Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра автоматизації та інтелектуальних інформаційних технологій

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Розробка сервісу пошуку житла для постраждалих громадян від російської агресії»

Виконав: студент 2-ого курсу, групи ЗАКІТ-22м
спеціальності 151 – Автоматизація та
комп'ютерно-інтегровані технології
_____ Віктор БІЛЕЦЬКИЙ
Керівник: к.т.н., доцент каф. АІІТ
_____ Володимир КОЦЮБИНСЬКИЙ
«4» грудня 2023 р.
Опонент: к.т.н., доцент каф. КСУ
_____ Микола БИКОВ
«8» грудня 2023 р.

Допущено до захисту
Зав. Кафедри АІІТ
д.т.н., проф.
Олег БІСІКАЛО
«11» грудня 2023 р.

Вінниця ВНТУ - 2023 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра автоматизації та інтелектуальних інформаційних технологій
Рівень вищої освіти ІІ-й (магістерський)
Галузь знань – 15 Автоматизація та приладобудування
Спеціальність 151 Автоматизація та комп'ютерно-інтегровані технології
Освітня програма – Інформаційні системи і Інтернет речей

ЗАТВЕРДЖУЮ
Завідувач кафедри АІІТ
д.т.н., проф. Олег БІСІКАЛО
«20» вересня 2023 року

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
Білецькому Віктору Володимировичу

1. Тема роботи: Розробка сервісу пошуку житла для постраждалих громадян від російської агресії
Керівник роботи к.т.н., доцент кафедри АІІТ Коцюбинський В.Ю. ,
Затверджені наказом ВНТУ від « 9 » грудня 2023 року № 247
2. Термін подання студентом роботи 5.12 2023 р.
3. Вихідні дані до роботи архітектура системи; дані про доступні пропозиції житла з інших сервісів;програмне середовище Visual Studio code; 8ГБ оперативної пам'яті; 512SSD;
4. Зміст текстової частини: вступ; аналіз предметної області існуючих сервісів пошуку житла; розроблення архітектури клієнт серверної системи та проектування сервісу пошуку житла; реалізація алгоритмічного та програмного забезпечення

автоматизованої системи підбору доступних пропозицій; висновки; пер використаних джерел.

5. Перелік ілюстративного матеріалу (з точним зазначенням обов'язкових креслень UML-діаграма варіантів використання, UML-діаграма діяльності,UML-діаг розгортання, архітектура системи, опис структури бази даних, тестування системи.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-3	К.т.н доцент кафедри АІІТ Володимир КОЦЮБИНСЬКИЙ	20.09.2023	05.12.2023

7. Дата видачі завдання 2023р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз потреб громадян та конкурентів	20.09-30.09	√
2	Обгрунтування методів реалізації та обробки	1.09-21.10	√
3	Розробка UML діаграм та дизайну проету	22.10-30.10	√
4	Розробка програмного забезпечення	31.10-15.11	√
5	Тестування та перевірка припущень	16.11- 01.11	√
6	Оформлення матеріалів до захисту МКР	02.12-12.12	√

Студент Віктор БІЛЕЦЬКИЙ

Керівник роботи Володимир КОЦЮБИНСЬКИЙ

АНОТАЦІЯ

УДК 004.45

Білецький В.В. Розробка сервісу пошуку житла для постраждалих громадян від російської агресії. Магістерська кваліфікаційна робота зі спеціальності 151 – Автоматизація та комп’ютерно-інтегровані технології, освітня програма – Інформаційні системи і Інтернет речей. Вінниця: ВНТУ, 2023. 89 с.

На укр.мові. Бібліогр.: 30 назв; рис.:33; табл.: 1.

Розроблюється інноваційний сервіс пошуку житла, спрямований на надання допомоги громадянам, що постраждали внаслідок російської агресії. Проект передбачає створення бази даних доступного житла, забезпечення зручних фільтрів пошуку, використання інтерактивної картографії для візуалізації об'єктів та встановлення системи зворотного зв'язку. Зокрема, враховуються важливі аспекти, такі як конфіденційність даних, співпраця з благодійними організаціями та кросплатформенність для максимальної доступності сервісу. Ця ініціатива спрямована на полегшення процесу переїзду та надання необхідної підтримки для постраждалих осіб.

Ключові слова: сервіс пошуку житла, постраждалі громадяни, російська агресія, розробка програмного забезпечення.

ANOTATION

Biletskiy V.V. Development of a housing search service for citizens affected by Russian aggression. Master's thesis on specialty 151 - Automation and computer-integrated technologies, educational program - Intelligent computer systems. Vinnytsia: VNTU, 2023. 86 p.

In the Ukrainian language. Bibliography: 30 titles; Fig.: 20; tab.: 11.

An innovative housing search service is being developed, aimed at providing assistance to citizens affected by Russian aggression. The project involves creating a database of affordable housing, providing convenient search filters, using interactive cartography to visualize objects, and establishing a feedback system. In particular, important aspects such as data privacy, cooperation with charitable organizations and the development of a mobile application for maximum accessibility of the service are taken into account. This initiative is aimed at facilitating the relocation process and providing the necessary support for affected persons.

Keywords: housing search service, affected citizens, Russian aggression.

ЗМІСТ

ВСТУП

Актуальність дослідження відзначаючи сталий вплив російської агресії на життя та благополуччя громадян, особливо тих, хто став жертвою конфлікту та втратив свої домівки, вступ до цієї магістерської роботи фокусується на необхідності та актуальності розробки сервісу пошуку житла. Цей сервіс має на меті відповідати конкретним потребам та викликам, які відображаються у постраждалих громадян внаслідок гуманітарної кризи.

Зважаючи на важливість забезпечення постраждалим не лише притулку, але й надійного та комфортного житла, розробка такого сервісу стає стратегічно важливою для полегшення процесу пошуку нового житлового простору. Робота спрямована на створення ефективного та інноваційного інструменту, який враховує спеціальні вимоги та умови, що виконуються у зв’язку з російською агресією.

У цьому вступі буде детально проаналізовано важливість розробки подібного сервісу, а також визначено основні виклики та завдання, які стоять перед дослідником у процесі вирішення цього актуального питання.

У країнах, які потерпають від російської агресії, ситуація гострих гуманітарних потреб та внутрішньо переміщених осіб стає все більш невтішною. Загроза фізичного поранення та стрес викликана внутрішньою житлом є серйозним викликом для постраждалих громадян, які втратили свої домівки та стали вимушеними переселенцями.

Розробка сервісу пошуку житла, спеціально адаптованого до умов гуманітарної кризи, має на меті забезпечити швидкий та ефективний механізм знаходження нового житла для тих, хто опинився в складних умовах конфлікту. Спрощення процесу пошуку та надання надійних інструментів вибору може вирішити практичні проблеми постраждалих та полегшити їхню реінтеграцію в новому середовищі.

Магістерська робота зосереджується на визначенні та розробці ключових функцій сервісу, що дозволяє надавати зручні та адаптовані рішення для потрібної групи населення, яка стала жертвою конфліктів. Очікується, що результати

дослідження та розробки дозволяють врахувати особливості гуманітарного контексту та надати практичний внесок у вирішення актуальних соціальних проблем.

Метою дослідження є розробка сервісу і створення ефективного інструменту для постраждалих громадян від російської агресії, який спростить та прискорить процес пошуку надійного житла. Сервіс націлений на надання зручного та адаптованого інтерфейсу для задоволення конкретних потреб груп населення, які втратили свої домівки через конфлікт, з метою полегшення їхнього переселення та реінтеграції.

Задачі досліджень магістерської кваліфікаційної роботи:

1. Огляд основних принципів функціонування систем пошуку житла з використанням принципів геолокації;

2. Аналіз існуючих на ринку систем пошуку житла;

3. Аналіз технологій та підходів для реалізації клієнтської та серверної частини системи пошуку житла;

4. Розробка алгоритмічного та програмного забезпечення для реалізації системи пошуку житла;

5. Тестування розробленого програмного забезпечення.

Об’єктом дослідження є процес обміну інформації між користувачами за допомогою існуючих систем пошуку житла на ринку.

Предметом дослідження є структура моделі комунікації між користувачами систем пошуку житла.

Методи дослідження. У роботі використано методи ідентифікації, збору і обробки даних, які отримуються від користувачів, та використання цих даних для покращення взаємодії між користувачами та підвищення достовірності інформації, яка публікується.

Новизна (Інноваційність) роботи вирізняється своєю новизною та інноваційністю через спеціалізовану адаптацію до умов гуманітарної кризи. Сервіс враховує унікальні потреби та виклики, які є у постраждалих, забезпечуючи

ефективний та зручний інструмент для пошуку житла, існуючого на ринку. Загалом, цей сервіс пропонує високотехнологічне та гуманітарно орієнтоване рішення для проблем пошуку житла в умовах гуманітарного кризового середовища.

Практична цінність полягає в розробленні алгоритмічного та програмного забезпечення сервісу для швидкого пошуку житла в умовах гуманітарної кризи. Він спрощує та прискорює процес вибору нового місця проживання, забезпечуючи безпеку, комфорт та індивідуальне врахування всіх побажань тимчасово переміщених осіб, які постійно ефективно реінтегруються в новому середовищі.

Апробація. Представлені в роботі результати апробовані в результаті участі в конференції «НАУКОВО-ТЕХНІЧНА КОНФЕРЕНЦІЯ ФАКУЛЬТЕТУ ІНТЕЛЕКТУАЛЬНИХ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА АВТОМАТИЗАЦІЇ»:

Публікації:Білецький В.В., Ковтун В.В. «Розробка сервісу пошуку житла для постраждалих громадян від російської агресії», «НАУКОВО-ТЕХНІЧНА КОНФЕРЕНЦІЯ ФАКУЛЬТЕТУ ІНТЕЛЕКТУАЛЬНИХ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА АВТОМАТИЗАЦІЇ», 2023. URL: – https://conferences.vntu.edu.ua/index.php/all-fksa/all-fksa-2023/index

1 АНАЛІЗ ІСНУЮЧИХ СИСТЕМ ПОШУКУ ЖИТЛА ТА ПРОБЛЕМИ РЕАЛІЗАЦІЇ ПРОЕКТУ

1.1 Аналіз сучасних систем в сфері пошуку житла в Україні

Україна, як країна, що зазнала впливу різкого соціального та політичного зриву, стикається зі зростанням потреби у надійних та ефективних рішеннях для пошуку житла, особливо для тих, хто став жертвою російської агресії. З урахуванням цього контексту аналіз конкурентів у сфері пошуку житла стає ключовим етапом розвитку нашого сервісу.

Сучасний ринок нерухомості в Україні включає різноманітні платформи та агентства, які пропонують різні підходи та послуги з пошуку та оренди житла. Аналізуючи їх підходи, технології та якість та ціни, звертаю увагу на переваги та недоліки конкурентів. Це надасть можливість нашому сервісу піднятися на новий рівень, пропонуючи інноваційні та конкурентоспроможні рішення для тих, хто потребує житла під час складних періодів [1].

Зростання технологічного розвитку також відзначається в сфері нерухомості, де онлайн-платформи та сервіси пропонують значущу роль у забезпеченні швидкого та зручного доступу до інформації про доступні житлові пропозиції. Аналіз конкурентів також дозволяє застосувати ті принципи та рішення, які користувачі вже відзначають як ефективні, а також визначити прогалини, які можна заповнити за допомогою новаторських підходів.

Україна має свої особливості та специфіку в контексті пошуку житла, і ретельний аналіз конкурентів враховує ці унікальні вимоги. Наша мета вбудована в створеному сервісі, який не відповідає сучасним стандартам, але враховує особливості української нерухомості та вигідно вирізняється на ринку існуючих конкурентів [2].

Безкоштовні сервіси пошуку житла можуть надавати зручні та доступні інструменти для ефективного знаходження пропозицій. Ось деякі способи, якими такий сервіс може здійснювати пошук та автоматизувати цей процес:

Фільтрація за Критеріями:

- Забезпечення можливості користувачам задавати конкретні критерії пошуку, такі як розташування, кількість кімнат, бюджет тощо.
- Використовує алгоритми фільтрації для автоматичного вибору пропозицій, які відповідають введеним критеріям.

Рекомендації та аналіз Повідомлення:

- Збір та аналіз інформації про вибір користувача та надання персоналізованих рекомендацій.
- Використання машинного навчання для автоматичного підбору пропозицій, які можуть зацікавити користувача.

Оповіщення та Сповіщення:

- Можливість досягти сповіщення про нові пропозиції, які відповідають критеріям користувача.
- Автоматизоване надсилання електронних листів або повідомлень для оперативного отримання інформації [8].

Картографічна інтеграція:

- Використання геолокації для точного визначення розташування пропозицій на карті.
- Можливість візуального огляду доступних об'єктів на карті з подальшим автоматичним оновленням результатів.

Аналітика та Звітність:

- Забезпечення користувачам аналізу ринку та динаміки цін на житло.

– Автоматичне формування звітів та статистики згідно із зазначеними критеріями.

Можливість Додавання Пропозицій:

– Функціонал для власників житла, який дозволяє додавати свої пропозиції до бази безкоштовного сервісу.

Шляхом інтеграції цих елементів та використання автоматизації, безкоштовні сервіси можуть значно полегшити процес пошуку житла для користувачів [9]. Також повинна відбуватись взаємодія з іншими сервісами, благодійними організаціями, безкоштовними дошками оголошень та недійними волонтерськими організаціями.

1.2 Дослідження науково-технічної задачі системи пошуку житла

Дослідження стану розв’язання науково-технічної задачі призведе до аналізу поточних підходів, методик та технологій, які використані в галузі, пов’язаних зі створенням сервісу пошуку житла для постраждалих громадян від російської агресії в Україні.

Це дослідження охоплює вивчення наукових публікацій, патентів, технічних звітів та інших джерел інформації для з’ясування існуючих розробок та інновацій у цій області. Також аналіз відомих вирішень, які вже застосовуються на ринку нерухомості та схожих галузях.

Методом цього дослідження є визначення того, які технології та підходи застосовуються конкурентами, які проблеми вони вирішують, а також виявлення можливостей для вдосконалення та впровадження новаторських рішень у наш проект. Дослідження стану галузі є ключовим етапом перед початком розробки, що дозволяє ефективно інтегрувати найбільш перспективні технології та методи в наш сервіс.Існують різні автоматизовані системи та застосунки для пошуку житла. Нижче

вказані різні приклади існуючих систем за типами послуг які вони надають своїм користувачам:

- сервіс світового пошуку житла:

Airbnb: Airbnb став популярною платформою завдяки своїм інноваціям та можливості надавати унікальні враження від проживання. Однак важливо бути важливим при виборі житла та отримати ризики, такі як безпека та конфлікти. Зростаюча конкуренція та вплив комісійних витрат також можуть вплинути на користувачів та власників нерухомості. Став популярною платформою завдяки своїм інноваціям та можливості надавати унікальні враження від проживання. Однак важливо бути важливим при виборі житла та отримати ризики, такі як безпека та конфлікти. Зростаюча конкуренція та вплив комісійних витрат також можуть вплинути на користувачів та власників нерухомості. Airbnb став популярною платформою завдяки своїм інноваціям та можливості надавати унікальні враження від проживання. Однак важливо бути важливим при виборі житла та отримати ризики, такі як безпека та конфлікти. Зростаюча конкуренція та вплив комісійних витрат також можуть вплинути на користувачів та власників нерухомості користувацький сервіс зображено на рисунку 1.1.

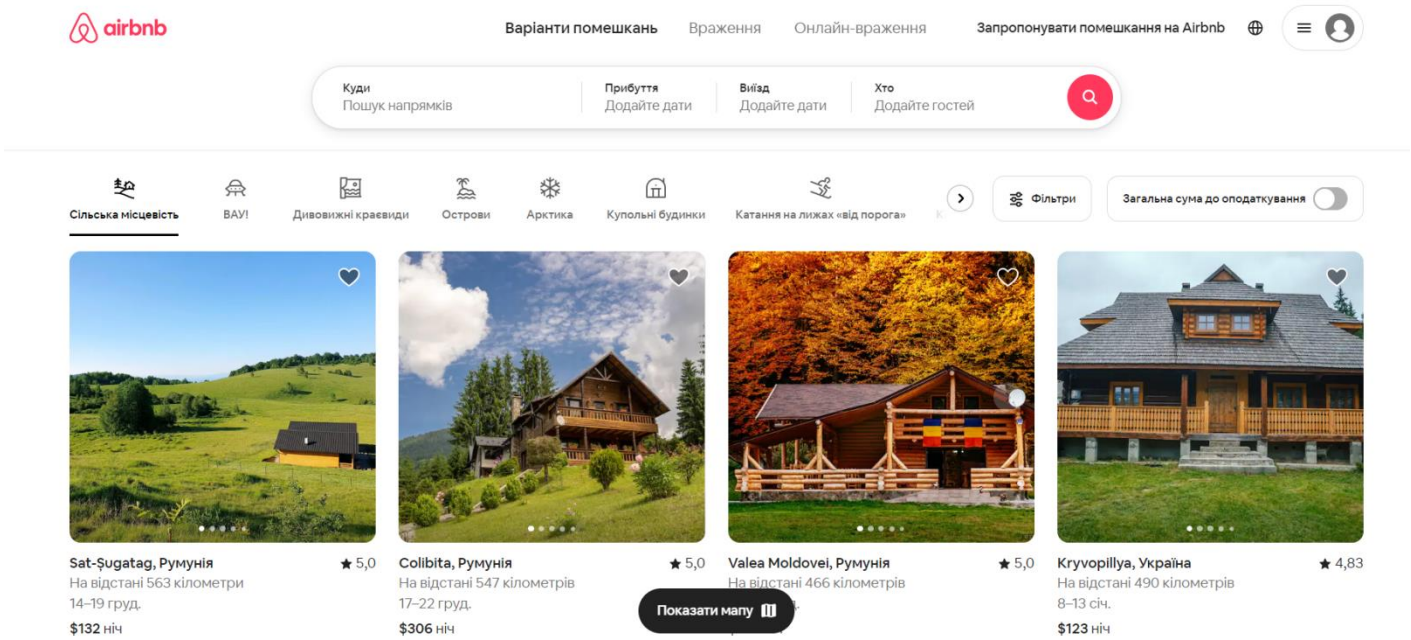

Рисунок 1.1 – Веб-застосування “ Airbnb ”

Переваги:

Широкий вибір варіантів: Airbnb пропонує величезний вибір житла в різних куточках світу, включаючи апартаменти, будинки, квартири та незвичайні об’єкти для проживання.

