МІНІСТЕРСТВО ФІНАНСІВ УКРАЇНИ
ДЕРЖАВНИЙ ПОДАТКОВИЙ УНІВЕРСИТЕТ

ФАКУЛЬТЕТ ФІНАНСІВ ТА ЦИФРОВИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ І СИСТЕМ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: Створення full-stack вебзастосунку для короткострокової оренди житла з використанням технологій React, Tailwind, Prisma, MongoDB та NextAuth.

Здобувача вищої освіти першого (бакалаврського) рівня,
IV курсу, групи ЦДБ-20-1, спеціальності 122 «Комп'ютерні науки»,
освітня програма «Технології цифрового дизайну»



(підпис)

Підька Андрія Сергійовича
(прізвище, ім'я, по батькові)

Науковий керівник 

(підпис)

Омельчук Антон Анатолійович к.т.н.,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Керівник освітньо-професійної програми 

(підпис)

Ратушняк Тетяна Володимирівна, к.ф.-м.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Допущено до захисту за рішенням кафедри комп'ютерних та інформаційних технологій і систем протокол від «22» травня 2024 № 15

Завідувач кафедри 

(підпис)

Філоненко Михайло Миколайович, к.ф.-м.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Ірпінь – 2024

Реєстрація № 48 31.05.2024 _____
(дата) (підпис відповідальної особи)

Результати перевірки до захисту 22.05.2024 _____
(до захисту, на доопрацювання) (дата) (підпис)

Результат захисту 5 (відмінно) 12.06.2024 _____
(оцінка) (дата) (підпис)

Голова комісії: _____ В.М. Франчук
(підпис)

Члени комісії: _____ М.М. Філоненко
(підпис)

_____ С.М. Грищенко
(підпис)

_____ В.О. Ніжегородцев
(підпис)

БЛАНК ОЦІНЮВАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Об'єкти контролю	Діапазон оцінки в балах	Фактична кількість балів	Примітки	Дата, П.І.Б.: керівника, голови комісії членів комісії,секретаря, (підписи)
ОЦІНЮВАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ КОМІСІЄЮ ПО ЗАХИСТУ				
1. Зміст випускної роботи (якість теоретичного матеріалу, його зв'язок з практичним, вміння представляти матеріал в логічній послідовності, наявність висновків та пропозицій)	від 0 до 30	29		12.06.2024 Франчук В.М. Філоненко М.М. Грищенко С.М. Ніжегородцев В.О. Суховецька Л.П.
2. Дотримання вимог щодо оформлення	від 0 до 10	9		
3. Вчасне виконання роботи на всіх етапах у встановлені кафедрою терміни	від 0 до 5	4		
4. Рецензент	від 0 до 5	5		
РАЗОМ БАЛІВ	від 0 до 50	47		

Оцінювання кваліфікаційної роботи науковим керівником - 47 балів.

ПЕРЕВЕДЕННЯ ОЦІНКИ ЗА 100-БАЛЬНОЮ ШКАЛОЮ ДО НАЦІОНАЛЬНОЇ ШКАЛИ ТА ШКАЛИ ЄКТС

Оцінка за 100-бальною шкалою	Оцінка за шкалоюЄКТС	Оцінка за національною шкалою	Дата	П.І.Б.: голови, членів комісії, секретаря, (підпис)
94	A	5(відмінно)	12.06.2024	Франчук В.М. Філоненко М.М. Грищенко С.М. Ніжегородцев В.О. Суховецька Л.П.

ПРАВИЛА ПЕРЕВЕДЕННЯ ОЦІНОК ДО НАЦІОНАЛЬНОЇ ШКАЛИ ТА ШКАЛИ ЄКТС

Оцінка за 100-бальною шкалою	Оцінка за шкалою ЄКТС	Оцінка за національною шкалою
90-100	A	5 (відмінно)
80-89	B	4 (добре)
70-79	C	
60-69	D	3 (задовільно)
50-59	E	
35-49	FX	2 (незадовільно)

Факультет фінансів та цифрових технологій
Кафедра комп'ютерних та інформаційних технологій і систем
Галузь знань: 12 «Інформаційні технології»
Спеціальність: 122 «Комп'ютерні науки»
Освітня програма: «Технології цифрового дизайну»

Затверджую
Завідувач кафедри _____ Філоненко М.М.
«23» жовтня 2023

Завдання на кваліфікаційну роботу
здобувачу вищої освіти
Підько Андрію Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи Створення full-stack вебзастосунку для короткострокової оренди житла з використанням технологій React, Tailwind, Prisma, MongoDB та NextAuth.

Затверджена наказом ректора від «20» жовтня 2023 № 1411
2. Строк здачі закінченої роботи «21» травня 2024
3. Мета роботи полягає у створенні повнофункціонального вебзастосунку, поглибленні знань та навичок у сфері веброзробки, отриманні досвіду.
Об'єкт дослідження вебдизайн та веброзробка.
Предмет дослідження процес створення дизайну та реалізація сайту.
Орієнтовна база дослідження вебзастосунок для платформи короткострокової оренди житла
4. Вихідні дані до роботи Теоретичні та практичні напрацювання вітчизняних і зарубіжних вчених у сфері розробки вебдодатків. Документація та навчальні матеріали щодо використання технологій React, Tailwind, Prisma, MongoDB та NextAuth. Статистичні дані та аналітичні звіти про стан і тенденції розвитку ринку короткострокової оренди житла.
5. Перелік графічного матеріалу Схеми архітектури вебзастосунку та взаємодії його компонентів, інфографіки конкуруючиї платформ, макети інтерфейсу, блок-схеми алгоритмів серверної логіки
6. Зміст кваліфікаційної роботи Розділ1. Аналіз предметної області та вибір технологій. 1.1. Огляд ринку короткострокової оренди житла. 1.2. Вибір стека технологій для реалізації вебзастосунку. 1.3. Аналіз подібних вебдодатків та їхніх функціональних можливостей. Розділ 2. Розробка клієнтської частини вебзастосунку. 2.1. Розробка інтерфейсу користувача за допомогою React і Tailwind CSS. 2.2. Імплементація аутентифікації та авторизації з використанням NextAuth. 2.3. Дизайн інтерфейсу та UX. Розділ 3. Розробка серверної частини вебзастосунку. 3.1. Конфігурація та робота з базою даних MongoDB. 3.2. Реалізація серверної логіки з використанням Prisma ORM. 3.3. Забезпечення безпеки та обробка помилок на сервері.

7. Календарний план виконання роботи

№ п/п	Назва етапів кваліфікаційноїроботи	Строк виконання етапів роботи	
		За планом	Фактично
1	2	3	4
1	Вибір теми, підбір літературних джерел, формування структури кваліфікаційної роботи	До 15.10.2023	До 15.10.2023
2	**І розділ** (розкрити сутність і стан дослідження наукової проблематики роботи, після чого навести її загальну характеристику; огляд літератури за темою і вибір конкретних напрямів досліджень)	29.12.2023	29.12.2023
3	**ІІ розділ** (аналіз експериментальної частини про методики досліджень)	16.02.2024	16.02.2024
4	**ІІІ розділ** (проведені теоретичні і (або) експериментальні дослідження та проектні рішення)	16.04.2024	16.04.2024
5	**Висновки**	30.04.2024	30.04.2024
6	Доопрацювання роботи з урахуванням зауважень наукового керівника	10.05.2024	10.05.2024
7	Отримання відгуку керівника	30.05.2024	30.05.2024
8	Отримання зовнішньої рецензії	03.06.2024	03.06.2024

8. Дата видачі завдання «23» жовтня 2023

9. Керівник кваліфікаційної роботи

(підпис) Омельчук А.А.
(прізвище, ініціали)

10. Керівник освітньо-професійної програми

(підпис) Ратушняк Т.В.
(прізвище, ініціали)

11. Завдання до виконання отримав здобувач вищої освіти

(підпис) Підько А.С.
(прізвище, ініціали)

АНОТАЦІЯ

Створення full-stack вебзастосунку для короткострокової оренди житла з використанням технологій React, Tailwind, Prisma, MongoDB та NextAuth.

Мета роботи полягає в розробці та реалізації вебзастосунку для короткострокової оренди житла з метою вивчення процесу розробки сучасних вебзастосунків з використанням великої кількості інструментів.

Завданнями роботи є:

- Проведення аналізу предметної області.
- Розробка естетичного та візуально привабливого дизайну.
- Створення функціонального вебсайту.
- Впровадження адаптивного дизайну.
- Оптимізація роботи вебсайту.

Предметом дослідження є процес створення дизайну та реалізація сайту.

Об'єктом дослідження є вебдизайн та веброзробка.

Базою дослідження є вебсайти провідних платформ для оренди житла, таких як Booking.com, Vacation Rentals by Owner, або Vrbo, Airbnb.

За результатами дослідження сформульовані методика та етапи реалізації вебсайту.

Одержані результати можуть бути використані компаніями, що надають послуги гаранта між власником житла та орендарем, а також для подальшого розвитку та вдосконалення подібних платформ, що сприятиме розширенню аудиторії та покращенню якості наданих послуг.

Кваліфікаційна робота містить 58 сторінок, 16 рисунків, список використаних джерел містить 52 найменування.

Ключові слова: короткострокова оренда, вебзастосунок, React, Tailwind CSS, Prisma, MongoDB, NextAuth, онлайн-платформа, онлайн, доступність.

Рік виконання кваліфікаційної роботи 2024

Рік захисту роботи 2024

ANNOUNCEMENT

Creation of a full-stack web application for short-term housing rental using React, Tailwind, Prisma, MongoDB, and NextAuth technologies.

The purpose of the work is to develop and implement a web application for short-term housing rental in order to study the process of developing modern web applications using a large number of tools.

The tasks of the work are:

- Analyzing the subject area.
- Development of aesthetic and visually appealing design.
- Creating a functional website.
- Implementation of adaptive design.
- Optimization of the website.

The subject of the study is the process of creating a design and implementing a website.

The object of research is web design and web development.

The basis of the study is the websites of the leading vacation rental platforms, such as Booking.com, Vacation Rentals by Owner, or Vrbo, Airbnb.

Based on the results of the study, the methodology and stages of website implementation were formulated.

The results can be used by companies that provide guarantor services between landlords and tenants, as well as for the further development and improvement of such platforms, which will help expand the audience and improve the quality of services provided.

The qualification work consists of 58 pages, 16 figures, and the list of references includes 52 titles.

Keywords: short-term rental, web application, React, Tailwind CSS, Prisma, MongoDB, NextAuth, online platform, online, accessibility.

Year of qualification work 2024

Year of work defense 2024

Зміст

Вступ 3
Розділ 1. Аналіз предметної області та вибір технологій 5
1.1.Огляд ринку короткострокової оренди житла 5
1.2. Вибір стека технологій для реалізації вебзастосунку 6
1.3 Аналіз подібних вебдодатків та їхніх функціональних можливостей 9
Розділ 2. Розробка клієнтської частини вебзастосунку 15
2.1. Розробка інтерфейсу користувача за допомогою React і Tailwind CSS 15
2.2. Імплементація автентифікації з використанням NextAuth 20
2.3. Дизайн інтерфейсу та UX 26
Розділ 3. Розробка серверної частини вебзастосунку 31
3.1. Конфігурація та робота з базою даних MongoDB 31
3.2 Реалізація серверної логіки з використанням Prisma ORM 37
3.3 Забезпечення безпеки та обробка помилок на сервері 42
Висновки 51
Список використаних джерел 53

Вступ

В умовах стрімкого розвитку інформаційних технологій та зростання попиту на послуги короткострокової оренди житла, створення повнофункціонального вебзастосунку для забезпечення ефективної взаємодії між орендодавцями та орендарями набуває особливої актуальності. Ця платформа буде розроблена з використанням сучасних технологій, включаючи React, Tailwind, Prisma, MongoDB та NextAuth. Розробка включає автентифікацію користувачів, ключові функції, такі як пошук, бронювання, управління орендою, управління даними та забезпечення безпеки для користувачів.

По суті, це повнофункціональний вебзастосунок короткострокової оренди, розроблений з використанням React, Tailwind, Prisma, MongoDB та NextAuth. Він буде містити в собі:

- Оцінка предметної області та вибір найкращого технологічного стека для розробки вебдодатку

- Окреслення архітектури та розробка клієнтських компонентів на React та Tailwind CSS

- Реалізація серверної частини додатка: робота з MongoDB та Prisma ORM

- Прив'язка аутентифікації та авторизації користувачів за допомогою NextAuth

- Проєктування доступного, інтуїтивно зрозумілого інтерфейсу користувача з дотриманням принципів UI/UX дизайну

- Тестування розробленого вебдодатка з аналізом його ефективності та надійності

Об'єкт: Процес розробки вебдодатка

Предмет дослідження: Використання технологій React, Tailwind, Prisma, MongoDB, NextAuth для створення повноцінного вебдодатка.

Методи дослідження: аналіз предметної області, моделювання архітектури додатка, розробка з використанням обраного стека технологій, тестування та оцінка результатів.

Це є однією з особливостей, що забезпечує наукову новизну — оригінальна архітектура вебдодатку, виконана з використанням актуальних технологій та підходів

до розробки, які дозволяють забезпечити високу продуктивність, масштабованість та надійність системи.

Практичне значення отриманих результатів визначається можливістю використання розробленого вебдодатку для автоматизації та оптимізації процесів, пов'язаних з короткостроковою орендою житла. Це підвищує зручність взаємодії з орендодавцями та орендарями на практиці.

