

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)

на тему:

**«Вебдодаток для онлайн-бронювання з функціями
порівняння»**

Виконала студентка групи ІПЗ-21-2
ІЛЬЧЕНКО Валерія Ільченко

Керівник роботи:
ЧИЖМОТРИЯ Олена Геннадіївна

Рецензент:
КОНЦИДАЙЛО Андрій Михайлович

Житомир – 2025

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»

Зав. кафедри інженерії
програмного забезпечення
Тетяна ВАКАЛЮК
«14» лютого 2025 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **ІЛЬЧЕНКО Валерія Ільченко**

Керівник роботи: **ЧИЖМОТРИ Олена Геннадіївна**

Тема роботи: **«Вебдодаток для онлайн-бронювання з функціями порівняння»**,

затверджена Наказом закладу вищої освіти від **«14» лютого 2025 р., №61/с**

Вихідні дані для роботи: предметом дослідження є веборієнтовані технології;
об'єктом дослідження є процес організації бронювання номерів у готелях.

Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів,
що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Чижмотря О.Г.	14.02.2025р.	14.02.2025р.
2	Чижмотря О.Г.	14.03.2025р.	14.03.2025р.
3	Чижмотря О.Г.	14.04.2025р.	14.04.2025р.

РЕФЕРАТ

Випускна кваліфікаційна робота бакалавра складається з вебдодатку для онлайн-бронювання з функціями порівняння та пояснювальної записки. Пояснювальна записка до випускної роботи містить 82 сторінок, 27 ілюстрацій, 13 таблиць, 5 додатків та 21 використане джерело.

Метою кваліфікаційної роботи є проектування та створення вебдодатку для онлайн-бронювання, що надає користувачам зручний інструмент для пошуку, бронювання номерів та написання відгуків. Платформа також дозволяє власникам готелів розміщувати інформацію про свої об'єкти, оновлювати дані та управляти бронюваннями.

У роботі сформульовано та розв'язано основні задачі, необхідні для створення вебсистеми, здійснено детальний аналіз та порівняння існуючих аналогічних онлайн-платформ.

Проведено обґрунтування вибору інструментальних засобів і технологій, що відповідають сучасним тенденціям у сфері веброзробки. Особливу увагу приділено аналізу варіантів використання та визначенню структури додатку. Було спроектовано та розроблено базу даних для зберігання необхідної інформації про користувачів, готелі, номери, бронювання, платежі, відгуки та порівняння.

Описано процес технічної реалізації вебдодатку, який повністю відповідає сучасним вимогам до вебсервісів у туристичній сфері та готовий до впровадження та експлуатації.

КЛЮЧОВІ СЛОВА: ВЕБДОДАТОК, БРОНЮВАННЯ, ПОРІВНЯННЯ, ГОТЕЛІ, LARAVEL, REACT, DOCKER

					ІПЗ.КР.Б – 121 – 25 – ПЗ			
Змн.	Арк.	№ докум.	Підпис	Дата	Вебдодаток для онлайн-бронювання з функціями порівняння	Літ.	Арк.	Аркушів
Розроб.		Гльченко В.О.						
Керівник		Чижмотря О. Г.					3	82
Рецензент		Концидайло А.М.				Житомирська політехніка, група ІПЗ		
Зав. каф.		Вакалюк Т.А.						

ABSTRACT

The bachelor's qualification work consists of a web application for online booking with comparison functionality and an explanatory note. The explanatory note accompanying the thesis comprises 82 pages, 27 illustrations, 13 tables, 5 appendices, and 21 references.

The purpose of the qualification work is the design and implementation of a web application for online booking that provides users with a convenient tool for searching, booking hotel rooms, and leaving reviews. The platform also enables hotel owners to publish information about their establishments, update data, and manage reservations.

The research formulates and resolves key tasks necessary for developing the web system, including a detailed analysis and comparison of existing similar online platforms.

The thesis provides a thorough justification for the selection of instrumental tools and technologies that meet contemporary trends in web development. Particular attention has been devoted to analyzing use cases and determining the application structure. A database has been designed and implemented to store essential data about users, hotels, rooms, reservations, payments, reviews, and comparisons.

Furthermore, the thesis describes the process of technical implementation of the web application, which fully meets contemporary requirements for web services in the tourism industry and is ready for deployment and operational use.

KEYWORDS: WEB APPLICATION, BOOKING, COMPARISON, HOTELS, LARAVEL, REACT, DOCKER

					ІІЗ.КР.Б – 121 – 25 – ІІЗ	Арк.
						4
Змн.	Арк.	№ докум.	Підпис	Дата		

ЗМІСТ

РЕФЕРАТ	3
ABSTRACT	4
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ.....	7
ВСТУП	8
РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ ВЕБДОДАТКУ ДЛЯ ОНЛАЙН- БРОНЮВАННЯ З ФУНКЦІЯМИ ПОРІВНЯННЯ	11
1.1. Опис та постановка задачі.....	11
1.2. Дослідження, аналіз та порівняння функціональних можливостей аналогів програмного продукту.....	13
1.3. Вибір архітектури вебдодатку	19
1.4. Обґрунтування вибору інструментальних засобів для розробки вебдодатку.....	21
Висновки до першого розділу.....	23
РОЗДІЛ 2. ПРОЄКТУВАННЯ ВЕБДОДАТКУ ДЛЯ ОНЛАЙН-БРОНЮВАННЯ З ФУНКЦІЯМИ ПОРІВНЯННЯ	25
2.1. Визначення варіантів використання та структури системи	25
2.2. Проєктування та розробка бази даних вебдодатку.....	29
2.3. Реалізація функціоналу роботи вебдодатку для онлайн-бронювання з функціями порівняння	35
Висновки до другого розділу	42
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ВЕБДОДАТКОМ ДЛЯ ОНЛАЙН-БРОНЮВАННЯ З ФУНКЦІЯМИ ПОРІВНЯННЯ.....	44
3.1. Порядок встановлення та налаштування системи.....	44
3.2. Структура інтерфейсу вебдодатку для онлайн-бронювання та порядок роботи	45
3.3. Тестування роботи вебдодатку	54
Висновки до третього розділу.....	58

ВИСНОВКИ.....	60
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	62
ДОДАТКИ.....	64

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						6
Змн.	Арк.	№ докум.	Підпис	Дата		

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API – Application Programming Interface

CRUD – Create, Read, Update, Delete

JSON – JavaScript Object Notation

JWT – JSON Web Token

REST – Representational State Transfer

SMTP – Simple Mail Transfer Protocol

SQL – Structured Query Language

СКБД – система керування базами даних

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						7
Змн.	Арк.	№ докум.	Підпис	Дата		

ВСТУП

У сучасному світі, де туризм стрімко розвивається та відіграє важливу роль у формуванні економіки багатьох країн, питання забезпечення зручного й швидкого доступу до послуг бронювання набуває особливої значущості. Туристична галузь демонструє сталий ріст попиту як на міжнародні, так і на внутрішні подорожі, що супроводжується підвищеними вимогами споживачів до якості та зручності сервісів. Водночас усе більшу частку на ринку туристичних послуг займають невеликі готелі, апартаменти, хостели та приватні об'єкти розміщення, які конкурують із великими мережами, шукаючи нові шляхи залучення клієнтів.

Саме малий бізнес у сфері гостинності потребує ефективних, доступних і простих у використанні цифрових інструментів, що дозволяють управляти власними об'єктами, приймати бронювання без посередників і знижувати операційні витрати. Створення вебдодатку для онлайн-бронювання з функцією порівняння відкриває нові можливості для таких власників, надаючи їм платформу для представлення своїх послуг широкій аудиторії та підвищення конкурентоспроможності на ринку.

Актуальність теми зумовлена не лише розвитком інформаційних технологій, а й змінами у поведінці споживачів, які все частіше віддають перевагу онлайн-сервісам із прямим бронюванням. Користувачі прагнуть самостійно обирати місце проживання, порівнювати ціни, умови, відгуки, отримуючи повний контроль над процесом планування подорожі. Це дозволяє заощаджувати кошти, час і уникати додаткових комісій традиційних туроператорів.