Локальний досвід: Користувачі можуть мати житло, яке відповідає їх уподобанням та дає можливість відчути себе як місцеве, отримуючи більш автентичний досвід.

Рецензії та рейтинги: Система відгуків та рейтингів дозволяє користувачам оцінювати якість та надійність власників нерухомості, що сприяє покращенню обслуговування та безпеки.

Різноманітні функції: Airbnb пропонує додаткові функції, такі як довгострокова оренда, Враження (Experiences), які розширюють спектр можливостей для користувачів.

Недоліки:

Проблем из безпекою: Іноді можуть виникати 15роблем из безпекою, особливо при оренді від недостатньо перевірених власників нерухомості.

Неоднакові стандарти: Якість та стандарти можуть змінюватися від одного об'єкта до іншого, після чого кожен власник може встановлювати свої умови.

Комісійні витрати: Airbnb встановлює комісійні витрати як для гостей, так і для власників, що може призвести до підвищення вартості оренди.

Проблем з відміною: Політика відміни може варіюватися, що може призвести до непорозумінь та незручностей для користувачів.

- сервіс пошуку житла Booking.com

Booking.com використовує середнє місце онлайн-платформи для бронювання за допомогою широкого вибору та зручного інтерфейсу. Проте важливо виконати те, що деякі готелі можуть віддавати перевагу прямим бронюванням через ваші власні веб-сайти, оминаючи комісії. Крім того, користувачам слід уважно читати умови бронювання та вивчати відгуки перед замовленням профіль за стосунку зображений на рисунку 1.2 [4].

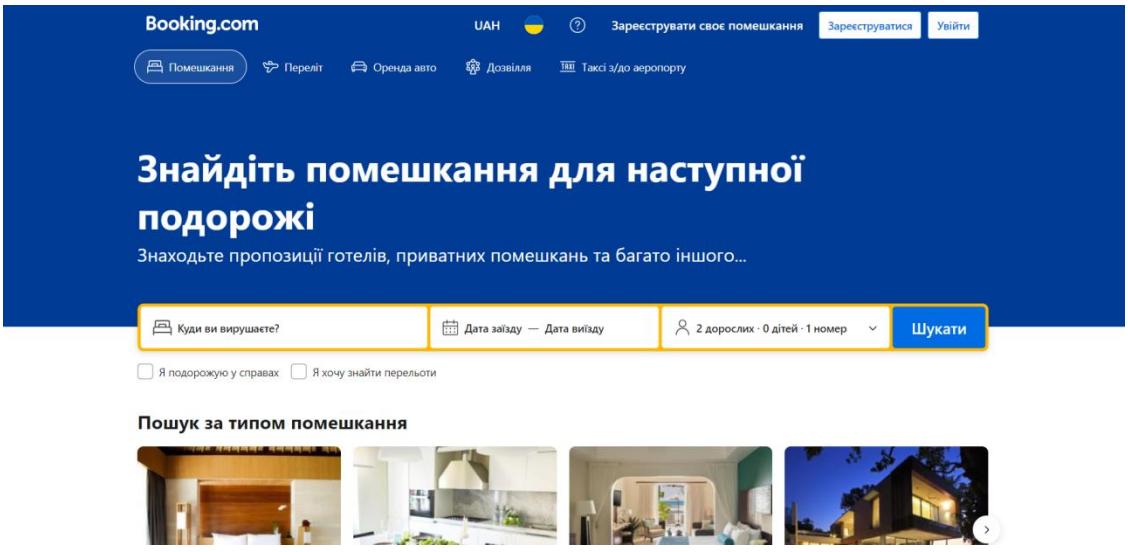

Рисунок 1.2 - Веб-застосунок “Booking.com”

Переваги:

Широкий вибір варіантів: Booking.com пропонує великий вибір готелів, апартаментів, гостьових будинків та інших варіантів проживання в різних країнах та містах.

Системних відгуків: користувачі можуть читати та залишати відгуки, які використовують обґрунтоване рішення та забезпечують якість обслуговування.

Програма лояльності: Booking.com пропонує програму лояльності зі знижками та бонусами для постійних користувачів.

Гнучкість умов бронювання: Багато варіантів пропонують гнучкі умови бронювання, які дозволяють змінювати або скасовувати бронювання без штрафів у певних випадках.

Недоліки:

Комісійні витрати: Власників часто платять високі готелі комісійні витрати Booking.com за кожне бронювання.

Неоднакові умови: Варіанти розміщення можуть мати різні умови, і це може створити невизначеність для користувачів.

Бронювання разом на сайті: У деяких випадках бронювання через сайт готелю може призвести до більших знижок чи додаткових переваг.

Допомагай: Допомагай надає рішення і послуги для громадян які шукають житло на території України, сервіс має здатність локально здійснювати пошук за регіоном його функціонал зображений на рисунку 1.3 [5].

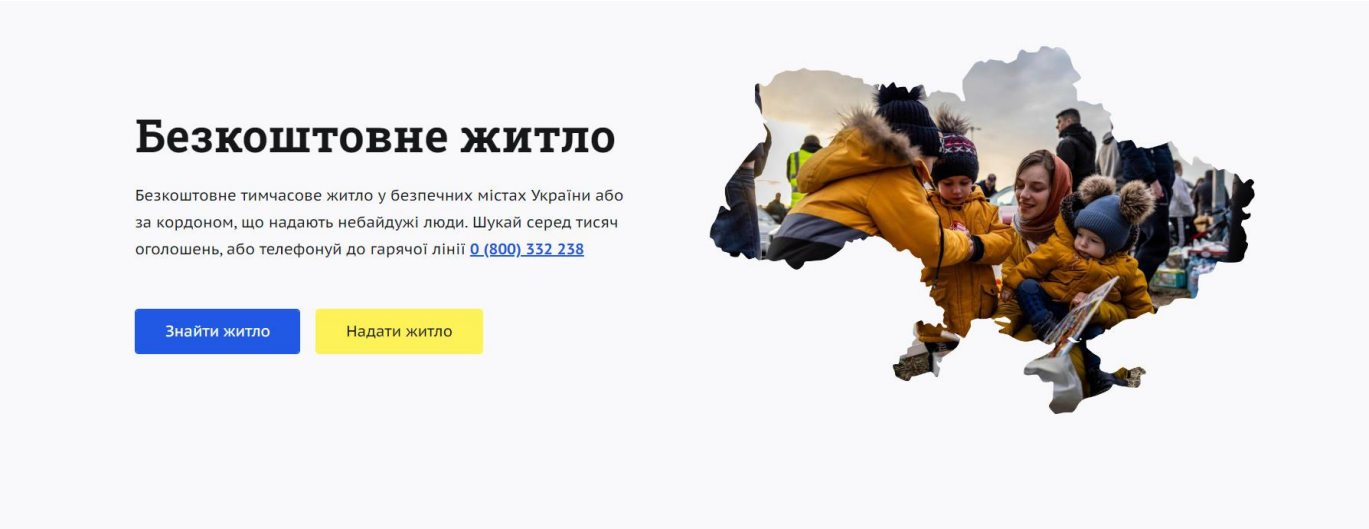

Рисунок 1.3 - Веб-застосунок “Допомагай”

Переваги:

Безкоштовність: Один з основних плюсів - використання таких сервісів є абсолютно безкоштовним для користувачів.

Широкий вибір: багато таких платформ мають велику кількість оголошень, що дозволяє користувачам вибирати серед різних варіантів житла.

Простота використання: звичайні, безкоштовні сервіси мають простий інтерфейс, що дозволяє користувачам легко шукати та переглядати доступні варіанти.

Недоліки:

Менша функціональність: на відміну від платних аналогів, безкоштовні сервіси можуть обмежувати функціональність, таку як фільтри, розширений пошук тощо.

Відсутність гарантій: Користувачам слід бути обережними, після чого платежі можуть також призвести до збільшення ризику виявлення реклам-шахрайств або недостовірної інформації.

Менший рівень підтримки: У безкоштовних службах може бути менше ресурсів для надання користувачам підтримки або розв'язання проблем.

Загально кажучи, безкоштовні сервіси пошуку житла можуть бути відмінним варіантом для тих, хто шукає доступні варіанти та готовий самостійно вирішувати питання з оренди житла.

1.3 Вибір і обґрунтування сучасних методів розв’язання науково-технічної задачі

Вибір і обґрунтування сучасних методів для розв’язання науково-технічної задачі фактично базується на специфіws цієї задачі, доступних ресурсах, актуальних технологічних тенденціях та інших факторах.

Детальний аналіз поставленої задачі з'ясувати її особливості, обмеження та ключові вимоги.

Визначення параметрів задачі, її складності та потреби у високоточних результатах.

Вивчення вже існуючих досліджень та методів, що використовують для таких завдань.

Виявлення сучасних тенденцій у відповідній галузі.

Розгляд актуальних технологій та інновацій, які можуть бути застосовані для вирішення завдань.

Оцінка можливостей нових методів та підходів.

Врахування доступних ресурсів, включаючи обладнання, програмне забезпечення та експертні знання.

Визначення обмежень часу, бюджету та інших факторів.

Експерименти та тестування:

Проведення експериментів для перевірки ефективності різних методів.

Тестування на реальних або симульованих даних.

Вибір або модифікація методів залежно від конкретних вимог та особливостей задачі.

Створення комбінації методів для досягнення оптимальних результатів.

Визначення критеріїв ефективності та встановлення метрики для оцінки досягнутих результатів.

Вибір сучасних методів має сприяти вирішенню завдань з використанням найбільш ефективних та інноваційних підходів. Фактори, такі як точність, швидкість, масштабованість та можливість взаємодії з існуючими технологіями, також можуть впливати на вибрані методи [6].

Створення сервісу пошуку житла для постраждалих громадян, які потребують швидкого та надійного доступу до тимчасового житла після російської агресії.

Методи та обґрунтування:

Геолокація та Картографія:

Обґрунтування: Використання геолокаційних технологій дозволяє користувачам швидко розмістити доступне житло в конкретних регіонах.

Методи: Інтеграція з геоспеціальними сервісами, використання API картографічних платформ.

Машинне Навчання для Рекомендацій:

Обґрунтування: Машинне навчання може аналізувати історію пошуків та пропозицій, щоб надавати персоналізовані рекомендації житла, враховуючи потреби кожного користувача.

Методи: Рекомендаційні системи, класифікація або кластеризація за допомогою алгоритмів машинного навчання.

Блок для безпеки та довіри:

Обґрунтування: Технологія блокчейн може забезпечити високий рівень безпеки, відстеження та довіри до транзакцій та даних користувачів.

Методи: Застосування смарт-контрактів для навчання угод та забезпечення їхньої надійності.

Інтерактивний Веб-Інтерфейс:

Обґрунтування: Сучасний та інтуїтивно зрозумілий інтерфейс дозволяє користувачам ефективно взаємодіяти з сервісом та здійснювати пошук зручно та швидко.

Методи: використання сучасних веб-технологій, анімації та реактивних фреймворків для покращення користувацького досвіду.

Інтеграція з Соціальними Мережами:

Обґрунтування: Взаємодія з соціальними мережами дозволяє втратити особливості вподобання та референції в процесі пошуку житла.

Методи: Інтеграція з API мережі, збір та аналіз соціальних даних.

набір методів, спрямованих на створення ефективного, безпечного та зручного сервісу пошуку житла для постраждалих громадян. Різноманітність застосованих методів дозволяє максимально підвищити якість обслуговування та забезпечити індивідуальні потреби користувачів [7].

1.4 Автоматизована система пошуку житла, актуальність проблеми

Автоматизована система пошуку житла представляє собою інноваційний підхід до пошуку та вибору нерухомості. Цей тип системи базується на використанні сучасних технологій та програмного забезпечення для автоматизації процесів перебування та вибору житла. Однією з ключових переваг є можливість швидко та ефективно фільтрувати та аналізувати великі обсяги інформації про доступні пропозиції на ринку. Ці системи пропонують різноманітні технології, такі як машинне навчання для надання персоналізованих рекомендацій, геолокація для точного визначення об’єктів нерухомості, а також блокчейн для забезпечення безпеки та довіри при угодах.

Автоматизовані системи пошуку житла дозволяють ефективно взаємодіяти з ринком нерухомості, забезпечуючи великий вибір варіантів та оптимізуючи процес прийняття рішень. Їхні можливості персоналізації, висока швидкість та використання сучасних технологій роблять їх актуальним та інструментом для споживачів нерухомості.

Автоматизована система пошуку житла базується на використання технологій та програмних засобів для швидкого та ефективного знаходження відповідних пропозицій з нерухомості.

Переваги:

- Швидкість та Ефективність: Автоматизація дозволяє швидко фільтрувати та порівнювати багато пропозицій, прискорюючи процес пошуку житла.
- Персоналізація рекомендацій: Система може контролювати індивідуальні вподобання та критерії користувачів для надання персоналізованих рекомендацій.
- Геолокація та Картографія: Інтеграція з картографічними сервісами дозволяє швидко локалізувати доступні об'єкти нерухомості в конкретних районах.
- Безпека та Довіра: Використання блокчейн-технологій може забезпечити безпеку та довіру при угодах та обмінах даними.

Недоліки:

- Залежність від Технічної Інфраструктури: Працездатність системи може бути обмежена технічними аспектами, такими як доступ до Інтернету та якості апаратного забезпечення.
- Відсутність Індивідуального Підходу: Алгоритми рекомендацій можуть бути обмежені при розпізнаванні унікальних потреб користувачів.
- Безпека даних: неможливо втратити питання конфіденційності та захисту особистих даних користувачів.

Актуальність: Розробка автоматизованої системи пошуку житла є актуальною через зростання запиту на зручні та ефективні інструменти для знаходження нерухомості. Застосування таких інноваційних технологій, як машинне навчання, геолокація та блокчейн, підтримує конкурентоспроможність таких систем на ринку нерухомості. Здатність швидко адаптуватися до змін у вимогах користувачів та ринкових тенденцій робить автоматизовані системи пошуку житла необхідним інструментом у сучасному світі нерухомості.

1.5 Висновки

У даному розділі було досліджено сучасні технології та реалізації систем пошуку житла. Висвітлено ключові аспекти області досліджень, що визначать основні напрямки які було реалізовано в наступних розділах роботи.

2 ВИБІР АРХІТЕКТУРИ ТА ТЕХНОЛОГІЇ ДЛЯ СТВОРЕННЯ СЕРВІСУ ПОШУКУ ЖИТЛА

2.1 Аналіз функцій системи

Сервіс пошуку житла, який буде розроблений, має включати в себе в себе наступні функції: реєстрація та авторизація на сайті, пошук житла за допомогою фільтру, бронювання та відміна бронювання житла, реалізація стороніх API, для підключення до інших сервісів та подальшої агрегації даних. Для візуалізації цих функцій було використано мову моделювання UML.

Мова моделювання UML (Unified Modeling Language) є стандартною мовою для візуального моделювання програмних систем. UML була розроблена для стандартизації та уніфікації процесу моделювання програмних систем і забезпечення чіткого засобу комунікації між розробниками та стейкхолдерами проекту. Основні елементи мови UML включають діаграми, які дозволяють відобразити різні аспекти системи:

- Діаграми класів (Class Diagrams): Ці діаграми використовуються для моделювання класів програми, їх атрибутів та методів. Вони показують структуру системи, включаючи спадкування (наслідування) між класами.
- Діаграми послідовності (Sequence Diagrams): Ці діаграми демонструють послідовність взаємодій між об'єктами системи в часі. Вони особливо корисні для моделювання взаємодій між різними об'єктами під час виконання програми.
- Діаграми активностей (Activity Diagrams): Вони використовуються для моделювання процесів, операцій і алгоритмів у системі. Діаграми активностей дозволяють показати послідовність дій та умовні розвітні пункти.
- Діаграми станів (State Diagrams): Вони дозволяють моделювати стани

об'єкта та переходи між ними відповідно до певних подій або умов. Вони корисні для моделювання поведінки системи, яка може знаходитися у різних станах.