Інформаційну базу дослідження складають наукові праці вітчизняних та закордонних авторів з питань веброзробки та використання технологій React, Tailwind, Prisma, MongoDB, NextAuth, а також офіційна документація за цими технологіями. Загальна структура та обсяг роботи. Кваліфікаційна робота складається зі вступу, трьох розділів, висновків, списку використаних джерел та додатків.

Під час підготовки своєї випускної роботи я представив результати своїх досліджень на конференції молодих вчених «Математичні методи та цифрові технології в економіці». Під час конференції мої тези отримали високу оцінку, а також було відзначено практичне значення та інноваційність запропонованих рішень.

Участь у конференції дала можливість не лише поділитися своїми дослідженнями з академічною громадськістю, а й отримати корисні поради та рекомендації від професіоналів у данній галузі. Це сприяло подальшому поліпшенню мого проекту, а також показало його актуальність та практичне значення у сучасних умовах цифровоії економіки.

Розділ 1. Аналіз предметної області та вибір технологій

1.1. Огляд ринку короткострокової оренди житла

Ринок короткострокової оренди, включаючи Airbnb, Vrbo та HomeAway, буквально вибухнув за останні кілька років. Він дозволяє власникам будинків або кімнат здавати їх в оренду мандрівникам на короткі терміни — як правило, менше ніж на 30 днів [1]. Розвиток економіки спільного використання та зростання попиту на справжні враження від подорожей - це два фактори, які розширюють ринок короткострокової оренди [55]. У цьому розділі кваліфікаційній роботі я зроблю огляд ринку короткострокової оренди, обговоривши його зростання, вплив на традиційну готельну індустрію та регуляторні виклики.

Зростання ринку

Ринок короткострокової оренди житла експоненціально зростає з моменту заснування Airbnb у 2008 році. Станом на 2018 рік на Airbnb було розміщено понад 6 мільйонів оголошень по всьому світу, у понад 81 000 містах та 191 країні [3]. Дохід компанії у 2017 році склав $2,6 млрд[24]. Vrbo та HomeAway не стали винятком з цього правила: Expedia Group повідомила про 24% зростання доходів від сегмента оренди житла для відпустки у 2018 році[17].

Зважаючи на явні ознаки зростання ринку короткострокової оренди житла, можна легко сказати, що це негативно впливає на стару традиційну готельну індустрію. Дослідження Zervas [55], наприклад, показує, що збільшення кількості оголошень на Airbnb у Техасі на 1% зменшило доходи готелів на 1,6%. Також стверджується, що зростання короткострокової оренди боляче вдарило по менш дорогих готелях і хостелах, які не орієнтувалися на бізнес-клієнтів. Однак критики вважають, що цільовим клієнтом у випадку ринку короткострокової оренди є зовсім інший сегмент мандрівників, і його зовсім не можна порівнювати з клієнтами готелів [22].

Але разом зі зростанням ринку короткострокової оренди настав і затяжний період регуляторних перевірок. Міста та місцеві органи влади запроваджують правила спільного використання житла, щоб захистити орендний фонд, забезпечити дохід та керувати використанням жилої площі в зоні їх повноважень [27]. Наприклад, у багатьох містах існують податки на проживання, вимоги щодо реєстрації та днів, на які можна здавати житло в оренду[35]. Цей новий набір правил створив складні умови як для платформ короткострокової оренди, так і для власників нерухомості.

У висновку можна заключити невпинно зростає останні роки й прогнозується що далі нічого у цьому тренді не зміниться. Сплеск короткострокової оренди житла розпочався зовсім нещодавно завдяки трьом факторам:створенням відповідних платформ, розвитку тренду спільного використання та мандрівникам, які прагнуть автентичних вражень. З одного боку, це вплинуло на традиційну готельну індустрію, виникли проблеми з державним регулюванням, оскільки місцеві органи влади намагаються контролювати ринок та надавати перевагу більш звичним типам взаємодії орендодавця та орендатора, але з розвитком ринку зацікавленим сторонам потрібно буде знайти баланс між перевагами короткострокової оренди та необхідністю її регулювання

1.2. Вибір стека технологій для реалізації вебзастосунку

Вибір відповідного технологічного стеку має критично важливе значення для розробки вебдодатку Для реалізації проєкту слід обрати відповідний стек технологій, який має забезпечити його успіх, супроводжуваність та масштабованість [19]. Технологічний стек — це мова програмування та набір інструментів для розробки додатків [50].

Як основну бібліотеку для реалізації фронтенд-частини було обрано React. React - це популярна бібліотека JavaScript з відкритим вихідним кодом, для створення користувацьких інтерфейсів, яка розроблена та підтримується Facebook [18]. Вона дозволяє дуже легко генерувати різні необхідні багаторазові компоненти інтерфейсу

та ефективно оновлювати їх при зміні даних[8]. Будучи компонентно-орієнтованим і використовуючи віртуальний DOM, React є дуже ефективним і гнучким [12].

По суті, React є декларативним, тому ви лише визначаєте, що має бути показано в інтерфейсі, коли відбувається якась подія у стані додатка [53]. Тому немає прямої необхідності маніпулювати DOM, що призводить до зменшення самописного коду та меншої кількості помилок [48]. Він містить багато бібліотек та інструментів, що дозволяє розробникам легко розширювати функціональність. [8].

CSS Framework: Tailwind

В якості CSS-фреймворку я буду використовувати Tailwind CSS, щоб спростити процес стилізації та забезпечити узгоджений дизайн у всьому вебдодатку. Tailwind - це CSS-фреймворк, що легко налаштовується та надає набір попередньо визначених класів для швидкої розробки користувацького інтерфейсу [49]. На відміну від популярних CSS-фреймворків, таких як Bootstrap або Material Design Lite, які постачаються з великою кількістю попередньо розроблених компонентів, створених на основі низькорівневого дизайну, Tailwind можна розглядати як лише набір утилітарних класів, які згодом можна компонувати для створення власного дизайну [29].

Однією з найбільших переваг використання Tailwind є гнучкість. Він дає розробникам можливість легко змінювати конфігурацію відповідно до конкретних вимог дизайну проєкту [49]. Підхід Tailwind, орієнтований насамперед на корисність, також сприяє більш зручному обслуговуванню та повторному використанню кодової бази, оскільки стилізація відокремлена від структури HTML [9].

Бекенд: Prisma та MongoDB

Основними технологіями бекенду, що використовуються при створенні платформи, є Prisma та MongoDB. " Prisma - це сучасний інструментарій для роботи з базами даних, який спрощує доступ до баз даних та управління ними для розробників додатків "[42]. Prisma надає розробнику інтуїтивно зрозумілий та безпечний за типом, API, для запитів та усуває поширені помилки, пов'язані з традиційними ORM. Prisma підтримує різні бази даних, серед яких MongoDB, популярну базу даних NoSQL, відому своєю масштабованістю, гнучкістю та продуктивністю [34].

Документна модель даних, яка підтримується MongoDB, добре підходить для платформи короткострокової оренди, оскільки це дозволить зберігати та отримувати складні, досить ієрархічні типи структур даних[7]. Поєднання Prisma з MongoDB дозволяє інтегрувати функції Prisma, такі як управління схемами баз даних, валідація даних та автоматична міграція баз даних, з масштабованістю та гнучкістю, які пропонує MongoDB [42].

Для обробки автентифікації та авторизації користувачів було обрано бібліотеку NextAuth. NextAuth - це універсальне рішення для автентифікації користувачів. Легко налаштовує потоки автентифікації, забезпечує безпечний спосіб автентифікації в додатках NextJs. Підтримує всі види автентифікації, від електронної пошти/пароля до кастомних провайдерів, що полегшує інтеграцію з існуючими механізмами автентифікації [38].

Також варто зазначити, що перевага NextAuth полягає в простоті та легкості інтеграції. Він абстрагує складнощі, пов'язані з реалізацією автентифікації, дозволяючи розробнику зосередитися на створенні головної частини вебдодатку. NextAuth також надає набір вбудованих функцій захисту, таких як захист CSRF та безпечна обробка сеансів, щоб гарантувати, що процес автентифікації є безпечним та захищеним від поширених атак [38].

Синергія між цими технологіями дозволяє спростити процес розробки функцій та можливостей для вебдодатку. Обраний стек також стимулюватиме повторне використання коду, підтримку та масштабованість, завдяки чому вебдодаток може трансформуватися та масштабуватися відповідно до потреб бізнесу. Обраний стек технологій— це React, Tw, Prisma, MongoDB та NextAuth, об'єднані в одне ціле, щоб сплести кастомне рішення, яке чітко відповідає вимогам, ефективне, приємне в роботі, обслуговуванні та масштабуванні. Цей стек технологій вбирає в себе найкращі моменти кожного з них і забезпечує безперебійну роботу додатку. Обраний стек технологій забезпечить хорошу основу для вебзастосунку, який ефективно та гнучко задовольнятиме сучасні потреби на ринку короткострокової оренди в майбутньому.

1.3 Аналіз подібних вебдодатків та їхніх функціональних можливостей

На шляху до розробки повнофункціонального вебдодатку для платформи короткострокової оренди житла необхідно дослідити наявні подібні платформи та їхні функціональні можливості. Це дослідження надасть цінну інформацію про галузеві стандарти та найкращі практики, що, в свою чергу, забезпечить потенційні можливості для вдосконалення. Вивчаючи функції, користувацький досвід і технічні фактори існуючих платформ, розробникам можуть приймати обґрунтовані рішення під час розробки та впровадження власного вебдодатку[11]. У цьому розділі ми зосередимося на аналізі трьох відомих платформ короткострокової оренди житла: Airbnb, Vrbo та Booking.com.

Airbnb - це, мабуть, одна з найвідоміших і найуспішніших платформ короткострокової оренди, яка має колосальне глобальне покриття і присутня у понад 220 країнах і регіонах світу[4]. Її унікальна перевага полягає в тому, що вона надає унікальний та автентичний досвід подорожей, поєднуючи господарів з мандрівниками, які потребують житла[23]. Платформа Airbnb має інтуїтивно зрозумілий, чистий дизайн і є дуже зручною для користувачів. Вона приваблює сучасним виглядом, що демонструє зображення нерухомості у високій роздільній здатності[30]. Користувачі можуть задавати фільтри для пошуку за місцем розташування, датою, ціною, типом проживання, зручностями тощо, щоб знайти житло найбільш відповідне до своїх вимог і вподобань[4]. Більше того, Airbnb надає можливість переглянути на карті результати пошуку та деталі про близькість житла до визначних місць[28]. Однією з ключових особливостей Airbnb є система відгуків, за допомогою якої господарі та гості діляться своїм досвідом. Це зміцнює довіру та прозорість серед користувачів і допомагає їм приймати обґрунтовані рішення при бронюванні. Airbnb також має систему обміну повідомленнями, яка дозволяє гостям спілкуватися з господарями, а також безпечну систему обробки платежів, яка обробляє транзакції та депозити.



Рис 1.1 Статистичні дані про Airbnb

Додаток Airbnb розроблений за допомогою сучасного вебміксу, який включає React, Node.js та GraphQL[5]. React був використаний для розробки інтерфейсу, що дозволяє створювати багаторазові компоненти інтерфейсу користувача та ефективно оновлювати його. Airbnb також використовує Mobile Fisrt підхід до дизайну(спочатку робиться мобільна версія додатку і вже потім маштабується до пк версії), що гарантує, що платформа є повністю адаптивною та доступною на різних пристроях[30]. Внутрішня частина Airbnb була розроблена на основі архітектури мікросервісів для створення легко масштабованих і підтримуваних сервісів[6]. Для обробки на стороні сервера використовується Node.js, а також GraphQL для ефективного пошуку та маніпулювання даними. Airbnb також використовує різні рішення для зберігання даних, зокрема MySQL та Apache Hadoop, для обробки великих обсягів даних, що генеруються платформою.

Vacation Rentals by Owner, або Vrbo, є ще одним великим гравцем на онлайн-ринку короткострокової оренди житла. Він в основному зосередився на наданні будинків для відпочинку, квартир і кондомініумів для мандрівників і туристів. Сайт був заснований у 1995 році, але пізніше, у 2015 році, був придбаний Expedia Group, яка ще більше розширила присутність сайту в Сполучених Штатах і в кількох країнах Європи[45].

Користувацький інтерфейс Vrbo такий самий дружній, як і в Airbnb, чистий,сучасний, і зосереджений на відображенні високоякісних зображень

перелічених об'єктів нерухомості. Платформа пропонує ряд фільтрів пошуку, таких як місце розташування, дати, ціновий діапазон, тип житла та зручності, що дозволяє знайти житло, яке відповідає їхнім потребам і вимогам. Врбо також має можливість перегляду результатів пошуку на карті, тому користувач може ознайомитися з розташуванням оголошень у порівнянні з визначними пам'ятками в цьому районі. Як і Airbnb, Vrbo має систему відгуків, яка дозволяє гостям залишати відгуки про свій досвід. Ця система відгуків підвищує довіру та прозорість між платформою та користувачами, а отже, дозволяє користувачам приймати обґрунтовані рішення під час бронювання житла[14]. Vrbo також включає систему обміну повідомленнями, за допомогою якої власники нерухомості та гості можуть легко спілкуватися. Платформа обробки платежів також є безпечною, що дозволяє обробляти транзакції та вносити гроші.

Вебдодаток Vrbo створено з використанням Java, JavaScript та HTML5. З точки зору фронтенду, він був розроблений з дотриманням/врахуванням принципів адаптивного вебдизайну, який гарантує, що користувацький інтерфейс є доступним та ідеально працює на максимальній лінійці пристроїв та екранів різного розміру[51]. На внутрішньому рівні Vrbo застосовує сервіс-орієнтовану архітектуру, яка дозволяє розробляти модульні та масштабовані сервіси. Платформа використовує Java для обробки на стороні сервера, і різні рішення для зберігання даних, включаючи Oracle і MySQL для обробки даних, які генеруються платформою.