Крім того, останні глобальні виклики, суттєво змінили туристичну індустрію, підкресливши важливість гнучких, адаптивних та незалежних платформ, здатних оперативно реагувати на зміну умов. У цих реаліях вебдодаток із можливістю порівняння пропозицій стає не лише інструментом для кінцевих користувачів, а й підтримкою малого бізнесу, що дозволяє локальним

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						8
Змн.	Арк.	№ докум.	Підпис	Дата		

власникам залишатися видимими, привабливими та доступними для клієнтів.

Таким чином, розробка вебдодатку для онлайн-бронювання з функцією порівняння відповідає актуальним потребам як мандрівників, так і власників невеликих об'єктів розміщення. Це рішення сприятиме підвищенню ефективності туристичної діяльності, розширенню можливостей малого бізнесу та розвитку цифрової інфраструктури у сфері гостинності.

Метою кваліфікаційної роботи є проектування та створення вебдодатку для онлайн-бронювання, що надаватиме користувачам зручний інструмент для пошуку, бронювання номерів та написання відгуків. Платформа також дозволить власникам готелів розміщувати інформацію про свої об'єкти, оновлювати дані та управляти бронюваннями.

Для досягнення поставленої мети визначено наступні завдання:

1. Провести аналіз існуючих онлайн-платформ для бронювання та виявити їх переваги та недоліки.
2. Сформулювати функціональні й нефункціональні вимоги до системи.
3. Визначити ролі користувачів і сценарії їх взаємодії з вебдодатком.
4. Спроекувати логічну структуру системи з основними модулями та зв'язками.
5. Розробити базу даних для зберігання та обробки інформації вебдодатку.
6. Створити прототип інтерфейсу користувача, зручного для взаємодії.
7. Реалізувати вебдодаток з функціоналом реєстрації, пошуку, бронювання та порівняння готелів.
8. Провести тестування роботи вебдодатку на відповідність вимогам.
9. Оцінити ефективність створеного рішення та можливості його вдосконалення.

Об'єктом дослідження є процес організації бронювання номерів у готелях. Розробка вебдодатку має на меті оптимізацію цього процесу шляхом створення зручних, ефективних та доступних інструментів як для користувачів, так і для власників помешкання.

Предметом дослідження є веборієнтовані технології для розробки

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						9
Змн.	Арк.	№ докум.	Підпис	Дата		

вебдодатку для онлайн-бронювання, що включає функцію порівняння варіантів помешкання.

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						10
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ ВЕБДОДАТКУ ДЛЯ ОНЛАЙН- БРОНЮВАННЯ З ФУНКЦІЯМИ ПОРІВНЯННЯ

1.1. Опис та постановка задачі

У сучасних реаліях туризм став не лише способом відпочинку, а й важливою складовою економічного розвитку багатьох країн. Щороку зростає кількість людей, які подорожують як всередині країни, так і за кордон. Водночас зміни у глобальній економіці, розвиток транспортної інфраструктури та доступність нових напрямків стимулюють збільшення попиту на послуги бронювання житла в онлайн-форматі. Для сучасного туриста швидкість, зручність та можливість самостійно контролювати процес пошуку і бронювання стали базовими вимогами. Саме тому цифрові сервіси займають провідне місце в організації подорожей, витісняючи традиційні способи бронювання через туристичні агентства.

Однак, попри значну кількість онлайн-платформ для бронювання, користувачі все частіше стикаються з обмеженнями, які ускладнюють вибір найкращого варіанту житла. Багато сервісів зосереджуються на великій кількості пропозицій, але не надають достатньо зручних інструментів для детального порівняння умов проживання, що особливо важливо для туристів із конкретними вимогами (наприклад, розташування, наявність певних зручностей, співвідношення ціни та якості). Крім того, більшість відомих платформ віддають пріоритет великим готельним мережам, унаслідок чого дрібні приватні об'єкти мають менші шанси бути поміченими потенційними клієнтами.

Окремої уваги потребує і питання доступності інформації. Не всі платформи забезпечують достатню прозорість щодо умов бронювання, додаткових витрат, політики скасування, що створює ризики для користувачів. З іншого боку, власники невеликих готелів часто мають обмежені можливості для самостійного керування пропозиціями на великих міжнародних платформах

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						11
Змн.	Арк.	№ докум.	Підпис	Дата		

через складність інтеграції, високі комісійні збори та відсутність індивідуального підходу.

У таких умовах актуальною є розробка вебдодатку, який об'єднуватиме зручність пошуку, прозорість інформації та функцію порівняння варіантів проживання. Такий додаток має створити рівні можливості як для користувачів, так і для малих готельних підприємств, сприяючи підвищенню їхньої видимості на ринку, залученню нових клієнтів і зміцненню позицій у конкурентному середовищі.

Для успішної реалізації вебдодатку для онлайн-бронювання з функцією порівняння необхідно вирішити комплекс завдань, що охоплюють усі етапи розробки програмного продукту: від дослідження предметної області до впровадження та тестування готової системи. Розробка такого вебдодатку вимагає врахування потреб різних категорій користувачів, забезпечення високої якості сервісу, функціональної повноти та захищеності даних. Водночас необхідно інтегрувати функціонал порівняння варіантів житла, що підвищує зручність користування платформою та допомагає приймати обґрунтовані рішення під час бронювання.

Особливу увагу під час виконання роботи слід приділити аналізу існуючих рішень на ринку, визначенню ключових вимог до системи, проєктуванню бази даних, розробці архітектури та інтерфейсу користувача, а також реалізації безпечної взаємодії між усіма компонентами платформи.

У межах даної кваліфікаційної роботи ставляться такі основні завдання:

1. Провести аналіз існуючих онлайн-платформ для бронювання з метою виявлення їхніх сильних і слабких сторін, недоліків та можливостей для удосконалення.
2. Сформулювати вимоги до системи, включаючи функціональні та нефункціональні вимоги, з урахуванням потреб кінцевих користувачів і власників об'єктів розміщення.
3. Визначити основні ролі користувачів системи та описати сценарії

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						12
Змн.	Арк.	№ докум.	Підпис	Дата		

їхньої взаємодії з вебдодатком (варіанти використання).

4. Розробити логічну структуру вебдодатку, визначивши основні модулі, компоненти та їх взаємозв'язки.

5. Спроекувати базу даних, що забезпечить зберігання, обробку та взаємодію всіх даних системи, дотримуючись принципів нормалізації та забезпечуючи цілісність даних.

6. Розробити прототип інтерфейсу користувача, що буде зручним та зрозумілим.

7. Реалізувати вебдодаток із використанням сучасних вебтехнологій, забезпечивши основні функції: реєстрацію та авторизацію користувачів, пошук житла, порівняння номерів, бронювання, управління контентом та перегляд відгуків.

8. Провести тестування функціональності вебдодатку, перевіривши коректність роботи всіх модулів, стабільність системи та її відповідність вимогам.

9. Оцінити отримані результати, визначити ефективність розробленого рішення, його переваги та можливості подальшого вдосконалення.

Зважаючи на зростаючий попит на цифрові сервіси у сфері туризму, особливий акцент зроблено на створенні інструменту, що поєднує зручність пошуку та порівняння варіантів проживання. Представлені завдання охоплюють усі ключові етапи розробки системи – від аналізу аналогів і формування вимог до реалізації, тестування та оцінки результатів, що закладає основу для комплексної реалізації проєкту.

1.2. Дослідження, аналіз та порівняння функціональних можливостей аналогів програмного продукту

Для створення конкурентоспроможного вебдодатку для онлайн-бронювання з функцією порівняння необхідно насамперед вивчити існуючі рішення, що вже представлені на ринку. Аналіз аналогічних платформ дозволяє

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						13
Змн.	Арк.	№ докум.	Підпис	Дата		

комплексно оцінити їхні функціональні можливості, сильні сторони, недоліки, а також визначити, які саме потреби користувачів залишаються недостатньо задоволеними.

Порівняння функціоналу існуючих систем допомагає виявити найбільш затребувані опції, вивчити підходи до реалізації ключових сервісів, а також знайти можливості для поліпшення користувацького досвіду. Крім того, детальний огляд конкурентних рішень є важливою передумовою для формування вимог до розробки власного додатку, оскільки дає змогу уникнути повторення слабких місць аналогів та запровадити нові функціональні можливості, що створять додаткову цінність для користувачів.

Таким чином, проведення аналізу аналогів є важливим етапом проєктування, що забезпечує обґрунтованість прийнятих рішень, сприяє формуванню оптимальної структури функціоналу та підвищує ймовірність успішної реалізації вебдодатку на конкурентному ринку.