- Діаграми співробітництва (Collaboration Diagrams): Вони показують взаємодію об'єктів та відносини між ними. Діаграми співробітництва допомагають відобразити, як об'єкти співпрацюють один з одним.
- Діаграми компонентів (Component Diagrams): Вони моделюють структуру компонентів системи та зв'язки між ними. Це корисний інструмент для проектування розподілених систем.
- Діаграми розгортання (Deployment Diagrams): Вони використовуються для моделювання фізичної структури системи, такої як сервери, обладнання та мережі.
- Діаграми прецедентів (Use Case Diagrams): Вони допомагають моделювати функціональність системи з точки зору користувача. Вони визначають ролі користувачів та їх взаємодії з системою.

Усі ці діаграми спільно створюють повний огляд програмної системи, дозволяючи розробникам, архітекторам і стейкхолдерам краще розуміти, проектувати та спілкуватися щодо системи. UML надає стандартні символи та правила для створення цих діаграм, що робить їх зрозумілими та стандартизованими для всіх учасників проекту. [9].

У цьому пункті описано існуючі упродукті діаграми варіантів використання, діяльності та розгортання, для більш детального відображення всього процесу взаємодії користувача автовокзалу із системою автовокзалу.

Діаграма варіантів використання (Use Case Diagram) - це графічний інструмент для моделювання функціональних можливостей і взаємодії користувачів (акторів) з системою. Вона дозволяє визначити основні функції системи та сценарії взаємодії між користувачами та системою. Елементи діаграми варіантів використання:

- Актори (Actors): Актори - це зовнішні сутності, які взаємодіють з системою. Вони можуть бути реальними користувачами системи або іншими системами, які взаємодіють з досліджуваною системою.
- Варіанти використання (Use Cases): Варіанти використання - це функціональні можливості системи, які можуть бути виконані акторами. Кожен варіант використання описує конкретний сценарій взаємодії між акторами та системою. Вони представлені у великих овалах на діаграмі.
- Відносини між акторами та варіантами використання: Відносини показують, які актори беруть участь у виконанні конкретного варіанту використання. Зазвичай використовуються наступні відносини:
- Включає (Includes): Вказує, що один варіант використання може включати інший.
- Розширює (Extends): Вказує, що один варіант використання може розширювати інший. [33]

Мета цієї діаграми полягає у тому, щоб визначити, як користувачі та інші системи (актори) будуть взаємодіяти з проектованою системою. Ця взаємодія описується через набір сценаріїв, відомих як "варіанти використання." Кожен варіант використання конкретизує, які дії виконує система під час спілкування з акторами.

На рисунку 2.1 наведена діаграма варіантів використання.

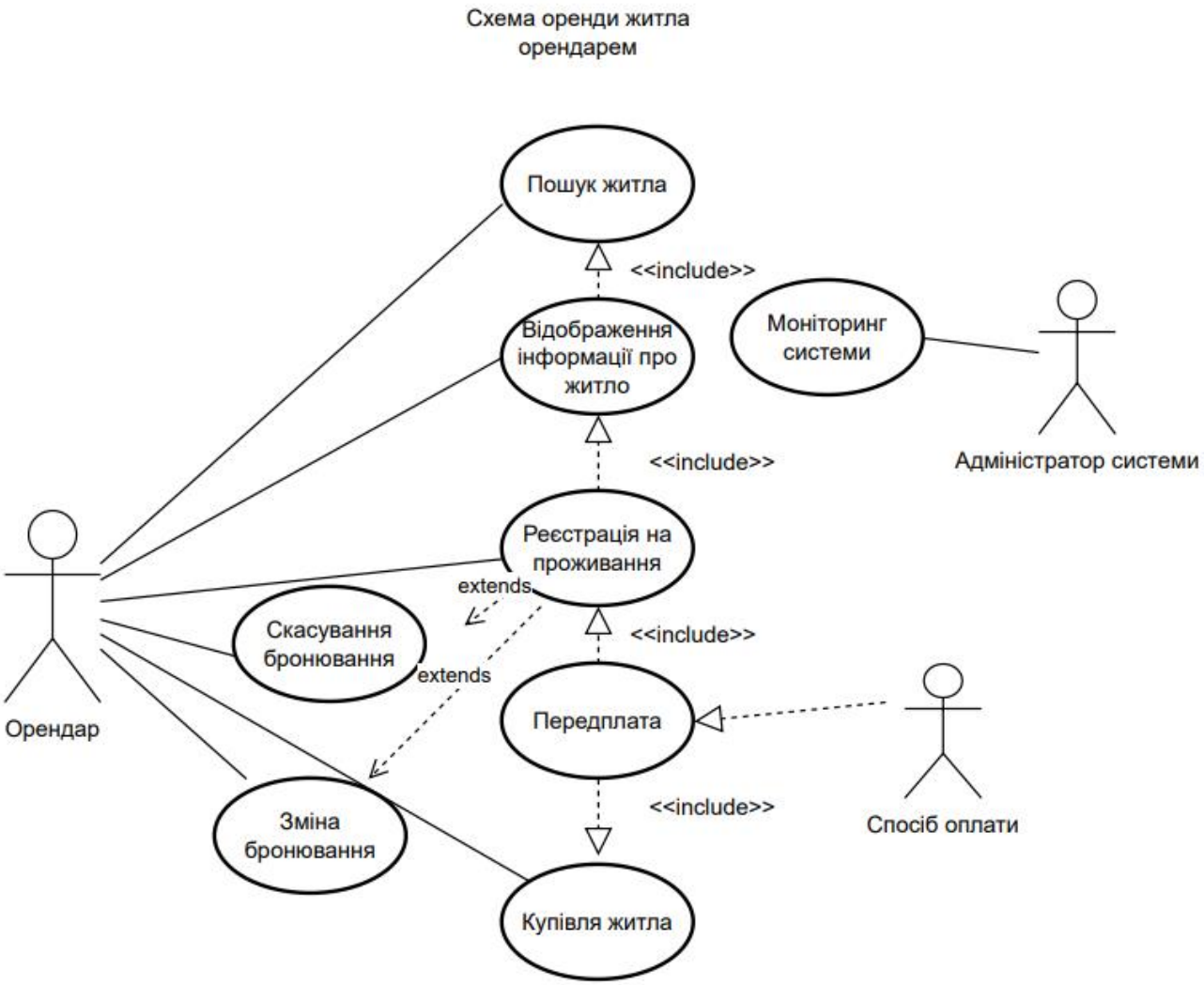

Рисунок 2.1- UML-діаграма варіантів використання

Опис взаємодії акторів із системою пошуку житла більш детально описано далі: Клієнт (Орендар) спочатку повинен зареєструватися на сайті, після чого він отримає доступ до функціоналу системи та зможе переглянути інформацію про житло та доступні варіанти за місцем проживання, або по Україні, після чого за допомогою фільтру обрати потрібний варіант із сформованого списку та забронювати його, у разі виникнення ситуації коли користувачу більше не знадобиться дане житло він зможе скасувати бронювання на нього, також у системі буде передбачено можливість оплати,

шляхом інтеграції із системою оплати тому в даній схемі вона виступає у вигляді окремого актора. У адміністратора системи є окремий вхід до системи, який дає йому можливість управляти всією системою, в свою чергу, менеджер системи має можливість моніторити бронювання житла та їх кількість які залишилися в окремому регіоні.

2.2 Проєктування архітектури системи

Архітектура системи для сервісу пошуку житла може бути побудована з урахуванням кількох ключових компонентів та рівнів. Враховуючи обсяг та склад завдань, описаних раніше, розглянемо архітектурний підхід, що включає кілька рівнів бізнес-логіки.

- Клієнтський рівень(Веб-інтерфейс):
- Веб-інтерфейс: Веб-додаток для користувачів, який дозволяє переглядати розклади рейсів, робити онлайн-платежі та отримувати інформацію про маршрути та послуги.
- Серверний рівень(Серверні застосунки):
- Серверні застосунки: Реалізація бізнес-логіки на сервері для обробки запитів від клієнтського додатку. Це може бути створене з використанням мови програмування C# та фреймворка ASP.NET.Серверні застосунки виконують бізнес-логіку, обробку запитів від клієнтського додатку та взаємодію з базою даних. Вони забезпечують обробку бронювання, планування рейсів тощо.
- Бізнес-логіка:

Рівень бізнес-логіки розділяє функціональність системи на логічні модулі, що полегшує розробку, тестування і супровід системи. Він включає в себе наступні модулі:

- керування рейсами та маршрутами: Модуль для планування, керування та моніторингу заявок житла.

- кронювання житла: Обробка запитів щодо бронювання житла.
- керування орендарями: Модуль для обліку та обслуговування клієнтів.
- Система управління базою даних (СУБД):
- Сховище даних: База даних є центральним сховищем інформації про позиції, регіон, орендарів та інші дані. Можливе використання СУБД, такої як MicrosoftSQLServer, дозволить забезпечити безпеку, цілісність та продуктивність бази даних.
- Можлива інтеграція з іншими системами(Платіжні системи, Моніторинг транспорту):
- Платіжні системи: Інтеграція з платіжними системами для оплати броні дозволить обробляти онлайн-платежі в системі.
- Забезпечення доступу до системи:
- Авторизація: Авторизація дозволяє контролювати доступ користувачів до різних функцій системи.
- Інфраструктура:

Хмарна архітектура: У подальшому як варіант можна розглянути хмарний хостинг для веб-додатку що забезпечить масштабованість та доступність.

Такий рівневий підхід дозволить розділити функціональність системи на логічні компоненти, полегшуючи розробку, супровід та масштабування. Архітектура системи описаної вище зображена на рисунку 2.3.

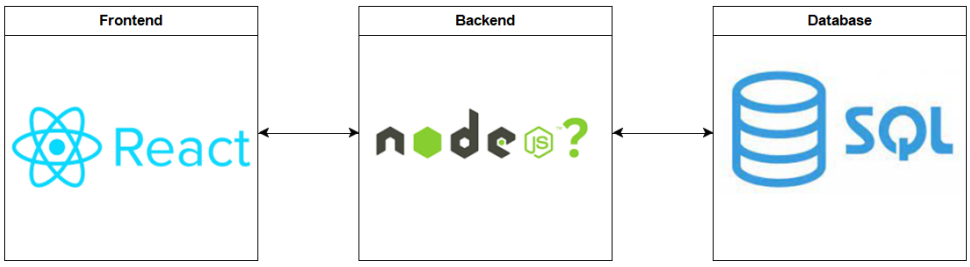

Рисунок 2.3 – Архітектура системи

У графічній формі цю архітектуру описує діаграма розгортання на рисунку 2.4.

Діаграма розгортання (DeploymentDiagram) в UML - це графічний інструмент, який використовується для моделювання фізичного розташування компонентів системи та їх взаємозв'язків на обладнанні (апаратурі) або в середовищі виконання (наприклад, серверах або віртуальних машинах). Діаграма розгортання дозволяє візуалізувати архітектурну конфігурацію системи та її фізичну структуру.Основні елементи діаграми розгортання включають в себе:

- Вузли (Nodes): Вони представляють фізичні або логічні вузли, де розташовані компоненти системи. Вузли можуть бути обчислювальними серверами, мережевими пристроями, апаратурою, віртуальними машинами тощо.
- Артефакти (Artifacts): Артефакти представляють файли, бібліотеки, програми тощо, які зберігаються внутрішньо в вузлах.
- Зв'язки (Connections): Зв'язки між вузлами показують, як компоненти системи спілкуються один з одним. Може використовуватися різний тип зв'язків, такий як асоціації, зв'язки залежності, агрегації тощо.
- Специфікації (Specifications): Для кожного артефакту можна вказати його специфікацію, тобто деталізований опис, що міститься в артефакті.
- Пакети (Packages): Групи компонентів або артефактів, які можуть бути об'єднані в пакети для кращої структуризації діаграми.

Діаграма розгортання допомагає розуміти, як фізичні компоненти системи розташовані та взаємодіють один з одним, що є важливим при проектуванні та реалізації системи [15].

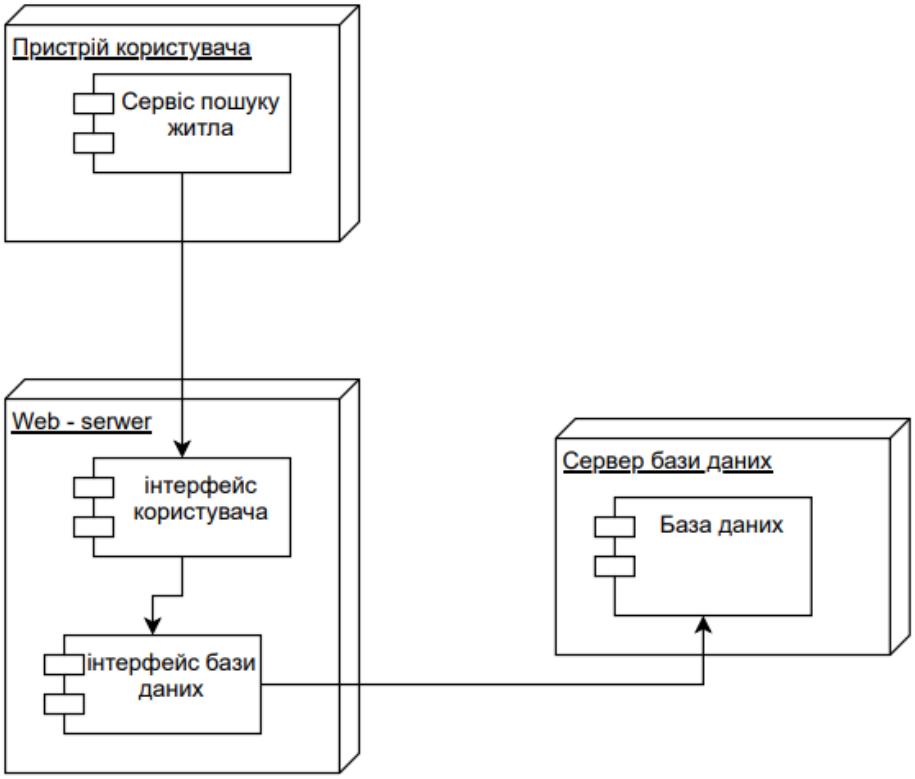

Рисунок 2.4- UML-діаграма розгортання

Діаграма розгортання показує обчислювальні вузли під час роботи програми, компоненти, та об'єкти, що виконуються на цих вузлах.

Діаграма діяльності (Activity Diagram) - це графічний інструмент для моделювання послідовності дій і процесів в системі або бізнес-процесі. Вона дозволяє візуалізувати, як окремі дії взаємодіють одна з одною та керують потоком даних. Основні елементи діаграми діяльності включають дії, функціональні блоки, рішення, паралельність та інші компоненти, які допомагають моделювати процеси та послідовності в системі. Діаграми діяльності використовуються для моделювання бізнес-процесів, проектування програмного забезпечення, аналізу та вдосконалення процесів, а також для документування та спілкування між учасниками проекту. Елементи діаграми діяльності:

- Дія (Action): Дії представлені у вигляді прямокутників і вказують на конкретні дії або обчислення, які відбуваються в системі. Наприклад, "Запит до бази даних" або "Вивід на екран".
- Функціональні блоки (Control Flows): Функціональні блоки - це стрілки, які показують потік виконання між діями. Вони з'єднують дії в послідовний порядок та вказують напрямок потоку виконання.
- Рішення (Decision): Рішення, або відсічки, використовуються для представлення умовних рішень в процесі. Вони визначають, які шляхи будуть обрані в залежності від певних умов.
- Паралельність (Fork/Join): Паралельність використовується для показу паралельних процесів, які відбуваються одночасно. Вона вказує, коли процеси розгалужуються (Fork) і коли об'єднуються (Join).
- Початок і кінець (Start/End): Позначення "Початок" і "Кінець" вказують на початок і завершення виконання діаграми. Вони є обов'язковими елементами [16].