Booking.com є одним з провідних світових туристичних агентств, що надає своїм клієнтам послуги з бронювання житла, включаючи готелі, хостели, гостьові апартаменти та будинки для відпочинку. Компанія була заснована в 1996 році і має глобальну присутність, пропонуючи житло в більш ніж 220 країнах і територіях по всьому світу[10].

Інтерфейс користувача Booking.com розроблений таким чином, щоб бути простим та інтуїтивно зрозумілим, з акцентом на наданні користувачам безперешкодного досвіду бронювання На порталі є низка фільтрів для пошуку за місцем розташування, датою, ціновим діапазоном, типом нерухомості та зручностями,

щоб знайти житло, яке точно відповідає потребам і зручностям. Сайт також пропонує перегляд результатів пошуку на карті та розташування об'єктів відносно різних визначних пам'яток. Однією з основних функціональних можливостей сайту є розгалужена система відгуків, де користувачі можуть залишити свій детальний відгук про конкретне місце проживання. Більше того, система пропонує зірковий рейтинг на основі відгуків гостей кожного об'єкта, на який користувачі можуть посилатися при бронюванні[33]. Для власників нерухомості та гостей сайт пропонує систему обміну повідомленнями між ними, а також безпечну систему обробки платежів за бронювання, включаючи транзакції та депозити.

Вебдодаток Booking.com розроблено з використанням набору вебтехнологій, таких як Java, JavaScript та HTML5. Розробка фронтенду виконана з використанням принципів адаптивного вебдизайну, що робить кожен можливий елемент візуалізації доступним на різних пристроях з різними характеристиками екрану. Бекенд базується на архітектурі мікросервісів, що дозволяє розробку масштабованих і підтримуваних сервісів. Платформа використовує Java для обробки на стороні сервера та різні рішення для зберігання даних, включаючи MySQL і Cassandra, для обробки великих обсягів даних, що генеруються платформою.

Порівняння трьох платформ короткострокової оренди показало, що ці три платформи мають дуже багато схожих функцій і можливостей. Усі три пропонують інтуїтивно зрозумілий користувацький інтерфейс, дуже багаті пошукові фільтри, перегляд карт, системи відгуків, системи обміну повідомленнями та безпечну обробку платежів. Усе це підкреслює набір галузевих стандартів і найкращих практик, які сформувалися на ринку короткострокової оренди.

Однак між трьома платформами є деякі важливі відмінності: Airbnb чітко орієнтована на те, щоб пропонувати унікальний та автентичний досвід проживання, що видно з диверсифікованих типів нерухомості та міцних стосунків між господарями та гостями на платформі [23]. Vrbo розміщує інформацію про будинки для відпочинку та кондомініуми, орієнтовані на сімейних та групових мандрівників, які шукають більш традиційні варіанти розміщення. Booking.com, як правило, будучи онлайн-

туристичною агенцією, пропонує ширший спектр типів житла, що включає готелі та апартаменти, приділяючи при цьому більше уваги можливостям рецензування та рейтингування.

З технологічної точки зору, всі три компанії використовують новітні вебтехнології та архітектури, щоб зробити додаток масштабованим, високопродуктивним і таким, що легко оновлюється та підтримується. Однак існують відмінності у конкретних технологіях, що використовуються. Airbnb використовує React і меншою мірою базується на архітектурі Node.js для фронтенду та бекенду відповідно, тоді як Vrbo та Booking.com значною мірою покладаються на Java як основну мову для обробки на стороні сервера [51,10].

Аналіз відомих платформ короткострокової оренди житла дозволяє зібрати важливу інформацію для розробки та впровадження нового вебзастосунку в цій ніші. При розробці та створенні нової платформи необхідно забезпечити використання галузевих стандартів і найкращих практик, отриманих в результаті аналізу, таких як зручний та інтуїтивно зрозумілий інтерфейс користувача, потужні можливості пошуку з фільтрами, перегляд карт, надійні відгуки та рейтинги, обмін повідомленнями та можливості обробки платежів.

Однак також важливо визначити потенційні сфери для вдосконалення та диференціації (це поняття означає процес виділення продукту або послуги на ринку шляхом надання йому унікальних властивостей або характеристик, які відрізняють його від конкурентів). Наприклад, нова платформа може зосередитися на конкретній ніші на ринку короткострокової оренди, наприклад, на екологічно чистому житлі або нерухомості, придатній для віддаленої роботи [15]. Крім того, слід використовувати нові технології, такі як штучний інтелект і машинний інтелект, які надають надзвичайно персоналізовані рекомендації, що, в свою чергу, покращує користувацький досвід, як описано в [2].

З технічної точки зору вибір технологій повинен враховувати й інші фактори, такі як масштабованість, продуктивність і здатність ефективно розвивати технології, як зазначає [16]. Представлений стек є сучасним. Він пропонує сучасний та

ефективний стек для створення повноцінного вебдодатку. Реалізація забезпечить нову платформу модульною та підтримуваною архітектурою, адаптивними та простими у використанні інтерфейсами, ефективним управлінням даними, безпечною та надійною автентифікацією та авторизацією.

Розділ 2. Розробка клієнтської частини вебзастосунку

2.1. Розробка інтерфейсу користувача за допомогою React і Tailwind CSS

Розробка та реалізація інтуїтивно зрозумілого та естетичного користувацького інтерфейсу є ключовою вимогою для залучення користувачів і забезпечення безперебійної роботи додатку. Успішна взаємодія та навігація користувачів через додаток вимагає такого інтерфейсу [29]. У цьому розділі буде розглянуто розробку користувацького інтерфейсу з використанням бібліотеки React JavaScript для побудови користувацьких інтерфейсів та Tailwind CSS для адаптивного дизайну. Завдяки використанню цих технологій вебдодаток буде наділений модульним, динамічним та адаптивним дизайном, що підвищить задоволеність користувачів та забезпечить вищий рівень утримання користувачів на платформі [18,49].

React - це бібліотека JavaScript з відкритим вихідним кодом, розроблена Facebook для створення користувацьких інтерфейсів [43]. Вона отримала широке визнання та використання серед розробників завдяки компонентній архітектурі, віртуальній Document Object Model, та декларативному синтаксису. React дозволяє розробникам створювати інкапсульовані компоненти інтерфейсу користувача, які керують власним станом і можуть бути скомпоновані разом для створення складних користувацьких інтерфейсів. Однією з головних особливостей React є його компонентна архітектура. У React користувацький інтерфейс розбивається на невеликі компоненти, які є самодостатніми та багаторазовими, кожен з яких керує власним станом та рендерингом. Кожен компонент інкапсулює власну структуру, логіку та стилізацію, що робить кодову базу більш модульною та зручною для підтримки [48]. React-компоненти можуть бути вкладені один в одного, що дозволяє створювати складні користувацькі інтерфейси, які збираються з простіших і менших компонентів.

У контексті платформи короткострокової оренди користувацький інтерфейс можна розбити на різні компоненти, такі як пошукові рядки, списки об'єктів нерухомості, перегляд мап і форми бронювання. Коли ці елементи розробляються

окремо, а потім компонуються, загальний інтерфейс стає гнучким і легко модифікується. Тому, якщо потрібно внести зміни, наприклад, в елемент списку нерухомості, це можна зробити, не впливаючи на інші елементи користувацького інтерфейсу, а отже, з меншим ризиком.

Ще однією значною перевагою використання React є його віртуальна DOM, яка сприяє підвищенню продуктивності та ефективності оновлення користувацького інтерфейсу згідно з Chodorow[12]. Віртуальний DOM - це полегшена версія оригінального DOM в пам'яті. Коли стан додатка змінюється, React спочатку оновлює віртуальний DOM, а потім, розрахувавши мінімально потрібну кількість змін для оновлення реального DOM, алгоритм переходить до виконання необхідних для цього кроків. Оскільки прямі маніпуляції з реальним DOM можуть бути обчислювально дорогими, React ефективно оновлює користувацький інтерфейс, не створюючи вузьких місць під час виконання. Це особливо важливо для платформи короткострокової оренди, оскільки користувачі матимуть справу з великою кількістю даних на своїх екранах: пропозиціями нерухомості, фільтрами, які потрібно застосувати, і, звичайно, відображенням карти, на якій має бути огляд деяких об'єктів нерухомості та їхнього розташування.

React підтримує декларативний синтаксис, за допомогою якого розробники можуть описувати, як повинен виглядати інтерфейс користувача в залежності від поточного стану програми. Це робить код дуже читабельним і зрозумілим, оскільки метою є опис бажаного вигляду, а не покрокове виконання процесу отримання цього вигляду.

React-компоненти зазвичай складаються з JSX - синтаксичного розширення для JavaScript, яке дозволяє записувати HTML-подібні структури у ваші JavaScript-файли. JSX було запроваджено для покращення створення елементів. Оскільки структура і зміст компонента стають більш інтуїтивно зрозумілими і виразними завдяки JSX, його використовують для визначення структури і змісту компонентів. Логіка та рендеринг подаються в одному конкретному файлі, тому розробник може підтримувати код у простий, послідовний та ефективний спосіб.

Наприклад, компонент списку нерухомості може бути написаний за допомогою JSX і містити зображення, заголовок, опис, ціну та кнопку бронювання - і все це в одному автономному блоці

Tailwind CSS - це низькорівневий CSS-фреймворк з можливістю швидкого налаштування, який використовує підхід що дозволяє швидко стилізувати користувацькі інтерфейси за допомогою попередньо розроблених класів. Це відрізняє його від інших традиційних CSS-фреймворків, де ви отримуєте заздалегідь розроблені визначені компоненти. Він робить більший акцент на створенні низькорівневих утилітарних класів, які можна об'єднувати та впроваджувати для створення унікальних дизайнів[9]. Tailwind має багато невеликих одноцільових класів, які можна безпосередньо комбінувати в HTML-документі, щоб отримати бажаний стиль елементів. Ілюстрацією таких класів utility-first є "text-lg", що означає великий розмір тексту, або "bg-blue-500", що означає певний відтінок синього кольору фону. Застосовуючи такі класи безпосередньо до елемента HTML без написання будь-якої форми CSS, розробники можуть швидко і без особливих зусиль стилізувати свої компоненти.

Утилітарні класи Tailwind допомагають розробляти різні елементи з однаковим користувацьким інтерфейсом на всій платформі короткострокової оренди. Наприклад, рядок пошуку, картка нерухомості та кнопки отримують однакову обробку завдяки цьому набору utility-first классів, що водночас посилює однорідність та згуртованість. Зменшуючи потребу у великій кількості кастомних CSS, що збереже кодову базу чистою та зручною для обслуговування, водночас зменшуючи ризик конфліктів стилів.

Tailwind CSS розроблено з урахуванням особливостей адаптивного вебдизайну, тому в ньому є набір адаптивних класів утиліт, які дозволяють розробникам легко створювати адаптивні сторінки. Ці класи створені відповідно до стандартів «mobile-first», де стилі спочатку створюються для роботи на найменшому розмірі екрану, а потім розширюються за допомогою точок розриву для більшого екрану. Завдяки використанню цих адаптивних класів утиліт платформа яка буде розроблена, гарантує, що інтерфейс залишатиметься доступним і функціональним на всіх пристроях - від

смартфона до стаціонарного комп'ютера. Це особливо важливо для вебдодатків в індустрії гостинності, де користувачі часто шукають і бронюють житло за допомогою своїх мобільних пристроїв Однією з важливих причин використання Tailwind CSS у цьому випадку є його високий рівень кастомізації. Tailwind надає конфігураційний файл, де розробники можуть визначити власну систему дизайну, включаючи кольори, шрифти, інтервали та точки розриву. Це дозволяє створити унікальну візуальну ідентичність, яка відповідає принципам бренду та вподобанням користувачів. Гнучкість Tailwind також поширюється на можливість його використання разом з іншими CSS-фреймворками або кастомними CSS. Розробники можуть легко інтегрувати Tailwind в існуючі проєкти або використовувати його в поєднанні з іншими підходами до стилізації (Lazaris, 2020). Така гнучкість гарантує, що платформа для короткострокової оренди може адаптувати свій стиль у міру розвитку проєкту, не прив'язуючись до конкретного фреймворку чи системи дизайну

React і Tailwind CSS можна інтегрувати разом, щоб створити дуже потужну та ефективну систему розробки. Потужність React в його архітектурі, заснованій на компонентах, можна об'єднати з підходом Tailwind, орієнтованим на утилітарність, для розробки модульних, багаторазових та візуально узгоджених користувацьких інтерфейсів. Таким чином, React-розробники можуть додавати класи утиліт Tailwind CSS безпосередньо до JSX-елементів, які використовуються всередині їхніх React-компонентів. Це дійсно краще для стилізації, оскільки структура та стилізація будь-якого компонента зберігаються в одному файлі, що спрощує процес стилізації, фактично збільшуючи швидкость у написанні стилів щонайменше на 30%. Наприклад, компонент картки нерухомості може бути створений на JSX за допомогою класів утиліти Tailwind, щоб стилізувати компонент, не відхиляючись від решти системи дизайну.

