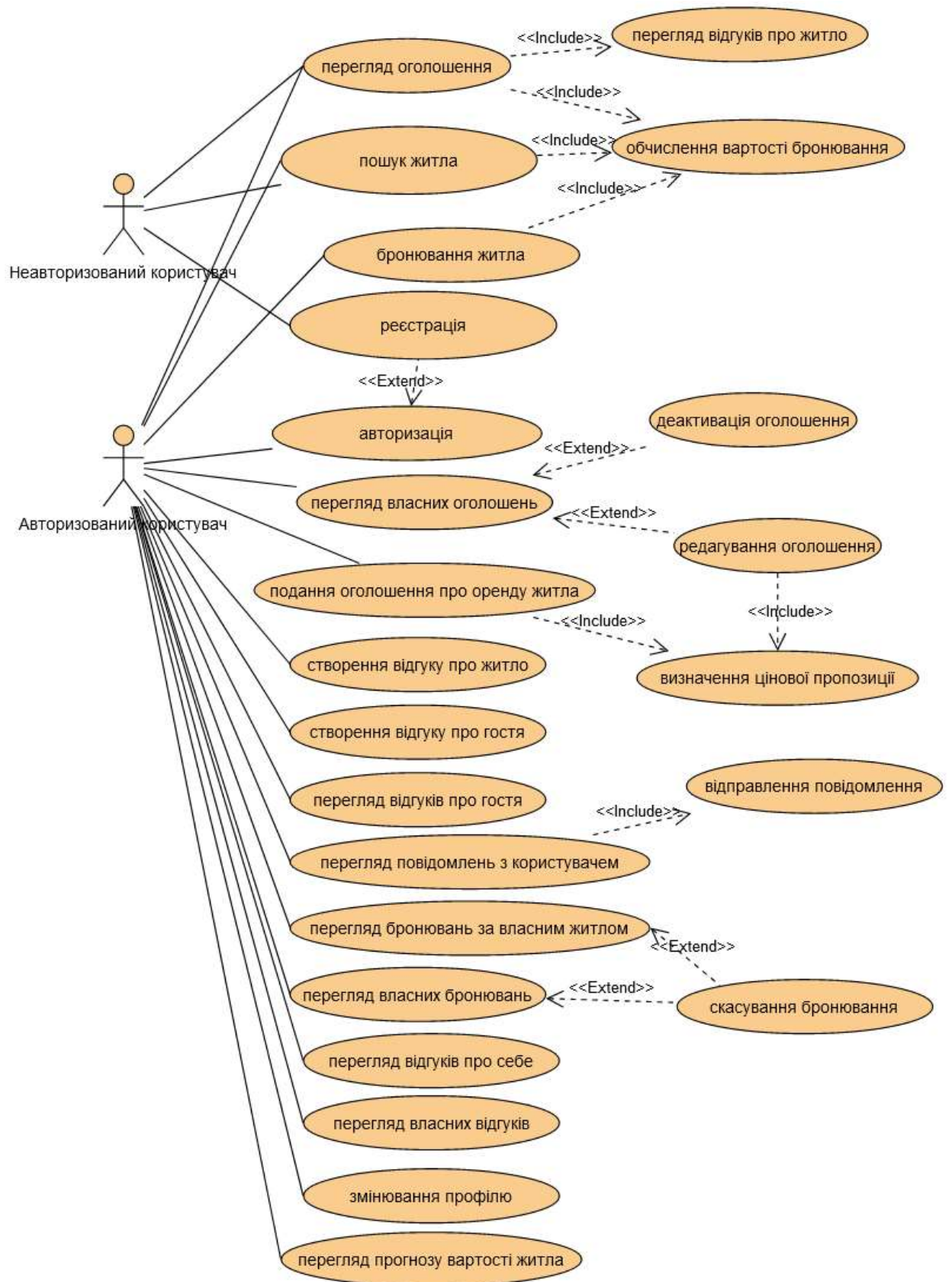


Рисунок 1.3 – Діаграма прецедентів

При організації потоків взаємодії користувачів з програмою було прийнято рішення не розділяти користувачів на групи. Тобто будь-який користувач повинен мати можливість і створити власне оголошення, стаючи таким чином власником житла, і забронювати житло для власного проживання, виступаючи таким чином гостем. Будь-який користувач може бути і власником, і гостем, при цьому для цього йому потрібен тільки один обліковий запис. Користувач застосунку може здавати власне житло, а в цей момент знаходитися в іншому місті одночасно і бронювати там житло у іншого користувача програми.

Але застосовуючи даний підхід, користувача можна класифікувати як авторизованого та неавторизованого.

Авторизований користувач може працювати з власними оголошеннями, створювати бронювання, керувати бронюваннями, неавторизований користувач може виконувати пошук і перегляд оголошень, тобто його сценарії використання програми обмежені.

1.3 Висновки за розділом 1

Завдання кваліфікаційної роботи:

- виконати аналіз програмних засобів підтримки замовлення короткострокової оренди житла;
- виконати проектування програмного забезпечення;
- реалізувати програмне забезпечення;
- створити новий метод прогнозування вартості короткострокової оренди житла;
- виконати експериментальне дослідження використання розробленого методу для розв'язання проблеми прогнозування вартості короткострокової оренди житла.

Виконаний огляд програмного забезпечення підтримки замовлення короткострокової оренди житла було завершено аналізом виявлених

програмних засобів і порівняння їх з розробкою, що виконується. Для виконуваної розробки визначено варіанти її використання неавторизованими і авторизованими користувачами після визначення функціональних вимог до програми.

2 МАТЕРІАЛИ І МЕТОДИ

2.1 Задача прогнозування вартості короткострокової оренди житла

Проблема прогнозування вартості короткострокової оренди житла є проблемою прогнозування часових рядів, коли за значенням у попередні інтервали часу, які формують таку послідовність значень, потрібно виконати прогнозування на майбутній проміжок часу.

Коли перед особою, яка бажає відправитися у мандрівку, виникає проблема планування цієї подорожі, то їй потрібно зрозуміти, в який саме момент варто виконати бронювання житла, адже на вартість оренди впливає багато різних факторів. Це, наприклад, і сезонність, і певні свята, але також і фактори, які визначають вартість за попередні періоди.

У момент, коли користувач заходить у відповідне програмне забезпечення, він зазвичай бачить актуальну вартість житла на найближчий момент, але якщо він хоче забронювати житло наперед, то він має розуміти, що поточна ціна на ці дати може змінитися. Зокрема і через те, що клієнти отримують досвід оренди, інформують про цей досвід інших клієнтів, залишаючи відгуки, а на основі збільшення попиту господар може підвищувати вартість або навпаки зменшувати її. Тому в даному випадку потрібно виконати прогнозування не на 1, 2 дні, коли такий прогноз дозволив би виявити спекулятивні коливання, або інші різкі види коливань, а виконати прогнозування на більший період. Наприклад, на місяць або декілька місяців.

Прогноз має виконуватися на основі вартості фактичних угод, укладених про оренду. Саме це є чітким показником того, як реальні люди в конкретних ситуаціях бронювали дане житло.

Якщо використовувати послідовність значень, то така послідовність дозволяє враховувати і сезонні коливання, і тенденції в зміні вартості на майбутнє на основі збільшення кількості позитивних відгуків, підвищення

рейтингу, адже кожне значення вартості в певний період вже виражає результуючий вплив таких факторів.

Тому зважаючи на описану логіку, задачу прогнозування вартості короткострокової оренди житла можна представити як задачу визначення залежності (2.1):

$$\langle V_{i+1}, \dots, V_{i+K} \rangle = f(V_i, \dots, V_{i-N}), \quad (2.1)$$

де V_i – середня вартість оренди житла на мінімальну одиницю часу за i -й період часу (періоди часу від i до $i - N$ належать до минулого, а періоди часу від $i + 1$ до $i + K$ належать до майбутнього);

f – функціональна залежність.

Оскільки це задача прогнозування вартості короткострокової оренди, то інтервал обирається мінімальним. У даному випадку буде розглядатися середнє значення вартості на добу (мінімальна одиниця) за період у місяць. Тобто фактично прогноз здійснюється на місяць вперед або на декілька місяців уперед.

Така постановка задачі визначає її як задачу прогнозування часового ряду. Існує набір моделей, які можуть використовуватися для розв’язання такої задачі:

- лінійні регресійні моделі [5];
- метод опорних векторів [6]-[7];
- нейронні мережі прямого поширення сигналу [8];
- рекурентні нейронні мережі, до яких належить архітектура long short-term memory (LSTM) [9].

З них найбільш перспективним є моделі на основі рекурентних нейронних мереж, зважаючи на активні дослідження в даній галузі, тому саме їх буде детально розглянуто в подальшій частині роботи [10]-[12].

2.2 Нейромережна архітектура Long Short-Term Memory

Рекурентна нейронна мережа на відміну від нейронних мереж прямого поширення сигналу зберігає пам'ять про те, що вона обробляла на своїх останніх попередніх кроках:

- реалізуються повторювані з'єднання, проходячи через петлі зворотного зв'язку: вихід попереднього кроку використовується як вхід для поточного кроку;

- приховані шари не тільки визначають вихідні дані, але й повертають їх як вхідні дані для навчання прихованого шару під час наступного спостереження: вони здійснюють навчання, регулюючи ваги синапсів у всій нейронній мережі, мережа перекалібровує вагові коефіцієнти як для поточних, так і для попередніх вхідних даних, множить вектор вхідних значень на вектор нових ваг (таким чином підвищуючи або знижуючи їх важливість по відношенню до мети навчання зниження помилки передбачення) і передає вектор результатів як вхід до наступного шару, адаптуючи вагові коефіцієнти, прихований шар поступово отримує функцію, яка перетворює вхідні значення у вихідні значення, які наближаються до фактичних спостережень у наборі навчальних даних [13].

LSTM або довга короткострокова пам'ять є розширеним алгоритмом рекурентних нейронних мереж, здатним зберігати довготривалу пам'ять і використовувати її для вивчення шаблонів у більш довгих послідовностях вихідних даних:

- рекурентні нейронні мережі могли зберегти пам'ять, але лише про етапи процесу в їхньому найближчому минулому, LSTM, навпаки, вводить цикли, які можуть генерувати тривалі градієнти, зберігаючи довгострокові закономірності;

- на кожному кроці LSTM може зв'язати три частини інформації: поточні вхідні дані, короткочасна пам'ять, яку мережа отримує від попередньої комірки (так званий прихований стан), і довгострокова пам'ять

від більш віддалених комірок (так званий стан чарунки), з якого чарунка мережі створює новий прихований стан;

– довготривалі градієнти вирішили проблему, яка називається зникаючим градієнтом, характерним для рекурентних нейронних мереж, коли модель припиняє навчання, оскільки нахил градієнта стає занадто малим для пошуку для подальшого покращення ваг, це трапляється, коли багато значень, які беруть участь у повторних обчисленнях градієнта, менші за 1, при цьому протилежна проблема, вибухаючий градієнт, виникає, коли багато значень перевищують 1 у повторюваних множеннях матриці, які виконує рекурентна нейронна мережа, тоді надмірно великі градієнти в кінцевому підсумку призведуть до нестабільності рекурентної нейронної мережі;

– проблема зникаючого градієнта обмежує пам'ять рекурентної нейронної мережі короткочасними залежностями, тоді як формулювання LSTM утримує градієнти досить крутими, щоб пошук не застряг у глухому куті: проблема зникаючого градієнта може виникнути, якщо моделі доведеться мати справу з довгими часовими рядами, що розтягуються на сотні періодів, кожен період повинен бути відображений мережевим рівнем, але глибокі мережі з численними шарами включають довгу послідовність множень матриці;

– чарунку в LSTM називають воротами: інформація вибірково додається або видаляється, коли проходить через ворота, відповідно чарунка діє як сито, яке визначає, скільки вхідної інформації захоплюється, а скільки зберігається;

– модель може вирішити, чи відкриває вона вхідні ворота для зберігання інформації, таким чином визначаючи, чи відхилити і видалити її з довготривалої пам'яті (ворота забуття), чи передає інформацію на наступний рівень (вихідний шлюз);

- затвори виконують матричні множення між значеннями інформації, які вони отримують як поточні вхідні дані, з короткочасної або довготривалої пам'яті;

- з часом LSTM дізнається, які фрагменти інформації ефективні для зменшення помилки прогнозування: модель відкриватиме та закриватиме ворота відповідним чином, призначаючи вищі або менші ваги від 0 до 1 інформаційним значенням, при цьому пропускаються корисні значення з більшою вагою через вихідні ворота для формування нової короткочасної пам'яті, але відкидаються значення з меншою вагою [10]-[13].

Під час обробки послідовних даних рекурентною нейронною мережею загалом або LSTM зокрема можна вибрати один з трьох типів побудови архітектур:

- архітектури багато-до-багатьох: у цьому випадку вхідні та цільові дані є послідовностями векторів, є два способи, як це можна реалізувати (кожен вектор послідовності дає відповідь у кожен момент часу t або вхідна послідовність спочатку крок за кроком обробляється рекурентним шаром, витягуючи щільне представлення вхідної послідовності, перш ніж наступний рекурентний шар починає генерувати вихідну послідовність);

- архітектура багато-до-одного: лише обробка останнього вектора ознак ініціює відповідь мережі, у такому випадку вхідними даними є послідовність векторів ознак, а вихідними є лише одне значення або вектор;

- архітектури один-до-багатьох: у цьому випадку вхідний сигнал є лише фіксованим тензором, а вихідний – послідовністю векторів, прикладом такого типу архітектури є підписи зображень [14].

2.3 Метод прогнозування вартості короткострокової оренди житла

У даному випадку у відповідності з залежністю (2.1) прогнозування загалом здійснюється на декілька місяців наперед. Це потрібно, оскільки дозволяє користувачу враховувати більший горизонт планування. У такому

випадку можна використати або архітектуру багато-до-багатьох або архітектуру багато-до-одного. Останню можна використати для прогнозування одного значення, але отримане в результаті значення прогнозу додати до нового часового ряду, формуючи вхідний екземпляр, зменшений на найстаріше значення з ряду та збільшений на прогнозоване значення. Відповідно це дозволить отримати прогнозоване значення на другий місяць. Такий варіант може використовуватися для зменшення складності моделі, за обмежених наборів даних. При цьому звичайно, що при варіанті прогнозування одного значення буде накопичуватися похибка, тому використовувати такий підхід для прогнозування на велику кількість періодів у майбутнє недоцільно.

Якщо використовувати архітектуру багато-до-багатьох, то у такому випадку потрібно буде більший обсяг даних для моделювання більш складної функціональної залежності. У даному випадку запропоновано використовувати архітектуру багато-до-багатьох, але за потреби можна використати спрощений варіант багато-до-одного.

На першому етапі роботи методу потрібно сформувати історичну вибірку даних, кожний елемент якої є часовим рядом, вхідними атрибутами якого є значення середньої добової вартості за $N + 1$ минулих місяців, а вихідними – значення середньої добової вартості за наступні K місяців.

Для послідовності вартості оренди житла за більшу ніж $N + K + 1$ періодів потрібно сформувати послідовність екземплярів вибірки даних, обираючи перші $N + K + 1$ значень, потім зсуваючи вікно на один місяць і обираючи наступні $N + K + 1$ періодів і т.д. до закінчення послідовності.

Для кожного елементу значення визначається як середня вартість за добу оренди житла. Для цього визначається сумарна вартість всіх замовлень в період, який охоплює всі дні місяця, а потім дане значення ділиться на кількість днів місяця, в які відбувалась оренда житла. Таким чином обчислюється значення кожного елементу вибірки даних.