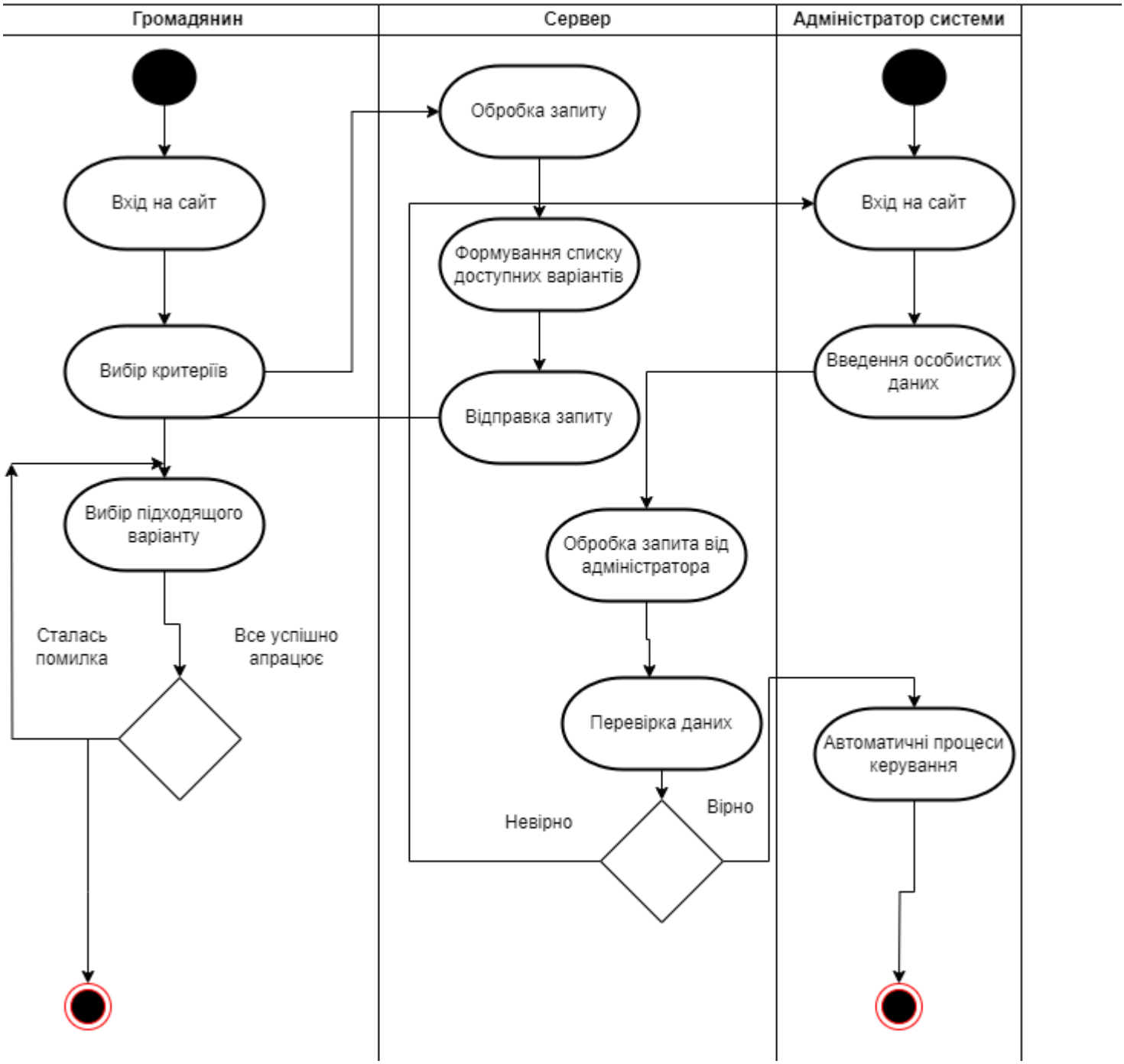

Рисунок 2.5- UML-діаграма діяльності

Діаграмі діяльності відображена на рисунку 2.5 пояснює роботу процесу пошуку клієнтом потрібного житла. Вона описує взаємодію між користувачем, адміністратором системи, та сервером за допомогою елементів діаграми діяльності.

2.3 Аналіз засобів створення програмного забезпечення для системи пошуку житла

Аналіз засобів створення програмного забезпечення для систем пошуку житла передбачає врахування різних аспектів, таких як мови програмування, фреймворки, бази даних, інструменти розробки та інші технології. Мова Програмування:

Використовуються популярні мови програмування, такі як JavaScript (зокрема Node.js для серверної частини та React або Angular для клієнтської частини), Python, Ruby або Java.

Вибір мови враховуючи зручність розробки, швидкість виконання та екосистему інструментів.

Фреймворки та Бібліотеки:

Використовуйте фреймворки для швидкої розробки, наприклад, Express або Django для серверної частини та React або Vue.js для клієнтської частини.

Застосування бібліотек для розв'язання конкретних задач, наприклад, Axios для HTTP-запитів або Leaflet для відображення карт.

Бази даних: Вибір баз даних залежно від необхідних систем, наприклад, MySQL, PostgreSQL або MongoDB.

Використовуйте ORM (Object-Relational Mapping) для спрощення взаємодії з базою даних.

Інструменти Розробки:

Використання інтегрованих середовищ розробок (IDE), таких як Visual Studio Code або PyCharm, для підвищення продуктивності розробників.

Використовується система контролю версій, наприклад, Git, для ефективного керування кодом.

Хмарні Послуги:

Застосування хмарних платформ, таких як AWS, Azure або Google Cloud, для забезпечення масштабованості, надійності та безпеки систем.

Автоматизація тестування та CI/CD:

Використання фреймворків для автоматизації тестування, таких як Jest або Selenium.

Налаштування системи Continuous Integration/Continuous Deployment (CI/CD) для автоматичної розгортки та оновлення програмного забезпечення.

Безпека:

Застосування кращих практичних заходів безпеки, включаючи валідацію введених даних, використання HTTPS, обробку аутентифікації та авторизації.

Мобільний доступ:

Врахування можливостей створення мобільних додатків за допомогою фреймворків, таких як React Native або Flutter.

Аналіз цих аспектів допоможе забезпечити високу якість та ефективність розробленого програмного забезпечення для систем пошуку [17].

JavaScript (JS) є мовою програмування, яка широко використовується для розробки веб-додатків, і вона має власні переваги та недоліки. Ось деякі з них та пояснення того, чому JS може бути ефективним вибором для розробки проекту пошуку житла:

Переваги:

Універсальність: JS підтримує браузери, що робить його універсальною мовою для розробки клієнтської частини веб-додатків. Також, завдяки платформі Node.js, він може бути використаний для розробки серверної частини.

Асинхронність: JS має асинхронну природу, що сприяє ефективній обробці багатопоточних операцій, що є числом для веб-додатків, які мають велику кількість одночасних запитів.

Велика Спільнота та Екосистема: JS має велику та активну спільноту розробників, а також розширену екосистему бібліотек і фреймворків, які полегшують розробку.

Швидкість розробки: динамічна типізація та можливість використання модернізованих фреймворків (наприклад, React або Angular) можуть швидко розробити ефективні та інтерактивні інтерфейси.

Реактивний підхід: Більшість фреймворків та бібліотек JS підтримують реактивний підхід, що дозволяє легко оновлювати інтерфейс у зв’язку зі змінами даних [18].

Недоліки:

Відсутність вбудованих засобів для роботи з БД: у порівнянні з іншими мовами JS має менше вбудованих засобів для роботи з базами даних. Це можна привести до використання сторонніх бібліотек або ORM.

Проблеми з безпекою: JavaScript використовується на клієнтській стороні, що може привести до питань з безпекою, особливо, якщо не вживаються заходи для захисту.

Різні Підходи до Програмування: JavaScript підтримує різні стандарти та підходи до програмування, що можна призвести до неоднорідності в коді, особливо у великих проектах [19].

JavaScript є ідеальним вибором для веб-додатків, включаючи систему пошуку житла, завдяки її універсальності, асинхронності та широкому спектру інструментів і бібліотек. Він надає можливість розробити як клієнту, так і серверну частину за допомогою однієї мови, що полегшує синхронізацію та розробку. Враховуючи активну спільноту та екосистему, можна швидко реалізувати функціональність і підтримувати проект протягом тривалого часу.

Python - це високорівнева, інтерпретована мова програмування, яка також має свої унікальні переваги та недоліки.

Переваги: Простота та читабельність коду: Синтаксис Python дуже простий та схожий на англійську мову, що полегшує розуміння та читабельність коду.

Багатофункціональність: Python підтримує різноманітні парадигми програмування, включаючи об'єктно-орієнтоване, функціональне та імперативне програмування.

Широке застосування: використання Python у різних сферах, включаючи веб-розробку, наукові дослідження, штучний інтелект, обробку даних, аналіз, ігрову розробку та багато іншого.

Велика Спільнота та Екосистема: Python має активну та дружню спільноту, а також широкий вибір бібліотек і фреймворків, що робить його потужним для різних проектів.

Швидкість розробки: Python дозволяє розробникам швидко створювати прототипи та реалізовувати функціональність за короткий час.

Недоліки: Швидкодія: У порівнянні з іншими мовами, такими як C++ або Java, Python може бути менш швидким через інтерпретацію.

Обмежена підтримка для розробки мобільних додатків: Python не є найкращим вибором для розробки мобільних додатків, хоча деякі фреймворки, такі як Kivy.

Глобальний інтерпретатор GIL: Global Interpreter Lock (GIL) обмежує використання багатьох процесів у деяких сценаріях.

Python також може бути відмінним вибором для розробки системи пошуку житла через свою простоту, гнучкість та велику спільноту. Завдяки різноманітним бібліотекам для обробки даних, штучного інтелекту та веб-розробки, Python дозволяє створювати потужні та ефективні рішення. Однак вибір між JavaScript і Python залежить від конкретних потреб проекту, вимог до швидкості та факторів інших.

2.4 Вибір засобів розробки системи пошуку житла

Вибір засобів розробки системи пошуку житла може бути обґрунтований наступним чином:

Мова програмування: JavaScript (JS) та React

Обґрунтування: JS є однією з найпоширеніших мов програмування, особливо в області веб-розробки. React, у своїй сторінці, є потужним фреймворком для створення інтерфейсів користувача. Вони забезпечать ефективну розробку високопродуктивних та динамічних веб-застосунків.

Середовище розробки: Visual Studio Code (VS Code)

Обґрунтування: VS Code є популярним та легким середовищем розробки, яке має велику кількість розширень для роботи з ефективними мовами програмування та технологій. Воно надає широкі можливості для автоматизації, відлагодження та інтеграції зі всіма інструментами.

База даних: Microsoft SQL Server

Обґрунтування: Microsoft SQL Server є потужною та надійною реляційною системою управління базами даних (RDBMS). Вона забезпечить стабільність, безпеку даних та ефективний доступ до інформації.

Фреймворки для розробки серверної частини: Node.js (з використанням Express або Koa)

Обґрунтування: Node.js дозволяє використовувати JS на серверній боці, забезпечуючи єдність мови програмування. Express або Koa є популярними фреймворками для створення веб-серверів та API.

Бібліотека для взаємодії з сервером: Axios

Обґрунтування: Axios є простою та ефективною бібліотекою для виконання HTTP-запитів, що дозволяє легко взаємодіяти з сервером.

Фреймворк для тестування: Jest

Обґрунтування: Є потужний фреймворк для тестування на мові JavaScript, включаючи автоматичні тести та тести одиниць. Він спрощує процес тестування та надає велику функціональність.

Ці засоби і технології дозволяють створити ефективну та масштабовану систему пошуку житла, забезпечуючи швидкість розробки, стабільність та високу продуктивність.

2.5 Вибір мови програмування

Вибір мови програмування та фреймворка для створення сервісу пошуку житла є важливим рішенням і залежить від ряду факторів. У цьому випадку обрано JavaScript (JS) і фреймворк React з такими основними перевагами:

- Веб-Орієнтовані Технології:
- JS та React є ключовими технологіями для веб-розробки, що забезпечують швидку та ефективну розробку веб-сайтів та веб-додаток.
- Однакова Мова на Клієнтському та Серверному Боці:
- Використання JS як мови програмування для клієнтської та серверної частин (за допомогою Node.js) запускає синхронізацію та обмін даними між клієнтом і сервером.
- React - Для Швидкої та Ефективної Розробки Інтерфейсу:
- React є бібліотекою, спеціалізованою на розробці користувацького інтерфейсу. Вона пропонує компонентний підхід та віртуальний DOM, що полегшує розробку, рефакторинг та управління станом.
- Широкий Вибір Бібліотек та Фреймворків:

- JavaScript має велику кількість бібліотек і фреймворків для різних потреб розробки. Це дає можливість вибрати та інтегрувати технології, які найкраще відповідають конкретним вимогам проекту.
- Спільнота та Екосистема:
- Як мова з великою спільнотою, JS має багато ресурсів, підтримку та інформацію для розробників. React також має активну спільноту та багато готових рішень.
- Гнучкість та Розширюваність:

JS дозволяє розробникам швидко реагувати на зміни у вимогах проекту та легко розширювати функціональність.

Узагальнюючи, вибір JS та React є логічним для веб-сервісу пошуку житла, орієнтованого на користувача, після чого ці технології спеціалізуються на швидкій та ефективній розробці інтерфейсів та взаємодії з користувачем [19].

2.6 Вибір системи управління базами даних (СУБД)

Microsoft SQL Server - це високопродуктивна система управління базами даних (СУБД), розроблена корпорацією Microsoft. Вона призначена для зберігання, керування, опрацювання та забезпечення доступності структурованих даних в різних застосунках і середовищах, від корпоративних систем до веб-додатків. SQL Server володіє великими можливостями щодо обробки даних, безпеки, резервного копіювання і відновлення даних, а також розвитку різноманітних додатків, що забезпечує його популярність серед організацій та розробників програмного забезпечення.Microsoft SQL Server є важливим інструментом для організацій і розробників, які працюють з базами даних і потребують надійного та продуктивного засобу для зберігання та обробки даних[20].

Основні переваги Microsoft SQL Server:

- Надійність: SQL Server відома своєю високою стабільністю і надійністю. Вона добре справляється з великими обсягами даних і має механізми для відновлення в разі аварій.
- Висока продуктивність: SQL Server може ефективно опрацьовувати складні запити та обробляти велику кількість транзакцій за короткий час.
- Широкі можливості: SQL Server надає розширені можливості для аналізу даних, створення звітів і роботи з даними. Вона підтримує роботу з текстовими, числовими, графічними і аудіо-візуальними даними.
- Зручний інтерфейс: Microsoft надає інтерфейси для роботи з SQL Server, включаючи SQL Server Management Studio (SSMS), що робить адміністрування та розробку додатків більш зручними.
- Інтеграція з іншими продуктами Microsoft: SQL Server легко інтегрується з іншими продуктами Microsoft, такими як Microsoft Azure, SharePoint, Excel і іншими.
- Висока безпека: SQL Server надає різні механізми безпеки, включаючи аутентифікацію, авторизацію, шифрування даних і моніторинг доступу.
- Спільнота та підтримка: Існує активна спільнота користувачів і розробників SQL Server, а також офіційна підтримка від Microsoft.
- Складові для аналізу даних: SQL Server має вбудовані складові для створення бізнес-аналітики і звітів, такі як SQL Server Analysis Services (SSAS) і SQL Server Reporting Services (SSRS).
- Резервне копіювання і відновлення: SQL Server надає засоби для резервного копіювання і відновлення даних, що допомагає забезпечити їхню безпеку.

- Клауд-інтеграція: SQL Server може легко інтегруватися з хмарними сервісами Microsoft Azure, що спрощує роботу з хмарними даними та обчисленнями [21].
- Незважаючи на багато переваг, Microsoft SQL Server також має свої недоліки і обмеження:
- Високі витрати: Ліцензійна вартість Microsoft SQL Server може бути високою, особливо для великих підприємств. Це може бути обмежувальним фактором для менших компаній або стартапів.
- Платформозалежність: SQL Server є платформозалежним продуктом, тобто він в основному призначений для операційних систем Windows. Це може створювати обмеження для розробки крос-платформових додатків.
- Обмеження версій: Різні версії SQL Server мають різні обмеження і функціональність. Деякі більш розширені можливості доступні тільки в дорогих версіях, що може обмежувати вибір для певних застосувань.
- Вимоги до апаратного забезпечення: Для оптимальної продуктивності і обробки великих обсягів даних SQL Server може вимагати потужний апаратний обладнання, що може збільшувати загальні витрати.
- Обмежена безкоштовна версія: Незважаючи на наявність безкоштовної версії SQL Server (Express Edition), вона має обмеження на розмір бази даних і ресурси сервера, що може бути недостатньо для деяких проектів[22].
- Врахувавши всі переваги недоліки при розробці автоматизованої системи автовокзалу було обрано СУБД MS SQL Server.

2.7 Вибір фреймворків

React - це бібліотека для створення інтерфейсів користувача веб-додатків. Вона дозволяє розробникам створювати ефективні та зручні застосунки, використовуючи компонентну архітектуру та віртуальний DOM. React використовується для швидкої та ефективної розробки інтерактивних інтерфейсів, а його однозначний логічний шар спрощує управління станом додатків. Бібліотека має велику та активну спільноту розробників, що полегшує вирішення завдань та обмін знаннями.

Вибір фреймворка React для розробки сервісу пошуку житла може бути обґрунтований кількома перевагами та особливостями цієї технології:

Переваги React:

- Компонентна архітектура:
- React базується на компонентній архітектурі, що полегшує розробку та управління складністю інтерфейсу. Кожен елемент може бути представлений як окремий компонент, що спрощує розширення та підтримку коду.
- Віртуальний DOM:
- Використання віртуального DOM дозволяє React оптимізувати швидкість відтворення змін у реальному DOM, що робить взаємодію з інтерфейсом швидше та ефективніше.
- Однозначний Логічний Шар:
- React сприяє створенню однозначного логічного шару (single-source-of-truth), що полегшує управління станом та взаємодію з даними.
- Розширюваність:
- React дозволяє легко розширити функціональність за допомогою сторонніх бібліотек та плагінів.
- Активна Спільнота:

- React має велику та активну спільноту розробників, що полегшує вирішення проблеми та забезпечує підтримку [23].
- Досвід Розробника:
- Я вже маю досвід використання React, це посприяє швидкій розробці та зменшить кількість можливих проблем.
- Недоліки React:
- Вивчення Кривої:
- Для новачків вивчення React може здаватися складним, особливо якщо вони раніше не працюють з концепціями, такими як JSX, компоненти та стейтуються.
- Зайва Складність для Простих Проектів:
- У невеликих проектах, де складність управління станом не така велика, використання React може здаватися зайвим.
- Велика кількість вибору:
- Велика кількість бібліотек та рішень, пов'язаних з React, може призвести до вибору не найкращого рішення або перенасичення проекту.
- Потреба в Додаткових Інструментах:
- Для повноцінної розробки іноді необхідно використовувати додаткові інструменти та бібліотеки [24].

Загалом, вибір React обґрунтований його перевагами, зокрема для складних та інтерактивних інтерфейсів, а також досвідом автора використання цієї технології.