Більше того, розробники можуть використовувати PostCSS разом з бібліотеками CSS-in-JS, які дозволяють налаштовувати інтеграцію між React і Tailwind CSS. PostCSS допомагає в подальшій обробці утилітарних класів Tailwind, пропонуючи такі

переваги, як автопрефіксинг та видалення невикористаних стилів. Для визначення стилів у React-компонентах будуть використовуватися бібліотеки на кшталт styled-components або Emotion, щоб забезпечити модульність та інкапсуляцію стилів. Щоб створити найкращий можливий інтерфейс, слід дотримуватися деяких практик і методів оптимізації продуктивності. Одна з найкращих практик включає створення невеликих, сфокусованих компонентів, які виконують одну функцію або відображають одну візуальну частину. Коли користувацький інтерфейс декомпозується на невеликі компоненти, які можна використовувати повторно, кодова база стає зручною для підтримки, а продуктивність додатку — зростає. І це також жахлива ідея — вкладати компоненти один в одного, де структура дуже глибока, тому що це призведе до зниження продуктивності, а код стане нечитабельним.

Ще одна річ, яку варто взяти до уваги - це оптимізація продуктивності під час рендерингу. Віртуальний DOM у React є ефективним, але розробники можуть зробити кроки для оптимізації продуктивності, мінімізуючи кількість непотрібних повторних рендерингів та використовуючи методи запам'ятовування та лінивого завантаження в процесі розробки[19]. Memoization , в свою чергу, - це спосіб кешування результатів дорогих(великих) обчислень або рендерингу. Ліниве завантаження використовується, коли компоненти завантажуються тільки тоді, коли це потрібно, таким чином зменшуючи початковий розмір пакета, що ще більше збільшує швидкість завантаження додатку. Це вбудовані функції оптимізації продуктивності, такі як очищення імен класів та струшування дерева. Очищення імен класів виконується у фінальній збірці, де видаляються всі невикористані класи CSS, зменшуючи розмір файлу CSS, тим самим зменшуючи час завантаження файлу. Під час «трясіння» дерева невикористаний JavaScript-код у пакунку видаляється щоб оптимізувати продуктивність додатку.

Без тестування немає жодних гарантій щодо якості реалізації інтерфейсу користувача. React "з коробки" оснащений купою утиліт для тестування, наприклад, Jest та React Testing Library. Розробники можуть писати модульні та інтеграційні тести для своїх компонентів. Тестування поведінки компонента та рендерингу в ізоляції та

комбінації дозволяє розробникам виявити якомога більше можливих проблем на ранніх стадіях розробки та підтримувати стабільний інтерфейс користувача без помилок.

Ще одним важливим фактором при розробці користувацького інтерфейсу є його доступність. Платформа короткострокової оренди повинна бути розроблена і впроваджена з урахуванням доступності, щоб гарантувати, що платформою може користуватися кожен, незалежно від його здатності використовувати допоміжні технології. React та Tailwind CSS пропонують технічні рішення та найкращі практики для створення доступного користувацького інтерфейсу в рамках використання семантичного HTML, впровадження альтернативного тексту для зображень та забезпечення належної клавіатурної навігації [43,49]. Дотримуючись рекомендацій щодо доступу до додатків, розробники можуть створити та протестувати інклюзивний досвід для всіх користувачів, зокрема й тих, що користуються допоміжними технологіями.

Важливим аспектом створення успішного вебдодатку є розробка користувацького інтерфейсу. Надійний і зручний інтерфейс необхідний для того, щоб платформа короткострокової оренди стала конкурентоспроможною і успішною на ринку. Компонентна архітектура, віртуальний DOM та декларативний синтаксис дають можливість створити модульний, ефективний та підтримуваний користувацький інтерфейс.

Інтеграція React в Tailwind CSS, разом з його кращими практиками для оптимізації продуктивності, допоможе забезпечити плавний, чистий дизайн на платформі короткострокової оренди. Користувацький інтерфейс буде інтуїтивно зрозумілим, чистим, адаптивним, інклюзивним і корисним для багатьох завдяки акценту на тестуванні та доступності.

2.2. Імплементація автентифікації з використанням NextAuth

Безпечна автентифікація та авторизація користувачів є двома найважливішими функціями будь-якого вебдодатку для оренди. NextAuth - одна з найпопулярніших бібліотек автентифікації для додатків на Next.js, оскільки вона дозволяє розробникам

реалізовувати потоки, пов'язані автентифікацією та авторизацією, з мінімальними зусиллями [38]. У цьому розділі ми обговоримо особливості, переваги та недоліки, а також найкращі практики реалізації автентифікації та авторизації.

NextAuth - це бібліотека з відкритим вихідним кодом, яка використовується для реалізації та роботи з автентифікацією в додатках Next.js. NextAuth має широкий спектр функцій і підтримку численних видів автентифікації, таких як: email/пароль, OAuth та інші. Він також постачається з інструментами та ключами, необхідними для управління сеансами та безпечного зберігання токенів. Основна причина, чому NextAuth є однією з найкращих для реалізації функцій автентифікації та авторизації на цій платформі короткострокової оренди, полягає в її гнучкості та простоті у використанні.

Як зазначено у навчальних ресурсах компанії Vercel, творців Next.js: "NextAuth.js - це повне рішення для автентифікації з відкритим вихідним кодом для додатків Next.js, розроблене з нуля для підтримки Next.js і Serverless". Іншими словами, він тісно інтегрований з Next.js, щоб отримати готові функції, такі як рендеринг на стороні сервера та маршрути API, одночасно гарантуючи, що потік автентифікації залишається безпечним і доцільним[1]. NextAuth підтримує широкий спектр провайдерів автентифікації, що дозволяє розробникам вибрати найбільш оптимальний для їхніх додатків. Для платформи короткострокової оренди можна використовувати автентифікацію за допомогою електронної пошти/пароля, а також OAuth-провайдерів, таких як Google і Facebook, щоб користувачі могли вибирати впізнаваних провайдерів, з якими їм зручно працювати, для полегшення доступу. Автентифікація за допомогою електронної пошти та пароля - це найпростіший тип автентифікації користувачів. Вона дозволяє користувачам створити обліковий запис, а потім отримати доступ до автентифікованої частини додатку, використовуючи свою електронну адресу та пароль, створений ними самими. NextAuth полегшує впровадження автентифікації надаючи вбудовані функції для обробки реєстрації користувачів, входу і навіть управління паролями.

Щоб реалізувати автентифікацію за електронною поштою/паролем на нашій платформі, нам потрібно використовувати EmailProvider від NextAuth і налаштувати його параметри, включаючи налаштування поштового сервера та шаблони листів. Деталі налаштування EmailProvider в NextAuth показані в наступному фрагменті коду:

```
import NextAuth from 'next-auth';
import EmailProvider from 'next-auth/providers/email';

export default NextAuth({
  providers: [
    EmailProvider({
      server: process.env.EMAIL_SERVER,
      from: process.env.EMAIL_FROM,
    }),
  ],
```

Рис 2.1 Налаштування EmailProvider в NextAuth

OAuth - це один із широко використовуваних варіантів використання протоколу автентифікації під час входу в додатки. Він дає можливість користувачам входити в додатки на основі облікового запису, який вони вже мають у різних постачальників онлайн-послуг, таких як Google, Facebook або Twitter [40]. NextAuth постачається з вбудованою підтримкою OAuth, що полегшує розробникам додавання входу до платформи короткострокової оренди за допомогою інших сервісів.

Щоб додати варіанти до OAuth, потрбіно внести потрібні сервіси у файлі конфігурації NextAuth. Наприклад, щоб увімкнути автентифікацію Google і Facebook, можна використовувати таку конфігурацію:

```
import NextAuth from 'next-auth';
import GoogleProvider from 'next-auth/providers/google';
import FacebookProvider from 'next-auth/providers/facebook';

export default NextAuth({
  providers: [
    GoogleProvider({
      clientId: process.env.GOOGLE_CLIENT_ID,
      clientSecret: process.env.GOOGLE_CLIENT_SECRET,
    }),
    FacebookProvider({
      clientId: process.env.FACEBOOK_CLIENT_ID,
      clientSecret: process.env.FACEBOOK_CLIENT_SECRET,
    }),
  ],
```

Рис 2.2 Імплементація автентифікації через Google і Facebook

Пропонуючи таку різноманітність механізмів автентифікації, платформа короткострокової оренди забезпечить кращий користувацький досвід для більшої кількості користувачів.

NextAuth постачається з надійними можливостями управлінням сеансами, що робить роботу з користувацькими сесіями безпечною та простою. Після автентифікації користувача NextAuth створить об'єкт сеансу з даними автентифікованого користувача, який, як мінімум, містить ідентифікатор користувача, його електронну пошту та будь-які інші дані, отримані від сервіса автентифікації.NextAuth зберігає дані сеансу в захищеному JWT-зашифрованому файлі cookie, який потім надсилається клієнту як HTTP-файл cookie. Цей метод забезпечує безпеку даних сеансу, коли дані зберігаються на стороні клієнта як недоступні і не піддаються вразливостям на стороні клієнта, наприклад, міжсайтовому скриптингу, за допомогою якого стає можливим виконання ворожого коду, таким чином зменшуючи ризики перехоплення сеансу та інших вразливостей безпеки. NextAuth.js пропонує доступ до об'єкта сесії на фронтенді - на сторінках і маршрутах API платформи короткострокової оренди за допомогою хука useSession і на бекенді за допомогою функції getSession. Нижче, у фрагментах коду, я показую, як здійснюється доступ до об'єкта сесії у фронтенді та бекенді:

```jsx
import { useSession } from 'next-auth/react';

function ProfilePage() {
  const { data: session, status } = useSession();

  if (status === 'loading') {
    return <div>Loading...</div>;
  }

  if (status === 'unauthenticated') {
    return <div>Please log in to view your profile.</div>;
  }

  return (
    <div>
      <h1>Profile Page</h1>
      <p>Welcome, {session.user.email}!</p>
    </div>
  );
}
```

Рис 2.3 Фронтенд (за допомогою хука useSession):

```jsx
import { getSession } from 'next-auth/react';

export default async function handler(req, res) {
  const session = await getSession({ req });

  if (!session) {
    res.status(401).json({ error: 'Unauthorized' });
    return;
  }
```

Рис 2.4 Бекенд (за допомогою функції getSession):

Окрім автентифікації, NextAuth забезпечує авторизацію та контроль доступу на основі ролей. Це робиться для того, щоб переконатися, що дозволи, надані користувачам, контролюються, а доступ до ресурсів обмежується певною роллю користувача. Впроваджуючи RBAC, цей захід гарантує, що користувачі можуть виконувати в системі лише ті операції, на які вони мають право відповідно до своїх ролей - власників нерухомості, орендарів або адміністраторів. NextAuth підтримує визначення користувацьких функцій зворотного виклику та хуків, які можна використовувати для застосування логіки авторизації. Наприклад, щоб обмежити доступ до деяких маршрутів API за роллю користувача, потрібно визначити кастомну

функцію проміжного програмного забезпечення, яка перевіряє роль користувача і відповідно авторизує або відхиляє доступ:

```
import { getSession } from 'next-auth/react';

const requireRole = (role) => async (req, res, next) => {
  const session = await getSession({ req });

  if (!session || session.user.role !== role) {
    res.status(403).json({ error: 'Forbidden' });
    return;
  }

  next();
};
```

Рис 2.5 Функція перевірки ролі користувача

Це стає можливим завдяки зворотним викликам і хукам NextAuth, які тепер будуть впроваджені для реалізації RBAC, щоб тільки відповідні ролі отримували дозвіл на відповідну реалізацію. Це підвищить безпеку на платформі короткострокової оренди, але при цьому вона все ще залишається інтегрованою в загальну програму.

Для захисту даних користувачів та запобігання несанкціонованому доступу необхідно дотримуватися найкращих практик щодо безпечної конфігурації, як описано в документації NextAuth. Сюди входить наступне:

- Безпечне шифрування та підпис сеансу, щоб переконатися, що ключи є надійними та різними кожного разу.
- Налаштування відповідного часу закінчення сеансу і терміну дії токенів.
- Зберігання та керування важливими значеннями конфігурації, наприклад, ключами API та секретами клієнта, у безпечний та відповідний спосіб за допомогою змінних середовища або безпечних систем керування ключами.

Щоб захистити дані користувача під час передачі, платформа використовує безпечні протоколи зв'язку, такі як HTTPS і SSL/TLS для захисту даних під час

передачі. За замовчуванням NextAuth підтримує HTTPS при розгортанні на таких платформах, як Vercel, або при використанні кастомного сервера з увімкненим HTTPS.

Також для безпеки важливі високі стандарти до паролів та їх зберігання/використання:

- Високі вимоги до паролів, мінімальна довжина та складність.
- Надійне хешування та солізування паролів перед тим, як зберігати їх у базі даних (щоб їх не можна було вгадати).
- Не зберігайте паролі у вигляді простого тексту.
- Реалізування потоків для безпечного скидання паролів за допомогою одноразових токенів/посилань.

У випадку NextAuth, безпечна робота з паролями вбудована, як хешування та безпечні потоки для скидання паролів, серед інших функцій.

Для безпеки важливо переконатися, що на платформі короткострокової оренди оновлені всі доступні пакети для NextAuth та його залежностей. Періодичний моніторинг повинен проводитися для підтвердження наявності повідомлень про вразливості або будь-які пов'язані з ними проблеми безпеки щодо NextAuth та пов'язаних з ним бібліотек.

При дотриманні всіх цих практик і належному виконанні міркувань безпеки, платформа короткострокової оренди зможе уникнути зливів даних користувачів та забезпечити безпечну і надійну систему автентифікації та авторизації для обробки даних користувачів і підтримувати довіру до користувачів в її межах.