Booking.com (рис. 1.1) є однією з найвідоміших та найбільш використовуваних у світі онлайн-платформ для бронювання житла [3]. Цей сервіс забезпечує посередницьку взаємодію між власниками об'єктів розміщення та користувачами, надаючи можливість готелям, апартаментам, хостелам та іншим об'єктам публікувати свої пропозиції, а клієнтам – самостійно здійснювати пошук і бронювання відповідно до власних уподобань та фінансових можливостей. Важливою особливістю платформи є її глобальне покриття, що дозволяє знаходити варіанти проживання практично в будь-якому куточку світу.

Платформа характеризується широким функціоналом, який включає детальні описи об'єктів, галерею фотографій, інтерактивну карту розташування, а також систему оцінювання та відгуків користувачів. Система рецензій виступає важливим інструментом підтримки прийняття рішень, оскільки дозволяє потенційним гостям ознайомитися з досвідом попередніх клієнтів. Інтерфейс вебверсії платформи вирізняється інтуїтивністю та зручністю, що забезпечує

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						14
Змн.	Арк.	№ докум.	Підпис	Дата		

простий доступ до основних функцій, зокрема пошуку, фільтрації та сортування результатів за різними критеріями.

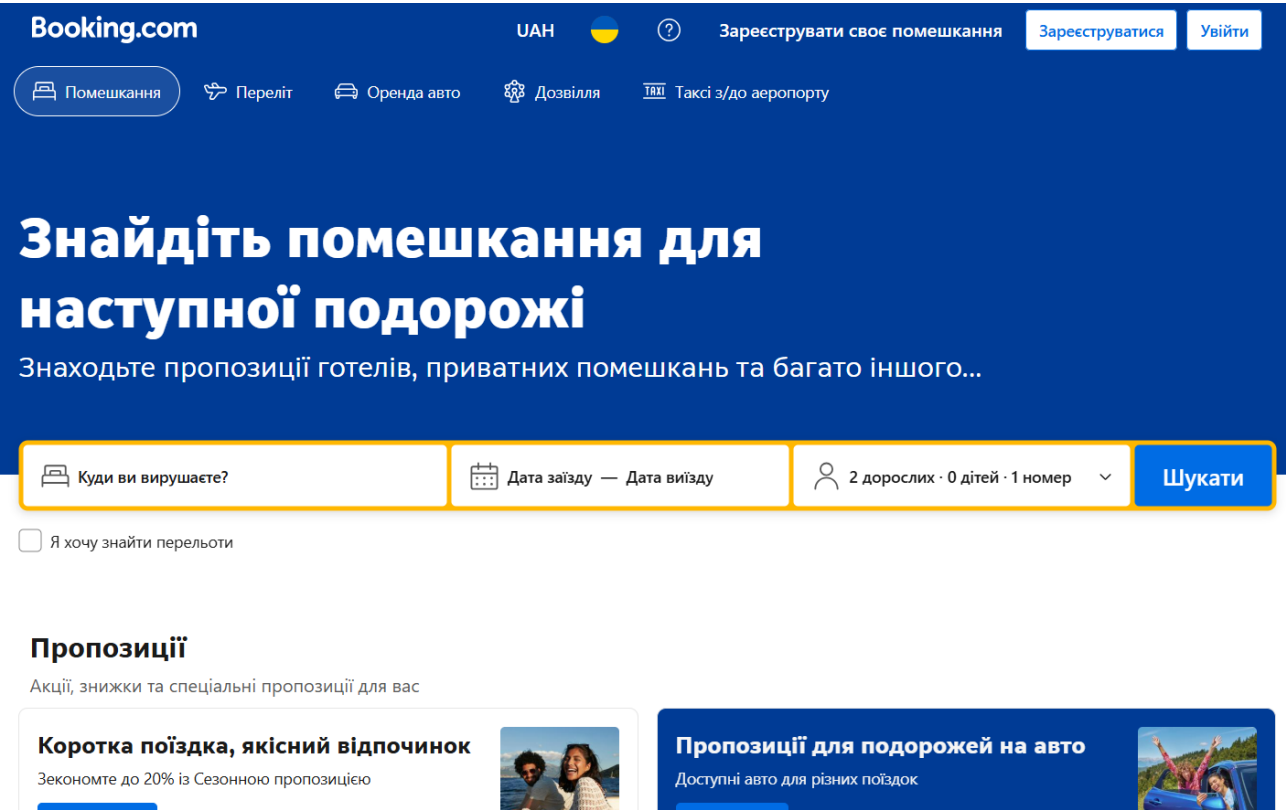


Рисунок 1.1 – Головна сторінка онлайн-платформи Booking.com

Водночас, функціональність Booking.com має певні недоліки, що можуть обмежувати користувацький досвід. Одним із ключових викликів є висока конкуренція серед об'єктів розміщення, що ускладнює просування пропозицій малих готелів на фоні великої кількості альтернатив. Крім того, обмежена можливість прямої комунікації між користувачами та власниками об'єктів знижує ефективність взаємодії, що у деяких випадках призводить до затримок у відповіді на запити або неможливості оперативно вирішити проблеми.

Також іноді фіксуються розбіжності між інформацією, представленою на сайті, та реальними умовами проживання, що може свідчити про обмежений контроль достовірності даних. Ще одним недоліком є обмежені можливості для одночасного порівняння кількох варіантів проживання за різними параметрами, що створює труднощі для користувачів, які прагнуть провести ґрунтовний аналіз перед бронюванням.

Таким чином, Booking.com є прикладом масштабної платформи з розвиненою системою онлайн-бронювання, однак її недоліки, зокрема обмежені інструменти порівняння та недостатня гнучкість у взаємодії з власниками, можуть бути враховані як напрямки удосконалення при розробці нового вебдодатку.

Expedia.com (рис. 1.2) є також однією з провідних онлайн-платформ для бронювання житла, що поєднує функції пошуку, порівняння цін і безпосереднього бронювання проживання, а також надає додаткові туристичні послуги [4]. Сервіс забезпечує користувачам можливість організації комплексних подорожей завдяки інтеграції таких функцій, як підбір авіаквитків, оренда автомобілів та формування пакетних пропозицій, що включають кілька елементів одночасно. Завдяки такій багатфункціональності платформа користується популярністю серед туристів, які прагнуть оптимізувати організацію своїх поїздок через єдиний ресурс.