2.8 Вибір середовища розробки

Visual Studio Code (VS Code) є популярним інтегрованим середовищем розробки (IDE) для веб-розробки, включаючи проекти з використанням React. Ось декілька аспектів, які почали актуальність та переваги VS Code:

Актуальність та переваги:

Безкоштовність та Відкритий Код:

VS Code є безкоштовним та відкритим для всіх, що робить його доступним для широкого кола розробників.

Розширюваність:

Велика кількість розширень (розширень), доступних для VS Code, дозволяє розробникам знайти інструменти під свої потреби та зручності.

Інтеграція з Git:

VS Code має вбудовану інтеграцію з Git, що полегшує ведення версії та спільну роботу над проектами.

Легкість використання та інтерфейс:

Інтерфейс VS Code є інтуїтивно зрозумілим, а його легкість робить використання його популярним серед розробників з різним рівнем досвіду.

Спільна робота з Node.js та JavaScript:

VS Code добре інтегрується з Node.js і підтримує розробку проектів на JavaScript, включаючи React.

Недоліки:

Велика кількість опцій:

Для новачків можна здатися, що VS Code має багато опцій та налаштувань, що можна призвести до невизначеності.

Високі Вимоги до Ресурсів:

Для великих проектів VS Code може вимагати більше ресурсів, що можна збільшити на продуктивність на менш потужних комп'ютерах.

Менше Функціональності для Деяких Технологій:

Деякі розширення можуть мати меншу функціональність або підтримку за допомогою інших IDE, особливо якщо розробляється на незвичайних технологіях на рисунку 2.4 зображено функціонал Visual Studio code.

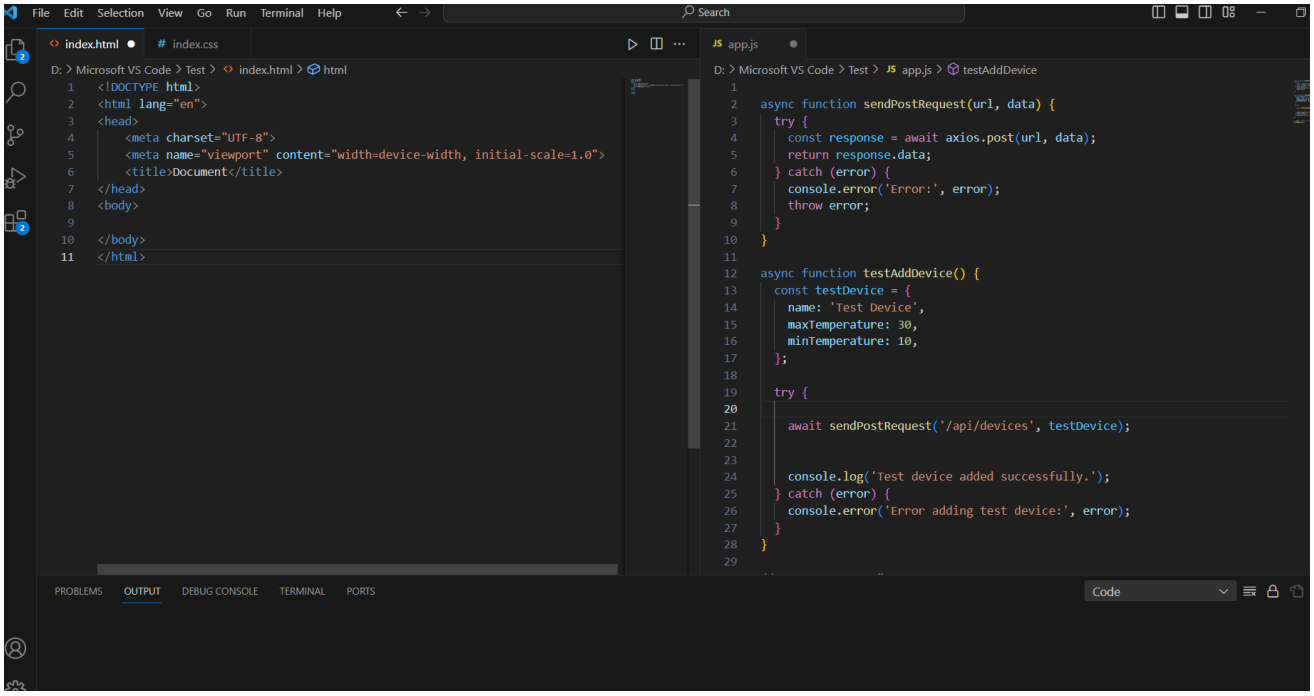

Рисунок 2.4 – Середовище розробки Visual Studio code.

Загалом, VS Code є популярним вибором, можливим для розробників React, завдяки своїм відчуттям, розширюваності та активній спільноті.

2.9 Висновки

У даному розділі здійснено вибір архітектури та технологій для реалізації даного проекту. Було прийнято рішення використовувати JavaScript як мову програмування для реалізації клієнтської частини. Для реалізації серверної частини було обрано технологію Node JS. В якості середовища розробки було обрано Visual Studio Code, завдяки його зручності, легкості використання та підтримки ряду розширень. Для

зберігання та обробки даних була обрана база даних Microsoft SQL Server, що забезпечує високу надійність і продуктивність, що важливо для ефективної роботи системи пошуку житла.

3 ПРОЕКТУВАННЯ СИСТЕМИ ПОШУКУ ЖИТЛА. ДИЗАЙН ТА ФУНКЦІОНАЛЬНІСТЬ ЗАСТОСУНКУ

3.1 Створення дизайну сервісу пошуку житла

При виборі дизайну створення застосунку сервісу пошуку житла для постраждалих громадян від російської агресії дотримувалися ряд основних принципів:

Користувальницька дружелюбність (User-Friendly):

Велика увага приділялася створенню інтуїтивно зрозумілого і легкого у використанні інтерфейсу. Мета полягала в тому, щоб користувачі будь-якого рівня могли легко користуватися сервісом без додаткових труднощів.

Мінімалізм та Простота:

Дизайн орієнтувався на мінімалістичний підхід, де кожен елемент має своє чітке визначення, а декоративні елементи були мінімізовані. Простий та лаконічний дизайн сприяє зосередженості користувача на основних функціях сервісу.

Адаптивність та Відповідність (Responsive Design):

Застосунок розроблений з урахуванням різних пристроїв та розмірів екранів, забезпечуючи користувачам комфортний досвід як на комп’ютерах, так і на мобільних пристроях.

Безпека та конфіденційність: Були прийняті заходи для забезпечення високого рівня безпеки та конфіденційності користувачів. Використання шифрування та безпекових стандартів гарантує захист особистої інформації [22].

Естетика та Привабливість:

Дизайн спроектований таким чином, щоб створити приємні враження та відповідати сучасним тенденціям. Гармонійне використання кольорів, шрифтів та графічних елементів підкреслює естетичний бік застосування.

Ці принципи дизайну спрямовані на створення ефективного та привабливого сервісу, який задовольняє потреби та очікування користувачів у пошуку житла після російської агресії на рисунку 3.1 зображено вигляд такого діалогово вікна.

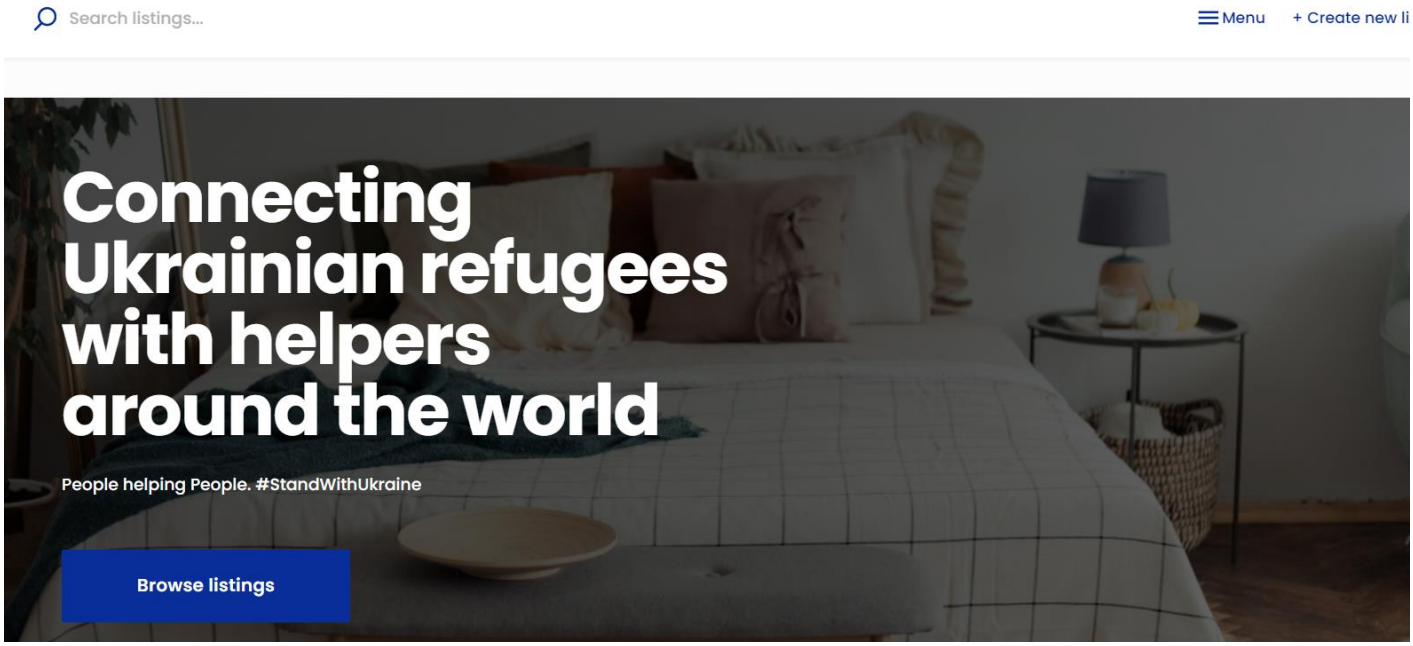

Рисунок 3.1– Загальний вигляд за стосунку сервісу пошуку житла

На рисунку 3.2 зображено форму реєстрації яка дозволяє користувачам безпечно відвідувати даний сервіс, який – зберігає логін, пароль, email.

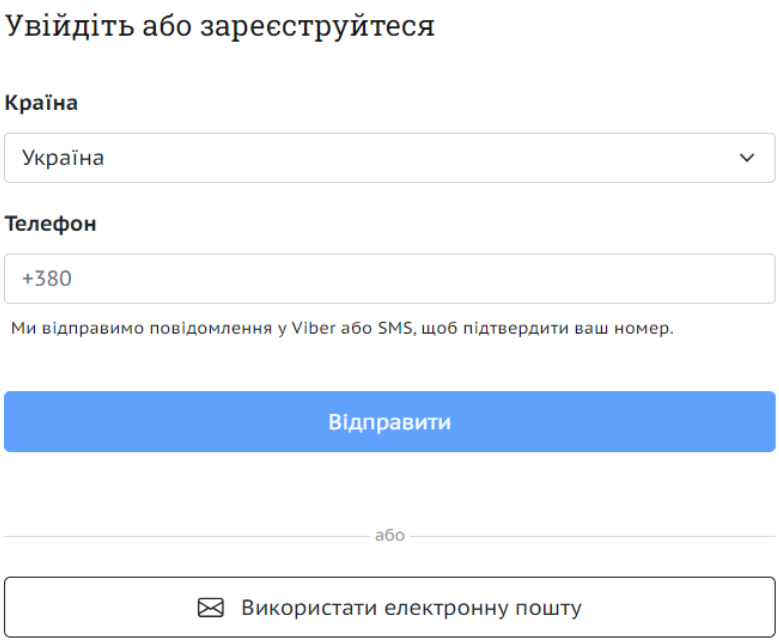

Рисунок 3.2 – Форма реєстрації

На рисунку 3.3 Зображені додаткові сервіси що містять категорії пов’язані з реабілітацією, допомогою знаходження роботи та сервісу пошуку житла, попередні 2 категорії знаходяться на стадії розробки та подальшого розвитку даного проекту.

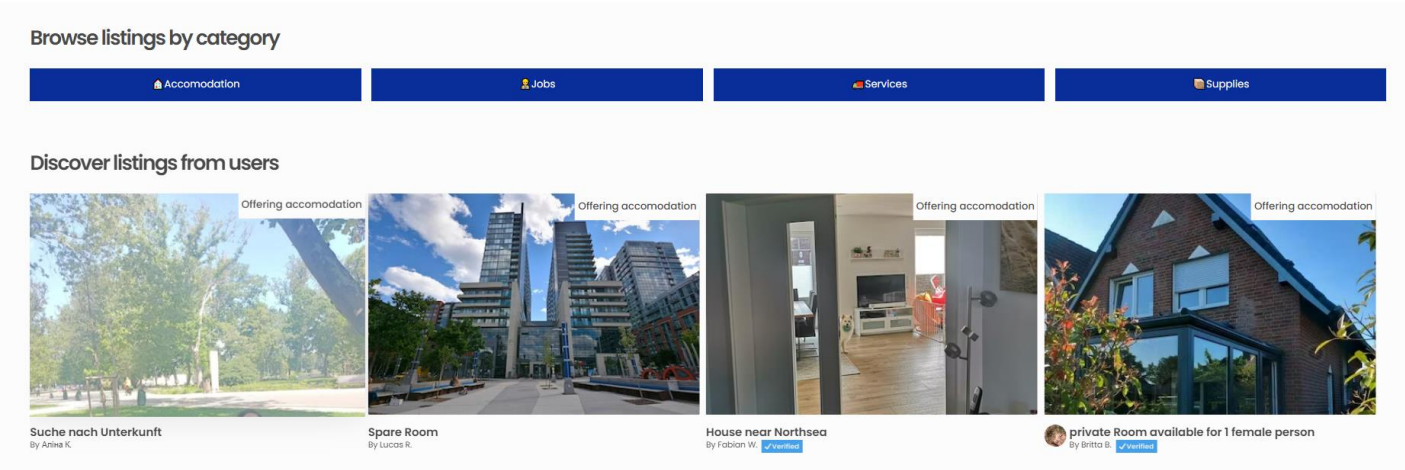

Рисунок 3.3 – Додатковий функціонал застосунку

На рисунку 3.4 зображено форму зворотнього зв’язку та соціальних мереж

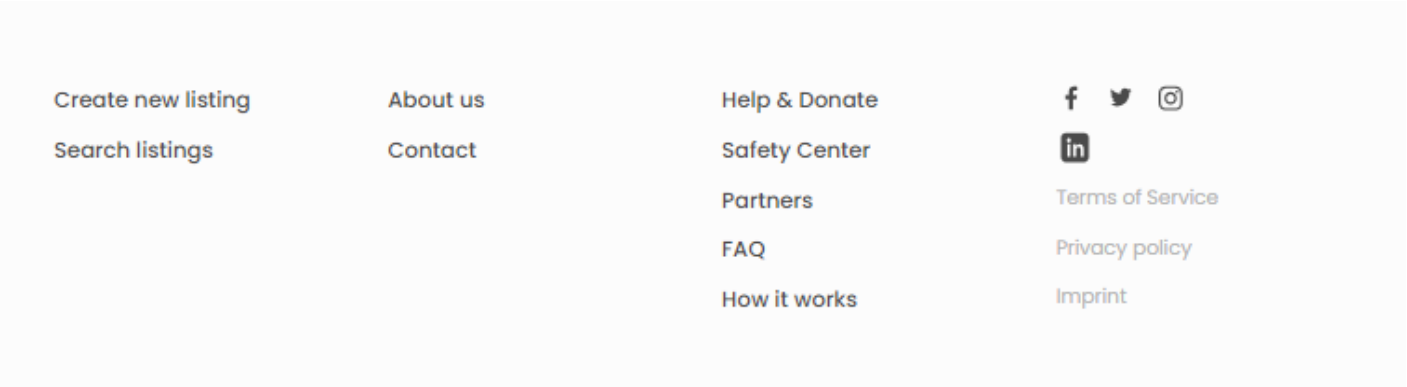

Рисунок 3.4 – Форма зворотнього зв’язку та соціальних мереж

Дизайн служби пошуку житла, орієнтований на простий і зрозумілий функціонал, спрямований на негайне задоволення потреб постраждалих громадян у пошуку житла після російської агресії. Проте він також відкриває можливості для розвитку та впровадження нових проектів із наданням додаткової підтримки та революційних рішень для постраждалих.

Простий та Зрозумілий Функціонал:

Інтуїтивно зрозумілі інтерфейс та функціональну користь користувачам легко шукати, оглядати та вибирати житло. Простота використання сприяє ефективному використанню сервісу, не завдаючи додаткових труднощів.

Можливості для Розвитку:

Система побудована з масштабування результатів, що дозволяє легко впроваджувати нові функції та розширювати функціональні залежно від необхідних користувачів.

Існує потенціал для покращення алгоритмів пошуку, впровадження інтеграцій з іншими соціальними та допоміжними сервісами.

Є можливість для запуску революційних проектів, спрямованих на вирішення конкретних потреб постраждалих, таких як програми соціальної підтримки, освітні ініціативи чи програми відновлення.

Такий підхід до дизайну забезпечує негайне задоволення потреб користувачів у пошуку житла, але й відкриває двері для майбутнього розвитку та інновацій у сфері допомоги тим, хто постраждав від російської агресії.