2.3. Дизайн інтерфейсу та UX

Дизайн інтерфейсу та користувацький досвід відіграють вирішальну роль у успіху та прийняті платформи користувачами, допомагають підкреслити зручність та інтерактивність вебодатку. Для створення дружнього користувацького інтерфейсу необхідно розробити дизайн інтерфейсу, який зробить користувача більш задоволеним, зацікавленим і лояльним[25]. У цьому розділі ми обговоримо принципи

та найкращі практики дизайну інтерфейсу та UX для платформи короткострокової оренди з метою створення зручного та стимулюючого користувацького інтерфейсу.

Дизайн інтерфейсу та користувацький досвід (UX) вебдодатку є дуже важливими факторами загального успіху додатку. Як пояснюють Hassenzahl and Tractinsky, «користувацький досвід (UX) - це створення змістовної і цінної взаємодії між користувачем і продуктом або послугою»[25](с. 95). У випадку платформи короткострокової оренди позитивний користувацький досвід може призвести до підвищення задоволеності користувачів, вищого рівня конверсії та кращого утримання користувачів [20].

Було проведено багато досліджень, які доводять, що перше враження від вебсайту формується протягом декількох секунд після його використання [31]. Пізніше дослідження, проведене, показало, що користувачеві потрібно максимум 50 мілісекунд, щоб сформувати первинну думку про візуальну привабливість вебсайту. Це підкреслює важливість гарного візуального інтерфейсу для платформи короткострокової оренди.

Хороший дизайн користувацького інтерфейсу зменшує когнітивне навантаження на розум і підвищує зручність використання, щоб користувач міг зорієнтуватися на платформі, знайти те, що йому потрібно, і, зрештою, виконувати заплановані дії без стресу. За словами Norman, D. [37], "Хороший дизайн насправді набагато важче помітити, ніж поганий, частково тому, що хороший дизайн настільки добре відповідає нашим потребам, що він непомітний" (с. 1).

Розробка ефективного та зручного інтерфейсу для платформи короткострокової оренди вимагає дотримання наступних принципів дизайну інтерфейсу:

1. Послідовність і стандарти. На платформі короткострокової оренди необхідно дотримуватися узгодженого та передбачуваного макета, кольорової гами, типографіки та схеми взаємодії. Стандарти в дизайні зменшать потребу людей замислюватися над елементами інтерфейсу та діями, а коли вони будуть впроваджені, то зменшать криву навчання та когнітивне навантаження [36].

2. Наочність і зворотній зв'язок. Інтерфейс повинен постійно показувати стан системи і в той же час надавати користувачеві своєчасний зворотній зв'язок про його дії. Якщо користувачі взаємодіють з додатком, наприклад, здійснюючи пошук квартири або бронюючи нерухомість, такі дії можуть бути показані в процесі виконання за допомогою відповідного візуального зворотного зв'язку та інформації. Це допомагає користувачам тримати систему під контролем, а також інформувати їх про наслідки своїх дій.

3. Запобігання помилкам та їх виправлення. Розробка інтерфейсу, який мінімізує виникнення помилок і забезпечує легкі механізми відновлення, має важливе значення для позитивного користувацького досвіду. Платформа короткострокової оренди повинна мати функцію валідації вводу, перевірки помилок і підказки для підтвердження, щоб запобігти ненавмисним або помилковим діям користувачів. Інтерфейс повинен стисло і чітко відображати повідомлення про помилки і підказувати користувачеві, як їх виправити [47].

4. Естетичний та мінімалістичний дизайн. Естетичний та мінімалістичний дизайн може покращити користувацький досвід за рахунок зменшення візуального безладу та зосередження уваги на основних елементах [36]. Платформа короткострокової оренди має бути спроектована просто і зрозуміло, щоб зменшити складність і представити інформацію у структурований, зрозумілий спосіб. Додаток повинен використовувати пробіли, типографіку та візуальну ієрархію, щоб привернути увагу користувачів до елементів і ключових дій, які користувач має виконати.

Окрім принципів дизайну інтерфейсу, для створення успішної платформи короткострокової оренди важливо враховувати фактори користувацького досвіду (UX). Юзабіліті охоплює загальний досвід і задоволеність користувачів при взаємодії

з платформою, включаючи такі аспекти, як зручність використання, доступність та емоційна залученість. Розробка користувацьких типів потребує проведення користувацьких досліджень, щоб зрозуміти потреби, цілі та вподобання цільових користувачів платформи[13]. Збираючи інформацію за допомогою опитувань, інтерв'ю та юзабіліті-тестування, команда розробників може визначити ключові вимоги користувачів і адаптувати функції та дизайн платформи для задоволення цих потреб. Персони користувачів, які є вигаданими зображеннями ідеальних користувачів можуть спрямовати дизайн, надаючи уявлення про те, чого хочуть користувачі, що їх мотивує, як вони поводяться та які їхні больові точки. Організація контенту та функціональності платформи в чітку та інтуїтивно зрозумілу інформаційну архітектуру має вирішальне значення для ефективної навігації користувачів [44]. Платформа короткострокової оренди повинна мати таку організаційну структуру, яка є логічною та ієрархічною; вона відображатиме ментальні моделі користувачів і задовольнятиме їхні потреби.

Система навігації має бути передбачуваною, послідовною та зручною, щоб користувачі могли легко знаходити потрібну інформацію або функції та отримувати до них доступ. Використання таких методів, як "хлібні крихти", чітке маркування та функція пошуку, покращує навігацію на платформі.

Адаптивний дизайн має вирішальне значення для забезпечення того, щоб платформа короткострокової оренди була зручною для мобільних пристроїв, враховуючи зростаючу поширеність мобільних пристроїв [32]. Адаптивний дизайн дозволяє користувацькому інтерфейсу адаптуватися та оптимізуватися під різні розміри та роздільну здатність екрану, забезпечуючи безперебійну роботу користувача на різних пристроях.

React і Tailwind CSS надають інструменти та найкращі практики для реалізації доступних інтерфейсів,. Завдяки інтеграції з адаптивними класами утиліт і гнучким структурам компонентів платформа гарантує узгодженість і простоту використання на кожному пристрої, незалежно від того, чи користувачі отримують доступ до платформи зі стаціонарних комп'ютерів, планшетів або смартфонів. Доступний дизайн

платформи короткострокової оренди гарантує, що нею зможуть користуватися люди з усіх верств суспільства, в тому числі з порушеннями зору, слуху, опорно-рухового апарату або когнітивними розладами [52].

React та Tailwind CSS включають засоби та найкращі практики для створення доступних інтерфейсів за допомогою семантичного HTML, надання атрибутів alt на зображеннях та забезпечення належної клавіатурної навігації. Також бажано при створенні вебдодатку проводити юзабіліті-тестування. Юзабіліті-тести, проведені репрезентативними користувачами, дозволять команді розробників отримати зворотній зв'язок і розуміння того, наскільки добре ця платформа відповідатиме потребам і очікуванням користувачів. Юзабіліті-тестування є важливим процесом для оцінки ефективності та зручності інтерфейсу платформи короткострокової оренди та користувацького досвіду [36]. Юзабіліті-тестування можна проводити на різних етапах процесу розробки - від ранніх прототипів до кінцевого продукту. Такі методи, як спостереження за користувачами, тести на виконання завдань та інтерв'ю з користувачами, можуть допомогти виявити проблеми юзабіліті, незрозумілі елементи або сфери для покращення. Такий цілісний підхід гарантує, що платформа стає кращою і змінюється з часом, залишається орієнтованою на своїх користувачів, а отже, стає більш ефективною.

Розділ 3. Розробка серверної частини вебзастосунку

3.1. Конфігурація та робота з базою даних MongoDB

Розробка сервісу короткострокової оренди включає в себе налаштування конфігурації внутрішньої бази даних для забезпечення ефективного зберігання, пошуку та управління даними. MongoDB, популярна база даних NoSQL, була обрана для цього проекту завдяки своїй масштабованості, гнучкості та високій продуктивності. У цьому розділі обговорюється налаштування бекенду за допомогою MongoDB, з акцентом на моделюванні даних, налаштуванні бази даних та інтеграції з Prisma, сучасним інструментарієм баз даних для Node.js і TypeScript.

MongoDB - це документно-орієнтована NoSQL-база даних, яка зберігає дані у гнучких JSON-подібних документах під назвою BSON (Binary JSON). На відміну від традиційних реляційних баз даних, які використовують таблиці та рядки, MongoDB організовує дані в колекції та документи, що дозволяє більш природно та інтуїтивно представляти складні структури даних [7].

Ключові особливості MongoDB:

- Масштабованість: MongoDB розроблена для горизонтального масштабування на декількох серверах, що дозволяє базі даних обробляти великі обсяги даних і високі навантаження на трафік.
- Гнучкість: Документна модель даних дозволяє створювати динамічні та еволюційні схеми, що полегшує адаптацію до мінливих вимог додатків.
- Продуктивність: MongoDB підтримує високопродуктивні операції читання та запису, має вбудовану індексацію, агрегацію та підтримку аналітики в реальному часі.

Як зазначає Chodorow, K. [12], "MongoDB розроблена для того, щоб полегшити створення і масштабування додатків, які повинні швидко розвиватися і обробляти великі обсяги даних" (стор. 5).

Моделювання даних - це процес проектування структури та організації даних у базі даних для підтримки вимог програми [26]. Моделювання даних за допомогою MongoDB передбачає визначення колекцій, документів і того, як вони будуть пов'язані між собою для представлення різних об'єктів і взаємодій у додатку для короткострокової оренди.

Для платформи короткострокової оренди можна розглянути наступні колекції:

1. Користувачі: Зберігає інформацію про користувачів, включно з даними їхніх профілів, обліковими даними для автентифікації та ролями (наприклад, господар, гість).
2. Списки: Містить інформацію про об'єкти оренди, таку як назва, опис, місцезнаходження, зручності, ціни та доступність.
3. Бронювання: Відображає транзакції бронювання, включаючи користувача, оголошення, дати заселення та виселення, а також платіжні реквізити.
4. Відгуки: Зберігає відгуки користувачів та рейтинги для об'єктів оренди та господарів.

При розробці моделі даних важливо враховувати взаємозв'язки між сутностями і те, як додаток буде запитувати і отримувати доступ до них. Наприклад, документ з оголошенням може містити вбудовану інформацію про користувача для полегшення ефективного пошуку інформації про власника, тоді як бронювання може посилатися як на користувача, так і на документ з оголошенням, використовуючи їхні унікальні ідентифікатори.

Налаштування та конфігурація MongoDB для платформи короткострокової оренди включає кілька кроків, зокрема встановлення, створення бази даних та налаштування підключення. Встановлення: MongoDB підтримує різні платформи, такі як Windows, macOS та Linux. Рекомендується завантажити його з центру завантажень

MongoDB. Створення бази даних: Після встановлення MongoDB необхідно створити нову базу даних. Це можна зробити за допомогою оболонки Mongo або інструменту з графічним інтерфейсом, наприклад, MongoDB Compass. Конфігурація підключення: Щоб підключити додаток до бази даних MongoDB, необхідно налаштувати URL-адресу з'єднання та облікові дані. Зазвичай це включає в себе вказівку хосту бази даних, порту, деталей автентифікації та імені бази даних.

```
const { MongoClient } = require('mongodb');

const uri = 'mongodb://localhost:27017';
const client = new MongoClient(uri, { useNewUrlParser: true, useUnifiedTopology: true });

async function connectToDatabase() {
  try {
    await client.connect();
    console.log('Connected to MongoDB');
    const db = client.db('short_term_rental');

  } catch (error) {
    console.error('Error connecting to MongoDB:', error);
  } finally {
    await client.close();
  }
}

connectToDatabase();
```

Рис 3.1 Підключення до MongoDB за допомогою драйвера MongoDB Node.js

Інтеграція з Prisma

Prisma - це передовий інструментарій для роботи з базами даних, який спрощує доступ до баз даних та управління ними для додатків на Node.js і TypeScript. Він надає безпечні типи API для запитів і мутації даних і включає потужні функції, такі як міграція баз даних, управління схемами і підписи в реальному часі.

Кроки для інтеграції Prisma з MongoDB

1. Встановіть Prisma та ініціалізуйте новий проект Prisma у бекенд-додатку за допомогою Prisma CLI.

2. Визначте схему Prisma, яка представляє модель даних і

відображення між додатком і базою даних. Схема, написана на мові схем Prisma (PSL), включає визначення моделі даних, конфігурацію бази даних і налаштування генератора.

Нижче наведено приклад схеми Prisma для платформи короткострокової оренди:

```
datasource db {
  provider = "mongodb"
  url      = env("DATABASE_URL")
}

generator client {
  provider = "prisma-client-js"
}

model User {
  id        String    @id @default(auto()) @map("_id") @db.ObjectId
  email     String    @unique
  name      String
  listings  Listing[]
  bookings  Booking[]
  reviews   Review[]
}

model Listing {
  id          String    @id @default(auto()) @map("_id") @db.ObjectId
  title       String
  description String
  location    String
  price       Float
  host        User      @relation(fields: [hostId], references: [id])
  hostId      String    @db.ObjectId
  bookings    Booking[]
  reviews     Review[]
}
```

Рис 3.2 моделі User, Listing зіставляються з відповідними колекціями MongoDB

```
model Booking {
  id         String   @id @default(auto()) @map("_id") @db.ObjectId
  listing    Listing  @relation(fields: [listingId], references: [id])
  listingId  String   @db.ObjectId
  guest      User     @relation(fields: [guestId], references: [id])
  guestId    String   @db.ObjectId
  checkIn    DateTime
  checkOut   DateTime
}

model Review {
  id        String   @id @default(auto()) @map("_id") @db.ObjectId
  listing   Listing  @relation(fields: [listingId], references: [id])
  listingId String   @db.ObjectId
  author    User     @relation(fields: [authorId], references: [id])
  authorId  String   @db.ObjectId
  rating    Int
  comment   String
}
```

Рис 3.3 Моделі Booking та Review зіставляються з відповідними колекціями MongoDB

Схема також включає необхідні поля та зв'язки між цими моделями. Prisma Client: Scaffold з Prisma Client, безпечний за типом клієнтом бази даних, який може прискорити здатність програми виконувати такі дії, як отримання, створення та оновлення значень у базі даних MongoDB, використовуючи API Prisma. Prisma має досить гарний та виразний API для більшості операцій з базою даних - створення, читання, оновлення та видалення записів.

```
const { PrismaClient } = require('@prisma/client');

const prisma = new PrismaClient();

async function createUser(email, name) {
  try {
    const user = await prisma.user.create({
      data: {
        email,
        name,
      },
    });
    console.log('Created user:', user);
  } catch (error) {
    console.error('Error creating user:', error);
  } finally {
    await prisma.$disconnect();
  }
}

createUser('john@example.com', 'John Doe');
```

Рис 3.4 Створення нового користувача в базі даних, за допомогою Prisma Client

Інтегруючи MongoDB з Prisma, платформа короткострокової оренди може отримати безпечний за типом та ефективний доступ до бази даних, тим самим мінімізуючи ризики, які можуть бути пов'язані з використанням власного драйвера, а також спрощуючи управління базою даних та забезпечуючи цілісність даних.