На практиці в якості значення N використано 5 ($N + 1$ дорівнює 6). В якості K використано значення 2. Тобто прогнозування відбувається на наступні 2 місяці за 6 попередніми.

На другому етапі забезпечується оброблення сформованої вибірки даних.

Для цього виконується аналіз пропущених значень, що можливо у тому випадку, коли житло не здавалось тимчасово. Наприклад, таке може траплятися, якщо реалізується ремонт. У такому випадку забезпечується пропорційне визначення вартості оренди за пропущені періоди. Для цього шляхом віднімання від наступного відомого значення останнього відомого значення і діленням отриманого значення на кількість пропущених періодів обчислюється значення кроку вартості, яке додається шляхом акумуляції до попереднього періоду.

Окрім того реалізується нормалізація значень вибірки даних.

На третьому етапі потрібно створити модель нейронної мережі на основі LSTM, що складається з вхідного шару, двох прихованих LSTM-шарів, повнозв'язного та вихідного шару. На вихідному шарі розташовується K нейронів.

На четвертому етапі потрібно реалізувати навчання створеної LSTM-моделі на основі вибірки даних, яка було визначено як результат виконання першого етапу.

Для цього на вхід моделі подається $N + 1$ значень, а на виході отримується K значення в наступному періоді. Ці дані порівнюються в процесі навчання з відповідними значеннями з навчальної вибірки даних (вихідними значеннями для відповідних екземплярів).

На п'ятому етапі реалізується прогнозування для нових періодів, для яких значення за наступний період невідоме, за допомогою моделі, навченої на попередньому етапі. Тобто за наявними значеннями середньої добової вартості оренди житла за попередні $N + 1$ періодів часу (місяців) необхідно виконати прогнозування її вартості на наступні K періодів часу (місяців). Для

цього подається на вхід моделі множина значень середньодобової вартості оренди житла за минулі $N + 1$ періодів часу, а на виході визначається вектор, який містить значення вартості на наступні K періодів

Прогнозування фактично виконувалось подаванням даних 6 місяців і визначенням вартості за сьомий і восьмий місяці.

2.4 Висновки за розділом 2

Виконана постановка задачі прогнозування вартості короткострокової оренди житла є задачею прогнозування часового ряду, для якої було виконано математичну формалізацію. Виділено коло методів, які можуть використовуватися для розв'язання цієї задачі, серед яких детально розглянуто рекурентні нейронні мережі і архітектуру LSTM.

Наукова новизна роботи полягає в запропонованому методі прогнозування вартості короткострокової оренди житла, який виконує прогноз на період з встановленим часовим інтервалом, використовуючи для цього штучну нейронну мережу на основі архітектури довгої короткострокової пам'яті, що дозволяє вибирати замовникам економічно обґрунтований момент для оформлення запланованого замовлення житла.

3 ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Вибір мови програмування

Логічним рішенням, зважаючи на актуальний стан галузі розробки застосунків для спільного застосування щодо замовлення короткострокової оренди житла, є створення вебзастосунку, але в даному випадку це буде особливий вебзастосунок, оскільки він повинен реалізовувати метод прогнозування вартості житла, а його основною особливістю є необхідність реалізації машинного навчання, зокрема моделей LSTM. За результатами попереднього аналізу даних рішень було вирішено створити програмну систему, яка об'єднає в своєму складі вебзастосунок та реалізацію методу прогнозування. Саме таке рішення було прийнято, зважаючи на те, що прогнозування не буде виконуватися при створенні кожного замовлення або спробі такого замовлення. Прогнозування має виконуватися періодично при накопиченні відповідних характеристик вартості житла за наступний період, щоб виконати майбутній прогноз. Наприклад, при прогнозуванні вартості житла на місяць вперед достатньо виконувати прогноз раз на місяць. Але при тому, що оголошення може бути не доступне певний час, то його вартість за такий період може бути не визначена, відповідно прогноз потрібно виконати буде тоді, коли значення вартості актуалізується. Проте навіть у такому випадку для вчасного оновлення в найгіршому випадку достатньо буде виконувати прогноз кожен день. Відповідно даний модуль може запускатися за розкладом, взаємодіяти через базу даних (БД) з вебзастосунком і не впливати фактично на роботу один одного безпосередньо.

Тому першим завданням був вибір мови програмування для реалізації вебзастосунку (табл. 3.1). Було проаналізовано мови програмування PHP та C# з використанням фреймворку веброзробки ASP.NET [15]-[17]. Це достатньо популярні мови програмування для веброзробки, тому вибір є логічним.

Таблиця 3.1 – Порівняння мов програмування для веброзробки

Критерій	PHP	C# ASP.NET
Відкритість	+	+
Легкість пошуку помилок	—+	+—
Придатність до розширення (зокрема створення десктопних застосунків)	—	+
Розробка застосунків корпоративного рівня	+—	++
Розробка вебзастосунків	+	+
Більший фокус створюваних рішень	Безпека та функціональність	Взаємодія з користувачами, користувацький інтерфейс

Проведене порівняння було виконано, враховуючи критерії, які актуальні для проблеми розробки вебзастосунку в межах кваліфікаційної роботи магістра, враховуючи зокрема і особливості досвіду розробника. Оскільки мова C# разом з вебфреймворком ASP.NET дозволяє легше шукати помилки при виникненні проблем з написанням коду, дозволяє створювати застосунки корпоративного рівня та придатна до розширення (це логічно для рішення даного рівня, з часом знадобиться створювати застосунки під інші операційні системи), то саме дану мову було обрано для веброзробки.

Для реалізації машинного навчання загалом і запропонованого методу прогнозування вартості зокрема було порівняно мови програмування R і Python, оскільки вони є достатньо відомими стосовно цих цілей [18]-[19]. При цьому саме мова програмування Python дозволяє легше починати створювати нові рішення, має засоби інтеграції до вебзастосунків, хоча враховуючи рішення щодо взаємодії через БД, даний критерій не був критичним, а також направленість у сфері аналізу даних на розробку

застосунків, тому саме її було обрано для реалізації засобів машинного навчання.

Таблиця 3.2 – Порівняння мов програмування для машинного навчання

Критерій	R	Python
Відкритість	+	+
Основне застосування	Аналіз даних та статистика	Розробка на основі даних
Легкість початкового входження	—	+
Інтеграція з вебзастосунками	—	+

Питання вибору системи керування БД було вирішено, враховуючи попередній вибір. Оскільки для вебзастосунку використано саме мову програмування C#, то рідним рішенням у даному випадку є використання Microsoft SQL Server, таке рішення буде більш оптимальним щодо швидкості, безпеки і зручності роботи, що важливо для рішень такого рівня.

3.2 Проєктування бази даних

Проєктування БД виконано на основі створення відповідної структури реляційної БД.

Таблиця Advertisement представляє конкретне оголошення про оренду житла, зберігаючи наступні його характеристики:

- AdvertisementId – ідентифікатор оголошення про оренду житла (первинний ключ);
- Name – назва оголошення;
- Info – повний опис житла;

- Floor – поверх, на якому розташовано житло;
- Floors – кількість поверхів житла;
- RentalType – тип оренди, який надається (будинок, квартира, спільна кімната, окрема кімната, готель);
- Guests – максимальна кількість гостей, яку може вмістити житло;
- Rooms – кількість кімнат;
- Beds – кількість ліжок;
- BathroomType – тип ванної кімнати (ванна, душ);
- Photos – кількість фотографій, що представляє житло, у папці з фотографіями;
- Bedrooms – кількість спалень;
- Score – середня поточна оцінка житла;
- CityId – зовнішній ключ з таблиці City, що визначає місто, в якому знаходиться житло;
- CheckinStart – час, починаючи з якого можна заселитися;
- CheckinEnd – час, до якого можна заселитися;
- Checkout – час, до якого потрібно виселитися.

Таблиця City представляє міста, оренда житла в яких відбувається, зберігаючи наступні його характеристики:

- CityId – ідентифікатор міста (первинний ключ);
- Name – назва міста.

Таблиця AdvertisementReview представляє відгук про оголошення, залишений користувачем. Дана сутність необхідна для накопичення інформації про оголошення на основі принципів зворотного зв'язку. Таблиця містить наступні характеристики:

- AdvertisementReviewId – ідентифікатор відгуку про житло (первинний ключ);
- AdvertisementId – зовнішній ключ з таблиці Advertisement, що визначає оголошення, до якого було залишено відгук;

– UserId – зовнішній ключ з таблиці User, що визначає користувача, який залишив відгук;

– Review – текст відгуку;

– RDate – дата, час, коли було залишено відгук;

– Score – оцінка відгуку.

Таблиця Message представляє повідомлення від гостя або потенційного гостя господарю житла, зберігаючи наступні характеристики повідомлення:

– MessageId – ідентифікатор повідомлення (первинний ключ);

– AuthorId – зовнішній ключ з таблиці User, що визначає автора повідомлення;

– UserId – зовнішній ключ з таблиці User, що визначає користувача, якому адресовано повідомлення;

– Mess – текст повідомлення;

– MDate – дата, час, коли було опубліковано повідомлення.

Таблиця User представляє обліковий запис користувача, при чому користувач може виступати і в ролі господаря, і в ролі гостя. Таблиця зберігає наступні характеристики облікового запису:

– UserId – ідентифікатор користувача (первинний ключ);

– Name – ім'я користувача;

– Phone – номер телефону користувача;

– Info – інформація про користувача;

– Photo – фотографія користувача;

– Languages – список мов, які знає користувач;

– Score – середня оцінка користувача;

– State – стан облікового запису (заборонено доступ або дозволено);

– Login – логін облікового запису;

– Password – пароль облікового запису.

Таблиця AdvertisementUser забезпечує зв'язок багато до багатьох між таблицями User і Advertisement, адже кожен користувач може виставляти

декілька оголошень, а з кожним оголошенням може бути пов'язано декілька користувачів: господар та його помічники. Таблиця зберігає наступні характеристики:

- AdvertisementUserId – ідентифікатор (первинний ключ);
- UserId – зовнішній ключ з таблиці User, що визначає користувача, який працює з оголошенням;
- AdvertisementId – зовнішній ключ з таблиці Advertisement, що визначає оголошення;
- PresenceType – тип запису (господар, помічник або без доступу).

Таблиця Order представляє замовлення на оренду житла, зберігаючи наступні характеристики замовлення:

- OrderId – ідентифікатор замовлення (первинний ключ);
- UserId – зовнішній ключ з таблиці User, що визначає користувача, який створив замовлення;
- AdvertisementId – зовнішній ключ з таблиці Advertisement, що визначає оголошення;
- ODate – дата, час, коли було створено замовлення;
- StartDate – дата, з якої орендовано житло;
- EndDate – дата, до якої орендовано житло;
- Price – вартість замовлення.

Для узагальнення характеристик житла було виділено окрему сутність Offer (рис. 3.1). Вона необхідна для визначення зручності, правил даного житла і дозволяє узагальнити набір характеристик загалом, з яких потім можна обирати відповідні для житла, при цьому не потрібно їх видумувати при створенні оголошення, а до того ж цей набір може бути в будь-який момент розширений на відміну від створення відповідних полів у складі таблиці Asvertisement.

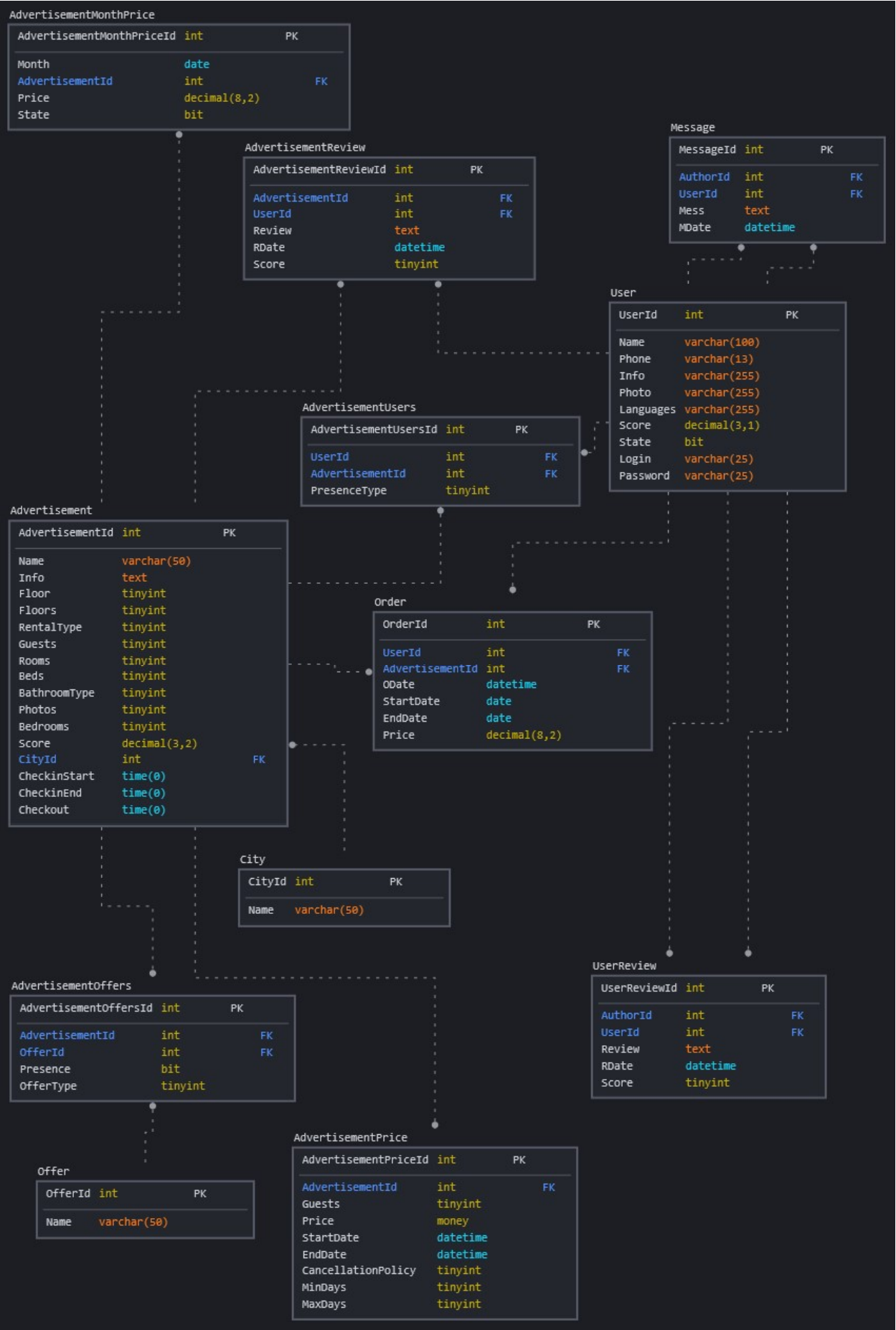


Рисунок 3.1 – Схема БД

Поля таблиці:

- OfferId – ідентифікатор характеристики оголошення (первинний ключ);
- Name – текст характеристики.