3.2 Проектування системи

У даному підрозділі описується створення та системи пошуку житла за допомогою фільтрів код програми показаний на рисунку 3.5 .

Рисунок 3.5 – Програмний опис структури побудови системи фільтрів програми

Дана система не розрахована тільки на територію України, але для взаємодії за кордоном потрібно задіяти більше ресурсів. На рисунку 3.6 зображено розробку додаткового функціоналу для системи фільтрів запитів пошуку житла.

```
<script>
    function applyFilters() {
      const criteria = document.getElementById('criteria').value;
      const duration = document.getElementById('duration').value;

      // Симуляція результатів
      const fakeResults = [
        { name: 'Апартаменти 1', criteria: ['anyone', 'family'], duration: 'a
        { name: 'Квартира 2', criteria: ['couples', 'pets'], duration: 'sever
        // Додайте інші об'єкти з результатами за потребою
      ];

      // Фільтрація результатів
      const filteredResults = fakeResults.filter(result => {
        return (
          (criteria === 'anyone' || result.criteria.includes(criteria)) &&
          (duration === 'any' || result.duration === duration)
        );
      });

      // Відображення результатів
      displayResults(filteredResults);
    }

    function displayResults(results) {
      const resultsContainer = document.getElementById('searchResults');
      resultsContainer.innerHTML = '';
```

Рисунок 3.6 – додатковий функціонал системи фільтрів

Детальний вигляд розробленого фільтру та категорій які можна обрати зображено на рисунку 3.7.

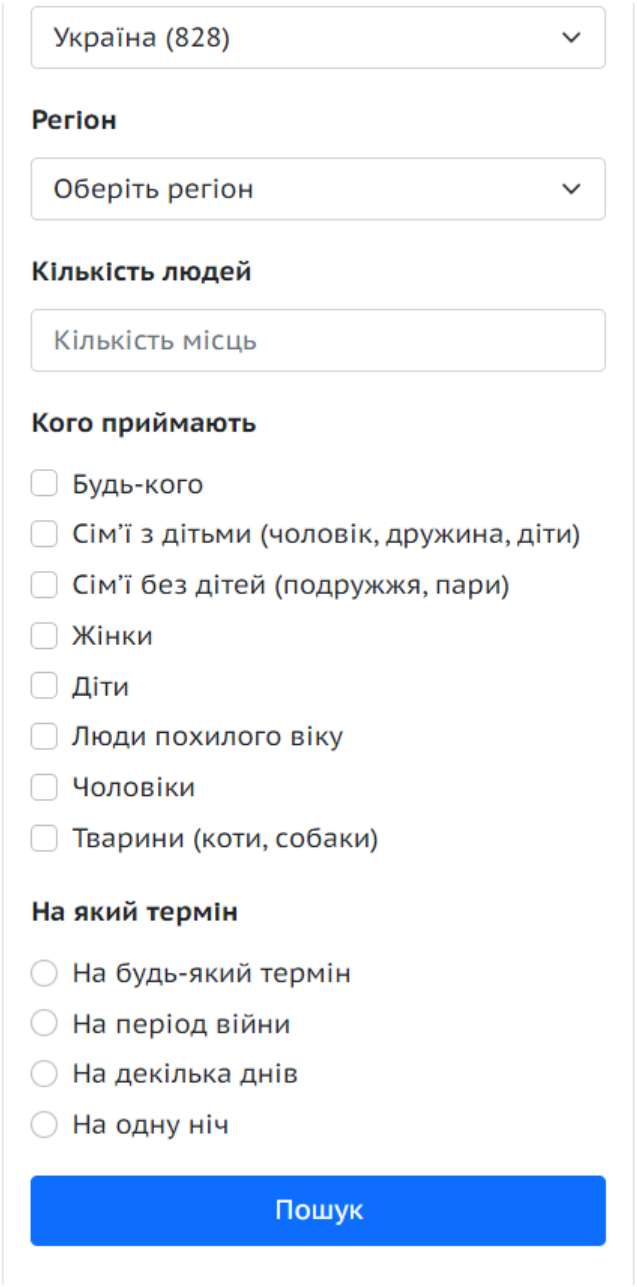

Рисунок 3.7 – Система фільтрів застосунку

3.3 Проектування логіки підключення свого сервісу до іншого сервісу в Україні

Підключення свого сервісу до іншого сервісу в Україні, який надає житло, може відбуватися за допомогою API (інтерфейсу програмування застосунків). Щоб об’єднати свій сервіс з іншими, описую кроки: Отримання ключа API:

Потрібно переконайтеся, що у вас є ключ API від іншого сервісу. Для отримання ключа можемо зареєструватися на їхньому веб-сайті або переглянути документацію API.

Ознайомлення з документацією API: Ознайомтеся з документацією API іншого сервісу. Вона містить інформацію про доступні ендпоінти, параметри запиту, формати даних і так далі.

Написаня коду з’єднання: написання коду на своєму сервері або клієнтському додатку, щоб виконати HTTP-запит до API іншого сервісу. Для цього можна використовувати мову програмування, знадобиться JavaScript (за допомогою бібліотеки axios).

Обробка відповіді API: Опрацюємо відповідь API від іншого сервісу у нашому коді. Нам може знадобитися розпарсити отримані дані і використовувати їх у своїй додатку.

Обробка помилок і забезпечення безпеки: Додаємо в код обробку помилок, перевіримо параметри запиту та відповіді для безпеки за стосунку [26].

Дотримання умов використання: Пам'ятайте про умови використання API іншого сервісу. Деякі сервіси можуть мати обмеження на кількість запитів, доступні функції та інші обмеження.

Встановлюємо бібліотеку axios за допомогою npm (Node Package Manager) на рисунку 3.8 показано процес встановлення бібліотеки.

```
npm install axios
```

Рисунок 3.8 - Встановлення бібліотеки axios

Створіть файл, apiRequest.js, і вводимо наступний код зображений на рисунку 3.9:

```
// Підключення бібліотеки axios
const axios = require('axios');

// URL API, який ви хочете викликати
const apiUrl = 'https://dopomagay.com/api/endpoint';

// Виклик GET-запиту до API
axios.get(apiUrl)
  .then(response => {
    // Обробка успішної відповіді
    console.log('Success:', response.data);
  })
  .catch(error => {
    // Обробка помилки
    console.error('Error:', error.message);
  });
```

Рисунок 3.9 – Виклик GET – запиту до API

Цей код отриманий GET-запитує до API і виводить результат або помилку в консоль на рисунку 3.10 зображену результат запиту в консолі.

```
node apiRequest.js
```

Рисунок 3.10 – Результат виводу в консоль

3.4 Опис модуля який виводить результати запитів пошуку житла

Модуль виведення результатів пошуку житла призначений для ефективного представлення інформації, отриманої в результатах використовуваних запитів до системи. Цей модуль грає ключову роль із забезпеченням користувачів зручною та інтуїтивно зрозумілою інтерактивною платформою для вибору відповідного житла.

Основні Характеристики: Модуль забезпечує виведення деталей житла, яке відповідає введеним критеріям користувача. Інформація включає зображення, опис, характеристики, а також доступні послуги та зручності [27].

Фільтрація та сортування:

Користувач має можливість використовувати фільтри та параметри сортування для точного визначення своїх вимог та отримати найбільш відповідні результати.

Інтерактивність:

Модуль забезпечує можливість взаємодії з результатами, дозволяючи користувачам переглядати деталі, визначати вибір за допомогою різних опцій та отримувати додаткову інформацію.

Картографічне Відображення:

Це актуально, тому модуль включає картографічний компонент, який показує розташування доступного житла на карті для зручності вибору.

Інформаційні повідомлення:

Система повідомляє користувачів про будь-які важливі аспекти, такі як наявність обмежень, зниження ціни, нові пропозиції тощо.

Підтримка Різних Платформ:

Модуль розроблений так, щоб забезпечити високу відзивливість та оптимальний вигляд на різних пристроях та браузерах.

Модуль для виведення результатів пошуку житла зручності та ефективності користування сервісом, створюючи процес вибору житла простим і приємним для

кожного користувача на рисунку 3.11 зображено загальний вигляд за стосунку для веб-системи.

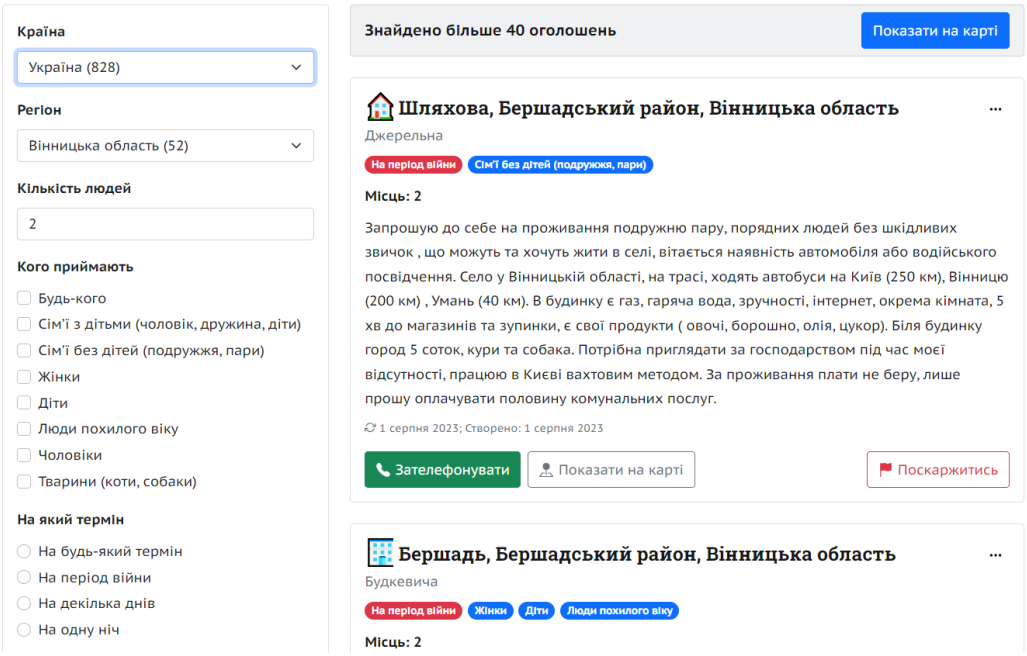

Рисунок 3.11 - Загальний вигляд блоку виведення інформації про житло

На Рисунку 3.12 показана версія для мобільних пристроїв

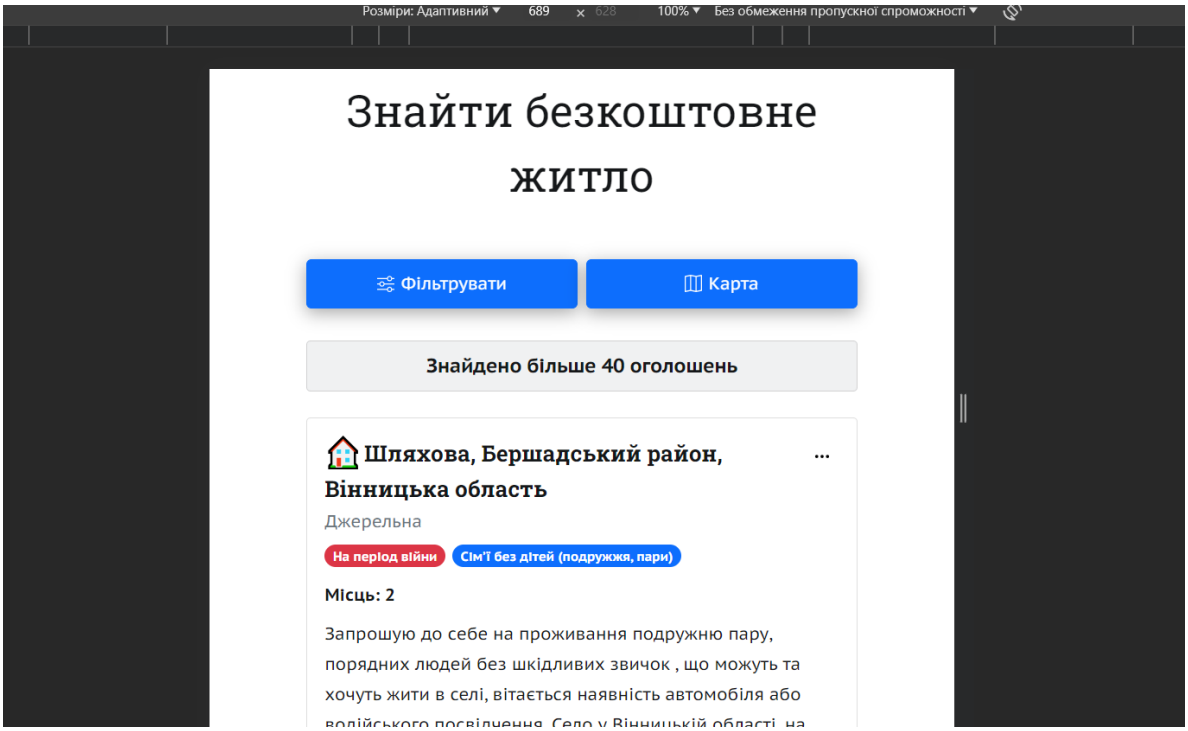

Рисунок 3.12- Версія за стосунку для мобільних девайсів

1.5 Картографічний компонент застосунку

Картографічний компонент у застосунку розширює можливості користувачів у виборі житла за допомогою візуального відображення його розташування на карті.

Основні Характеристики:

Доступні варіанти житла позначаються маркерами на карті. Кожен маркер може являти собою окремий об'єкт або групу об'єктів з певними характеристиками.

Інтерактивність:

Користувач може взаємодіяти з картою, наводячи на маркери, отримувати коротку інформацію та переходити до повного перегляду деталей об'єкта.

Зручне Визначення Розташування:

Користувач може швидко налаштувати розташування об'єктів житла в різних частинах міста чи на різних вулицях, що полегшує процес вибору оптимального варіанту [29].

Комплементарність з іншими функціями:

Картографічний компонент взаємодіє з іншими частинами застосунку, такими як фільтри, що дозволяють користувачам точно використовувати своє уподобання.

Використання картографічного компонента в застосуванні не лише полегшує орієнтацію користувачів, а й покращує процес вибору житла більш захоплюючим та інтерактивним результат роботи даного модуля зображений на рисунку 3.13.

Рисунок 3.13 Вигляд картографічного помічника пошуку житла

3.6 Тестування розробленої системи

Тестування програмного забезпечення – це процес перевірки відповідності роботи ПЗ вимогам, специфікаціям та очікуваним результатам. Цей етап розробки виявляє помилки та дефекти, покращує якість продукту, підвищує безпеку та впевненість у його роботі. Тестування включає виконання тест-кейсів, порівняння реальних результатів з очікуваними та виявлення помилок і дефектів.

Тестування є важливим елементом для забезпечення якості програмного забезпечення та впевненості в його надійності. Тестувальники допомагають розробникам виявляти помилки та дефекти, що можуть призвести до збоїв або неправильної роботи ПЗ. Вони також сприяють підвищенню безпеки продукту та поліпшенню його функціональності.

Тестування може бути виконане як вручну, так і з використанням автоматизованих інструментів. Автоматизація тестування – це використання програмних засобів для виконання тестових випадків, оцінки результатів і створення звітів. В залежності від ступеня автоматизації, тестування може бути ручним, напівавтоматичним або повністю автоматичним[26].

Ручне та автоматизоване тестування є важливими на різних рівнях тестування та можуть бути використані в різних типах та видів тестування. Обидва підходи можуть бути включені в різні етапи процесу тестування та сприяють досягненню різних цілей.

Автоматизація тестування має велику перевагу в збереженні часу, зусиль та ресурсів. Один раз створений автоматизований тест можна запускати повторно без значних зусиль.

Хоча мануальне тестування може бути застосовано до практично будь-якого додатка, автоматизоване тестування рекомендується проводити лише для стабільних систем. Автоматизоване тестування часто використовується для регресійного тестування, але не підходить для всіх видів тестування, наприклад, ad-hoc або дослідницького, які краще виконувати вручну.

Мануальне тестування може бути монотонним та повторюваним, але автоматизація допомагає уникнути цього, забезпечуючи ефективність завдяки автоматизованому виконанню завдань.

У практиці найчастіше використовується комбінація ручного та автоматизованого тестування, причому ступінь автоматизації залежить від типу проекту та особливостей виробничих процесів у конкретній компанії[27].

Під час проведення мануального тестування, тестувальник виступає в ролі користувача та намагається знайти в програмному забезпеченні помилки та/або неточності в роботі програми.

Вимогами до даної системи є: робочий вхід на сайт як для адміністратора так і для користувача, робочий фільтр пошуку та їх коректний пошук із бази даних, робоча можливість бронювання та відміни бронювання користувачем та відображення цих змін у загальному табло квитків як з боку адміністратора сайту так і зі сторони користувача, “user-friendly” інтерфейс [30].

Даний сценарій перевір правильність роботи форми реєстрації та перевіряє, що введені дані зберігаються правильно та користувач може успішно увійти в систему після реєстрації Результати перевірки занесені в таблицю 3.1.