Оптимізація продуктивності бази даних MongoDB має вирішальне значення для забезпечення безперебійної та швидкої роботи користувачів на платформі короткострокової оренди. Для підвищення продуктивності бази даних можна використовувати кілька методів:

1. Індексування: Створення правильних індексів для запитуваних полів може полегшити інкрементне виконання запитів. Припускаючи, що MongoDB

підтримує кілька типів індексів - індекси з одного поля, складені індекси, побудовані з декількох полів, і багатоключові індекси зі списків документів.

2. Оптимізація запитів: Аналіз та оптимізація шаблонів запитів може бути використана для скорочення часу запиту та ресурсів, необхідних для отримання даних з бази даних. Це стосується таких методів, як використання ефективних операторів запитів, повернення лише необхідних полів та уникнення пошуку непотрібних даних.

3. Кешування: Покращуйте механізм кешування, впроваджуючи кешування на рівні додатків або використовуючи вбудований кеш MongoDB WiredTiger, щоб зменшити навантаження на базу даних і пришвидшити час відгуку.

4. Шардинг: В ідеалі, коли додаток великий, використовується шардинг для розподілу даних між різними машинами, що дозволяє легко масштабувати систему та підвищити продуктивність. MongoDB підтримує власний шардинг і дозволяє базі даних додатків горизонтально масштабуватися і справлятися зі збільшеним навантаженням.

Забезпечення працездатності, продуктивності та доступності бази даних MongoDB передбачає виконання таких завдань, як моніторинг. Для цього використовуються різні інструменти управління та моніторингу. Наприклад, MongoDB Atlas - це хмарний сервіс, який автоматизує моніторинг баз даних, процеси резервного копіювання та масштабування. Він надає інформацію про продуктивність бази даних, сповіщення про проблеми в режимі реального часу та інструменти для підвищення ефективності. Ще одним зручним інструментом є MongoDB Compass, який пропонує користувацький інтерфейс для взаємодії з базами даних MongoDB. Він допомагає у візуалізації даних, оптимізації запитів та моніторингу продуктивності. Крім того, функції логування та профілювання MongoDB допомагають відстежувати активність бази даних, щоб точно визначити запити та діагностувати вузькі місця в продуктивності. Правильна конфігурація налаштувань

логування та профілювання дозволяє користувачам збирати інформацію про поведінку бази даних та усувати проблеми. Регулярне створення резервних копій баз даних та впровадження стратегій аварійного відновлення мають вирішальне значення для забезпечення довговічності даних та безперервності бізнесу. MongoDB має вбудовані інструменти резервного копіювання та відновлення, а також інтеграцію зі стороннім програмним забезпеченням для резервного копіювання.

Налаштування бекенду з MongoDB є важливою частиною процесу розробки при створенні вебдодатку для платформи короткострокової оренди. MongoDB може бути найкращим рішенням для обробки складних вимог платформи до даних, враховуючи її масштабування, гнучкість і характеристики продуктивності. Створивши відповідну модель даних, налаштувавши та конфігурувавши базу даних та інтегрувавши її з Prisma, бекенд може забезпечити ефективне та надійне зберігання та пошук даних.

Оптимізація продуктивності бази даних за допомогою таких методів, як індексування, оптимізація запитів, кешування та шардінг, забезпечує безперебійну та швидку роботу користувачів. Регулярний моніторинг та обслуговування бази даних MongoDB за допомогою таких інструментів, як MongoDB Atlas та Compass, допомагає підтримувати її працездатність, продуктивність та доступність.

У випадку динамічного зростання платформи гнучкість і масштабованість MongoDB забезпечить адаптивність додатку до постійно мінливих вимог, обсягів даних, що надходять, і навантажень трафіку. Використовуючи можливості MongoDB та Prisma, бекенд платформи короткострокової оренди тепер може стати потужним каталізатором для створення потужних та ефективних повностекових вебдодатків.

3.2 Реалізація серверної логіки з використанням Prisma ORM

Prisma представляє новий спосіб доступу до баз даних та керування ними. Це різновид ORM, який за допомогою безпечного за типом і простого у використанні API дозволяє легко працювати із запитами до даних та їхньою мутацією в базі даних. У цьому проекті детально розглядається реалізація серверної логіки щодо моделювання даних та виконання CRUD операцій, а також її використання для інтеграції з базою даних MongoDB.

Prisma - це новий тип інструментарію для роботи з базами даних, що складається з декількох основних частин. Клієнт Prisma це автоматично згенерований конструктор запитів, який надає виразний API, за допомогою якого розробники можуть взаємодіяти з базою даних, щоб забезпечити типо-безпечні запити та мутацію бази даних у безпечний спосіб. Prisma Migrate інструмент для управління ефективною міграцією бази даних, управління схемами та версіями бази даних. Prisma Studio повнофункціональний візуальний редактор баз даних з графічним інтерфейсом.

Ключові особливості Prisma

1. Тип безпеки API: Prisma Client створює строго типізований API на основі схеми бази даних. Це забезпечує безпеку під час компіляції та автоматичне завершення запитів до бази даних.
2. Інтуїтивно зрозуміла мова запитів: Виразна мова запитів Prisma розроблена таким чином, щоб бути дружньою, з можливістю написання складних запитів з мінімальним акцентом на шаблони в процесі роботи.
3. Агностичність до баз даних: Наразі Prisma підтримує PostgreSQL, MySQL, SQLite та MongoDB, що означає, що розробники можуть вибрати відповідну базу даних для свого додатку.

За словами Lawson (2020)[28], "Prisma змінює те, як ми взаємодіємо з базами даних у сучасній розробці додатків. Вона забезпечує безпечний для типів, інтуїтивно зрозумілий і ефективний спосіб роботи з даними, що полегшує створення масштабованих і підтримуваних додатків" (с. 5).

Моделювання даних передбачає визначення структури та взаємозв'язків між об'єктами даних у базі даних. Prisma дозволяє виконувати декларативне моделювання даних, де схема бази даних визначається за допомогою мови схем Prisma Schema Language (PSL). PSL змушує розробників використовувати формат, який є достатньо читабельним і виразним для опису моделей даних, полів і зв'язків.

Взаємодіяти з базою даних та реалізовувати серверну логіку для платформи короткострокової оренди житла легко за допомогою Prisma Client - безпечного за типами та інтуїтивно зрозумілого API для виконання CRUD-операцій над базою даних.

```
async function getListingById(listingId) {
  try {
    const listing = await prisma.listing.findUnique({
      where: {
        id: listingId,
      },
      include: {
        host: true,
        bookings: true,
        reviews: true,
      },
    });
    console.log('Retrieved listing:', listing);
    return listing;
  } catch (error) {
    console.error('Error retrieving listing:', error);
    throw error;
  }
}
```

Рис 3.5 Отримання списку за його ідентифікатором(ID)

```
async function updateListingPrice(listingId, newPrice) {
  try {
    const updatedListing = await prisma.listing.update({
      where: {
        id: listingId,
      },
      data: {
        price: newPrice,
      },
    });
    console.log('Updated listing:', updatedListing);
    return updatedListing;
  } catch (error) {
    console.error('Error updating listing:', error);
    throw error;
  }
}
```

Рис 3.6 Приклад оновлення ціни оголошення в базі даних за допомогою методу update

```
async function deleteBooking(bookingId) {
  try {
    const deletedBooking = await prisma.booking.delete({
      where: {
        id: bookingId,
      },
    });
    console.log('Deleted booking:', deletedBooking);
    return deletedBooking;
  } catch (error) {
    console.error('Error deleting booking:', error);
    throw error;
  }
}
```

Рис 3.7 Приклад видалення бронювання оголошення в базі даних за допомогою методу delete

Все вище перелічене це логічні операції сервера, які дозволяють розробникам створювати, читати, оновлювати та видаляти дані в базі даних платформи, абстрагуючись від бази даних за допомогою інтуїтивно зрозумілого API Prisma Client.

Інтеграція з MongoDB

Інструмент також може співпрацювати з MongoDB, популярною базою даних NoSQL, яка відома своєю масштабованістю, гнучкістю та продуктивністю. Щоб налаштувати Prisma для роботи з MongoDB, потрібно вказати URL-адресу підключення до MongoDB та увімкнути коннектор MongoDB у схемі Prisma. При роботі з MongoDB, Prisma автоматично відображає визначені моделі даних у колекції в MongoDB та автоматично транслює запити Prisma у специфічні для MongoDB операції. Така абстракція дозволяє скористатися перевагами інтуїтивно зрозумілої мови запитів Prisma та безпечного щодо типів API, не втрачаючи при цьому переваг масштабованості та гнучкості MongoDB. Оптимізація серверної логіки важлива для покращення користувацького досвіду платформи короткострокової оренди, забезпечуючи її чуйність та безперебійність.

Мова запитів Prisma надає розробникам засоби для чіткого визначення потреб у даних, тим самим зменшуючи обсяг даних, які можуть бути передані між базою даних і додатком. Але за допомогою опцій select і include розробник може точно визначити, які поля і вхідні сутності повинні бути отримані, щоб не отримати нерелевантні дані.

Пагінація і фільтрація інтегровані спеціально для підвищення продуктивності запитів у випадку великих наборів даних і для зменшення передачі даних. Prisma підтримує пагінацію за допомогою опцій skip і take, а фільтрацію - за допомогою опції where. Пакетні операції виконуються в ситуаціях, коли багато операцій з базою даних потрібно виконати разом. Згідно з документацією Prisma, ці операції допомагають мінімізувати обхідні шляхи між додатком і базою даних. Prisma підтримує пакетні операції createMany, updateMany та deleteMany. Використовуючи ці методи оптимізації, можливо підвищити продуктивність серверної логіки та забезпечити ефективний пошук даних і маніпуляції з ними.

Належна обробка помилок та ведення журналу є важливими кроками у забезпеченні надійності та доступності платформи. Prisma забезпечує надійний механізм обробки помилок, за допомогою якого вона видає загальний тип помилки відповідно до проблеми, що виникла[42]. Розробники можуть за допомогою блоків try-catch відстежувати та обробляти такі помилки, і, в свою чергу, відповідати на них найбільш відповідною помилкою для відповідного клієнта.

```
try {
  const listing = await prisma.listing.findUnique({
    where: {
      id: listingId,
    },
  });
  if (!listing) {
    throw new Error('Listing not found');
  }
  return listing;
} catch (error) {
  console.error('Error retrieving listing:', error);
  throw new Error('Internal server error');
}
```

Рис 3.8 Приклад обробки помилок у Prisma.

Окрім обробки помилок, ведення журналу в додатку є ще одним аспектом, який сприяє моніторингу та налагодженню програми. Prisma має хорошу інтеграцію з популярними зовнішніми бібліотеками логування, такими як Winston або Pino, в яких ви можете логувати запити, помилки або будь-яку іншу важливу інформацію.

Налаштування серверної логіки є однією з найважливіших частин процесу розробки повноцінних вебдодатків для платформи короткострокової оренди. Prisma пропонує безпечний за типом і дуже інтуїтивно зрозумілий API, який звільняє розробників від обробки більшості випадків доступу до бази даних, щоб більше зосередитися на основній логіці програми. Завдяки можливостям моделювання даних і CRUD-операцій, які пропонує Prisma, а також можливостям інтеграції з MongoDB, розробники можуть створювати ефективну і масштабовану серверну логіку. Належна обробка помилок і ведення журналів гарантують надійність і зручність обслуговування додатків, що допомагає розробникам швидко виявляти і виправляти помилки.

У випадку якщо платформа короткострокової оренди почне зростати і розвиватись, гнучкість і розширюваність Prisma дозволить адаптуватися до мінливих вимог і безперешкодно масштабувати додаток. Використовуючи можливості Prisma ORM, я зможу створити надійну та ефективну серверну основу для платформи короткострокової оренди, що забезпечить її успіх на конкурентному ринку.

3.3 Забезпечення безпеки та обробка помилок на сервері

Забезпечити безпеку та належну обробку помилок при розробці вебдодату, призначеного для короткострокової оренди, є надзвичайно важливим. Вразливості в безпеці та обробці помилок призводять до витоку даних, несанкціонованого доступу та компрометації інформації користувачів. У цьому розділі розглядаються найкращі практики та способи забезпечення безпеки та обробки помилок на серверній стороні платформи короткострокової оренди, з акцентом на перевірці вхідних даних, захисті від поширених вебвразливостей та правильних стратегіях обробки помилок.