Таблиця UserReview представляє також як і таблиця AdvertisementReview відгук, але відгук не на житло, а на користувача, тобто гостя зі сторони господаря. Використання одразу двох типів відгуків дозволяє клієнтам розуміти фактичний рівень житла, а господарям отримувати інформацію про своїх потенційних гостей. Таблиця зберігає наступні характеристики відгуку про користувача:

- UserReviewId – ідентифікатор відгуку про користувача (первинний ключ);
- AuthorId – зовнішній ключ з таблиці User, що визначає користувача, який створив відгук;
- UserId – зовнішній ключ з таблиці User, що визначає користувача, про якого створено відгук;
- Review – текст відгуку;
- RDate – дата, час, коли було створено відгук;
- Score – оцінка відгуку.

Таблиця AdvertisementOffer забезпечує зв'язок багато до багатьох між таблицями Advertisement і Offer, адже кожне житло може мати цілий набір характеристик, а кожна характеристика у свою чергу може характеризувати одразу декілька оголошень. Таблиця зберігає наступні характеристики:

- AdvertisementOfferId – ідентифікатор (первинний ключ);
- AdvertisementId – зовнішній ключ з таблиці Advertisement, що визначає оголошення;
- OfferId – зовнішній ключ з таблиці Offer, що визначає характеристику оголошення;
- Presence – присутність або відсутність даної характеристики;
- OfferType – тип характеристики.

Кожне житло може мати різну вартість у різні періоди часу. Тому було введено таблицю AdvertisementPrice для збереження даних цінових пропозицій щодо оренди житла. Поля таблиці:

- AdvertisementPriceId – ідентифікатор цінової пропозиції (первинний ключ);
- AdvertisementId – зовнішній ключ з таблиці Advertisement, що визначає оголошення;
- Guests – кількість гостей;
- Price – вартість за добу оренди житла;
- StartDate – дата початку дії пропозиції;
- EndDate – дата завершення дії пропозиції;
- CancellationPolicy – політика скасування оренди;
- MinDays – мінімальна кількість діб, на яку можна орендувати житло;
- MaxDays – максимальна кількість діб, на яку можна орендувати житло.

Таблиця БД AdvertisementMonthPrice є окремою спеціалізованою таблицею, яка необхідна для внесення результатів роботи модуля, написаного мовою програмування Python, після чого вебзастосунок тільки запитує відповідні дані з даної таблиці БД, ніяк не змінюючи і не наповнюючи дану таблицю. Дана таблиця містить дані про середню вартість конкретного житла (оголошення) на добу протягом місяця, що може представляти історичні дані, а може – прогнозований результат. Поля даної таблиці:

- AdvertisementMonthPriceId – ідентифікатор (первинний ключ);
- AdvertisementId – зовнішній ключ з таблиці Advertisement, що визначає номер оголошення, до якого належить дане значення середньої вартості;
- Price – середня вартість на добу протягом місяця;
- State – стан запису (прогнозоване або історичне значення);

– Month – дата, що представляє місяць, за яким визначено дане значення вартості оренди.

3.3 Структура програмного забезпечення

До структури програмного забезпечення повинні входити наступні одиниці:

– тека Models, куди входять моделі вебзастосунку, розроблені у відповідності зі структурою БД;

– тека Controllers, куди входять контролери вебзастосунку, що охоплюють основні процеси взаємодії користувача з даними системи на основі використання відповідних функцій (для зручності були об'єднані в групи);

– тека Tools, куди входять базові засоби оброблення, що включають роботу з контекстом даних, де дані об'єднуються у відповідні переліки, оброблення дат тощо;

– тека Views, куди входять представлення вебзастосунку, які можна об'єднати до груп вебсторінок оголошень, бронювань, повідомлень, відгуків та профілю;

– тека Method, куди входять файли, написані мовою Python: model.py, який забезпечує створення, навчання і збереження моделі, processor.py, який забезпечує оброблення даних на основі описаної логіки запропонованого методу, predictor.py, що безпосередньо виконує прогнозування на основі навченої моделі зі збереженням результатів до БД;

– тека wwwroot, куди входять основні ресурси проєкту, включаючи файли зображень, стилів, бібліотек.

Розроблена структура програмного забезпечення приведена на рис. 3.2.

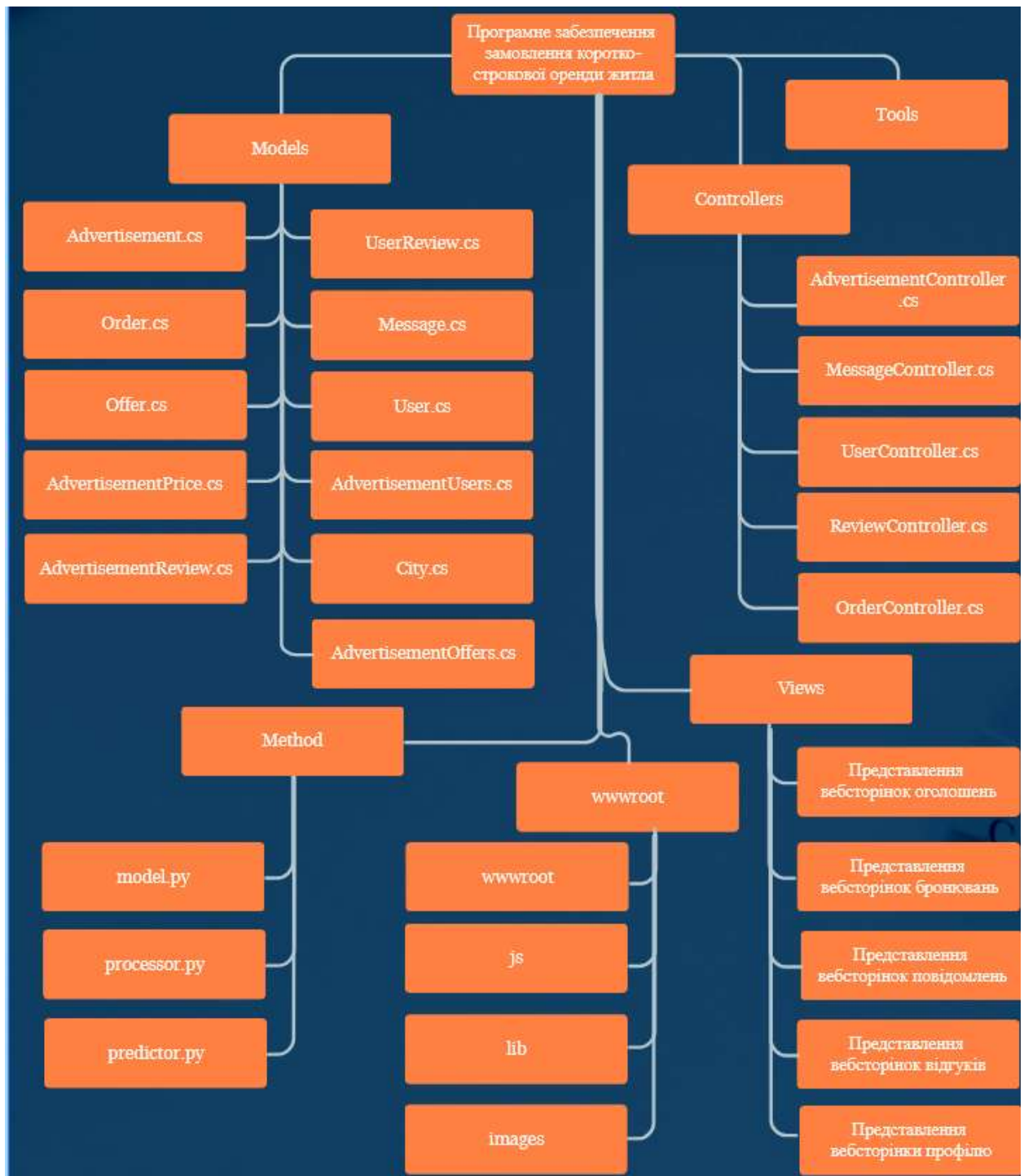


Рисунок 3.2 – Структурна схема програми

3.4 Висновки за розділом 3

Виконано вибір мов програмування C# і Python для реалізації відповідно вебзастосунку та запропонованого методу, визначено структуру БД і програми загалом.

4 ОСНОВНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ

4.1 Функціональна схема програмної системи

У відповідності з визначеними сценаріями роботи з програмою, можлива взаємодія з програмою як авторизованих, так і неавторизованих користувачів. Зрозуміло, що робота програми розпочинається зі взаємодії з неавторизованим користувачем. На початку роботи програми неавторизованому користувачу відображається вебсторінка пошуку оголошень для оренди житла (рис. 4.1).

Після того, як отримано результати пошуку, користувач може обрати для перегляду конкретне оголошення і воно відкриється окремо, а може виконати авторизацію для роботи далі в повноцінному режимі. Якщо користувач обрав відкрити окреме оголошення, то він може для цього оголошення переглянути щомісячну середню вартість оренди його (історичну і прогнозовану), може визначити вартість оренди. Після цього користувач може знову повернутися до пошуку, виконуючи пошук інших оголошень, а може авторизуватися.

Після авторизації користувачу надається можливість керування власними оголошеннями, роботи з повідомленнями, керування власним профілем, роботи з відгуками або можливість виходу з власного облікового запису. Окрім того користувач може переглянути власні бронювання, при цьому йому надається можливість змінити їх. Якщо оренда за бронюванням вже розпочата, то можна тільки створити відгук про житло. Якщо вона ще не розпочата, то надається можливість скасування бронювання. Ще одна можливість авторизованого користувача пов'язана з виконанням пошуку оголошень. Але на відміну від неавторизованого користувача, авторизований користувач може забронювати житло.

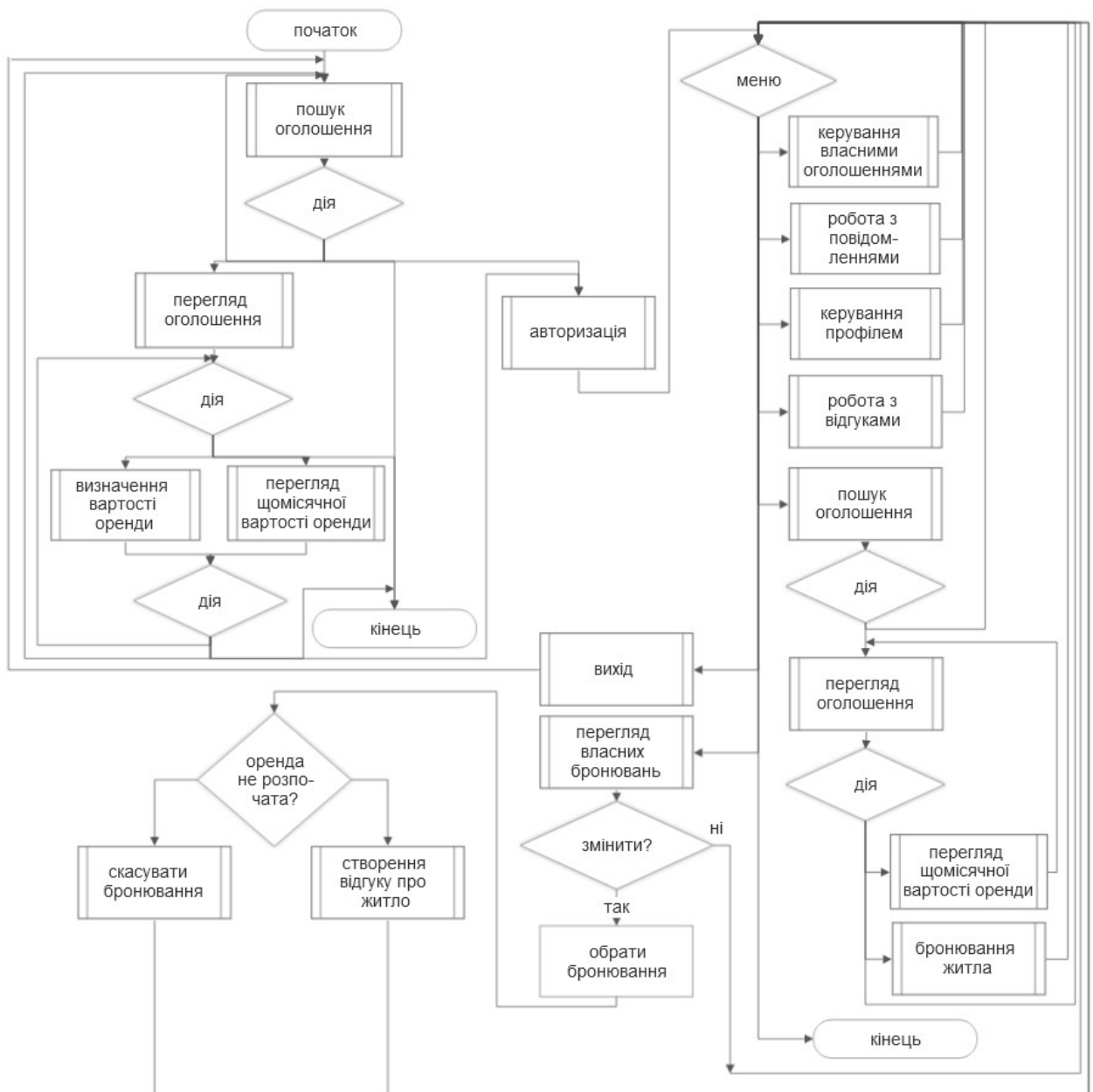


Рисунок 4.1 – Функціональна схема програмної системи

4.2 Опис модулів реалізованої програмної системи

Реалізацію моделей було виконано за допомогою модуля Entity Framework Core, який дозволяє співставляти об'єкти в коді програми, з якими зазвичай і працюють моделі, з таблицями БД. Такий підхід дозволяє не виконувати подвійні дії, коли потрібно створити спочатку структуру БД, а потім в коді прописати структуру об'єктів, яка фактично повторює структуру таблиць БД. За використання Entity Framework Core потрібно тільки створити

класи моделей, а таблиці БД були створено автоматично на основі даного коду. Окрім безпосереднього перетворення між об'єктами та сутностями БД Entity Framework Core надає зручний інструмент роботи над вибиранням відповідних даних, дозволяючи виконувати запити Language-INtegrated Query (LINQ).

Окрім безпосередньо визначення атрибутів класів, які стають полями таблиць БД, класи містять також додаткові засоби, що підвищують зручність роботи з ними, тому розглянемо принципи створення таких класів на основі класу Advertisement, який містить в собі всі потрібні дані для роботи з оголошеннями окрім безпосередньо вмісту структури самої таблиці БД, розглянутої в розділі 3.