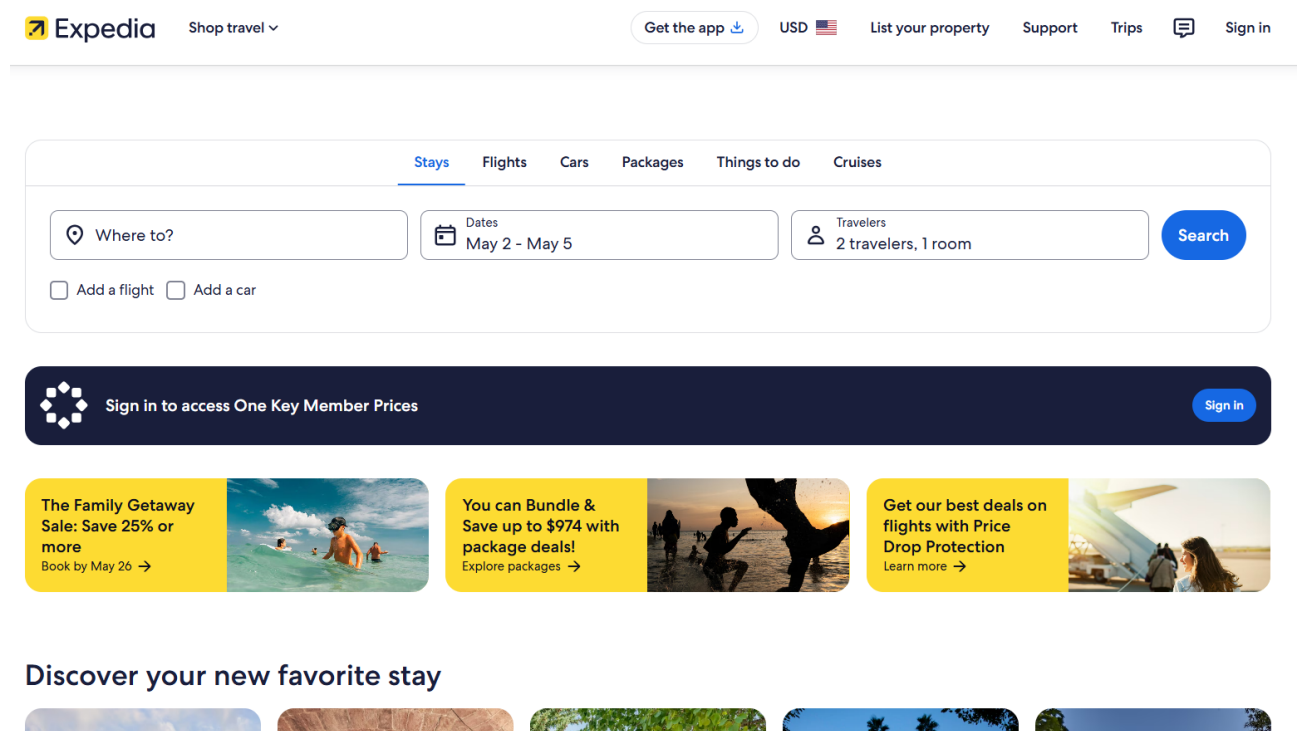


Рисунок 1.2 – Домашня сторінка сторінка онлайн-системи

Серед переваг Expedia можна виділити широкий вибір варіантів розміщення – від готелів і апартаментів до вілл у різних країнах світу. Платформа пропонує інструменти для порівняння цін між різними постачальниками послуг,

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						16
Змн.	Арк.	№ докум.	Підпис	Дата		

що дозволяє користувачам знаходити більш вигідні умови проживання. Важливою функцією є можливість ознайомитися з відгуками та оцінками попередніх гостей, що створює додаткову інформаційну підтримку для прийняття рішень. Крім того, зареєстрованим користувачам сервіс надає знижки, що підвищує їхню лояльність до платформи.

Проте, попри широкий функціонал, Expedia має низку обмежень, які впливають на якість користувацького досвіду. Однією з проблем є недостатня деталізація інформації про готелі: на сторінках окремих об'єктів відсутні повні описи, а можливість переглянути додаткові деталі про конкретні номери обмежена або взагалі відсутня. Це ускладнює процес ознайомлення з умовами проживання та змушує користувачів приймати рішення на основі обмежених даних.

Ще однією проблемою є орієнтація сервісу на користувачів окремих країн, що проявляється у виборі мови інтерфейсу та валюти – обидва параметри часто встановлюються автоматично, без можливості зміни, що створює незручності для туристів з інших регіонів. Додатково користувачі відзначають недосконалість інтерфейсу, зокрема труднощі в пошуку окремих функцій або фільтрів, що ускладнює навігацію по сайту.

Варто також зазначити, що Expedia стягує окремі комісійні збори за бронювання, що може збільшувати кінцеву вартість послуг. Окрім того, на платформі час від часу виникають ситуації, коли інформація про наявність номерів або умови проживання не відповідає дійсності, що може бути наслідком недостатнього контролю за даними або рекламного впливу з боку готелів. Обмеженість вибору в окремих регіонах через ексклюзивні угоди з постачальниками також впливає на різноманітність пропозицій.

Таким чином, Expedia є багатофункціональною платформою, що надає користувачам комплексний набір послуг для організації подорожей, проте має обмеження, пов'язані з деталізацією інформації, інтерфейсними рішеннями, доступністю даних та комунікацією з користувачами.

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						17
Змн.	Арк.	№ докум.	Підпис	Дата		

Airbnb є наступною цифровою платформою для короткострокової оренди житла, яка забезпечує взаємодію між власниками житлових об'єктів та мандрівниками [1]. Цей сервіс, що поєднує вебсайт та мобільний додаток, дозволяє користувачам знаходити, бронювати та оцінювати житло по всьому світу. На відміну від традиційних платформ бронювання готелів, Airbnb спеціалізується переважно на приватних варіантах розміщення: квартирах, будинках, кімнатах, а також унікальних об'єктах на кшталт будиночків на деревах чи навіть замків.

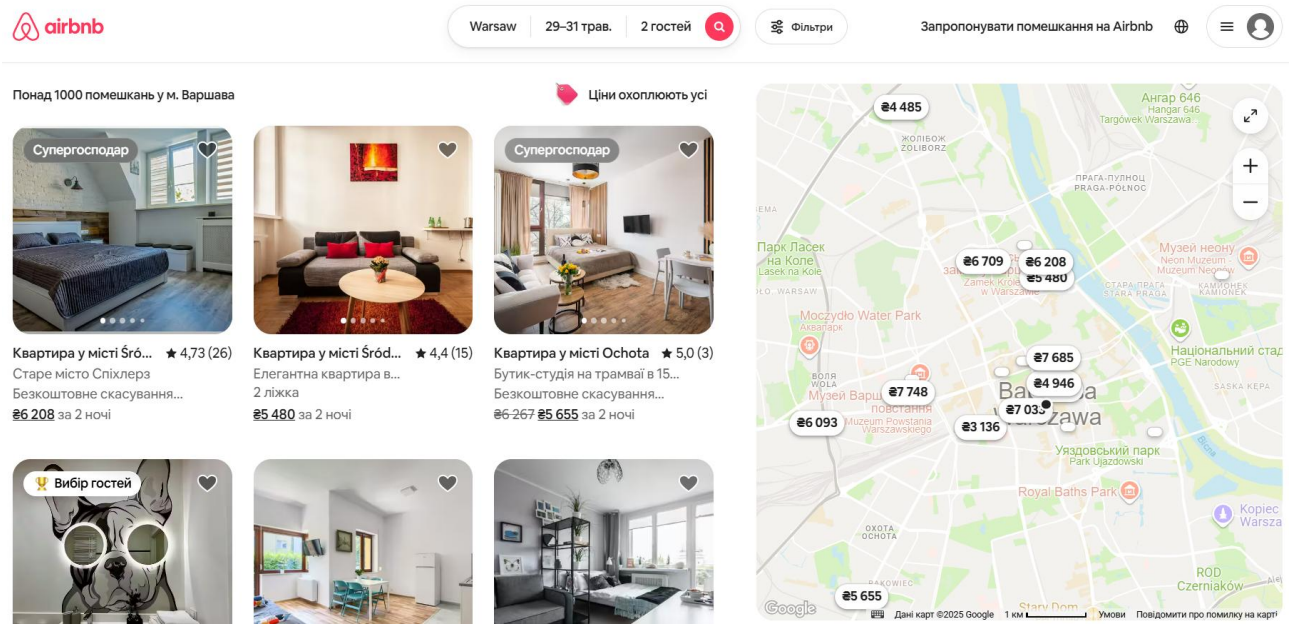


Рисунок 1.3 – Сторінка для пошуку помешкання на платформі Airbnb

Платформа відзначається широким функціоналом для користувачів, включаючи розгалужену систему фільтрів, можливість створювати списки бажаних варіантів, інтеграцію мапи для зручної навігації, а також систему рейтингу й відгуків. Серед технічних переваг слід відзначити наявність API, що дозволяє стороннім розробникам інтегрувати функціонал Airbnb в інші сервіси або створювати власні додатки на його основі.

Airbnb також вирізняється високим рівнем уваги до питань довіри та безпеки, що реалізовано через систему страхування, гарантії захисту власників, перевірку особистих даних користувачів та захищену платіжну платформу.

Завдяки цим заходам платформа підвищує рівень довіри між учасниками сервісу та створює передумови для лояльності клієнтів.

Однак, попри значні переваги, Airbnb має і низку обмежень. Одним із ключових недоліків є орієнтація платформи переважно на приватних власників житла, що обмежує можливості готелів щодо ефективного розміщення своїх пропозицій та просування послуг. Крім того, інтерфейс платформи демонструє окремі недоліки при зміні мови, що проявляється, зокрема, в некоректному відображенні елементів дизайну при виборі окремих мовних версій, наприклад української.

Серед функціональних обмежень слід зазначити відсутність можливості вибору країни для попереднього фільтрування міст, що ускладнює пошук для користувачів, які шукають варіанти лише в межах конкретної країни. Окрім цього, платформа демонструє обмежений набір інструментів для готелів, що звужує їхні можливості керування контентом порівняно з приватними орендодавцями.