Безпека на стороні сервера важлива для захисту конфіденційних даних користувача, запобігання несанкціонованому доступу та підтримки цілісності вебдодатку. Нижче наведені деякі з основних міркувань безпеки, які будуть взяті до уваги під час розробки короткострокової платформи на стороні сервера:

1. Перевірка вхідних даних: Дуже важливим компонентом безпеки вебдодатків є перевірка вхідних даних, яка допомагає запобігти ін'єкційним атакам, включаючи SQL-ін'єкції та міжсайтовий скриптинг (XSS)[39].

Семантична перевірка вхідних даних розглядає всі вхідні дані користувача як ненадійні та забезпечує їхню коректність перед використанням.

2. Автентифікація та авторизація: Належна реалізація процесів автентифікації та авторизації забезпечить доступ до захищених ресурсів лише авторизованим користувачам. Безпечні методи, такі як вебтокени JSON та автентифікація на основі сеансу, були б належним чином впроваджені в платформу короткострокової оренди, з дотриманням належних перевірок авторизації на основі ролей та дозволів користувачів.

3. Безпечна комунікація: Уся комунікація між клієнтом і сервером має бути зашифрована за допомогою HTTPS, щоб захистити інформацію від підслуховування і втручання. Сервер повинен бути налаштований належним чином і мати дійсний сертифікат SSL/TLS, щоб забезпечити безпечний зв'язок.

4. Безпека даних: Конфіденційні дані, такі як пароль користувача та його персональні дані, повинні надійно зберігатися на сервері. Паролі слід хешувати за допомогою надійних алгоритмів хешування, таких як bcrypt та Argon2, перед тим, як зберігати їх у базі даних. Конфіденційні дані мають бути обов'язково зашифровані під час передачі та у стані спокою, щоб уникнути несанкціонованого доступу.

Як зазначає OWASP [39], «Вразливості безпеки можуть бути впроваджені на будь-якому етапі процесу розробки, від проектування до реалізації та розгортання. Важливо виявити та усунути ці вразливості на ранній стадії процесу розробки, щоб мінімізувати ризик експлуатації».

Методи перевірки вхідних даних

Перевірка вхідних даних є критично важливим аспектом безпеки на стороні сервера. Вона передбачає перевірку та санітарну обробку даних користувача, щоб переконатися, що вони відповідають очікуваному формату і не містять шкідливого контенту. Нижче наведені методи перевірки для платформи короткострокової оренди:

1. Перевірка типу даних: Це гарантує, що тип даних введених користувачем, відповідає очікуваному. Наприклад, якщо певне поле має

приймати лише цілі значення, то перевірка має бути виконана, щоб переконатися, що користувач дійсно ввів ціле число.

2. Перевірка довжини: Переконайтеся, що користувачі можуть вводити коректні дані, а також запобігти переповненню буфера. Переконайтеся, що введені дані не перевищують очікувану межу.

3. Перевірка формату: Вхідні дані перевіряються відповідно до формату за допомогою регулярних виразів або заздалегідь визначених шаблонів. Це може бути будь-що, наприклад, електронна пошта, номери мобільних телефонів тощо, для яких потрібно, щоб ці значення були у певному визначеному форматі.

4. Перевірка за білим списком: Вхідні дані перевіряються за заздалегідь визначеним набором допустимих значень або символів. Цей спосіб є більш суворим і безпечним, ніж перевірка за чорним списком.

5. Параметризовані запити: Використання підготовлених операторів, або параметризованих запитів, у будь-якій взаємодії з базою даних зробить вебдодаток захищеним від атак SQL-ін'єкцій. При використанні параметризованого запиту вхідні дані користувача розглядаються як дані і ніколи не виконуються як частина SQL-запиту.

```
const { PrismaClient } = require('@prisma/client');
const prisma = new PrismaClient();

async function createListing(title, description, price) {
  try {
    const listing = await prisma.listing.create({
      data: {
        title,
        description,
        price,
      },
    });
    console.log('Created listing:', listing);
  } catch (error) {
    console.error('Error creating listing:', error);
  } finally {
    await prisma.$disconnect();
  }
}

createListing('Cozy Apartment', 'A cozy apartment in the heart of the city', 100.0);
```

Рис 3.9 Приклад параметризованих запитів із використанням Prisma ORM.

У наведеному прикладі метод createListing використовується для створення нового списку в базі даних за допомогою методу create, в той час як користувацькі дані для назви, опису та ціни (title, description, and price) передаються як окремі параметри, щоб вони розглядалися як дані, а не виконувалися як частина SQL-запиту.

На додаток до перевірки вхідних даних, дуже важливо захистити платформу короткострокової оренди від поширених вебуразливостей. Нижче наведені деякі з найбільш критичних вразливостей та методи їх усунення:

1. Cross-Site Scripting (XSS): Міжсайтовий скриптинг дозволяє зловмиснику впроваджувати шкідливі скрипти на сторінки в Інтернеті, які використовуються для виконання скриптів з пристроїв користувачів які нічого не підозрюють. Вхідні дані користувачів повинні бути закодовані і належним чином очищені перед відображенням у браузері. Прикладом є використання бібліотек для dompurify, які допомагають очистити будь-який створений користувачем контент.

2. Cross-Site Request Forgery (Міжсайтова підробка запитів(CSRF)): Атака, яка змушує користувача виконати дії у вебдодатку, які він не мав наміру

ініціювати. Запобігання CSRF-атаці полягає у додаванні сервером токенів CSRF у форми та їх перевірці під час публікації форми. Доступні бібліотеки для генерації та перевірки токенів для захисту від CSRF-атаки, такі як csurf.

3. Порушена аутентифікація та управління сесіями: Це призводить до несанкціонованого доступу та перехоплення сесії через погано реалізовану аутентифікацію та управління сесіями. Слід використовувати безпечні механізми автентифікації - наприклад, NextAuth або автентифікацію на основі сеансу – і слідкувати за належним закінченням терміну дії сеансу та його анулюванням.

4. Небезпечні прямі посилання на об'єкти (Insecure Direct Object References, IDOR): Це відбувається, коли сервер розкриває свої внутрішні посилання на об'єкти, такі як ідентифікатори баз даних, через URL-адреси або форми. Як правило, IDOR потрібно запобігати на всіх серверах по всьому світу, впроваджуючи модель непрямих посилань на об'єкти та перевіряючи доступ користувачів до запитуваних об'єктів.

5. Недостатня реєстрація та моніторинг: реєстрація та моніторинг є критично важливими елементами у виявленні та вирішенні проблем безпеки. Сервер повинен мати механізми реєстрації та моніторингу, які дозволять відстежувати активність користувачів, виявляти підозрілу поведінку та своєчасно реагувати на підозрілі події.

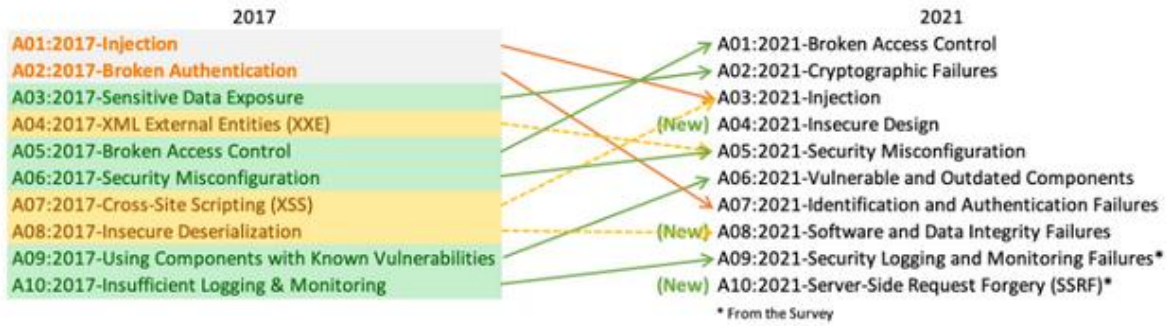


Рис 3.10 Поширеність вебуразливостей 2017-2021 на основі рейтингу OWASP Top 10. [39]

Kaspersky Top 10	OWASP ranking
1 Broken Access Control	A01
2 Sensitive Data Exposure	A02
3 Server-Side Request Forgery (SSRF)	A10
4 SQL Injection	A03
5 Cross Site Scripting (XSS)	A03
6 Broken Authentication	A07
7 Security Misconfiguration	A05
8 Insufficient Protection from Brute Force Attacks	A07
9 Weak User Password	A07
10 Using Components with Known Vulnerabilities	A06

Рис 3.11 Поширеність вебуразливостей за 2021-2023 на основі рейтингу securelist. [46]

Реєстрація та обробка помилок

Належна обробка помилок і ведення журналу помилок - це частина забезпечення стабільної та безпечної роботи платформи короткострокової оренди. Ось деякі з найкращих практик, пов'язаних з обробкою помилок і веденням записів:

1. Обробка помилок: Впровадження централізованих механізмів обробки помилок у додатку, які можуть відловлювати та обробляти помилки в уніфікований спосіб. Використовуйте блоки перехоплення, щоб перехоплювати винятки та відображати користувачеві інформативні повідомлення про помилки, не допускаючи витоку конфіденційної інформації.

2. Журналювання помилок: Реєструйте помилки та винятки з достатньою кількістю деталей, наприклад, повідомлення про помилки,

трасування стеку та відповідну інформацію про запит. Використовуйте бібліотеку журналювання, таку як Winston або Bunyan, яка надає можливості для структурованого ведення журналів і керування ними.

3. Log Levels: Рівні логів включають відповідні рівні, такі як налагодження, інформація, попередження та помилки, і використовуються для класифікації рівня повідомлень. Налаштуйте фреймворки журналів для реєстрації повідомлень відповідно до середовища - розробка, постановка або виробництво - і налаштуйте серйозність рівня журналу відповідно до повідомлення.

4. Ротація і збереження журналів: Він також повинен пропонувати політики ротації та збереження журналів, які запобігають поглинанню журналами занадто багато місця на диску. Ротація журналів, як на основі розміру, так і у встановлений час, повинна бути достатньою для того, щоб файли журналів могли бути використані для усунення несправностей і проведення аудиту.

5. Моніторинг та оповіщення: Слід розробити систему виявлення та оповіщення, щоб інформувати системних адміністраторів про проблеми, пов'язані з критичними помилками, продуктивністю та подіями, що стосуються безпеки. Використовуйте інструменти моніторингу для перевірки журналів і метрик додатку.

Тестування та аудит безпеки

Повний спектр тестування та аудиту безпеки застосовується до платформи короткострокової оренди, щоб забезпечити безпеку та надійність її елементів. Ось основні аспекти тестування та аудиту безпеки:

Модульне тестування: Перевірка правильності кожного елемента і функціональності, до кожної деталі перевірки вхідних даних, всілякі перевірки помилок і логіки, пов'язаної з безпекою. Для юніт-тестування можна використовувати фреймворки, наприклад Jest.

Інтеграційне тестування: Це тестування, яке проводиться для перевірки взаємодії між різними складовими та модулями додатку; водночас воно також забезпечує нормальну інтеграцію/тестування серверного коду з точки зору інтеграції з базами даних, зовнішніми сервісами та інтеграції з клієнтською стороною додатку.

Тестування безпеки: Процес виявлення будь-якої вразливості або слабкості в додатку за допомогою тестування на проникнення, сканування вразливостей і методів перегляду коду. Це можна автоматизувати за допомогою деяких інструментів -OWASP ZAP, Burp Suite, Acunetix.

Аудит залежностей: Ця діяльність передбачає виконання періодичних перевірок залежностей, що використовуються в додатку, на наявність відомих вразливостей. Аудит залежностей зазвичай виконується за допомогою таких інструментів, як npm-аудит або Snyk, які сканують залежності проекту і виявляють будь-які уразливості безпеки в встановлених пакетах.

Аудит безпеки: Періодичний аудит безпеки слід проводити, щоб отримати загальний статус безпеки програми. Аудит повинен включати перевірку коду, перевірку конфігурацій, аудит інфраструктури для виявлення та усунення прогалин та інших лазівок у засобах контролю безпеки на відповідність галузевим стандартам.

Забезпечення безпеки та належної обробки помилок на стороні сервера має вирішальне значення для успіху та надійності платформи. Саме тому регулярне тестування та аудит безпеки є важливими кроками у напрямку пошуку ймовірних проблем безпеки та їх завчасного вирішення. Таким чином, модульне тестування, інтеграційне тестування, тестування безпеки, аудит залежностей та аудит безпеки допоможуть знайти будь-яку ймовірну вразливість, яку можна вчасно усунути, перш ніж нею скористаються зловмисники.

Таблиця 3.1

Типи тестування та їх основні задачі

Тип тестування	**Область фокусу**
Модульне тестування	Окремі компоненти та функції
Інтеграційне тестування	Взаємодія між компонентами та модулями
Тестування безпеки	Виявлення вразливостей та слабких місць
Аудит залежностей	Вразливості безпеки у залежностях проекту
Аудит безпеки	Загальний стан безпеки та відповідність вимогам

З розвитком і зростанням платформи, необхідно переглядати і постійно впроваджувати оновлені заходи і практики безпеки. Створення середовища розробки, в якому основна увага приділяється заходам безпеки та обробці помилок протягом усього процесу розробки, призведе до створення безпечної та надійної платформи короткострокової оренди для користувачів для обміну цінностями.

Висновки

Результатом роботи над кваліфікаційною роботою став успішно розроблений повнофункціональний вебдодаток платформи для короткострокової оренди житла. Було використано такі технології, як React, Tailwind, Prisma, MongoDB та NextAuth. Проведене дослідження допомогло мені досягти мети та вирішити поставлені завдання.