- `public int AdvertisementId` – атрибут класу, необхідний для визначення однойменного поля таблиці БД з ідентифікатором оголошення, який є первинним ключем таблиці;

- `public string Name` – атрибут класу, необхідний для визначення однойменного поля таблиці БД з назвою оголошення, що використовується як заголовок під час відображення результатів пошуку, виведення інформації про оголошення, при цьому дане поле є обов'язковим, а його довжина обмежена 50 символами;

- `public string Info` – атрибут класу, необхідний для визначення однойменного поля таблиці БД з повним описом оголошення;

- `public byte Floor` – атрибут класу, необхідний для визначення однойменного поля таблиці БД з номером поверху, на якому розташовано житло;

- `public byte Floors` – атрибут класу, необхідний для визначення однойменного поля таблиці БД з кількістю поверхів у житлі;

- `public byte RentalType` – атрибут класу, необхідний для визначення однойменного поля таблиці БД з типом оренди житла, що представляє собою обмежене ціле число, тобто для компактності збереження не зберігається повна назва типу оренди, окрім того, оскільки варіанти оренди є стабільною

множиною, що визначається всією галуззю загалом і рідко змінюється, то сенсу виділяти це поле в окрему сутність не було, при цьому дане поле є обов'язковим;

- public byte Guests – атрибут класу, необхідний для визначення однойменного поля таблиці БД з максимальною кількістю гостей, яку здатне прийняти житло, при цьому дане поле є обов'язковим;

- public byte Rooms – атрибут класу, необхідний для визначення однойменного поля таблиці БД з кількістю кімнат у житлі, при цьому дане поле є обов'язковим;

- public byte Beds – атрибут класу, необхідний для визначення однойменного поля таблиці БД з кількістю ліжок, розміщеною в житлі для проживання гостей, при цьому дане поле є обов'язковим;

- public byte BathroomType – атрибут класу, необхідний для визначення однойменного поля таблиці БД з типом і варіантом розміщення ванної кімнати, при цьому дане поле є обов'язковим;

- public byte Photos – атрибут класу, необхідний для визначення однойменного поля таблиці БД з кількістю фотографій житла, при цьому дане поле є обов'язковим, але безпосередньо сам набір фотографій не міститься в БД: для цього використовується папка з зображеннями у структурі проекту, куди завантажуються фотографії, надаючи їм відповідне ім'я (воно складається з ідентифікатора оголошення і за порядком номера фотографії);

- public byte Bedrooms – атрибут класу, необхідний для визначення однойменного поля таблиці БД з кількістю спальних кімнат житла, при цьому дане поле є обов'язковим;

- public float Score – атрибут класу, необхідний для визначення однойменного поля таблиці БД з сумарною оцінкою, виставленою житлу за результатами відгуків всіх користувачів, які виконували оренду даного житла шляхом усереднення значення;

– `public int CityId` – атрибут класу, необхідний для визначення однойменного поля таблиці БД з ідентифікатором міста, в якому розташоване житло, при цьому дане поле є обов’язковим, є зовнішнім ключем;

– `public TimeSpan CheckinStart` – атрибут класу, необхідний для визначення однойменного поля таблиці БД з часом, з якого найраніше може початися оренда житла;

– `public TimeSpan CheckinEnd` – атрибут класу, необхідний для визначення однойменного поля таблиці БД з часом найпізнішого початку оренди житла;

– `public TimeSpan Checkout` – атрибут класу, необхідний для визначення однойменного поля таблиці БД з часом завершення оренди;

– `public virtual City City` – віртуальний атрибут, який визначає повні дані міста, в якому розташоване житло, виставлене за оголошенням;

– `public virtual ICollection<AdvertisementUser> AdvertisementUsers` – віртуальний атрибут, який визначає колекцію, що складається з користувачів, які є власниками житла або допомагають у виконанні його оренди, і містить повністю відповідні об’єкти;

– `public virtual ICollection<AdvertisementReview> AdvertisementReviews` – віртуальний атрибут, який визначає колекцію, що складається з відгуків користувачів про дане оголошення і містить повністю відповідні об’єкти;

– `public virtual ICollection<AdvertisementOffer> AdvertisementOffers` – віртуальний атрибут, який визначає колекцію, що складається з характеристик житла і містить повністю відповідні об’єкти;

– `public virtual ICollection<AdvertisementPrice> AdvertisementPrices` – віртуальний атрибут, який визначає колекцію, що складається з цінових пропозицій щодо даного оголошення про оренду житла і містить повністю відповідні об’єкти;

– `public virtual ICollection<Order> Orders` – віртуальний атрибут, який визначає колекцію, що складається з замовлень даного житла і містить повністю відповідні об'єкти.

Клас також містить атрибути, які не відображаються на поля таблиць БД. Для них окремо встановлено відповідні методи для отримання значень та для окремих таких атрибутів – для встановлення значень.

Атрибут `CheckinStartStr` містить текстове представлення часу, з якого найраніше може початися оренда житла, і надає метод для повернення його значення.

Атрибут `CheckinEndStr` містить текстове представлення часу найпізнішого початку оренди житла і надає метод для повернення його значення.

Атрибут `CheckoutStr` містить текстове представлення часу завершення оренди і надає метод для повернення його значення.

Атрибут `RentalTypeStr` містить текстове представлення типу оренди житла, формуючи його на основі відповідного цілого числа, або навпаки, перетворюючи текстове представлення на число. Встановлено наступну відповідність:

- тип оренди «цілий будинок» відповідає значенню 1;
- тип оренди «ціла квартира» відповідає значенню 2;
- тип оренди «готельний номер» відповідає значенню 3;
- тип оренди «спільна кімната» відповідає значенню 4;
- тип оренди «окрема кімната» відповідає значенню 5.

Атрибут `BathroomTypeStr` містить текстове представлення типу ванної кімнати і варіанту її розміщення, формуючи його на основі відповідного цілого числа, або навпаки, перетворюючи текстове представлення на число. Встановлено наступну відповідність:

- тип ванної кімнати «ванна» відповідає значенню 1;
- тип ванної кімнати «душ» відповідає значенню 2;
- тип ванної кімнати «душ та ванна» відповідає значенню 3;

- тип ванної кімнати «душ в блоці» відповідає значенню 4;
- тип ванної кімнати «душ на поверсі» відповідає значенню 5;
- тип ванної кімнати «ванна в блоці» відповідає значенню 6;
- тип ванної кімнати «ванна на поверсі» відповідає значенню 7.

Атрибут `ScoreStr` визначає текстове представлення середньої оцінки, виставленої за відгуками даному оголошенню. Оцінка при цьому округлюється до 2 знаку після коми.

Контролери реалізовано з використанням запитів LINQ. Для заповнення представлень відповідними даними. Розглянемо дані принципи створення контролерів на прикладі класу `AdvertisementController`, який забезпечує заповнення представлень даними стосовно оголошень про оренду житла.

Метод `Task<IActionResult> Details(int? ad)` визначає заповнення представлення перегляду повної інформації про оголошення даними оголошення з номером `ad`.

Метод `Task<IActionResult> SearchAdvertisements(int? rt, int? guests, int? rooms1, int? rooms2, int? beds1, int? beds2, int? bath, int? bedrooms1, int? bedrooms2, int? city, int? price1, int? price2, int? d1, int? m1, int? y1, int? d2, int? m2, int? y2, List<Offer> offers)` реалізує пошук оголошень з широкою кількістю налаштувань, які включають наступні:

- тип оренди житла `rt`, для якого має бути задано конкретне числове значення;
- кількість гостей `guests`;
- кількість кімнат, яка повинна знаходитися в проміжку від `rooms1` до `rooms2`;
- кількість ліжок, яка повинна знаходитися в проміжку від `beds1` до `beds2`;
- тип ванної кімнати `bath`;
- кількість ліжок, яка повинна знаходитися в проміжку від `bedrooms1` до `bedrooms2`;

- місто, в якому повинно знаходитися житло, що задається ідентифікатором відповідного міста на основі параметра city;
- вартість оренди на добу, яка повинна знаходитися в проміжку від price1 до price2;
- проміжок дат, на які планується оренда: задається окремими значеннями, що включають для дати початку день d1, місяць m1, рік y1, а для дат завершення відповідно день d2, місяць m2, рік y2;
- перелік характеристик, які повинно мати житло, offers, що включає ідентифікатори відповідних характеристик.

Перевірка потрібних характеристик житла реалізується наступним чином. Для цього перебираються всі потрібні значення в масиві характеристик і якщо хоча би одна з них відсутня у фактичного оголошення (перевіряється запитом LINQ для вибору значень, що відповідають умові співпадіння ідентифікатора та наявності даної характеристики, для чого значення Presence має дорівнювати 1, таким чином визначаючи оголошення, які потрібно залишити), то його вилучають з переліку:

```
foreach (var off in offers)
{
    var aof = adv.AdvertisementOffers.Where(a => a.OfferId = off &&
a.Presence == 1);
    if (aof == null)
    {
        off = false;
    }
}
```

Для перевірки доступності житла на задані дати виконується пошук замовлень за даним оголошенням на вказаний проміжок дат. Якщо такі

замовлення існують, то оголошення виключається з переліку оголошень, які повертаються в результаті:

```

DateTime ld1 = DateTime(y1, m1, d1);
DateTime ld2 = DateTime(y2, m2, d2);
var orders = db.Orders.Where(o => o.Advertisement ==
adv.AdvertisementId);
orders = orders.Where(o => o.StartDate >= ld1 && o.StartDate <= ld2 ||
o.EndDate >= ld1 && o.EndDate <= ld2 || o.StartDate <= ld1 && o.EndDate >=
ld2)
if (orders != null)
{
    off = false;
}

```

Метод `Task<IActionResult> AdvertisementPrice(int? ad, int? d1, int? m1, int? y1, int? d2, int? m2, int? y2, int? guests)` виконує обчислення вартості доби оренди житла за оголошенням з номером `ad` на дати, починаючи з дня `d1`, місяця `m1`, року `y1`, завершуючи днем `d2`, місяцем `m2`, роком `y2`, для кількості гостей `guests`.

Для реалізації виконується пошук усіх цінових пропозицій, які стосуються даного оголошення і включають заданий часовий проміжок. При цьому враховується, що такий часовий проміжок може охоплювати одразу декілька цінових пропозицій:

```

DateTime cld1 = ap.StartDate;
DateTime cld2 = ap.EndDate;
if (cld1 < ld1)
{
    cld1 = ld1;
}

```

```

    }
    if (cld2 > ld2)
    {
        cld2 = ld2;
    }
    cld1 = cld2 - cld1;
    int d = cld1.TotalDays;

```

Цінова пропозиція використовується тільки в тому випадку, коли мінімальна і максимальна тривалості оренди в ній відповідають кількості днів з заданого проміжку дат:

```

if (ap.MaxDays >= ld && ap.MinDays <= ld)
{
    days = days + d;
    result = result + d * ap.Price;
    cp = ap.CancellationPolicy;
}

```

Метод `IActionResult Create()` виконує підтримку початкового відкриття вебсторінки створення нового оголошення.

Метод `Task<IActionResult> Create([Bind("AdvertisementId, Name, Info, Floor, Floors, RentalType, Guests, Rooms, Beds, BathroomType, Offers, Photos, Bedrooms, CityId, CheckinStart, CheckinEnd, Checkout")] Advertisement adv)` вносить в БД дані нового оголошення, включаючи назву `Name`, опис `Info`, номер поверху `Floor`, кількість поверхів `Floors`, тип оренди `RentalType`, максимальну кількість гостей `Guests`, кількість кімнат `Rooms`, кількість ліжок `Beds`, тип ванної кімнати `BathroomType`, набір характеристик `Offers`, кількість ліжок `Bedrooms`, номер міста `CityId`, час початку заселення `CheckinStart`, час завершення заселення `CheckinEnd`, час виселення `Checkout`.

Метод `Task<IActionResult> Update(int? adv)` виконує підтримку початкового відкриття вебсторінки редагування існуючого оголошення з ідентифікатором `adv`.

Метод `Task<IActionResult> Update(int adv, [Bind("AdvertisementId, Name, Info, Floor, Floors, RentalType, Guests, Rooms, Beds, BathroomType, Offers, Photos, Bedrooms, CityId, CheckinStart, CheckinEnd, Checkout")] Advertisement adv)` вносить в БД дані оголошення з номером `adv`, змінюючи значення на назву `Name`, опис `Info`, номер поверху `Floor`, кількість поверхів `Floors`, тип оренди `RentalType`, максимальну кількість гостей `Guests`, кількість кімнат `Rooms`, кількість ліжок `Beds`, тип ванної кімнати `BathroomType`, набір характеристик `Offers`, кількість фотографій `Photos`, кількість ліжок `Bedrooms`, номер міста `CityId`, час початку заселення `CheckinStart`, час завершення заселення `CheckinEnd`, час виселення `Checkout`.

4.3 Висновки за розділом 4

Функціональні можливості програмного забезпечення впорядковано за допомогою відповідної схеми функціонування програми, після чого їх було реалізовано на основі створення відповідного набору класів, за якими представлено принципи їх створення і основні рішення.

5 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

5.1 Призначення програми

Основним призначенням програми є забезпечення бронювань житла гостями та створення пропозицій для бронювання господарями з наданням можливостей зворотного зв'язку між даними сторонами. Особливістю програми є надання даних щодо середньої вартості оренди житла за місяць на основі історичних даних та результатів прогнозування.

5.2 Умови виконання програми

Апаратне забезпечення сервера повинно відповідати наступним вимогам:

- процесор: чотири ядра, тактова частота – 2,9 ГГц;
- оперативна пам'ять: 10 Гб;
- жорсткий диск: 300 Мб доступного простору.

Вимогою доступу до програмного забезпечення є його запуск на сервері, відповідно має підтримуватися високошвидкісний доступ через мережу інтернет до сервера з наявністю встановленого програмного забезпечення: інтерпретатора Python, бібліотеки Keras, пакетів Pandas, засобів середовища Visual Studio версії не нижче 2017 (або без встановлення самого середовища встановлення засобів платформи .NET, пакету SDK, які використовуються для забезпечення роботи середовища Visual Studio відповідної версії).

5.3 Виконання програми

Графічний інтерфейс вебзастосунку було створено з використанням шаблону [20].

Основою взаємодії користувача з програмою є вебсторінка пошуку житла (рис. 5.1). Вона доступна і авторизованому, і неавторизованому користувачу.

На вебсторінці пошуку, яка доступна через пункт меню «Пошук», потрібно задати параметри пошуку, що включають наступні:

- місто, в якому знаходиться житло: для цього використовується поле «Місто», яке має випадаючий перелік, у поле потрібно вводити назву міста, а випадаючий перелік звужується при цьому до назв, які відповідають заданій частині;
- дату заїзду, яку потрібно задати введенням в заданому форматі або вибором з календаря;
- дату виїзду, яку потрібно задати введенням в заданому форматі або вибором з календаря;
- кількість гостей, що вибирається з випадаючого переліку (дане значення може впливати на вартість житла);
- вартість оренди доби в проміжку від мінімального значення до максимального значення, які потрібно ввести у відповідні поля;
- тип житла, що потрібно обрати з випадаючого переліку (окрім конкретних типів доступний пункт «усі», що використовується за замовчуванням);
- кількість кімнат в проміжку від мінімальної кількості до максимальної, значення яких потрібно обрати з випадаючих переліків;
- кількість ліжок в проміжку від мінімальної кількості до максимальної, значення яких потрібно обрати з випадаючих переліків;
- кількість спалень в проміжку від мінімальної кількості до максимальної, значення яких потрібно обрати з випадаючих переліків;
- тип ванної кімнати, який потрібно задати з випадаючого переліку (за замовчуванням доступний варіант «усі»);

– масив характеристик житла, для яких потрібно встановити, чи є обов’язковою їх наявність у оголошенні (даний перелік може через БД розширюватися).

AGRentaller Оголошення **Пошук** Бронювання Повідомлення Відгуки Профіль Вийти

Місто: Мюнхен

Дата заїзду: 22.12.2021

Дата виїзду: 25.12.2021

Гостей: 3

Вартість від до: 10 100

☒ кухня
☒ wi-fi
☒ пральна машина
☐ праска
☐ опалення
☐ гаряча вода
☒ кондиціонер
☐ фен
☐ телевізор

☐ базові зручності
☐ детектор диму
☐ чайник
☐ кавоварка
☐ духовка шафа
☐ посудомийка
☐ холодильник
☐ мікрохвильовка
☐ самостійне прибуття

Тип житла: усі

Кімнат від до: 1 3

Ліжок від до: 2 3

Тип ванної кімнати: Душ

Спальень від до: 1 2

Шукати

Результати пошуку

Назва	Тип житла	Оцінка	Вартість
Сендлінг сан	ціла квартира	★4.9	90 \$
Антиквартура	ціла квартира	★4.8	70 \$
Тівольський світ	ціла квартира	★4.8	75 \$

Рисунок 5.1 – Вебсторінка пошуку житла

Після визначення всіх цих параметрів необхідно натиснути на кнопку «Шукати». Результати відображаються нижче в переліку на вебсторінці. Там перелічуються всі варіанти житла, що відповідають заданим параметрам. Виведені варіанти відсортовані за середньою оцінкою гостей.