Таким чином, Airbnb має обмеження функціональності для готелів, технічні недоліки інтерфейсу, залежність від господарів та труднощі у користувацькому сервісі, що можуть розглядатися під час власної розробки.

1.3. Вибір архітектури вебдодатку

Вибір відповідної архітектури є критично важливим етапом проектування будь-якої програмної системи, оскільки від нього залежить ефективність взаємодії між компонентами додатку, масштабованість, стабільність та можливість подальшого розвитку. З огляду на функціональні вимоги вебдодатку для онлайн-бронювання з функцією порівняння, було прийнято рішення про впровадження архітектури клієнт-серверного типу з чітким розподілом ролей між фронтендом і бекендом, що взаємодіють через RESTful API. Такий підхід забезпечує гнучкість системи, полегшує її підтримку та дозволяє масштабувати окремі компоненти залежно від потреб. [10]

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						19
Змн.	Арк.	№ докум.	Підпис	Дата		

Серед варіантів архітектур, що розглядалися для реалізації системи, можна виділити:

- Монолітну архітектуру, яка характеризується об'єднанням усіх компонентів у єдиній програмі. Перевагами такого підходу є простота розгортання, проте він має обмеження у масштабованості й гнучкості.
- Архітектуру з розділенням на фронтенд і бекенд (клієнт-серверна модель), що передбачає незалежну розробку інтерфейсу користувача та серверної частини з комунікацією через API. Цей підхід забезпечує більшу гнучкість і дозволяє масштабувати частини системи окремо.
- Мікросервісну архітектуру, що передбачає розподіл системи на окремі автономні сервіси. Такий підхід є складнішим у розробці та потребує значних ресурсів для налаштування взаємодії й моніторингу.

Зважаючи на функціональні та нефункціональні вимоги системи, а також обмеженість часу й ресурсів, для розробки було обрано архітектуру клієнт-серверного типу з RESTful API, як оптимальне рішення.

Вибір архітектури клієнт-серверного типу з використанням RESTful API обґрунтований потребою у чіткому розмежуванні фронтенду та бекенду для забезпечення гнучкості й масштабованості системи [9]. Такий підхід дозволяє окремо оновлювати інтерфейс користувача та серверну частину без негативного впливу на інші компоненти. Крім того, використання REST API забезпечує можливість інтеграції з іншими сервісами та платформами у майбутньому.

Застосування сучасного технологічного стеку: Laravel, React, MySQL, MinIO – дозволяє реалізувати надійну, безпечну та продуктивну систему, а використання Docker-контейнерів забезпечує стабільність роботи застосунку незалежно від середовища розгортання.

Архітектура вебдодатку реалізована на основі клієнт-серверної моделі, що передбачає розділення системи на такі основні компоненти:

1. Frontend – односторінковий застосунок (SPA), реалізований на React. Відповідає за взаємодію з користувачем, відображення даних та надсилання

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						20
Змн.	Арк.	№ докум.	Підпис	Дата		

запитів до серверу через REST API.

2. Backend – серверна частина, побудована на фреймворку Laravel (PHP), що обробляє бізнес-логіку, взаємодіє з базою даних MySQL, файловим сховищем MinIO, а також відповідає за відправку email-повідомлень через SMTP.

3. База даних – MySQL, використовується для зберігання основних даних додатку (користувачі, готелі, кімнати, бронювання, відгуки тощо).

4. Файлове сховище – MinIO, що використовується для зберігання медіафайлів (фото готелів і кімнат).

5. Поштовий сервіс – SMTP сервер, через який здійснюється надсилання повідомлень користувачам.

Усі сервіси розгорнуті у Docker-контейнерах, що забезпечує їх ізоляцію, полегшує налаштування, автоматизує розгортання та спрощує масштабування системи в майбутньому [12].

Фронтенд і бекенд працюють як незалежні сервіси, обмін даними між якими здійснюється через стандартизовані HTTP-запити до RESTful API. Це дозволяє підвищити гнучкість системи, спрощує модифікацію окремих компонентів та відкриває можливості для інтеграції з іншими клієнтськими застосунками. Така архітектура відповідає актуальним вимогам ринку та завданням системи онлайн-бронювання з функціями порівняння.

1.4. Обґрунтування вибору інструментальних засобів для розробки вебдодатку

Зважаючи на функціональні потреби вебдодатку для онлайн-бронювання з функціями порівняння, а також на вимоги щодо гнучкості, безпеки та інтеграційних можливостей, було обрано сучасний набір технологій, що забезпечує реалізацію усіх необхідних аспектів системи.

У якості серверної частини (backend) було використано фреймворк Laravel (мова програмування PHP). Laravel є одним із найпопулярніших PHP-фреймворків, що надає розвинений набір функцій для швидкої розробки, зокрема

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						21
Змн.	Арк.	№ докум.	Підпис	Дата		

реалізацію маршрутизації, системи аутентифікації, роботи з базою даних через ORM Eloquent, обробки запитів REST API та middleware [6]. Laravel також відомий своєю зручністю у підтримці масштабованих вебрішень, що критично важливо для платформи онлайн-бронювання, яка має обробляти численні запити користувачів.

Фронтенд додатку реалізовано з використанням бібліотеки React, що дозволяє створити односторінковий застосунок (SPA) з динамічним, інтерактивним інтерфейсом. React забезпечує високу швидкодію завдяки віртуальному DOM, а також підтримує компонентний підхід, що полегшує розширення та обслуговування інтерфейсу [18]. Вибір React обґрунтовано необхідністю реалізації гнучкого й зручного інтерфейсу, що дозволяє користувачам швидко здійснювати пошук, порівняння та бронювання об'єктів розміщення.

У якості системи керування базами даних обрано MySQL – надійну та широко використовувану реляційну СКБД, яка забезпечує стабільну роботу, підтримку транзакцій, високий рівень сумісності з PHP та Laravel, а також масштабованість для обробки великих обсягів даних. [7] MySQL дозволяє ефективно зберігати структуровані дані про користувачів, об'єкти розміщення, номери, бронювання та інші сутності, що використовуються у системі.

Для адміністрування бази даних використовується phpMyAdmin, що забезпечує зручний вебінтерфейс для роботи з MySQL. Це рішення дозволяє швидко виконувати основні операції з даними, включаючи створення таблиць, налаштування зв'язків, виконання SQL-запитів, резервне копіювання та відновлення даних [5].

З метою зберігання медіа-файлів (зображень готелів та номерів) інтегровано файлове сховище MinIO. Використання MinIO дозволяє зберігати файли у розподіленому об'єктному сховищі з високим рівнем доступності, що спрощує керування медіа-контентом та інтеграцію з Laravel через стандартні драйвери.

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						22
Змн.	Арк.	№ докум.	Підпис	Дата		

Для обробки електронної пошти застосовується SMTP-сервер, через який реалізується відправлення email-повідомлень користувачам, зокрема підтвердження реєстрації, сповіщення про статус бронювання та інші сервісні повідомлення. Використання SMTP забезпечує сумісність із більшістю поштових сервісів та підтримує стандартні протоколи безпечної передачі даних.

Важливою частиною інфраструктури вебдодатку є застосування Docker-контейнерів для розгортання усіх компонентів системи. Кожен сервіс (бекенд, фронтенд, база даних, файлове сховище, phpMyAdmin, SMTP) розгорнуто в окремому контейнері, що забезпечує ізоляцію, повторюваність середовища та легкість масштабування. Docker дозволяє стандартизувати процес налаштування, спрощує оновлення та розгортання додатку на різних хостингових платформах [8].

Таким чином, обраний набір інструментальних засобів повністю відповідає вимогам до системи онлайн-бронювання, забезпечуючи комплексне рішення з високим рівнем продуктивності, надійності, масштабованості та підтримкою сучасних стандартів та вимог веброзробки.

Висновки до першого розділу

Проведене у першому розділі дослідження дозволило системно оцінити передумови, потреби та сучасні тенденції, що формують вимоги до вебдодатку для онлайн-бронювання з функціями порівняння. Було проаналізовано як загальносвітові процеси цифровізації туристичної галузі, так і локальні виклики, пов'язані з обмеженими можливостями малих готельних підприємств у просуванні власних послуг через великі міжнародні платформи.