У першому розділі роботи детально описано предметну область та обрано оптимальний стек технологій для розробки вебдодатку. Огляд ринку свідчить про великий попит та майбутній розвиток такої платформи. Аналіз подібних вебдодатків допоміг визначити основні функціональні можливості та вимоги, необхідні для розробки конкретної платформи. Обраний стек технологій, серед яких React, Tailwind, Prisma, MongoDB та NextAuth, забезпечує швидку та ефективну розробку сучасного вебдодатку з високою продуктивністю та масштабованістю.

Другий розділ буде присвячено створенню клієнтської частини вебдодатку. За допомогою бібліотеки React та CSS фреймворку Tailwind був розроблений інтуїтивно зрозумілий та привабливий користувацький інтерфейс. Принципи адаптивного дизайну дозволили забезпечити коректне відображення та роботу додатку на різних пристроях. Інтеграція з NextAuth для аутентифікації та авторизації забезпечує безпеку даних користувачів та обмежує доступ до функціоналу відповідно до ролей користувачів.

Третій розділ детально описує процес розробки серверної частини вебдодатку: створення чогось цілісного навколо конфігурації бази даних MongoDB для ефективного зберігання та управління даними на платформі. Prisma ORM допомогла спростити нашу взаємодію з базою даних і дозволила швидко реалізувати необхідну бізнес-логіку. Особливу увагу було приділено безпеці вебдодатку, враховуючи належну валідацію вхідних даних, захист від найпоширеніших вебвразливостей та правильну обробку помилок на сервері.

Результати тестування показали, що розроблений вебдодаток для платформи короткострокової оренди житла відповідає сучасним стандартам веброзробки і

відповідає потребам його користувача. Застосований стек технологій дозволяє розробляти високопродуктивні, масштабовані та надійні додатки з привабливим та зручним інтерфейсом.

Розроблена платформа має великі перспективи для вдосконалення та подальшого розвитку. Нові можливі напрямки розвитку можуть охоплювати зміни у функціоналі - за рахунок покращення системи оцінювання та відгуків, інтеграції з платіжними системами, розширених фільтрів пошуку тощо. Найголовніше, що вебдодаток також потребує подальшого тестування реальними користувачами для отримання зворотного зв'язку та внесення відповідних змін.

Досвід, отриманий під час виконання кваліфікаційної роботи, створив умови для поглиблення моїх знань та навичок у сфері веброзробки, управління базами даних, безпеки вебдодатків та дизайну користувацького інтерфейсу. Участь у розробці реального проекту зверху донизу, відповідна взаємодія з набором технологій та практичне вирішення проблем підготували мене до майбутньої роботи в IT-сфері.

Список використаних джерел

1. Agarwal, A. (2021). Implementing authentication in Next.js with NextAuth.js. URL: https://medium.com/geekculture/why-and-how-to-get-started-with-next-auth-61740558b45b (дата звернення 27.02.2024)

2. Abu Daqar, M., Smoudy, A., (2019). The Role of Artificial Intelligence on Enhancing Customer Experience. URL: https://www.researchgate.net/publication/334315006_The_Role_of_Artificial_Intelligence_on_Enhancing_Customer_Experience (дата звернення 25.10.2023)

3. Airbnb About us. URL: https://news.airbnb.com/about-us/ (дата звернення 10.11.2023)

4. Airbnb homepage. URL: https://www.airbnb.com.ua/ (дата звернення 02.11.2023)

5. Airbnb Engineering & Data Science. (2013). Isomorphic JavaScript: The future of web apps. The Airbnb Tech Blog. URL: https://medium.com/airbnb-engineering/isomorphic-javascript-the-future-of-web-apps-10882b7a2ebc (дата звернення 20.11.2023)

6. Airbnb Engineering & Data Science. (2016). Отримано з URL: https://airbnb.io/ (дата звернення 05.12.2023)

7. Banker, K. MongoDB in action. Manning Publications Co., 2011. 312 с.

8. Banks, A., & Porcello, E. (2017). Learning React: Functional web development with React and Redux. O'Reilly Media, Inc. 2017. 348 с.

9. Bhaskar, V. (2023). Tailwind CSS: A Comprehensive Introduction and Setup Guide. URL: https://tailgrids.com/blog/tailwind-css-introduction-and-setup-guide (дата звернення 29.10.2023)

10. Booking.com homepage. URL: https://www.booking.com/ (дата звернення 13.03.2024)

11. Canziba, E. (2018). Hands-on UX design for developers: Design, prototype, and implement compelling user experiences from scratch. Packt Publishing Ltd. 2018. 350 с.

12. Chodorow, K. MongoDB: The definitive guide: Powerful and scalable data storage. O'Reilly Media, Inc. 2019. 511 с.

13. Cooper, A., Reimann, R., Cronin, D., & Noessel, C.. About face: The essentials of interaction design (4th ed.). John Wiley & Sons. 2014. 720 с.

14. Cherdouh S., Kherri A., Kebir S. (2022). Using Sentiment Analysis of Online Hotel Reviews To Explore the Effect of Information and Communication Technologies on Hotel Guest Satisfaction. URL: https://www.researchgate.net/publication/360942416_Using_Sentiment_Analysis_of_Online_Hotel_Reviews_To_Explore_the_Effect_of_Information_and_Communication_Technologies_on_Hotel_Guest_Satisfaction (дата звернення 24.11.2023)

15. Dolnicar, S. (2019). A review of research into paid online peer-to-peer accommodation: Launching the Annals of Tourism Research curated collection on peer-to-peer accommodation. Annals of Tourism Research, 75, 248-264. URL: https://www.researchgate.net/publication/345485079_A_review_of_research_into_paid_online_peer-to-peer_accommodation_Launching_the_Annals_of_Tourism_Research_Curated_Collection_on_peer-to-peer_accommodation (дата звернення 10.01.2024)

16. Elmansy, R.. Teach yourself visually search engine optimization (SEO): A complete guide to search engine optimization. John Wiley & Sons. 2015. 307 с.

17. Expedia Group. Expedia Group reports fourth quarter and full year 2010 results. URL: https://www.sec.gov/Archives/edgar/data/1324424/000132442420000006/earningsrelease-q42019.htm (дата звернення 03.01.2024)

18. Fedosejev, A.. React.js essentials. Packt Publishing Ltd. 2015. 183 с.

19. Gackenheimer, C.. Introduction to React. Apress. 2015. 144 с.

20. Garrett, J. J.. The elements of user experience: User-centered design for the web and beyond (2nd ed.). New Riders. 2010. 192 с.

21. Guttentag, D. (2015). Airbnb: Disruptive innovation and the rise of an informal tourism accommodation sector. Current Issues in Tourism, 18(12), 1192-

1217. URL: https://www.researchgate.net/publication/271624904_Airbnb_Disruptive_innovation_and_the_rise_of_an_informal_tourism_accommodation_sector (дата звернення 24.02.2024)

22. Guttentag, D. A., & Smith, S. L.. Assessing Airbnb as a disruptive innovation relative to hotels: Substitution and comparative performance expectations. International Journal of Hospitality Management, 64, 1-10. URL: https://www.researchgate.net/publication/315907012_Assessing_Airbnb_as_a_disruptive_innovation_relative_to_hotels_Substitution_and_comparative_performance_expectations (дата звернення 25.02.2024)

23. Guttentag, D.. Progress on Airbnb: A literature review. Journal of Hospitality and Tourism Technology, 10(4), 814-844. URL: https://www.researchgate.net/publication/333880155_Progress_on_Airbnb_a_literature_review (дата звернення 23.12.2023)

24. Hartmans, A. (2018). Airbnb made $93 million in profit on $2.6 billion in revenue, but an internal clash sent the CFO out the door. Business Insider. URL: https://www.businessinsider.com/airbnb-profit-revenue-2018-2 (дата звернення 12.04.2024)

25. Hassenzahl, M., & Tractinsky, N.. User experience - A research agenda. Behaviour & Information Technology, 25(2), 91-97. URL: https://www.researchgate.net/publication/233864602_User_experience_-_A_research_agenda (дата звернення 16.02.2024)

26. Hoberman, S.. Data modeling for MongoDB: Building well-designed and supportable MongoDB databases. Technics Publications. 2014. 226 с.

27. Kaplan, R. A., & Nadler, M. L.. Airbnb: A case study in occupancy regulation and taxation. University of Chicago Law Review Dialogue, 82(1), 7. URL: https://chicagounbound.uchicago.edu/cgi/viewcontent.cgi?article=1038&context=uclrev_online&httpsredir=1&referer= (дата звернення 30.12.2023)

28. Ke, Q.. Service providers of the sharing economy: Who joins and who

benefits?. Proceedings of the ACM on Human-Computer Interaction, 1(CSCW), 1-17. URL: https://www.researchgate.net/publication/320014417_Service_Providers_of_the_Sharing_Economy_Who_Joins_and_Who_Benefits (дата звернення 02.03.2024)

29. Likhanova,V.. What is Tailwind CSS and how to implement it in a website or React application? URL: https://shorturl.at/2YMfr (дата звернення 08.04.2024)

30. Law, R., Schuckert, M., Xianwei, L.,. Hospitality and Tourism Online Reviews: Recent Trends and Future Directions, 30(3), 1825-1844. URL: https://www.researchgate.net/publication/264238827_Hospitality_and_Tourism_Online_Reviews_Recent_Trends_and_Future_Directions (дата звернення 08.04.2024)

31. Lindgaard, G., Fernandes, G., Dudek, C., & Brown, J.. Attention web designers: You have 50 milliseconds to make a good first impression! Behaviour & Information Technology, 25(2), 115-126. URL: https://www.researchgate.net/publication/220208334_Attention_web_designers_You_have_50_milliseconds_to_make_a_good_first_impression_Behaviour_and_Information_Technology_252_115-126 (дата звернення 05.02.2024)

32. Marcotte, E.. Responsive web design. A Book Apart. 2011. 150 с.

33. Mellinas, J. P., María-Dolores, S. M. M., & García, J. J. B. . Effects of the Booking.com scoring system. Tourism Management, 57, 80-83. URL: https://www.academia.edu/80553679/Effects_of_the_Booking_com_scoring_system?f_ri=943 (дата звернення 02.02.2024)

34. MongoDB documentation. URL: https://docs.mongodb.com/ (дата звернення 19.10.2023)

35. Nieuwland, S., & van Melik, R.. Regulating Airbnb: How cities deal with perceived negative externalities of short-term rentals. Current Issues in Tourism, 1-15. URL: https://www.researchgate.net/publication/326814900_Regulating_Airbnb_how_cities_deal_with_perceived_negative_externalities_of_short-term_rentals (дата звернення 04.04.2024)

36. Norman, D. (2013). The design of everyday things: Revised and expanded edition. Basic Books. 2013. 368 с.

37. NextAuth.js documentation. URL: https://next-auth.js.org/ (дата звернення 02.11.2023)

38. OWASP. OWASP Top Ten. URL: https://owasp.org/www-project-top-ten/ (дата звернення 02.12.2023)

39. Parecki, A.. OAuth 2.0 simplified. OAuth. 2018. 240 с.

40. Prisma documentation. URL: https://www.prisma.io/docs/ (дата звернення 16.12.2023)

41. React documentation.Accessibility. URL: https://reactjs.org/docs/accessibility.html (дата звернення 10.03.2024)

42. Rosenfeld, L., Morville, P., & Arango, J.. Information architecture: For the web and beyond (4th ed.). O'Reilly Media. 2015. 483 с.

43. Schaal, D.. Expedia Acquires HomeAway for $3.9 Billion Отримано з URL: https://skift.com/2015/11/04/expedia-acquires-homeaway-for-3-9-billion/ (дата звернення 22.10.2023)

44. Securelist.(2024). Top 10 web application vulnerabilities in 2021–2023. URL: https://securelist.com/top-10-web-app-vulnerabilities/112144/ (дата звернення 17.11.2023)

45. Shneiderman, B., Plaisant, C., Cohen, M., Jacobs, S., & Elmqvist, N.. Designing the user interface: Strategies for effective human-computer interaction (6th ed.). 2016. 616 с.

46. Stefankrause. JS web frameworks benchmark. URL: https://www.stefankrause.net/wp/?p=431 (дата звернення 12.12.2023)

47. Tailwind CSS documentation. URL: https://tailwindcss.com/docs/ (дата звернення 25.01.2024)

48. Vrbo homepage. URL: https://www.vrbo.com/ (дата звернення 14.01.2024)

49. W3C.. Web Content Accessibility Guidelines (WCAG) 2.1.. URL:

https://www.w3.org/TR/WCAG21/ (дата звернення 14.02.2024)

50. Wieruch, R.. The road to learn React: Your journey to master plain yet pragmatic React.js. Robin Wieruch. 2018. 286 с.

51. Zervas, G., Proserpio, D., & Byers, J. W.. A first look at online reputation on Airbnb, where every stay is above average. SSRN Electronic Journal, 1-22. URL: https://www.researchgate.net/publication/346494211_A_first_look_at_online_reputation_on_Airbnb_where_every_stay_is_above_average (дата звернення 18.01.2024)

52. Zervas, G., Proserpio, D., & Byers, J. W.. The rise of the sharing economy: Estimating the impact of Airbnb on the hotel industry. Journal of Marketing Research, 54(5), 687-705. URL: https://www.researchgate.net/publication/272302080_The_Rise_of_the_Sharing_Economy_Estimating_the_Impact_of_Airbnb_on_the_Hotel_Industry (дата звернення 18.10.2023)