За кожним варіантом житла виводиться його перша фотографія, назва, тип житла, середня оцінка від гостей та вартість однієї доби оренди за

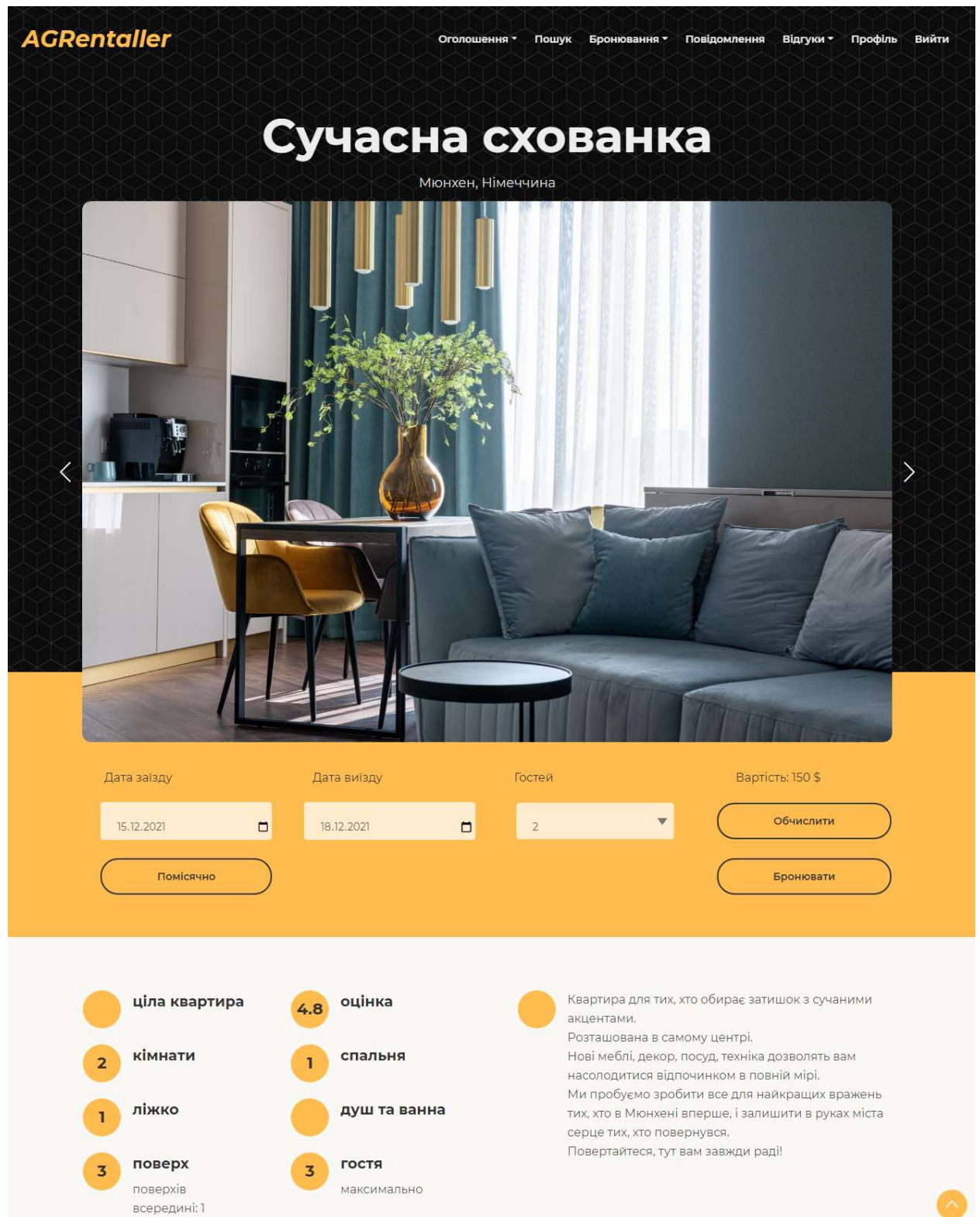


Рисунок 5.2 – Вебсторінка оголошення

За кожним оголошенням відтворюється перелік фотографій, між якими можна переключатися, назва житла, місто проживання. Нижче доступні основні характеристики житла, що включають його тип, кількість кімнат, кількість ліжок, поверх розміщення, середню оцінку, тип ванної кімнати, максимальну кількість гостей, текстовий опис квартири, окрім того нижче розташований набір характеристик, наявних у квартири, перелік відгуків користувачів.

Окрім того можна одразу змінити параметри пошуку і переглянути вартість проживання в даному житлі за заданий період заданої кількості гостей. Звідси можна також перейти до вебсторінки перегляду статистики і прогнозу вартості оренди житла (рис. 5.3). На цій сторінці можна задати піотрбний початковий місяць і кінцевий місяць інтервалу, для якого потрібно відобразити середньомісячну вартість доби оренди. В лівому стовпчику відображаються історичні дані, а в правому – прогнозовані.

Загалом розширений варіант меню, доступний авторизованому користувачу, включає пункти:

- «Оголошення» (недоступний неавторизованому користувачу), що дозволяє працювати з власними оголошеннями і включає підпункти «Додати», «Усі»;
- «Пошук»;
- «Бронювання» (недоступний неавторизованому користувачу), що дозволяє працювати з власними бронюваннями;
- «Повідомлення» (недоступний неавторизованому користувачу), що дозволяє переглянути переписку з власниками житла;
- «Відгуки» (недоступний неавторизованому користувачу), що дозволяє переглядати різні відгуки, містить підпункти «Житло», що містить відгуки про житло авторизованого користувача як господаря, «Гість», що містить відгуки про авторизованого користувача як про гостя, «Мої», що містить відгуки авторизованого користувача;
- «Профіль» (недоступний неавторизованому користувачу);

– «Вийти».

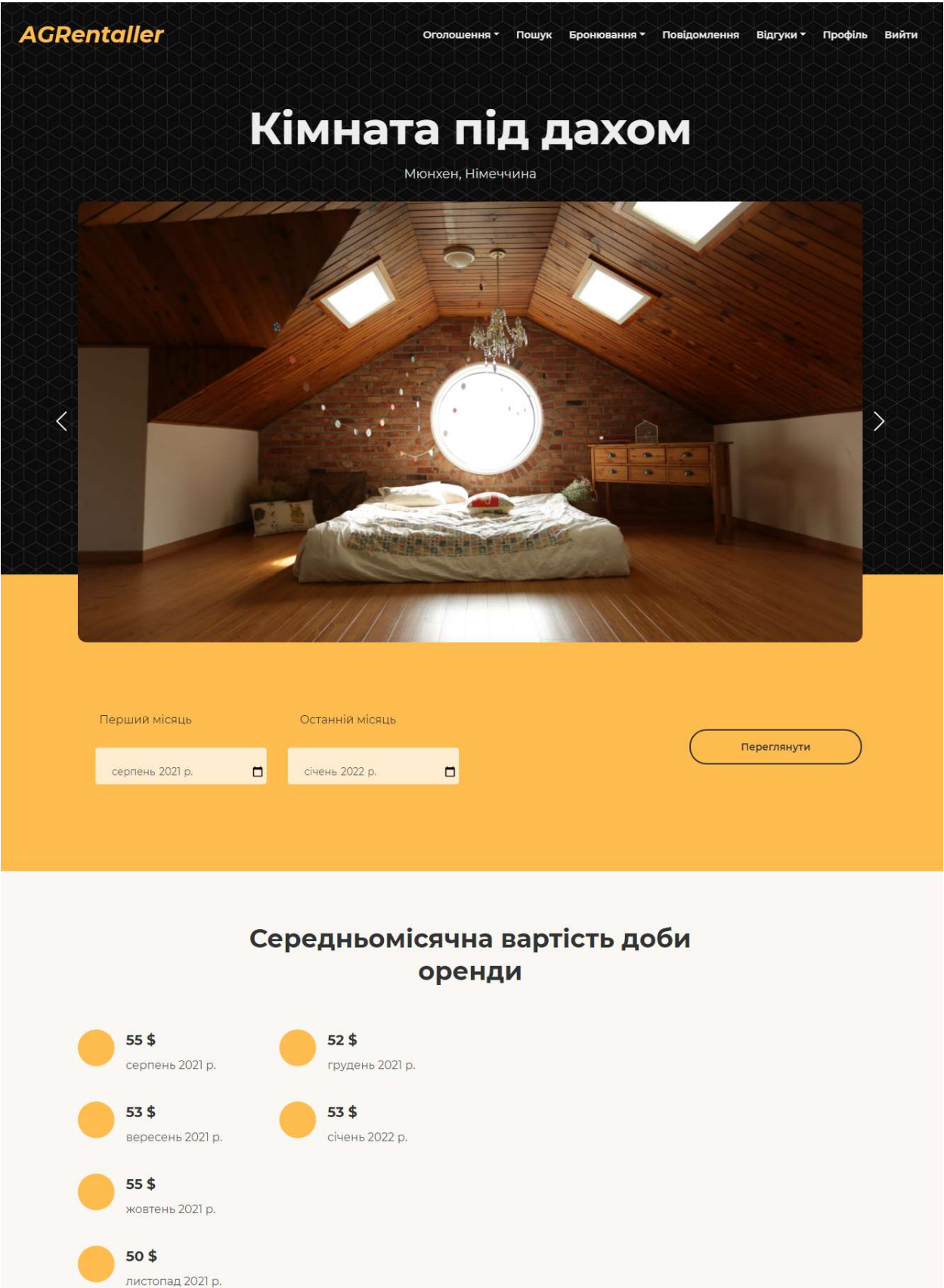


Рисунок 5.3 – Вебсторінка середньомісячної вартості оренди житла

5.4 Експериментальне дослідження проблеми прогнозування вартості короткострокової оренди житла

Експериментальне дослідження було проведено на основі вибірки даних з джерела [21], де розміщено дані про вартість оренди житла на Airbnb за містами світу. Вибірка даних складається з екземплярів, кожен з яких представляє вартість оренди за відповідну дату для зазначеного номеру оголошення. В якості основи для дослідження було використано вибірку, яка містить дані по місту Мюнхен і включає загалом більше 1 мільйона позицій. Відповідні дані були за допомогою методу прогнозування вартості короткострокової оренди житла перетворені до форми часових рядів, формуючи відповідну вибірку даних, після чого її було оброблено на основі другого етапу роботи методу, а отримані в результаті дані використовувались для роботи відповідних моделей прогнозування (табл. 5.1).

Таблиця 5.1 – Результати експериментального дослідження

Метод / модель	Середньоквадратична похибка
Лінійна регресія	0,3164
Метод опорних векторів	0,2861
Нейронні мережі прямого поширення сигналу	0,1929
Метод прогнозування вартості короткострокової оренди житла	0,1285

Найкращі результати прогнозування на даній вибірці даних дозволив отримати запропонований метод, маючи середньоквадратичну похибку в 0,1285 на нормалізованих значеннях вартості оренди житла на добу. Ці результати виявились на 33 % кращими за результати, отримані на основі звичайних нейронних мереж прямого поширення сигналу. Інші моделі дозволили отримати гірші результати.

5.5 Висновки за розділом 5

Описано основні режими роботи користувачів з програмою, представлено логіку комунікації користувачів. Проведено експериментальне дослідження на вибірці даних про вартість оренди житла на Airbnb. За результатами дослідження запропонований метод дозволив отримати середньоквадратичну похибку в 0,1285, що на 33 % краще за нейронні мережі прямого поширення сигналу.

ВИСНОВКИ

У кваліфікаційній роботі магістра розроблено програмне забезпечення замовлення короткострокової оренди житла для підтримки замовлень на основі результатів прогнозування вартості житла в майбутньому періоді. Для досягнення даного результату було вирішено завдання роботи:

- виконано аналіз програмних засобів підтримки замовлення короткострокової оренди житла: виконано огляд і аналіз програмних аналогів, за результатами якого було визначено особливості програми, що розробляється, порівняно з аналогами, визначено варіанти використання програми неавторизованими і авторизованими користувачами після визначення функціональних вимог до програми;

- створено новий метод прогнозування вартості короткострокової оренди житла;

- виконано проєктування програмного забезпечення: виконано вибір мов програмування C# і Python для реалізації відповідно вебзастосунку та запропонованого методу, визначено структуру бази даних і програми загалом;

- реалізовано програмне забезпечення у вигляді вебзастосунку, який дозволяє подавати оголошення про оренду житла, бронювати житло, виконувати його пошук, керувати власними бронюваннями і керувати господарю бронюваннями власного житла, створювати відгуки на житло і гостей, відправляти повідомлення від гостя до господаря і навпаки, переглядати дані про вартість житла в минулому та результати прогнозування вартості житла на майбутні 2 місяці;

- виконано експериментальне дослідження використання розробленого методу для розв'язання проблеми прогнозування вартості короткострокової оренди житла, де на основі вибірки даних, що характеризує вартість житла на Airbnb, підтверджено, що запропонований метод дозволив отримати найвищу точність прогнозування порівняно з лінійною регресією,

методом опорних векторів, нейронними мережами прямого поширення сигналу.

Таким чином, слід констатувати, що всі завдання, які було поставлено на роботу, виконано повністю.

Наукова новизна роботи полягає в запропонованому методі прогнозування вартості короткострокової оренди житла, який виконує прогноз на період з встановленим часовим інтервалом, використовуючи для цього штучну нейронну мережу на основі архітектури довгої короткострокової пам'яті, що дозволяє вибирати замовникам економічно обґрунтований момент для оформлення запланованого замовлення житла.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Airbnb [Електрон. ресурс] : неймовірні помешкання та враження. – Режим доступу : <https://www.airbnb.com.ua/>.
2. Airbnb vs. Vrbo [Electronic resource] : Key Differences Explained. – Access mode : <https://www.igms.com/airbnb-or-vrbo/>.
3. Шикова, Е. Airbnb или booking, что лучше по нашему мнению [Электрон. ресурс] / Е. Шикова. – Режим доступа : <https://trip-together.ru/airbnb-booking-rent-apartment>.
4. Booking.com. Офіційний сайт. Найкращі готелі та помешкання [Електрон. ресурс]. – Режим доступу : <https://www.booking.com/>.
5. Linear Regression [Electronic resource]. – Access mode : <http://www.stat.yale.edu/Courses/1997-98/101/linreg.htm>.
6. Sethi, A. Support Vector Regression Tutorial for Machine Learning [Electronic resource] / A. Sethi. – Access mode : <https://www.analyticsvidhya.com/blog/2020/03/support-vector-regression-tutorial-for-machine-learning/>.
7. Sharp, T. An Introduction to Support Vector Regression (SVR) [Electronic resource] / T. Sharp. – Access mode : <https://towardsdatascience.com/an-introduction-to-support-vector-regression-svr-a3ebc1672c2>.
8. Gupta, V. Understanding Feedforward Neural Networks [Electronic resource] / V. Gupta. – Access mode : <https://learnopencv.com/understanding-feedforward-neural-networks/>.
9. Знакомство с архитектурой LSTM-сетей [Электрон. ресурс]. – Режим доступа : <http://datareview.info/article/znakomstvo-s-arhitekturoy-lstm-setey/>.
10. Gonzalez, J. Non-linear system modeling using LSTM neural networks [Text] / J. Gonzalez, Wen Yu // IFAC-PapersOnLine. – 2018. – Vol. 51, Issue 13. – Pp. 485-489.