Таблиця 3.1 Результат перевірки тестування форми реєстрації

Крок	Дії користувача	Очікуваний результат	Результат
1	Відкрити сторінку реєстрації	Відображення форми реєстрації з відповідними полями	Успіх
2	Заповнити всі обов’язкові поля	Поля заповнені коректними даними	Успіх
3	Вибрати опцію ‘Зареєструватись’	Система приймає дані відображає повідомлення про успішну реєстрацію	Успіх
4	Використати створені облікові дані для входу	Вхід в систему виконано успішно	Успіх

Для тестування форми фільтрів ви можете використати наступний код на мові JavaScript. У цьому тесті використовується бібліотека Jest, яка дозволяє створювати тести та перевіряти їх результати, що зображено на рисунку 3.14.

```
// Підключення Jest
const { test, expect } = require('@jest/globals');

// Підключення функції applyFilters та displayResults
const { applyFilters, displayResults } = require('./yourFileName'); // Заміні

// Тест для функції applyFilters
test('applyFilters correctly filters results', () => {
  // Підготовка вхідних даних
  const criteria = 'anyone';
  const duration = 'any';
  const fakeResults = [
    { name: 'Апартаменти 1', criteria: ['anyone', 'family'], duration: 'any'
    { name: 'Квартира 2', criteria: ['couples', 'pets'], duration: 'severalDa
    // Додайте інші об'єкти з результатами за потребою
  ];

  // Виклик функції applyFilters
  const filteredResults = applyFilters(criteria, duration, fakeResults);

  // Перевірка результатів
  expect(filteredResults).toHaveLength(1); // Очікується, що залишиться лише
  expect(filteredResults[0].name).toBe('Апартаменти 1'); // Очікується, що це
});

// Тест для функції displayResults
test('displayResults correctly displays results', () => {
  // Підготовка вхідних даних
```

Рисунок 3.14 Перевірка автоматичним тестуванням

У цьому коді ми створили два тести. Перший тест перевіряє, чи функція applyFilters правильно фільтрує результати за заданими критеріями. Другий тест перевіряє, чи функція displayResults правильно виводить результат у відповідний контейнер. При перевірці ми отримали бажані результати тестування.

3.7 Висновки

У даному розділі описано розробку програмного забезпечення для сервісу пошуку житла, що проміжним є етапом у сучасному інформаційному середовищі.

Створення такого сервісу вимагає комплексного підходу та врахування різних аспектів, таких як функціональність, зручність використання, ефективність та безпека.

У результаті розробки додаткового програмного забезпечення для пошуку житла можна отримати не тільки безкоштовний та доступний інструмент для користувачів, але й сприяти вирішенню соціальних проблем, що забезпечують житлові умови для постраждалих громадян. Використання сучасних технологій, таких як веб-розробка та бази даних, дозволяє підняти рівень сервісу та надати ефективний інструмент для вирішення житлових питань.

Застосування модуля для виведення результатів пошуку житла на карті створює потужний інструмент для користувачів, які шукають оптимальне житло. Це все до покращення роботи сервісу та задоволення користувачів, надаючи їм зручний та інноваційний інструмент для вибору нерухомості.

Завершуючи розробку, важливо зазначити, що високоякісне програмне забезпечення пошуку житла може стати необхідним кроком у розвитку соціальної сфери та вирішенні гострих проблем національного рівня.

ВИСНОВКИ

У магістерській кваліфікаційній роботі було проведено аналіз проблеми розробки сервісу пошуку житла для постраждалих громадян від російської агресії. Після цього було вирішено створити власну систему із відкритим вихідним кодом.

Для розробки було використано наступні технології: JavaScript, React, Node JS Microsoft SQL Server.

У першому розділі було виконано аналіз проблеми, пов'язаної з розробкою програмного застосунку. Було детально розглянуто стан дослідження цієї науково-технічної задачі, спираючись на аналіз існуючих конкурентів як приватних так і волонтерськх організацій, також досліджена проблема постраждалих громадян що налічує 2000000 жителів, тимчасово і постійно переселених осіб.

Далі, було розглянуто сучасні методи вирішення цього завдання та обрано оптимальний варіант для подальшого дослідження і розробки. Крім того, був проведений аналіз можливих нових підходів та методів для вирішення цієї задачі, що підсилить виконання цієї роботи та забезпечить її інноваційність.

Загалом, на основі проведеного аналізу, було визначено, що розробка програмного продукту є належним та обґрунтованим напрямком дослідження та розвитку, сприаючись на точні дані отриманих заявок на житло що налічую від 150-220 на день, при потребі в 400-600 заявок.

У другому розділі були створені докладні UML-діаграми, включаючи діаграми варіантів використання, діяльності та розгортання, які надають більш детальний опис системи. Дані діаграми детально висвітлили різні аспекти функціонування системи включаючи процеси взаємодії з користувачами та компонентами системи.

Крім того, був зроблений аналіз та вибір конкретного програмного забезпечення, мови програмування, системи управління базами даних (СУБД) та фреймворків для подальшої розробки автоматизованої системи для автовокзалу. Цей вибір здійснювався з урахуванням усіх вимог та потреб системи з метою забезпечення її ефективності.

У третьому розділі було описано розробку програмного забезпечення за стосунку її модулів та опис подальшого розвитку, а саме описано модуль фільтрів для ефективного підбору та пошуку найкращого варіанту з багатьма можливими варіантами для комфортного життя, враховано багато аспектів та потреб людей. Також розроблений модуль показу результатів пошуку, адаптив та кросплатформенність.

Результатом виконання магістерської кваліфікаційної роботи є розроблене програмне забезпечення, яке відповідає поставленим вимогам – система має робочий вхід на сайт як для адміністратора так і для користувача, має робочий модуль фільтрів та прекрасний користувацький інтерфейс, статистично яким в день може користуватись від 0 до 10000 людей одночасно.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. What is the Information Revolution? URL :https://study.com/learn/lesson/information-revolution-global-economy-effects-importance-examples.html
2. Alter S., "Work System Theory: Overview of Core Concepts, Extensions, and Challenges for the Future" (2013). Business Analytics and Information Systems. P. 35.
3. Adner R., Kapoor, R. (2016). Right Tech, Wrong Time. Harvard business review, 94(11), P 60-67.
4. Олейник А. І., Сізов А. В. (2012) IT-інфраструктура 110-170 с.
5. Мішин, Г. А. Словники та енциклопедії на академіку. IT-інфраструктура URL: http://dic.academic.
6. Сімченко, Н. Н. IT-інфраструктура підприємства: робоча програма навчальної дисципліни / Н. Н. Сімченко. -
7. What is Web Application? URL: https://www.educba.com/what-is-web-application/
8. Скороходов В.А. Автоматизоване робоче місце : Навчальний посібник / В.А. Скороходов, І.М. Худякова. - К.: ВД «Професіонал», 2007. - 416 с.
9. Cloud computing. Principles and Paradigms. / Edited by Rajkumar Buyya, James Broberg, Andrzej Goscinski. — New Jersey: John Wiley & Sons, Inc.,P. 2011. — 641.
10. ITIL Service Operation. Best Management Practice Product, UK:TCO, 2011 «Organizing for service operation»
11. ITIL Service Transition. Best Management Practice Product, UK:TCO, 2011
12. What is the Voice search technology? AT Internet's definition URL :https://www.atinternet.com/en/glossary/voice-search/
13. "Eloquent JavaScript" авторства Marijn Haverbeke P. 58-112.
14. "Node.js Design Patterns" авторства Mario Casciaro P. 112.
15. "React Up and Running" авторства Стояна Стефанова та "Learning React" авторства Alex Banks & Eve Porcello
16. "Effective Software Test Automation" авторства Kanglin Li P. 50-70.

17. "Agile Testing: A Practical Guide for Testers and Agile Teams" авторства Lisa Crispin та Janet Gregory P. 40-67.
18. INIT URL:https://www.initse.com/ende/company/about-init/
19. Artificial Intelligence as a factor of public transportations system development URL :https://www.sciencedirect.com/science/article/pii/S2352146522005312
20. Основи QA - Тестування програмного забезпечення URL :https://lemon.school/blog/osnovy-qa
21. Ручне та автоматизоване тестування – QALight URL:https://qalight.ua/baza-znaniy/ruchne-ta-avtomatizovane-testuvannya/
22. OOP Concepts for Beginners: What is PolymorphismURL:https://stackify.com/oop-concept-polymorphism/ (дата звернення 07.10.2023)
23. Microsoft SQL Server URL:https://www.techtarget.com/searchdatamanagement/definition/SQL-Server (дата звернення 10.10.2023)
24. MS SQL SERVER HISTORY AND ADVANTAGES URL:https://bytescout.com/blog/2014/09/ms-sql-server-history-and-advantages.html (дата звернення 10.10.2023)
25. Коли тестування повинно бути автоматизованим? URL: https://www.stickyminds.com/article/when-should-test-be-automated
26. Практичний досвід автоматизованого тестування URL: http://www.methodsandtools.com/archive/archive.php?id=33.
27. Демків Л. І. Дослідження впливу параметрів функції належності на якісні показники функціонування системи з двома коренями в правій півплощині // Вісник Нац. ун-ту «Львівська політехніка». – 2012. – № 736. – 36–43 с.
28. Основи QA - Тестування програмного забезпечення URL:https://lemon.school/blog/osnovy-qa (дата звернення 01.11.2023)
29. Ручне та автоматизоване тестування – QALight URL:https://qalight.ua/baza-znaniy/ruchne-ta-avtomatizovane-testuvannya/ (дата звернення 10.11.2023)

ДОДАТКИ

Додаток А

(обов'язковий)

Технічне завдання

ЗАТВЕРДЖУЮ
Зав. кафедри АІІТ,
д.т.н., проф.
_____ Олег БІСІКАЛО

"____" _____ 2023 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на магістерську кваліфікаційну роботу

«Розробка сервісу пошуку житла для постраждалих громадян від російської агресії»

08-31.МКР.03.02.000 ТЗ

Керівник: к.т.н., доцент каф. АІІТ
_____ Володимир КОЦЮБИНСЬКИЙ
«____» _____ 2023 р.

Виконавець: студент групи 3АКІТ-22М
_____ Віктор БІЛЕЦЬКИЙ
«____» _____ 2023 р.

Вінниця ВНТУ - 2023

1.Назва та галузь застосування

1.1. Назва – Розробка сервісу пошуку житла для постраждалих громадян від російської агресії.

1.2. Галузь застосування – Автоматизація та комп'ютерно-інтегровані технології.

2. Підстава для проведення розробки.

Підставою для виконання роботи є наказ №__ по ВНТУ від «__» _____2023р., та індивідуальне завдання на МКР, затверджене протоколом №__ засідання кафедри АІІТ від «__» _____ 2023р.

Термін виконання робіт:

3. Мета та призначення розробки.

Метою магістерської кваліфікаційної роботи є задоволення потреб постраждалих громадян в наслідок російської агресії на території України.

4. Джерела розробки.

1. Методичні вказівки до виконання магістерських кваліфікаційних робіт для студентів спеціальностей 126 – «Інформаційні системи та технології», 151 – «Автоматизація та комп’ютерно-інтегровані технології» / Уклад. Р. Н. Квєтний, О. М. Бевз, О. В. Бісікало., Р. В. Маслій – Вінниця : ВНТУ, 2020.

2. Основи QA - Тестування програмного забезпечення URL:https://lemon.school/blog/osnovy-qa (дата звернення 01.11.2023)

3. OOP Concepts for Beginners: What is PolymorphismURL:https://stackify.com/oop-concept-polymorphism/ (дата звернення 07.10.2023)

5 Технічні дані.

5.1 визначення потреб постраждалих близько 500 000 осіб ;

5.2 визначення пропозицій на ринку 100-150 на день;

5.3 оцінка потреб громадян близько 300-600 пропозицій на день.

6 Стадії розробки.

а) Аналіз потреб громадян та конкурентів 20.09 – 30.09

б)Обгрунтування методів реалізації та розробки 01.09 – 21.10

в)Розробка UML діаграм та дизайну проекту 22.10 – 30.10

г) Розробка прогармного забезпечення 31.10 – 15.11

д)Тестування та перевірка припущень 16.11 – 01.11

є)Оформлення матеріалів до захисту МКР 02.12 – 12.12

7 Порядок контролю та приймання.

Рубіжний контроль провести до «___» _____ 2023 р.

Попередній захист МКР провести «___» _____ 2023 р.

Захист МКР провести до «___» _____ 2023 р.

Додаток Б

(обов’язковий)