Результати аналізу підтверджують, що сучасний туристичний ринок вимагає від цифрових сервісів не лише базових функцій пошуку та бронювання, а й розширених можливостей для детального порівняння варіантів розміщення за різними параметрами. Більшість існуючих платформ орієнтовані на масовий сегмент, надаючи перевагу великим готельним мережам, що обмежує видимість

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						23
Змн.	Арк.	№ докум.	Підпис	Дата		

невеликих об'єктів на ринку. Це створює потребу в інноваційних рішеннях, що поєднуюватимуть високий рівень користувацького досвіду, зручність взаємодії та рівні можливості для всіх учасників.

Аналіз аналогів показав, що такі сервіси, як Booking.com, Expedia та Airbnb, хоча й володіють потужним функціоналом, мають низку недоліків, які впливають на зручність користування та доступність інформації. Зокрема, виявлено обмеження щодо гнучкості фільтрації, відсутність ефективних інструментів порівняння варіантів проживання, складності у прямій комунікації з власниками об'єктів, а також значні фінансові бар'єри для участі малих готелів. Ці аспекти вказують на існуючу нішу для нових рішень, здатних поєднати гнучкість, відкритість і персоналізацію сервісів.

У результаті проведеної роботи було сформовано перелік завдань, необхідних для створення конкурентоспроможного вебдодатку, що відповідає актуальним потребам ринку. Особливий акцент зроблено на необхідності інтеграції функцій порівняння номерів, підтримці гнучкого налаштування пошукових параметрів, забезпеченні прозорості інформації щодо умов проживання та фінансових умов, а також наданні зручних інструментів управління для власників об'єктів.

Також було обґрунтовано вибір архітектурного підходу, що передбачає клієнт-серверну модель із розділенням фронтенду та бекенду, взаємодіючих через RESTful API. Така архітектура забезпечує гнучкість масштабування, можливість інтеграції сторонніх сервісів та зручність супроводу. Використання сучасних технологій (Docker, Laravel, MySQL, React, MinIO) дозволяє створити надійну, продуктивну та безпечну систему, адаптовану до потреб різних користувачів.

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						24
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 2. ПРОЄКТУВАННЯ ВЕБДОДАТКУ ДЛЯ ОНЛАЙН-БРОНЮВАННЯ З ФУНКЦІЯМИ ПОРІВНЯННЯ

2.1. Визначення варіантів використання та структури системи

Для успішної реалізації вебдодатку для онлайн-бронювання з функціями порівняння необхідно чітко визначити сценарії взаємодії користувачів із вебсайтом та сформувати його логічну структуру. Це дозволить не лише забезпечити відповідність вимогам кінцевих користувачів, а й реалізувати гнучку, масштабовану та ефективну архітектуру платформи.

Проектування будь-якої інформаційної системи починається з визначення її функціональних можливостей та способів взаємодії користувачів із системою. Одним із ефективних методів опису таких взаємодій є варіанти використання (use cases). Варіанти використання – це опис сценаріїв, які показують, як різні типи користувачів (актори) взаємодіють із системою для досягнення своїх цілей [13]. Вони відображають функціональність системи «з точки зору користувача», дозволяючи чітко визначити очікувані дії, результати та умови виконання тих чи інших операцій.

Використання use-case діаграм на етапі проектування є важливим, оскільки вони дозволяють системно й структуровано підійти до розробки програмного продукту. Це дає змогу:

- визначити усі необхідні функції системи відповідно до ролей користувачів;
- уникнути пропусків чи надмірного дублювання функціоналу;
- спростити комунікацію між розробникам та кінцевими користувачами завдяки зрозумілому опису роботи системи;
- закласти основу для майбутнього технічного завдання та розробки.

Вебдодаток для онлайн-бронювання з функцією порівняння орієнтований на обслуговування кількох категорій користувачів, кожна з яких має свої завдання, права доступу та функціональні можливості. У системі визначено три

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						25
Змн.	Арк.	№ докум.	Підпис	Дата		

основні ролі: користувач (клієнт), власник об'єкта розміщення та адміністратор. Кожна з ролей взаємодіє з системою відповідно до своїх потреб і призначення. Для кожної з цих ролей визначено типові сценарії використання, які були також візуалізовані у вигляді діаграми варіантів використання.

Користувач – це кінцевий споживач послуг платформи, який шукає місце для тимчасового проживання. Його роль передбачає взаємодію з системою для здійснення пошуку житла, перегляду пропозицій, порівняння номерів за обраними критеріями, а також оформлення бронювання. Після завершення проживання користувач має можливість залишати відгуки та оцінки про об'єкт розміщення.

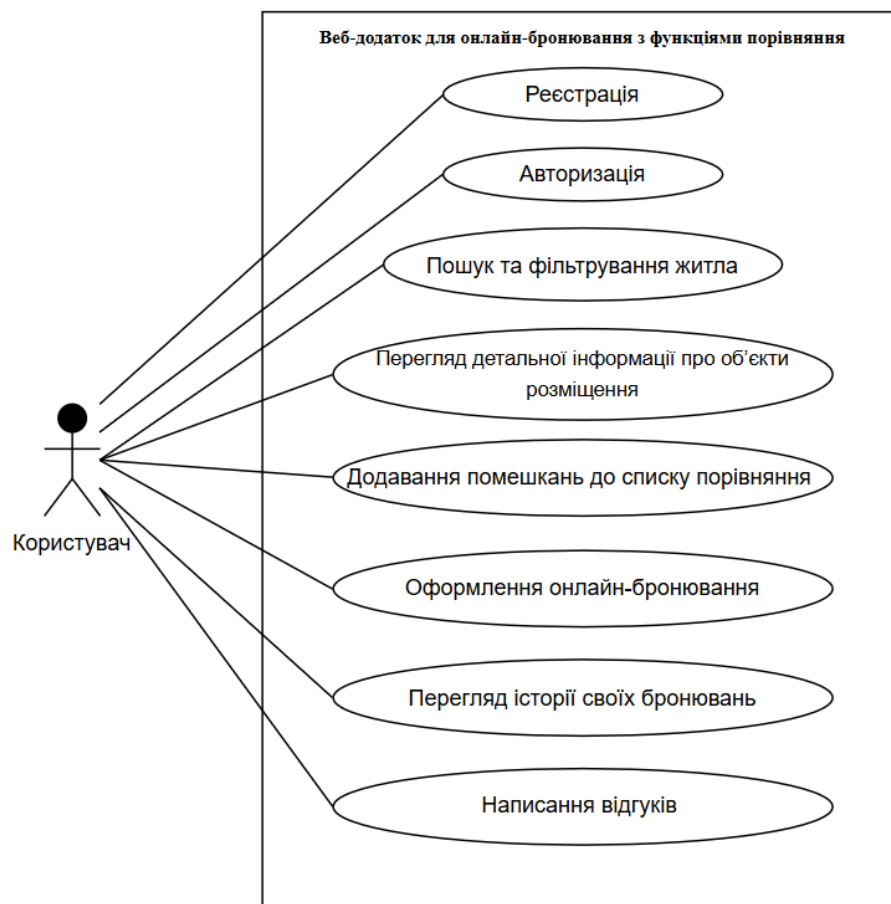


Рисунок 2.1 – Діаграма варіантів використання для користувачів
Основні функції користувача (див. рис. 2.1):

- реєстрація та авторизація в системі;

- пошук та фільтрація житла за місцем розташування, ціною, кількістю спальних місць, наявністю зручностей тощо;
- перегляд детальної інформації про об'єкти розміщення та окремі кімнати;
- додавання кімнат до списку порівняння для зручного вибору;
- оформлення онлайн-бронювання;
- перегляд історії своїх бронювань;
- написання відгуків та виставлення оцінок після проживання.

Власник об'єкта розміщення – це особа, яка керує власними готелями чи іншими об'єктами проживання, представленими на платформі. Його основне завдання – підтримка актуальності інформації про свої об'єкти, управління пропозиціями та моніторинг бронювань.