11. Phi, M. Illustrated Guide to LSTM's and GRU's [Electronic resource] : A step by step explanation / M. Phi. – Access mode : <https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21>.
12. Saxena, S. Introduction to Long Short Term Memory (LSTM) [Electronic resource] / S. Saxena. – Access mode : <https://www.analyticsvidhya.com/blog/2021/03/introduction-to-long-short-term-memory-lstm/>.
13. Onnen, H. Temporal Loops [Electronic resource] : Intro to Recurrent Neural Networks for Time Series Forecasting in Python / H. Onnen. – Access mode : <https://towardsdatascience.com/temporal-loops-intro-to-recurrent-neural-networks-for-time-series-forecasting-in-python-b0398963dc1f>.
14. Melcher, K. Multivariate Time Series Analysis with an LSTM based RNN [Electronic resource] / K. Melcher. – Access mode : <https://www.kdnuggets.com/2021/10/multivariate-time-series-analysis-lstm-based-rnn.html>.
15. Pitt, Ch. Pro PHP 8 MVC [Text] : Model View Controller Architecture-Driven Application Development / Ch. Pitt. – New York : Apress, 2021. – 388 p.
16. Learn C# Programming [Text] : A guide to building a solid foundation in C# language for writing efficient programs / Marius Bancila, Raffaele Rialdi, Ankit Sharma, Dino Esposito. – Birmingham : Packt Publishing, 2020. – 636 p.
17. Murach, J. Murach's ASP.NET Core MVC [Text] / J. Murach, M. Delamater. – Fresno, CA : Mike Murach & Associates, 2020. – 780 p.
18. Wickham, H. R for Data Science [Text] : Import, Tidy, Transform, Visualize, and Model Data / H. R. Wickham, G. Grolemund. – Massachussets : O'Reilly Media, 2017. – 520 p.
19. Lutz, M. Programming Python [Text] : Powerful Object-Oriented Programming / M. Lutz. – Massachussets : O'Reilly Media, 2011. – 1632 p.
20. Curso Free Website Tepmlate. Free CSS Tepmplates. Free CSS [Electronic resource]. – Access mode : <https://www.free-css.com/free-css->

templates/page273/corso.

21. Get the Data – Inside Airbnb. Adding data to the debate [Electronic resource]. – Access mode : <http://insideairbnb.com/get-the-data.html>.

ДОДАТОК А
Технічне завдання

Вступ

Короткострокова оренда житла характеризується зміною вартості оренди. На такі зміни впливають різні фактори, які загалом визначаються зміною попиту. Для користувачів таких програмних систем завжди актуальною проблемою постає момент виконання бронювання, тобто чи це потрібно зробити сьогодні, чи через певний період. Засоби прогнозування вартості мають допомогти у вирішенні таких проблем.

Розроблюване програмне забезпечення у складі функцій бронювання житла повинна надавати можливість отримання прогнозу вартості.

A.1 Підстави для розробки

Підставою для розробки є завдання на кваліфікаційну роботу магістра, визначене за темою «Дослідження та програмна реалізація методів підтримки замовлення короткострокової оренди житла», яку затверджено наказом № 435 від 5 листопада 2021 р. за Національним університетом «Запорізька політехніка».

A.2 Призначення розробки

Основним призначенням програми є забезпечення бронювань житла гостями та створення пропозицій для бронювання господарями з наданням можливостей зворотного зв'язку між даними сторонами.

A.3 Основні вимоги до програми

A.3.1 Вимоги до функціональних характеристик

Для реалізації призначення програми вона повинна виконувати наступні функції:

- подання оголошення про оренду житла;
- редагування оголошення;
- визначення цінової пропозиції для оголошення на задані дати;
- перегляд оголошення про оренду житла;
- бронювання житла на задані дати;
- створення відгуку про житло;
- створення відгуку про гостя;
- перегляд відгуків про себе;
- перегляд власних відгуків;
- перегляд повідомлень з власником житла;
- відправлення повідомлення власнику житла;
- пошук житла в заданому місті на задані дати з заданими додатковими параметрами;
- обчислення вартості бронювання житла;
- перегляд власних оголошень;
- деактивація оголошення;
- перегляд бронювань житла власником;
- перегляд власних бронювань;
- скасування бронювання житла;
- обчислення сумарного рейтинга житла;
- обчислення сумарного рейтинга користувача;
- прогнозування вартості оренди житла на наступні місяці;
- обчислення середньої вартості оренди житла за місяць;
- змінювання профілю користувача;
- авторизація і реєстрація користувачів.

A.3.2 Вимоги до складу та параметрів технічних засобів

Апаратне забезпечення роботи вебсервера повинно відповідати вимогам роботи з щонайменше 200 користувачами одночасно. Для підтримки

цих вимог апаратне забезпечення повинно мінімально включати:

- процесор: чотири ядра, тактова частота – 2,9 ГГц;
- оперативна пам'ять: 10 Гб;
- жорсткий диск: 300 Мб доступного простору.

A.4 Порядок контролю та приймання

Календарний план є основою для контролю виконання роботи, у свою чергу приймання роботи здійснюється під час її захисту перед екзаменаційною комісією.

ДОДАТОК Б
Текст програми

Б.1 Текст файла /Models/Advertisement.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;

namespace Rentaller.Models
{
    public class Advertisement
    {
        [Key]
        public int AdvertisementId
        {get; set;}
        [Required] [MaxLength(50)]
        public string Name
        {get; set;}
        public string Info
        {get; set;}
        public byte Floor
        {get; set;}
        public byte Floors
        {get; set;}
        [Required]
        public byte RentalType
        {get; set;}
        [Required]
        public byte Guests
        {get; set;}
        [Required]
        public byte Rooms
        {get; set;}
        [Required]
        public byte Beds
        {get; set;}
        [Required]
        public byte BathroomType
        {get; set;}
        [Required]
        public byte Photos
        {get; set;}
        [Required]
        public byte Bedrooms
```

```

        {get; set;}
        public float Score
        {get; set;}
        [Required]
        public int CityId
        {get; set;}
        public TimeSpan CheckinStart
        {get; set;}
        public TimeSpan CheckinEnd
        {get; set;}
        public TimeSpan Checkout
        {get; set;}
        public virtual City City
        {get; set;}
        public virtual ICollection<AdvertisementUser>
AdvertisementUsers
        {get; set;}
        public virtual ICollection<AdvertisementMonthPrice>
AdvertisementMonthPrices
        {get; set;}
        public virtual ICollection<AdvertisementReview>
AdvertisementReviews
        {get; set;}
        public virtual ICollection<AdvertisementOffer>
AdvertisementOffers
        {get; set;}
        public virtual ICollection<AdvertisementPrice>
AdvertisementPrices
        {get; set;}
        public virtual ICollection<Order> Orders
        {get; set;}
        [NotMapped]
        public string CheckinStartStr
        {
            get
            {
                dt = this.CheckinStart;
                string css = dt.ToString("dd:hh");
                return css;
            }
        }
        [NotMapped]
        public string CheckinEndStr

```

```

{
    get
    {
        dt = this.CheckinEnd;
        string css = dt.ToString("dd:hh");
        return css;
    }
}
[NotMapped]
public string CheckoutStr
{
    get
    {
        dt = this.Checkout;
        string css = dt.ToString("dd:hh");
        return css;
    }
}
[NotMapped]
public string RentalTypeStr
{
    get
    {
        switch (this.RentalType)
        {
            case 1:
                return "цілий будинок";
            case 2:
                return "ціла квартира";
            case 3:
                return "готельний номер";
            case 4:
                return "спільна кімната";
            case 5:
                return "окрема кімната";
            default:
                return "";
        }
    }
    set
    {
        switch (value)
        {

```

```

        case "цілий будинок":
            this.RentalType = 1;
            break;
        case "ціла квартира":
            this.RentalType = 2;
            break;
        case "готельний номер":
            this.RentalType = 3;
            break;
        case "спільна кімната":
            this.RentalType = 4;
            break;
        case "окрема кімната":
            this.RentalType = 5;
            break;
        default:
            this.RentalType = 0;
    }
}
}
[NotMapped]
public string BathroomTypeStr
{
    get
    {
        switch (this.BathroomTypeStr)
        {
            case 1:
                return "ванна";
            case 2:
                return "душ";
            case 3:
                return "душ та ванна";
            case 4:
                return "душ в блоці";
            case 5:
                return "душ на поверсі";
            case 6:
                return "ванна в блоці";
            case 7:
                return "ванна на поверсі";
            default:
                return "немає";
        }
    }
}

```

```

    }
}
set
{
    switch (value)
    {
        case "ванна":
            this.BathroomTypeStr = 1;
            break;
        case "душ":
            this.BathroomTypeStr = 2;
            break;
        case "душ та ванна":
            this.BathroomTypeStr = 3;
            break;
        case "душ в блоці":
            this.BathroomTypeStr = 4;
            break;
        case "душ на поверсі":
            this.BathroomTypeStr = 5;
            break;
        case "ванна в блоці":
            this.BathroomTypeStr = 6;
            break;
        case "ванна на поверсі":
            this.BathroomTypeStr = 7;
            break;
        default:
            this.BathroomTypeStr = 0;
    }
}
}
[NotMapped]
public string ScoreStr
{
    get
    {
        double result = Math.Round(this.Score, 2);
        return result.ToString();
    }
}
}
}

```

Б.2 Текст файла /Models/Order.cs

```

using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;

namespace Rentaller.Models
{
    public class Order
    {
        [Key]
        public int OrderId
        {get; set;}
        [Required]
        public int UserId
        {get; set;}
        [Required]
        public int AdvertisementId
        {get; set;}
        [Required]
        public DateTime ODate
        {get; set;}
        [Required]
        public DateTime StartDate
        {get; set;}
        [Required]
        public DateTime EndDate
        {get; set;}
        [Required]
        public float Price
        {get; set;}
        public virtual User User
        {get; set;}
        public virtual Advertisement Advertisement
        {get; set;}
        [NotMapped]
        public string ODateStr
        {
            get
            {
                return ((DateTime)this.ODate).ToString("dd.MM.yyyy
hh:mm");
            }
        }
    }
}

```

```

    }
}
[NotMapped]
public string StartDateStr
{
    get
    {
        return
((DateTime)this.StartDate).ToString("dd.MM.yyyy");
    }
}
[NotMapped]
public string EndDateStr
{
    get
    {
        return
((DateTime)this.EndDate).ToString("dd.MM.yyyy");
    }
}
[NotMapped]
public string PriceStr
{
    get
    {
        double result = Math.Round(this.Price, 2);
        return result.ToString();
    }
}
}
}

```

Б.3 Текст файла /Models/AdvertisementPrice.cs

```

using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;

namespace Rentaller.Models
{
    public class AdvertisementPrice
    {

```

```

[Key]
public int AdvertisementPriceId
{get; set;}
[Required]
public int AdvertisementId
{get; set;}
[Required]
public byte Guests
{get; set;}
[Required]
public float Price
{get; set;}
[Required]
public DateTime StartDate
{get; set;}
[Required]
public DateTime EndDate
{get; set;}
[Required]
public byte CancellationPolicy
{get; set;}
[Required]
public byte MinDays
{get; set;}
[Required]
public byte MaxDays
{get; set;}
public virtual Advertisement Advertisement
{get; set;}
[NotMapped]
public string StartDateStr
{
    get
    {
        return
((DateTime)this.StartDate).ToString("dd.MM.yyyy");
    }
}
[NotMapped]
public string EndDateStr
{
    get
    {

```

```

        return
        ((DateTime)this.EndDate).ToString("dd.MM.yyyy");
    }
}
[NotMapped]
public string PriceStr
{
    get
    {
        double result = Math.Round(this.Price, 2);
        return result.ToString();
    }
}
}
}

```

Б.4 Текст файла /Models/AdvertisementReview.cs

```

using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;

namespace Rentaller.Models
{
    public class AdvertisementReview
    {
        [Key]
        public int AdvertisementReviewId
        {get; set;}
        [Required]
        public int AdvertisementId
        {get; set;}
        [Required]
        public int UserId
        {get; set;}
        [Required]
        public float Score
        {get; set;}
        public string Review
        {get; set;}
        [Required]
        public DateTime RDate
    }
}

```

```

        {get; set;}
        public virtual Advertisement Advertisement
        {get; set;}
        public virtual User User
        {get; set;}
        [NotMapped]
        public string RDateStr
        {
            get
            {
                return ((DateTime)this.RDate).ToString("dd.MM.yyyy
hh:mm");
            }
        }
    }
}

```

Б.5 Текст файла /Models/UserReview.cs

```

using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;

namespace Rentaller.Models
{
    public class UserReview
    {
        [Key]
        public int UserReviewId
        {get; set;}
        [Required]
        public int UserId
        {get; set;}
        [Required]
        [ForeignKey("User")]
        public int AuthorId
        {get; set;}
        [Required]
        public float Score
        {get; set;}
        public string Review
        {get; set;}
    }
}

```

```

        [Required]
        public DateTime RDate
        {get; set;}
        public virtual User Author
        {get; set;}
        public virtual User User
        {get; set;}
        [NotMapped]
        public string RDateStr
        {
            get
            {
                return ((DateTime)this.RDate).ToString("dd.MM.yyyy
hh:mm");
            }
        }
    }
}

```

Б.6 Текст файла AdvertisementController.cs

```

using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Rendering;
using Microsoft.EntityFrameworkCore;
using Rentaller.Tools;
using Rentaller.Models;
using System;
using System.Collections.Generic;
using System.Globalization;
using System.Linq;

namespace Rentaller.Controllers
{
    public class AdvertisementController : Controller
    {
        private readonly RentallerData db;

        public AdvertisementController (RentallerData dbc)
        {
            db = dbc;
        }
    }
}

```

```

        public async Task<IActionResult> Details(int? ad)
        {
            var adv = await db.Advertisements.Include(a =>
a.AdvertisementId).Include(a => a.Name).Include(a => a.Info).Include(a =>
a.Floor).Include(a => a.Floors).Include(a => a.RentalType).Include(a =>
a.Guests).Include(a => a.Rooms).Include(a => a.Beds).Include(a =>
a.BathroomType).Include(a => a.Photos).Include(a => a.Bedrooms).Include(a
=> a.Score).Include(a => a.CityId).Include(a =>
a.CheckinStartStr).Include(a => a.CheckinEndStr).Include(a =>
a.CheckoutStr).Include(a => a.AdvertisementUsers).Include(a =>
a.AdvertisementReviews).Include(a =>
a.AdvertisementOffers).FirstOrDefaultAsync(a => a.AdvertisementId == ad);
            if (ad == null)
            {
                return NotFound();
            }
            return View(adv);
        }

        public async Task<IActionResult> SearchAdvertisements(int?
rt, int? guests, int? rooms1, int? rooms2, int? beds1, int? beds2, int?
bath, int? bedrooms1, int? bedrooms2, int? city, int? price1, int?
price2, int? d1, int? m1, int? y1, int? d2, int? m2, int? y2, List<Offer>
offers)
        {
            var advs = await db.Advertisements.Include(a =>
a.AdvertisementId).Include(a => a.Name).Include(a => a.Info).Include(a =>
a.Floor).Include(a => a.Floors).Include(a => a.RentalType).Include(a =>
a.Guests).Include(a => a.Rooms).Include(a => a.Beds).Include(a =>
a.BathroomType).Include(a => a.Photos).Include(a => a.Bedrooms).Include(a
=> a.Score).Include(a => a.CityId).Include(a =>
a.CheckinStartStr).Include(a => a.CheckinEndStr).Include(a =>
a.CheckoutStr).Include(a => a.AdvertisementUsers).Include(a =>
a.AdvertisementReviews).Include(a => a.AdvertisementOffers).Where(a =>
a.CityId == city && a.Guests <= guests && a.RentalType == rt &&
a.BathroomType == bath);
            advs.Where(a => a.Rooms >= rooms1 && a.Rooms <= rooms2
&& a.Beds >= beds1 && a.Beds <= beds2 && a.Bedrooms >= bedrooms1 &&
a.Bedrooms <= bedrooms2);
            advs.OrderBy(a => a.Score);
            List<float> prices = new List<float>();
            foreach (var adv in advs)
            {

```