ІЛЮСТРАТИВНА ЧАСТИНА

РОЗРОБКА СЕРВІСУ ПОШУКУ ЖИТЛА ДЛЯ ПОСТРАЖДАЛИХ ГРОМАДЯН ВІД РОСІЙСЬКОЇ АГРЕСІЇ

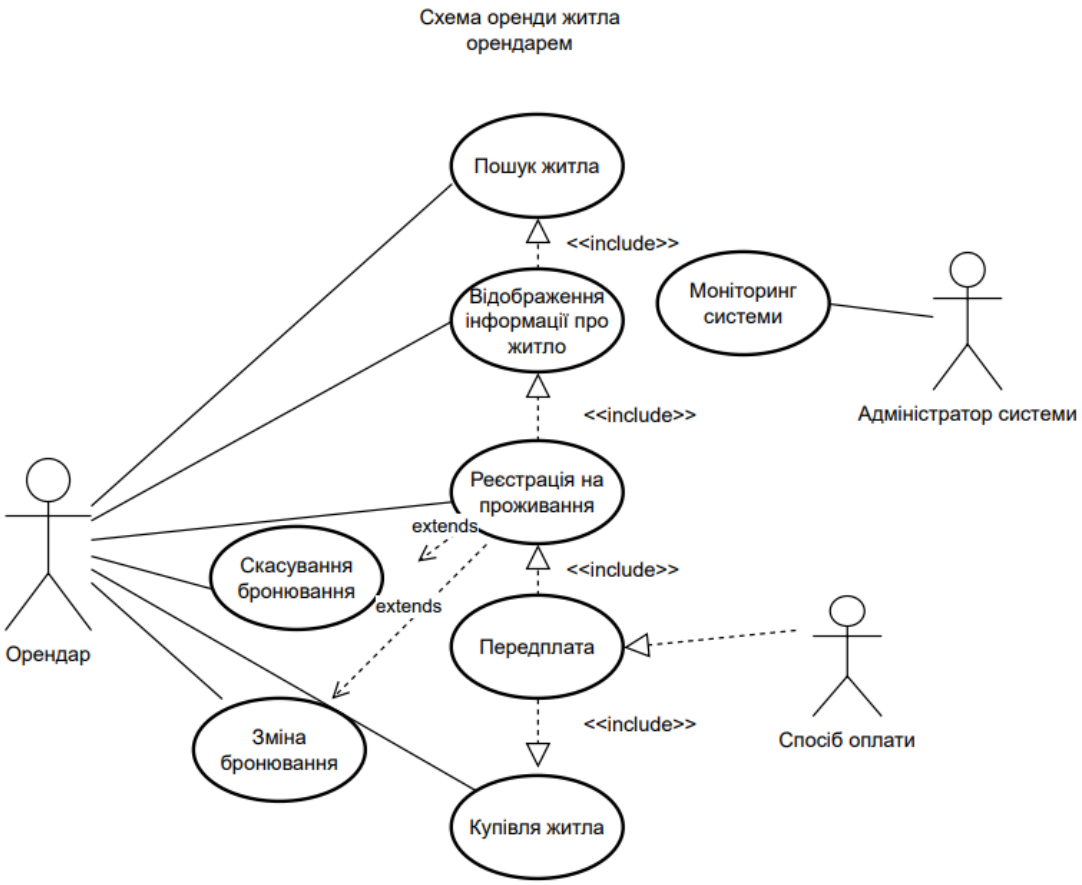

Рисунок Б.1- UML-діаграма варіантів використання

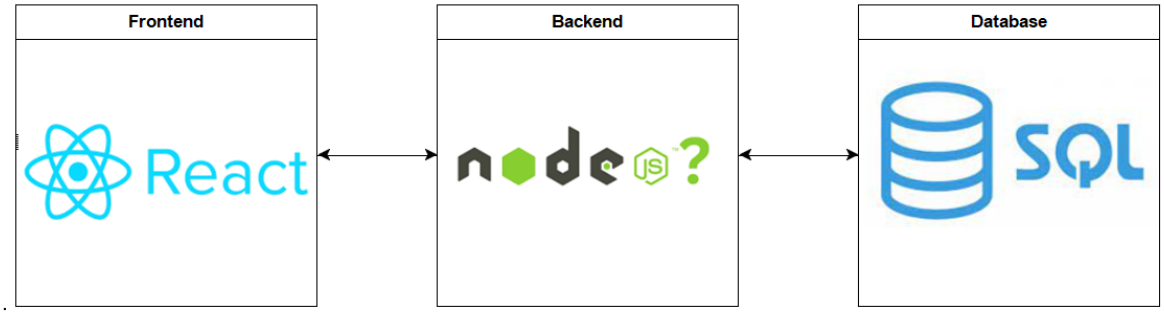

Рисунок Б.2 – Архітектура системи

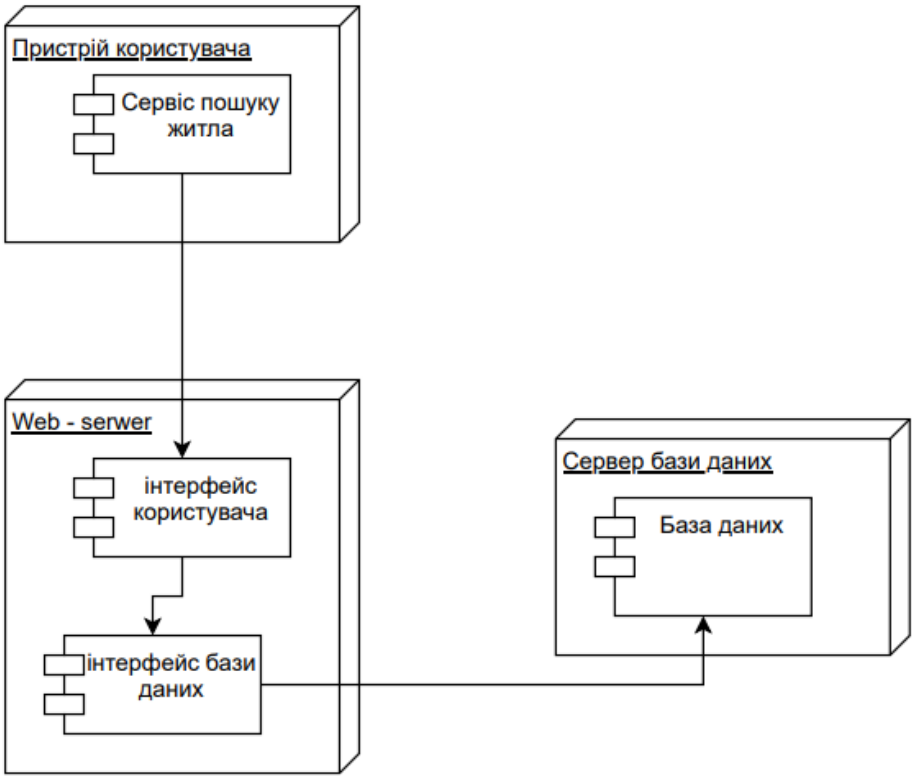

Рисунок Б.3- UML-діаграма розгортання

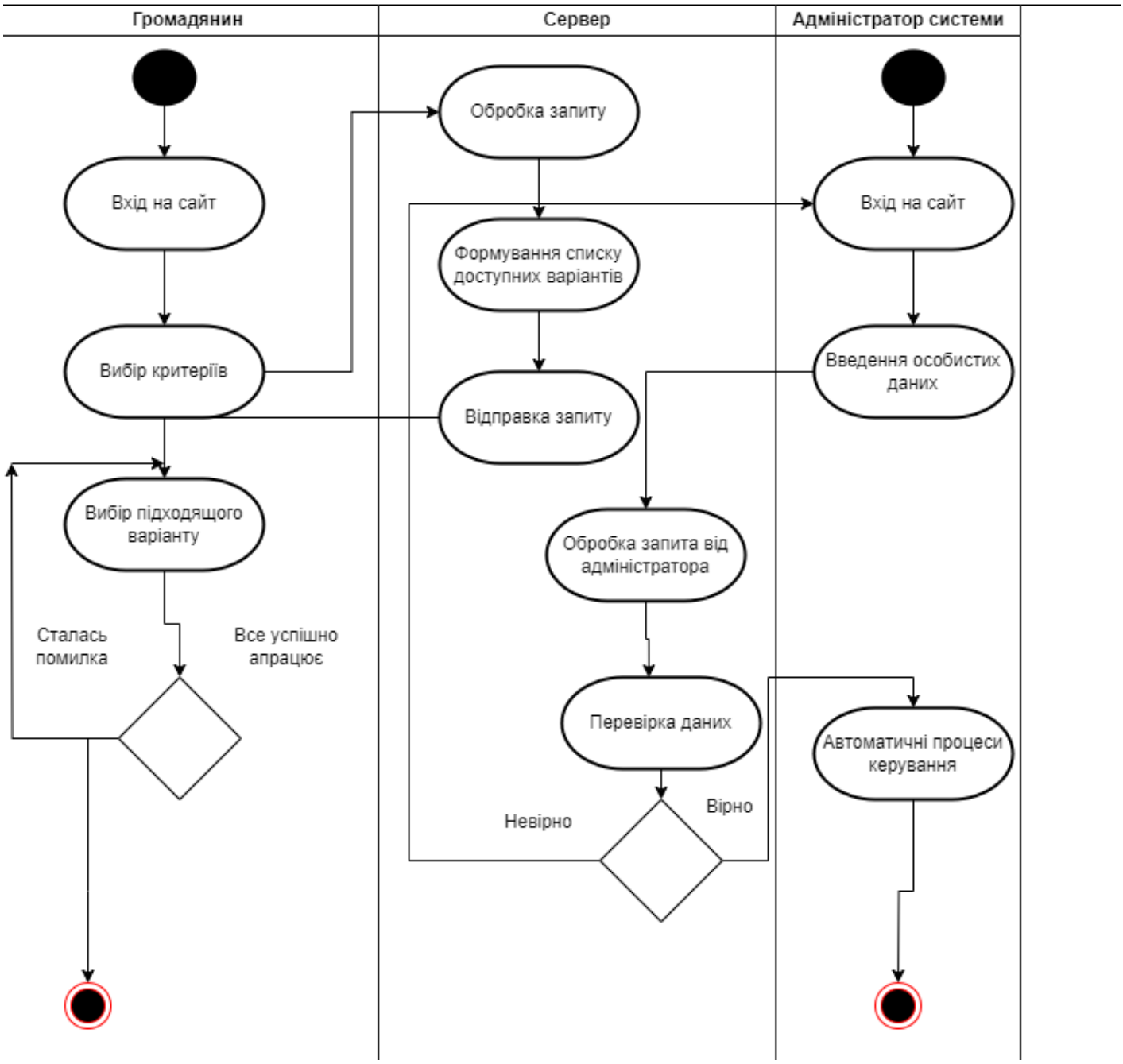

Рисунок Б.4- UML-діаграма розгортання

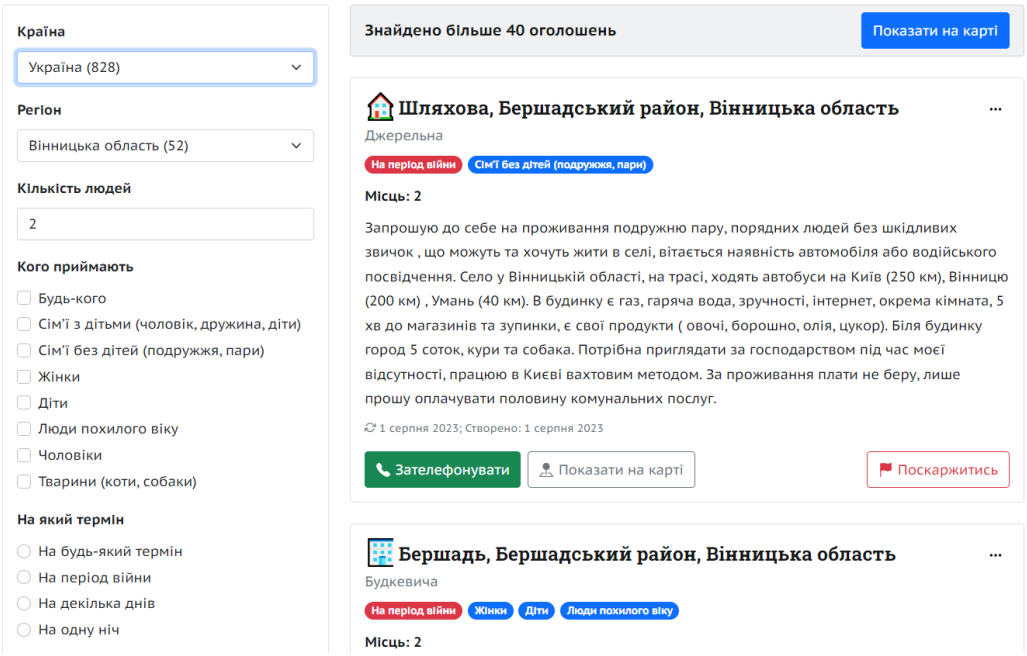

Рисунок Б.5 - Загальний вигляд блоку виведення інформації про житло

Рисунок Б.6- Модуль картографічного помічника пошуку житла

Таблиця Б.1 Результат перевірки тестування форми реєстрації

Крок	Дії користувача	Очікуваний результат	Результат
1	Відкрити сторінку реєстрації	Відображення форми реєстрації з відповідними полями	Успіх
2	Заповнити всі обов’язкові поля	Поля заповнені коректними даними	Успіх
3	Вибрати опцію ‘Зареєструватись’	Система приймає дані відображає повідомлення про успішну реєстрацію	Успіх
4	Використати створені облікові дані для входу	Вхід в систему виконано успішно	Успіх

Додаток В

(обов’язковий)

Лістинг програми

```
<сценарій>
    функція applyFilters() {
      // Отримання значень фільтрів
      const region = document.getElementById('region').value;
      const price = document.getElementById('price').value;
      const duration = document.getElementById('duration').value;

      // Симулюємо відправку запиту на сервер і отримуємо результати
      const fakeResults = [
        { назва: 'Апартаменти 1', регіон: 'північ', ціна: 1200, тривалість: 7 },
        { назва: 'Квартира 2', регіон: 'південь', ціна: 800, тривалість: 14 },
        // Додайте інші об'єкти з результатами за потребою
      ];

      // Фільтрація результатів
      const filteredResults = fakeResults.filter(result => {
        повернення (
          (регіон === 'усі' || result.region === регіон) &&
          (ціна === '' || result.price <= parseInt(ціна)) &&
          (тривалість === '' || result.duration <= parseInt(тривалість))
        );
      });

      // Відображення результатів
      displayResults(filteredResults);
    }

    функція displayResults(результати) {
      const resultsContainer = document.getElementById('searchResults');
      resultsContainer.innerHTML = '';

      if (results.length === 0) {
        resultsContainer.innerHTML = '<p>Не має результатів за вказаними критеріями.</p>';
        повернення;
```

```
    }

    results.forEach(результат => {
      const resultElement = document.createElement('div');
      resultElement.innerHTML = `<strong>${result.name}</strong> - Регіон: ${result.region}, Ціна: ${result.price} грн, Термін: ${result.duration} днів`;
      resultsContainer.appendChild(resultElement);
    });
  }
 </script>
```

```
<сценарій>
  функція applyFilters() {
    критерії const = document.getElementById('criteria').value;
    const duration = document.getElementById('duration').value;

    // Симуляція результатів
    const fakeResults = [
      { назва: 'Апартаменти 1', критерії: ['будь-хто', 'сім'я'], тривалість: 'будь-який' },
      { назва: 'Квартира 2', критерії: ['пари', 'домашні тварини'], тривалість: 'кілька днів' },
      // Додайте інші об'єкти з результатами за потребою
    ];

    // Фільтрація результатів
    const filteredResults = fakeResults.filter(result => {
      повернення (
        (критерії === 'будь-хто' || result.criteria.includes(критерії)) &&
        (тривалість === 'будь-який' || result.duration === тривалість)
      );
    });

    // Відображення результатів
    displayResults(filteredResults);
  }

  функція displayResults(результати) {
```

```
      const resultsContainer = document.getElementById('searchResults');
      resultsContainer.innerHTML = '';

      if (results.length === 0) {
        resultsContainer.innerHTML = '<p>Не має результатів за вказаними критеріями.</p>';
        повернення;
      }

      results.forEach(результат => {
        const resultElement = document.createElement('div');
        resultElement.innerHTML = `<strong>${result.name}</strong> - Критерії: ${result.criteria.join(', ')}, Термін: ${result.duration}`;
        resultsContainer.appendChild(resultElement);
      });
    }
  </script>


// Підключення Jest
const { перевірити, очікувати } = вимагати ('@jest/globals');

// Підключення функції applyFilters та displayResults
const { applyFilters, displayResults } = require('./yourFileName'); // Замініть 'yourFileName' на назву файлу з функціями

// Тест для функції applyFilters
test('applyFilters правильно фільтрує результати', () => {
  // Підготовка вхідних даних
  критерії const = 'будь-хто';
  const duration = 'будь-який';
  const fakeResults = [
    { назва: 'Апартаменти 1', критерії: ['будь-хто', 'сім'я'], тривалість: 'будь-який' },
    { назва: 'Квартира 2', критерії: ['пари', 'домашні тварини'], тривалість: 'кілька днів' },
   // Додайте інші об'єкти з результатами за потребою
  ];

  // Виклик функції applyFilters
  const filteredResults = applyFilters(критерії, тривалість, fakeResults);

  // Перевірка результатів
```

```
  очікувати(filteredResults).toHaveLength(1); // Очікується, що залишилося лише один результат
  expect(filteredResults[0].name).toBe('Апартаменти 1'); // Очікується, що це буде 'Апартаменти 1'
});

// Тест для функції displayResults
test('displayResults правильно відображає результати', () => {
  // Підготовка вхідних даних
  const resultsContainer = document.createElement('div');
  результати const = [
    { назва: 'Апартаменти 1', критерії: ['будь-хто', 'сім'я'], тривалість: 'будь-який' },
    { назва: 'Квартира 2', критерії: ['пари', 'домашні тварини'], тривалість: 'кілька днів' },
  ];

  // Виклик функції displayResults
  displayResults(результати, resultsContainer);

  // Перевірка результатів
  expect(resultsContainer.innerHTML).toContain('Апартаменти 1'); // Очікується, що виведено 'Апартаменти 1'
  expect(resultsContainer.innerHTML).toContain('Квартира 2'); // Очікується, що виведено 'Квартира 2'
});



body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
    background-color: #7cd4df;
  }

  .container {
    max-width: 800px;
    margin: 50px auto;
    background-color: #fff;
    padding: 20px;
    border-radius: 5px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
```

```
  }

  h1, h2 {
    text-align: center;
    color: #333;
  }

  form {
    margin-top: 20px;
  }

  label {
    display: block;
    margin-bottom: 5px;
  }

  input, button {
    margin-bottom: 10px;
    padding: 8px;
  }

  button {
    background-color: #4caf50;
    color: #fff;
    cursor: pointer;
    border: none;
    border-radius: 3px;
  }

  button:hover {
    background-color: #45a049;
  }



// Suppose you have a function to send requests to the server/
// An example of the function of sending a POST request to the server
async function sendPostRequest(url, data) {
  try {
    const response = await axios.post(url, data);
    return response.data;
```

```
  } catch (error) {
    console.error('Error:', error);
    throw error;
  }
}

// Test program for adding a device
async function testAddDevice() {
  const testDevice = {
    name: 'Test Device',
    maxTemperature: 30,
    minTemperature: 10,
  };

  try {
    // Send a POST request to add the device
    await sendPostRequest('/api/devices', testDevice);

    // After successfully adding the device, display a message
    console.log('Test device added successfully.');
  } catch (error) {
    console.error('Error adding test device:', error);
  }
}

// Start the test program
testAddDevice();
const express = require('express');
const bodyParser = require('body-parser');
const mongoose = require('mongoose');

const app = express();
const PORT = process.env.PORT || 3000;

app.use(bodyParser.json());

mongoose.connect('mongodb://localhost:27017/temperature_control', { useNewUrlParser:
true, useUnifiedTopology: true });

const deviceSchema = new mongoose.Schema({
  name: String,
```

```
  maxTemperature: Number,
  minTemperature: Number,
});

const temperatureLogSchema = new mongoose.Schema({
  device: { type: mongoose.Schema.Types.ObjectId, ref: 'Device' },
  temperature: Number,
  timestamp: { type: Date, default: Date.now },
});

const Device = mongoose.model('Device', deviceSchema);
const TemperatureLog = mongoose.model('TemperatureLog', temperatureLogSchema);

app.post('/api/devices', async (req, res) => {
  try {
    const device = await Device.create(req.body);
    res.json(device);
  } catch (error) {
    res.status(500).json({ error: error.message });
  }
});

app.get('/api/devices', async (req, res) => {
  try {
    const devices = await Device.find();
    res.json(devices);
  } catch (error) {
    res.status(500).json({ error: error.message });
  }
});

app.get('/api/temperature-log', async (req, res) => {
  try {
    const temperatureLog = await TemperatureLog.find().sort('-
timestamp').limit(10).populate('device');
    res.json(temperatureLog);
  } catch (error) {
    res.status(500).json({ error: error.message });
  }
app.listen(PORT, () => {
  console.log(`Server is running on http://localhost:${PORT}`
```

ДодатокГ

(обов'язковий)

ПРОТОКОЛ
ПЕРЕВІРКИ МАГІСТЕРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи:« Розробка сервісу пошуку житла для постраждалих громадян від російської агресії»

Тип роботи: магістреська кваліфікаційна робота
(БДР, МКР)

Підрозділ: кафедра АІІТ, ФІІТА, ЗАКІТ-22м
(кафедра, факультет, навчальна група)

Науковий керівник: Коцюбинський В.Ю., доц. каф. АІІТ
(прізвище, ініціали, посада)

Показники звіту подібності Unicheck

Оригінальність 97.6% Схожість 2.4%

Аналіз звіту подібності (відмітити потрібне)

■ Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату

□ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її автора. Роботу направити на розгляд експертної комісії кафедри.

□ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень

Особа, відповідальна за перевірку Роман МАСЛІЙ
(підпис) (прізвище, ініціали)

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck щодо роботи

Автор ВікторБІЛЕЦЬКИЙ
(підпис) (прізвище, ініціали)

Керівник роботи Володимир КОЦЮБИНСЬКИЙ
(підпис) (прізвище, ініціали)