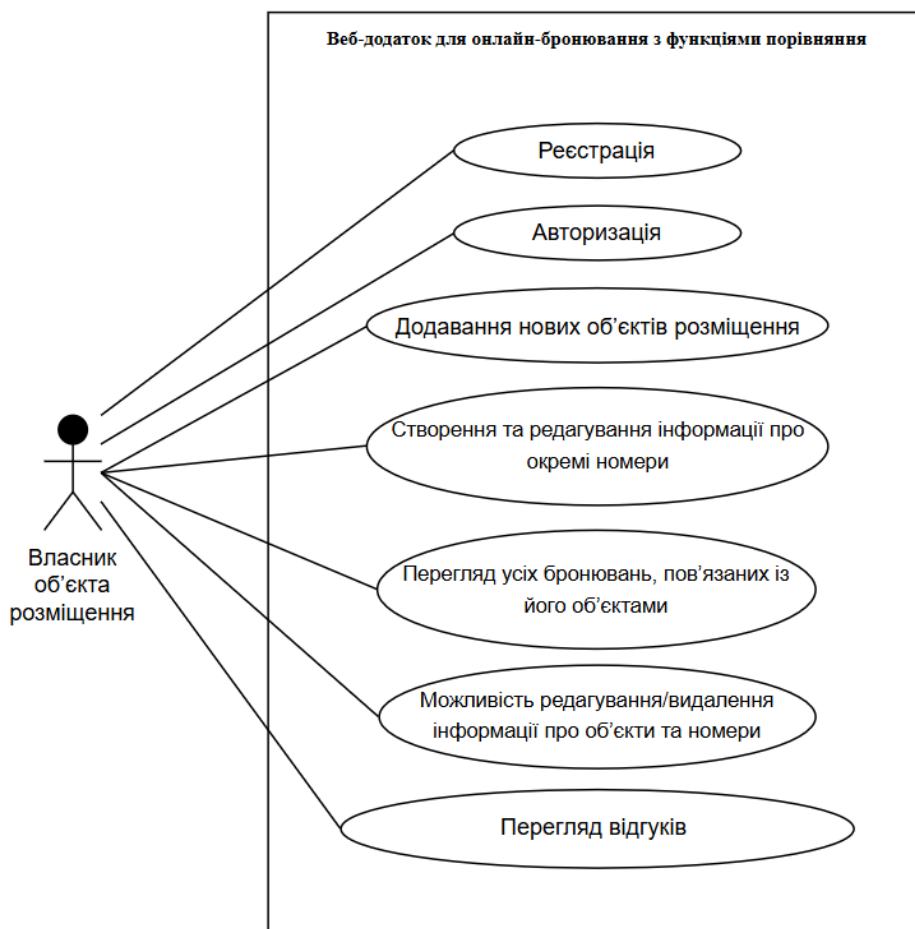


Рисунок 2.2 – Діаграма варіантів використання для власників помешкання
Власник має доступ до наступного функціоналу (див. рис. 2.2):

- реєстрація власного облікового запису;
- додавання нових об'єктів розміщення, включаючи їхні описи, фотографії, зручності, місце розташування;
- створення та редагування інформації про окремі кімнати (кількість місць, ціна, площа, наявність ванної кімнати, тощо);
- перегляд усіх бронювань, пов'язаних із його об'єктами;
- можливість редагування або видалення інформації про об'єкти та номери;
- модерація відгуків, залишених користувачами про його об'єкти (наприклад, подання скарги на неправдивий відгук).

Адміністратор – це особа, відповідальна за підтримку стабільної роботи платформи, модерацію контенту та управління глобальними даними. Його роль передбачає найширший рівень доступу до системи. Основні завдання адміністратора:

- модерація та підтвердження нових об'єктів розміщення перед їхньою публікацією на платформі;
- управління довідниками системи (типи готелів, типи переваг, доступні зручності);
- редагування та видалення записів довідників у разі потреби;
- контроль за запитами власників щодо додавання нових зручностей чи населених пунктів;
- забезпечення дотримання правил використання платформи.

Кожна роль у системі наділена конкретними функціональними можливостями та рівнем доступу, що гарантує необхідний рівень захисту даних, безпеки та ефективного управління платформою. Такий підхід до розподілу прав доступу відповідає стандартам розробки багатокористувацьких вебдодатків і забезпечує упорядковану взаємодію між усіма учасниками системи.

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						28
Змн.	Арк.	№ докум.	Підпис	Дата		

2.2. Проектування та розробка бази даних вебдодатку

Етап проектування та розробки бази даних відіграє ключову роль у процесі створення будь-якої інформаційної системи, адже саме база даних формує основу для структурування та обробки інформації, з якою працюватиме програмне забезпечення. Якісно спроектована база даних забезпечує стабільну роботу системи та оперативний доступ до необхідних даних.

Раціональне проектування структури бази даних підвищує її адаптивність і здатність до масштабування. Завдяки цьому надалі можна безперешкодно розширювати функціонал вебдодатку відповідно до змін у вимогах чи умовах його використання.

Для реалізації бази даних у межах даного проєкту було обрано систему керування MySQL, що обумовлено її численними перевагами. По-перше, ця система керування базами даних (СКБД) є однією з найпоширеніших у світі, широко застосовується в проєктах різного масштабу й активно підтримується спільнотою, що забезпечує її стабільність і надійність.

MySQL відома своєю високою продуктивністю та здатністю до масштабування, що особливо важливо для вебдодатків, які працюють з великим обсягом даних [2]. Крім того, MySQL надає розвинені засоби керування правами доступу, що дозволяє забезпечити високий рівень безпеки збережених даних. Усі ці фактори разом роблять MySQL оптимальним рішенням для створення надійної, масштабованої та безпечної бази даних.

У процесі проектування були враховані всі необхідні вимоги, що дозволило сформулювати логічно впорядковану та добре організовану структуру бази даних (рис. 2.3).

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						29
Змн.	Арк.	№ докум.	Підпис	Дата		