```

        var off = true;
        foreach (var off in offers)
        {
            var aof = adv.AdvertisementOffers.Where(a =>
a.OfferId = off && a.Presence == 1);
            if (aof == null)
            {
                off = false;
            }
        }
        if (off)
        {
            DateTime ld1 = DateTime(y1, m1, d1);
            DateTime ld2 = DateTime(y2, m2, d2);
            var orders = db.Orders.Where(o =>
o.Advertisement == adv.AdvertisementId);
            orders = orders.Where(o => o.StartDate >= ld1
&& o.StartDate <= ld2 || o.EndDate >= ld1 && o.EndDate <= ld2 ||
o.StartDate <= ld1 && o.EndDate >= ld2)
            if (orders != null)
            {
                off = false;
            }
            else
            {
                var aps = await
db.AdvertisementPrices.Include(a => a.Guests).Include(a =>
a.Price).Include(a => a.StartDate).Include(a => a.EndDate).Include(a =>
a.CancellationPolicy).Include(a => a.MinDays).Include(a =>
a.MaxDays).Where(a => a.StartDate <= ld1 && a.EndDate >= ld1 && a.Guests
== guests && a.AdvertisementId == adv.AdvertisementId || a.StartDate <=
ld2 && a.EndDate >= ld2 && a.Guests == guests && a.AdvertisementId ==
adv.AdvertisementId);

                var days = 0;
                DateTime ld = ld2 - ld1;
                int allperiod = ld.TotalDays;
                float result = 0;
                foreach (var ap in aps)
                {
                    DateTime cld1 = adv.StartDate;
                    DateTime cld2 = adv.EndDate;
                    if (cld1 < ld1)
                    {

```

```

        cld1 = ld1;
    }
    if (cld2 > ld2)
    {
        cld2 = ld2;
    }
    cld1 = cld2 - cld1;
    int d = cld1.TotalDays;
    if (ap.MaxDays >= ld && ap.MinDays <=

ld)

    {
        days = days + d;
        result = result + d * ap.Price;
    }
}
if (days != ld)
{
    off = false;
}
result = result / days;
if (result > price2 || result < price1)
{
    off = false;
}
else
{
    prices.Add(Math.Round(result));
}
}
}
if (!off)
{
    advs.Remove(adv);
}

}
Object[] res = new Object[] {advs,prices};
if (advs == null)
{
    return NotFound();
}
return View(res);
}

```

```

        public async Task<IActionResult> AdvertisementPrice(int?
ad, int? d1, int? m1, int? y1, int? d2, int? m2, int? y2, int? guests)
        {
            DateTime ld1 = DateTime(y1, m1, d1);
            DateTime ld2 = DateTime(y2, m2, d2);
            var aps = await db.AdvertisementPrices.Include(a =>
a.Guests).Include(a => a.Price).Include(a => a.StartDate).Include(a =>
a.EndDate).Include(a => a.CancellationPolicy).Include(a =>
a.MinDays).Include(a => a.MaxDays).Where(a => a.StartDate <= ld1 &&
a.EndDate >= ld1 && a.Guests == guests && a.AdvertisementId == ad ||
a.StartDate <= ld2 && a.EndDate >= ld2 && a.Guests == guests &&
a.AdvertisementId == ad);
            var days = 0;
            var cp;
            DateTime ld = ld2 - ld1;
            int allperiod = ld.TotalDays;
            float result = 0;
            foreach (var ap in aps)
            {
                DateTime cld1 = ap.StartDate;
                DateTime cld2 = ap.EndDate;
                if (cld1 < ld1)
                {
                    cld1 = ld1;
                }
                if (cld2 > ld2)
                {
                    cld2 = ld2;
                }
                cld1 = cld2 - cld1;
                int d = cld1.TotalDays;
                if (ap.MaxDays >= ld && ap.MinDays <= ld)
                {
                    days = days + d;
                    result = result + d * ap.Price;
                    cp = ap.CancellationPolicy;
                }
            }
            if (days == ld)
            {
                result = Math.Round(result / days);
            }
            else

```

```

        {
            result = 0;
        }
        var adv = await db.Advertisements.Include(a =>
a.AdvertisementId).Include(a => a.Name).Include(a => a.Info).Include(a =>
a.Floor).Include(a => a.Floors).Include(a => a.RentalType).Include(a =>
a.Guests).Include(a => a.Rooms).Include(a => a.Beds).Include(a =>
a.BathroomType).Include(a => a.Photos).Include(a => a.Bedrooms).Include(a
=> a.Score).Include(a => a.CityId).Include(a =>
a.CheckinStartStr).Include(a => a.CheckinEndStr).Include(a =>
a.CheckoutStr).Include(a => a.AdvertisementUsers).Include(a =>
a.AdvertisementReviews).Include(a =>
a.AdvertisementOffers).FirstOrDefaultAsync(a => a.AdvertisementId == ad);
        Object[] res = new Object[] {adv,result, cp};
        if (res == 0)
        {
            return NotFound();
        }
        return View(res);
    }

    [Authorize]
    public IActionResult Create()
    {
        ViewData["Offers"] = new SelectList(db.Offers,
"OfferId", "Name");
        ViewData["Citys"] = new SelectList(db.Citys, "CityId",
"Name");

        return View();
    }

    [HttpPost] [ValidateAntiForgeryToken]
    public async Task<IActionResult>
Create([Bind("AdvertisementId, Name, Info, Floor, Floors, RentalType,
Guests, Rooms, Beds, BathroomType, Offers, Bedrooms, CityId,
CheckinStart, CheckinEnd, Checkout")] Advertisement adv)
    {
        if (ModelState.IsValid)
        {
            var user = await db.Users.FirstOrDefault(u =>
u.Login == User.Identity.Name);
            adv.RentalTypeStr = adv.RentalType;
            adv.BathroomTypeStr = adv.BathroomType;

```

```

        adv.Score = 0;
        db.Advertisements.Add(adv);
        AdvertisementUser au = new AdvertisementUser();
        au.UserId = user.UserId;
        au.AdvertisementId = adv.AdvertisementId;
        au.PresenceType = 1;
        db.AdvertisementUsers.Add(au);
        var adoffers = adv.Offers;
        foreach (var offer in adoffers)
        {
            AdvertisementOffer ao = new
AdvertisementOffer();
            ao.AdvertisementId = adv.AdvertisementId;
            ao.OfferId = offer.Id;
            ao.Presence = true;
            var o = db.Offers.Include(o =>
o.OfferId).Include(o => o.OfferType).FirstOrDefaultAsync (o => o.OfferId
== offer.Id);
            ao.OfferType = o.OfferType;
            db.AdvertisementOffers.Add(ao);
        }
        await db.SaveChangesAsync();
        return RedirectToAction(nameof(Index));
    }
    return View(adv);
}

[Authorize]
public async Task<IActionResult> Update(int? adv)
{
    var ad = await db.Advertisements.FindAsync(adv);
    if (ad == null)
    {
        return NotFound();
    }
    ViewData["Offers"] = new SelectList(db.Offers,
"OfferId", "Name");
    ViewData["Citys"] = new SelectList(db.Citys, "CityId",
"Name");
    return View(ad);
}

[HttpPost]

```

```

        [ValidateAntiForgeryToken]
        public async Task<IActionResult> Update(int adv,
[Bind("AdvertisementId, Name, Info, Floor, Floors, RentalType, Guests,
Rooms, Beds, BathroomType, Offers, Photos, Bedrooms, CityId,
CheckinStart, CheckinEnd, Checkout")] Advertisement adv)
        {
            if (ModelState.IsValid)
            {
                db.Advertisements.Update(adv);
                AdvertisementUser au = new AdvertisementUser();
                au.UserId = user.UserId;
                au.AdvertisementId = adv.AdvertisementId;
                au.PresenceType = 1;
                db.AdvertisementUsers.Update(au);
                var adoffers = adv.Offers;
                foreach (var offer in adoffers)
                {
                    AdvertisementOffer ao = new
AdvertisementOffer();
                    ao.AdvertisementId = adv.AdvertisementId;
                    ao.OfferId = offer.Id;
                    ao.Presence = true;
                    var o = db.Offers.Include(o =>
o.OfferId).Include(o => o.OfferType).FirstOrDefaultAsync (o => o.OfferId
== offer.Id);

                    ao.OfferType = o.OfferType;
                    db.AdvertisementOffers.Update(ao);
                }
                await db.SaveChangesAsync();
            }
            ViewData["Offers"] = new SelectList(db.Offers,
"OfferId", "Name");
            ViewData["Citys"] = new SelectList(db.Citys, "CityId",
"Name");
            return View(adv);
        }

        [Authorize]
        public async Task<IActionResult>
AdvertisementMonthPrices(int? adv, int? m1, int? y1, int? m2, int? y2)
        {
            DateTime ld1 = DateTime(y1, m1, 1);
            DateTime ld2 = DateTime(y2, m2, 1);

```

```
        var adps = await db.AdvertisementMonthPrices.Include(a
=> a.AdvertisementId).Include(a => a.Month).Include(a =>
a.Price).Include(a => a.State).Where(a => a.Month >= ld1 && a.Month <=
ld2).OrderBy(a => a.Month);
        return View(adps);
    }
}
}
```

ДОДАТОК В
Слайди презентації

Національний університет «Запорізька політехніка»
кафедра програмних засобів

Дипломна кваліфікаційна робота

**Дослідження та програмна реалізація
методів підтримки замовлення
короткострокової оренди житла**

Виконав:

студент групи КНТ-220м

А. О. Гусєв

Керівник:

к.т.н, доцент

С. Ю. Скрупський

2021

Рисунок В.1 – Слайд 1

2

Об'єкт дослідження – процес замовлення короткострокової оренди житла.

Предмет дослідження – методи підтримки замовлення короткострокової оренди житла.

Мета роботи – розробити програмне забезпечення замовлення короткострокової оренди житла для підтримки замовлень на основі результатів прогнозування вартості оренди житла в майбутньому періоді.

Рисунок В.2 – Слайд 2

Завдання роботи

3

- виконати аналіз програмних засобів підтримки замовлення короткострокової оренди житла;
- виконати проєктування програмного забезпечення;
- реалізувати програмне забезпечення;
- створити новий метод прогнозування вартості короткострокової оренди житла;
- виконати експериментальне дослідження використання розробленого методу для розв'язання проблеми прогнозування вартості короткострокової оренди житла.

Рисунок В.3 – Слайд 3

Порівняння програм-аналогів

4

Критерій	Airbnb	Booking.com	Розробка
Короткострокове бронювання	+	+	+
Відправлення повідомлень господарю	+	–	+
Відгуки про гостей	+	–	+
Прогнозування вартості оренди житла	–	–	+

Рисунок В.4 – Слайд 4

Діаграма варіантів використання

5

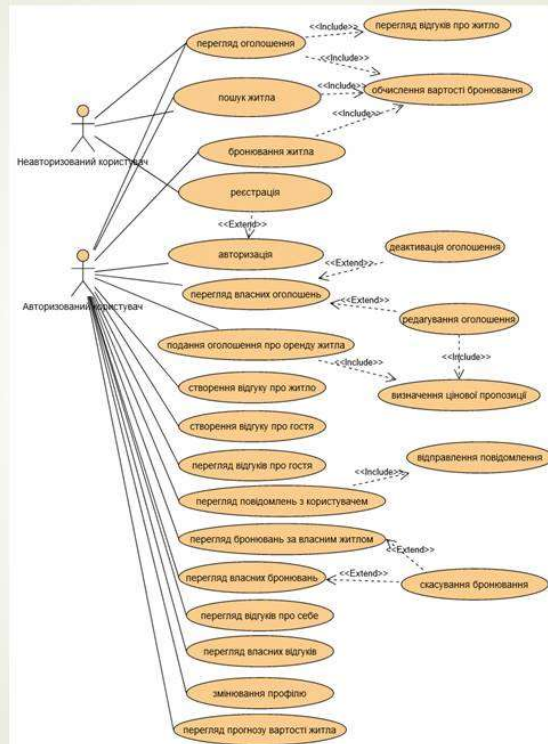


Рисунок В.5 – Слайд 5

Метод прогнозування вартості короткострокової оренди житла

6

Задача прогнозування вартості короткострокової оренди житла:

$$\langle V_{i+1}, \dots, V_{i+K} \rangle = f(V_i, \dots, V_{i-N}),$$

де V_i – середня вартість оренди житла на мінімальну одиницю часу за i -й період часу (періоди часу від i до $i - N$ належать до минулого, а періоди часу від $i + 1$ до $i + K$ належать до майбутнього);

f – функціональна залежність.

Перший етап. сформувати історичну вибірку даних, кожний елемент якої є часовим рядом, вхідними атрибутами якого є значення середньої добової вартості за $N + 1$ минулих місяців, а вихідними – значення середньої добової вартості за наступні K місяців.

Другий етап. Оброблення сформованої вибірки даних шляхом заміщення пропущених даних і нормалізації.

Третій етап. Створити модель нейронної мережі на основі LSTM, що складається з вхідного шару, двох прихованих LSTM-шарів, повнозв'язного та вихідного шару з K нейронами.

Четвертий етап. Реалізувати навчання створеної LSTM-моделі на основі вибірки даних, яка було визначено як результат виконання першого етапу.

П'ятий етап. Реалізувати прогнозування для нових даних.

Рисунок В.6 – Слайд 6

Вибір мов програмування			7
Критерій	PHP	C# ASP.NET	
Відкритість	+	+	
Легкість пошуку помилок	→+	→+	
Придатність до розширення (зокрема створення десктопних застосунків)	—	+	
Розробка застосунків корпоративного рівня	→+	++	
Розробка вебзастосунків	+	+	
Більший фокус створюваних рішень	Безпека та функціональність	Взаємодія з користувачами, користувацький інтерфейс	
Критерій	R	Python	
Відкритість	+	+	
Основне застосування	Аналіз даних та статистика	Розробка на основі даних	
Легкість початкового входження	—	+	
Інтеграція з вебзастосунками	—	+	

Рисунок В.7 – Слайд 7

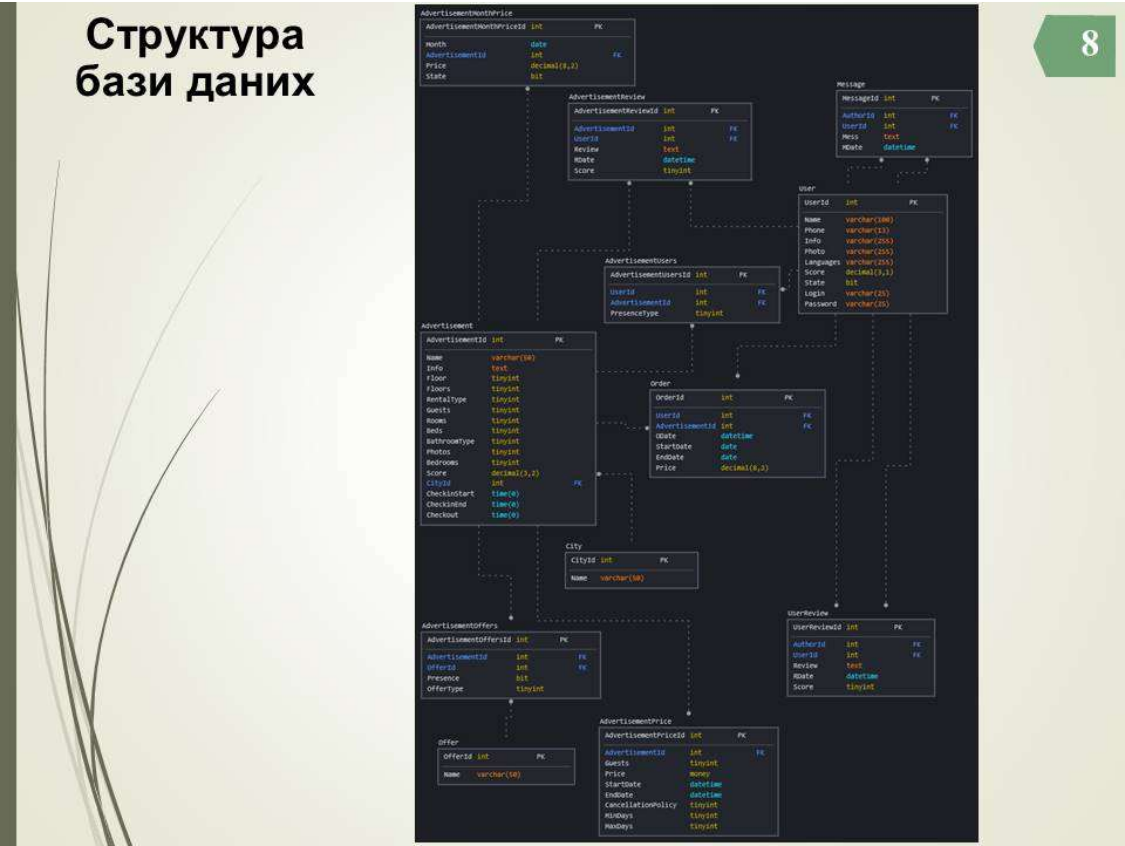


Рисунок В.8 – Слайд 8

Структурна схема програми

9



Рисунок В.9 – Слайд 9

Функціональна схема програмної системи

10

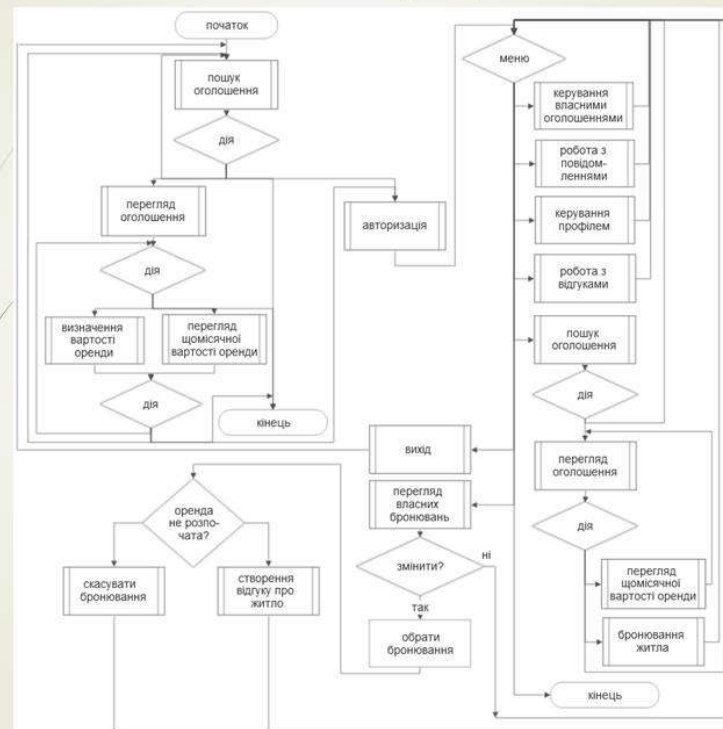


Рисунок В.10 – Слайд 10

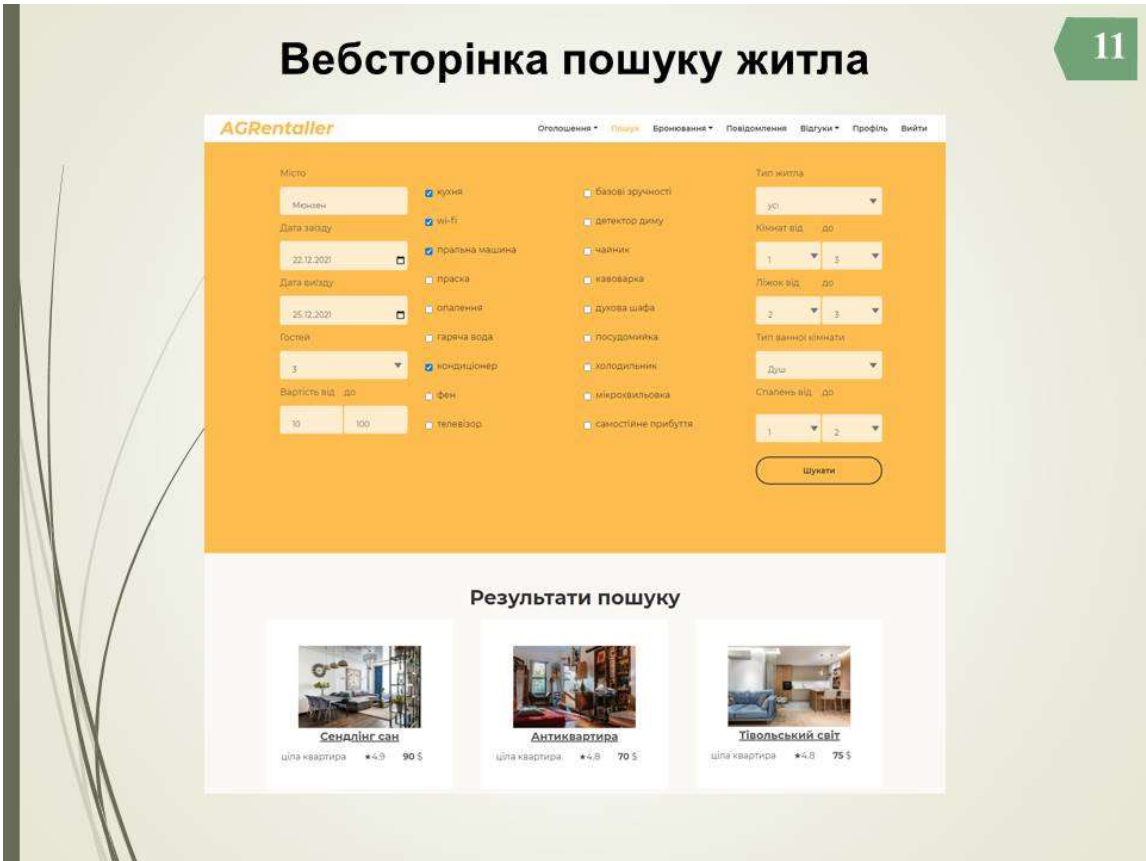


Рисунок В.11 – Слайд 11

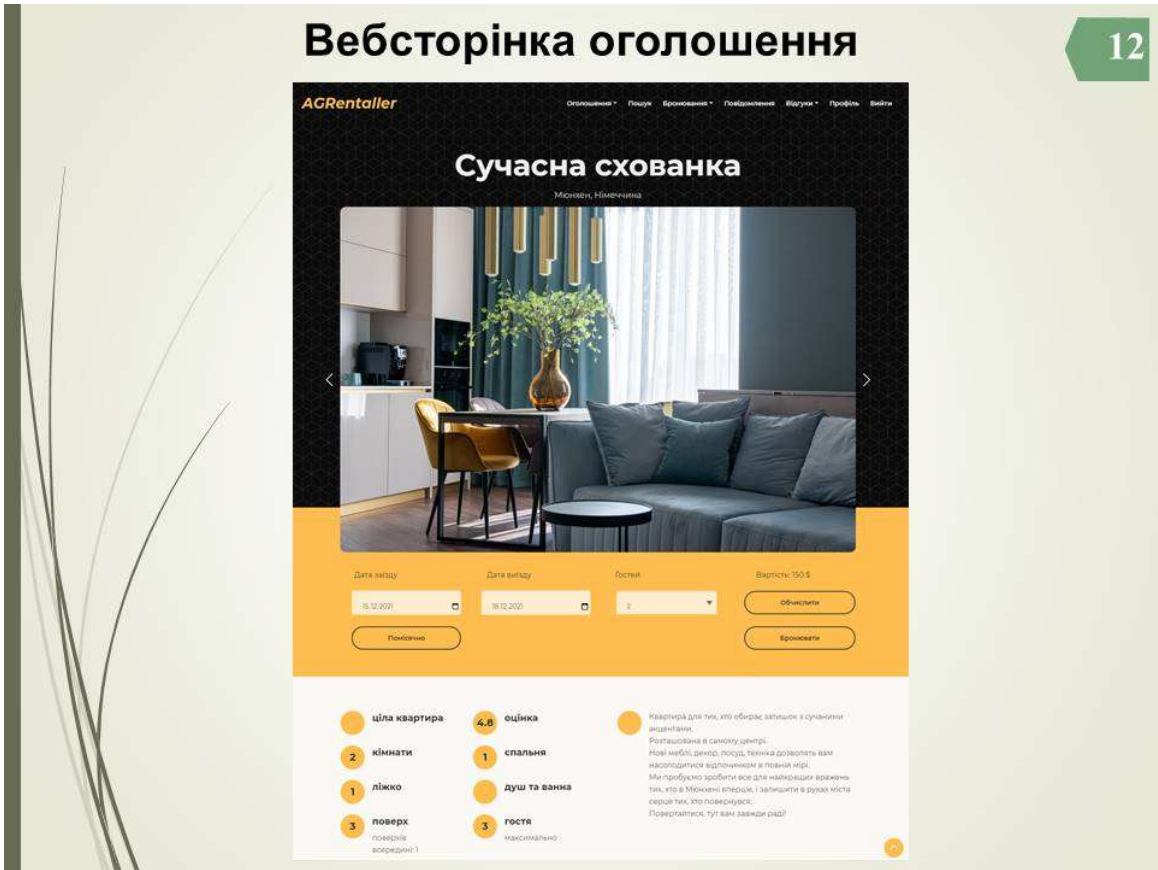


Рисунок В.12 – Слайд 12

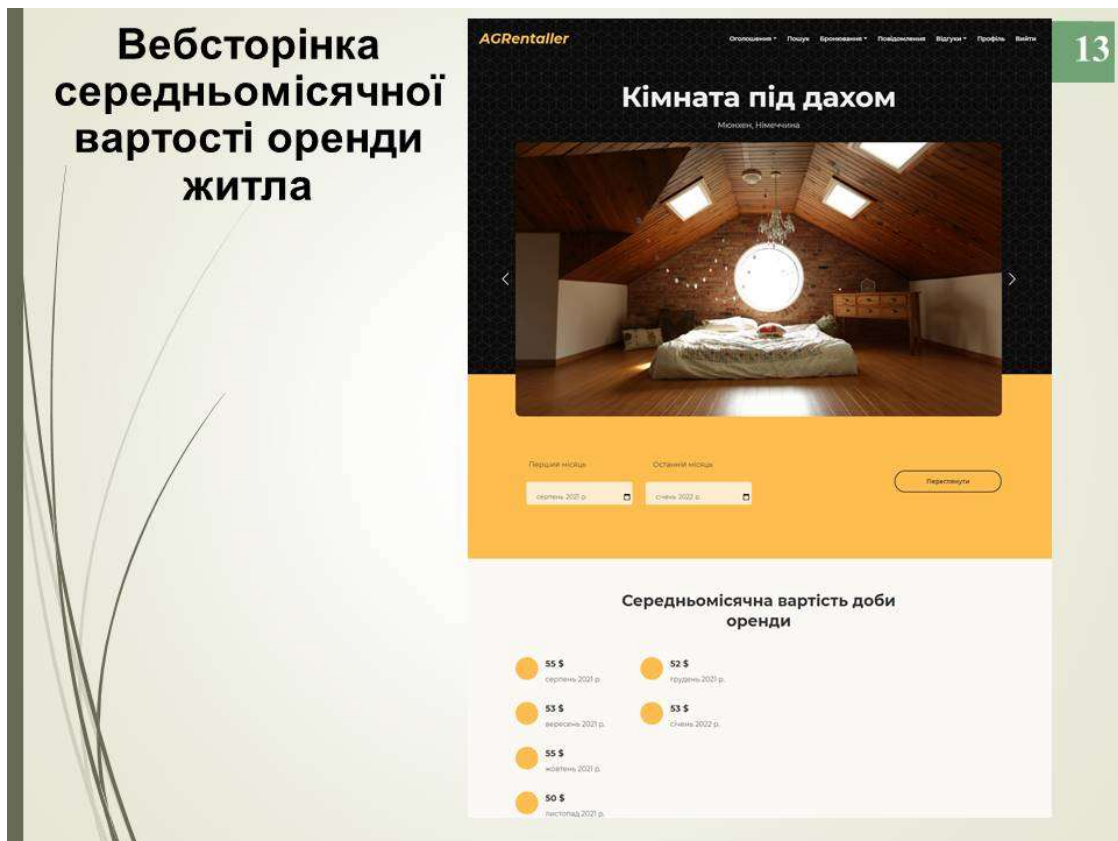


Рисунок В.13 – Слайд 13

Експериментальне дослідження

Метод / модель	Середньоквадратична похибка
Лінійна регресія	0,3164
Метод опорних векторів	0,2861
Нейронні мережі прямого поширення сигналу	0,1929
Метод прогнозування вартості короткострокової оренди житла	0,1285

Рисунок В.14 – Слайд 14

Висновки

15

Реалізовано програмне забезпечення у вигляді вебзастосунку, який дозволяє подавати оголошення про оренду житла, бронювати житло, виконувати його пошук, керувати власними бронюваннями і керувати господарю бронюваннями власного житла, створювати відгуки на житло і гостей, відправляти повідомлення від гостя до господаря і навпаки, переглядати дані про вартість житла в минулому та результати прогнозування вартості житла на майбутні 2 місяці.

Наукова новизна роботи полягає в запропонованому методі прогнозування вартості короткострокової оренди житла, який виконує прогноз на період з встановленим часовим інтервалом, використовуючи для цього штучну нейронну мережу на основі архітектури довгої короткострокової пам'яті, що дозволяє вибирати замовникам економічно обґрунтований момент для оформлення запланованого замовлення житла.

Рисунок В.15 – Слайд 15