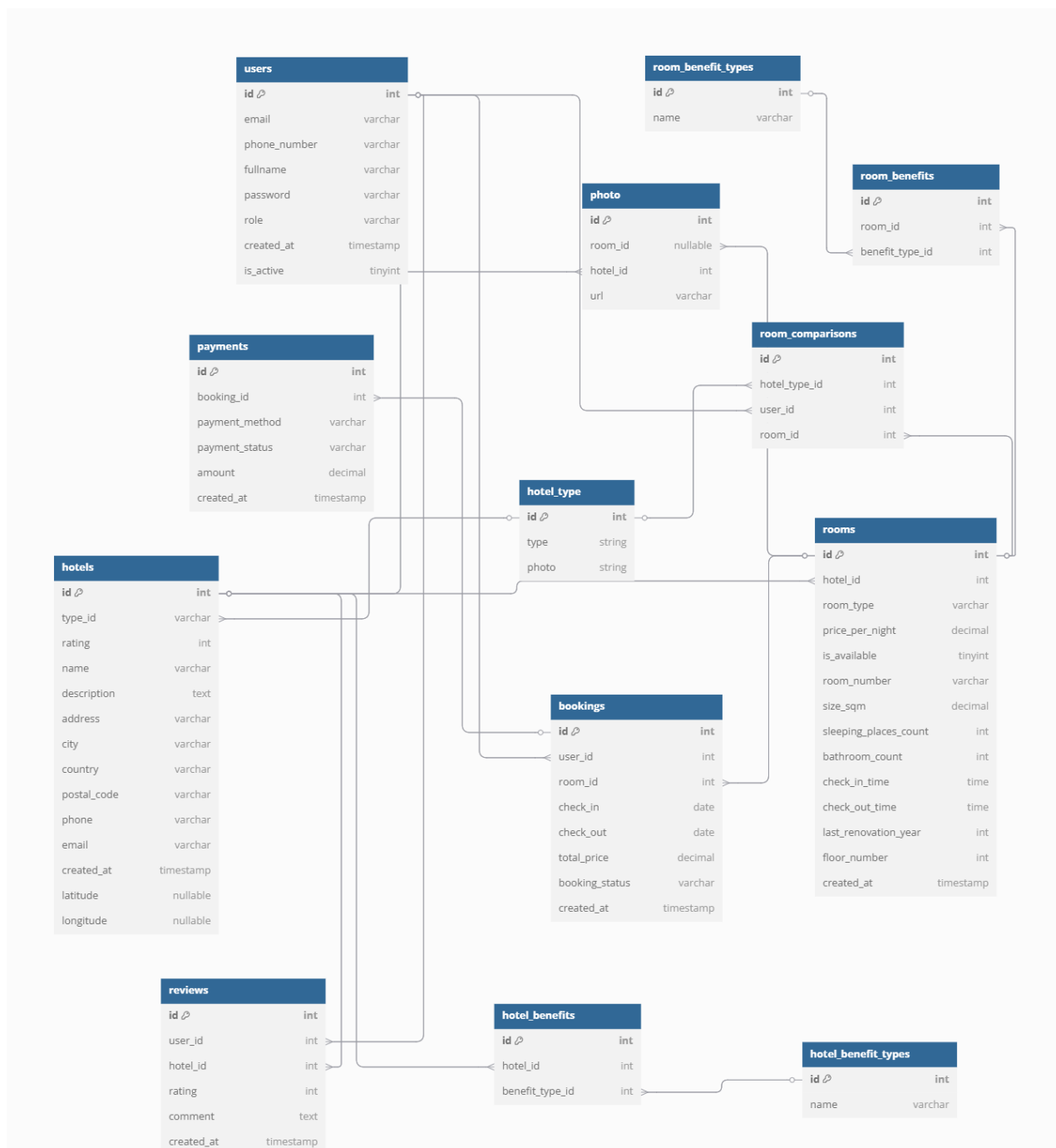


Рисунок 2.3 – Структура бази даних вебдодатку

Таблиця Users відповідає за зберігання облікових записів користувачів (табл. 2.1). У ній містяться дані про електронну пошту, номер телефону, повне ім'я, роль у системі (користувач, власник або адміністратор), дату створення профілю, пароль та статус активності.

Таблиця 2.1

Таблиця Users

Назва поля	Тип даних	РК	FK	Примітка
id	int	+	-	унікальний ідентифікатор
email	varchar(255)	-	-	електронна пошта
phone_number	varchar(255)	-	-	номер телефону
fullname	varchar(255)	-	-	прізвище та ім'я
password	varchar(255)	-	-	пароль
role	varchar(255)	-	-	роль
created_at	timestamp	-	-	коли створено користувача
is_active	tinyint	-	-	чи активний користувач

Таблиця Hotels містить розширену інформацію про об'єкти розміщення (табл. 2.2). Вона включає такі поля, як тип готелю, рейтинг, назва, опис, адреса, місто, країна, поштовий індекс, контактні дані, координати (широта і довгота) та дата створення запису.

Таблиця 2.2

Таблиця Hotels

Назва поля	Тип даних	РК	FK	Примітка
id	int	+	-	унікальний ідентифікатор
type_id	int	-	+	ідентифікатор типу готелю
rating	int	-	-	рейтинг
name	varchar(255)	-	-	назва
description	text	-	-	детальний опис
address	varchar(255)	-	-	адреса
city	varchar(255)	-	-	місто
country	varchar(255)	-	-	країна
postal_code	varchar(255)	-	-	поштовий індекс
phone	varchar(255)	-	-	номер телефону
email	varchar(255)	-	-	електронна пошта
created_at	timestamp	-	-	час створення
latitude	varchar(255)	-	-	широта
longitude	varchar(255)	-	-	довгота

Таблиця Rooms зберігає дані про окремі кімнати, прив'язані до відповідних готелів (табл. 2.3). У ній фіксуються характеристики номерів, зокрема тип кімнати, площа, кількість спальних місць та ванних кімнат, вартість за добу, час заселення/виселення, останній рік ремонту, доступність до бронювання та дата додавання.

Таблиця Rooms

Назва поля	Тип даних	РК	FK	Примітка
id	int	+	-	унікальний ідентифікатор
hotel_id	int	-	+	ідентифікатор готелю
room_type	varchar(255)	-	-	тип кімнати
price_per_night	decimal	-	-	ціна за добу
is_available	tinyint	-	-	чи доступна для бронювання
room_number	varchar(255)	-	-	номер кімнати
size_sqm	decimal	-	-	площа кімнати
sleeping_places_count	int	-	-	кількість спальних місць
bathroom_count	int	-	-	кількість ванних кімнат
check_in_time	time	-	-	час заселення
check_out_time	time	-	-	час виселення
last_renovation_year	int	-	-	рік останнього ремонту
created_at	timestamp	-	-	час створення

Таблиця Photo призначена для зберігання шляхів до зображень (табл. 2.4), що ілюструють як готелі, так і окремі кімнати, забезпечуючи візуальну підтримку під час вибору житла.

Таблиця 2.4

Таблиця Photo

Назва поля	Тип даних	РК	FK	Примітка
id	int	+	-	унікальний ідентифікатор
room_id	int	-	+	ідентифікатор кімнати
hotel_id	int	-	+	ідентифікатор готелю
url	varchar(255)	-	-	шлях до зображення

Таблиця Bookings фіксує операції бронювання (табл. 2.5), які здійснюють користувачі. Вона містить інформацію про користувача, кімнату, дати заїзду і виїзду, загальну вартість бронювання, його статус та дату створення замовлення.

Таблиця 2.5

Таблиця Bookings

Назва поля	Тип даних	РК	FK	Примітка
1	2	3	4	5
id	int	+	-	унікальний ідентифікатор
user_id	int	-	+	ідентифікатор користувача
room_id	int	-	+	ідентифікатор кімнати

Продовження таблиці 2.5

1	2	3	4	5
check_in	date	-	-	дата заселення
check_out	date	-	-	дата виселення
total_price	decimal	-	-	загальна вартість
booking_status	varchar(255)	-	-	статус бронювання
created_at	timestamp	-	-	час створення

Таблиця Payments реалізує логіку зберігання платіжної інформації (табл. 2.6). У ній міститься спосіб оплати, статус транзакції, сума та зв'язок із конкретним бронюванням.

Таблиця 2.6

Таблиця Payments

Назва поля	Тип даних	РК	ФК	Примітка
id	int	+	-	унікальний ідентифікатор
booking_id	int	-	+	ідентифікатор бронювання
payment_method	varchar(255)	-	-	спосіб оплати
payment_status	varchar(255)	-	-	статус оплати
amount	decimal	-	-	сума
created_at	timestamp	-	-	час створення

Таблиця Reviews акумулює відгуки користувачів про готелі (табл. 2.7). Вона включає оцінку, текстовий коментар, посилання на користувача та готель, а також дату створення відгуку.

Таблиця 2.7

Таблиця Reviews

Назва поля	Тип даних	РК	ФК	Примітка
id	int	+	-	унікальний ідентифікатор
user_id	int	-	+	ідентифікатор користувача
hotel_id	int	-	+	ідентифікатор готелю
rating	int	-	-	рейтинг
comment	text	-	-	коментар
created_at	timestamp	-	-	час створення

Таблиця Room_comparisons підтримує функціональність порівняння номерів (табл. 2.8), що дає змогу користувачам зберігати вибрані кімнати для подальшого аналізу. Вона включає зв'язок із користувачем, типом готелю і конкретною кімнатою.

Таблиця Room_comparisons

Назва поля	Тип даних	РК	FK	Примітка
id	int	+	-	унікальний ідентифікатор
user_id	int	-	+	ідентифікатор користувача
hotel_type_id	int	-	+	ідентифікатор типу готелю
room_id	int	-	+	ідентифікатор кімнати

Таблиця Hotel_type містить класифікацію типів готелів (табл. 2.9), таких як готель, апартаменти, хостел тощо, і слугує довідником типів розміщення, включаючи зображення до кожного типу.

Таблиця 2.9

Таблиця Hotel_type

Назва поля	Тип даних	РК	FK	Примітка
id	int	+	-	унікальний ідентифікатор
type	varchar(255)	-	-	назва типу
photo	varchar(255)	-	-	зображення

Таблиця Hotel_benefit_types містить перелік типів переваг готелів (табл. 2.10), зокрема Wi-Fi, паркінг, сніданок тощо. Ця таблиця використовується для організації структури зручностей, доступних у різних помешканнях.

Таблиця 2.10

Таблиця Hotel_benefit_types

Назва поля	Тип даних	РК	FK	Примітка
id	int	+	-	унікальний ідентифікатор
name	varchar(255)	-	-	назва типу переваг

Таблиця Room_benefits реалізує зв'язок між окремими кімнатами та їх перевагами (табл. 2.11). Вона слугує проміжною таблицею для реалізації відношення багато-до-багатьох.

Таблиця 2.11

Таблиця Room_benefits

Назва поля	Тип даних	РК	FK	Примітка
id	int	+	-	унікальний ідентифікатор
room_id	int	-	+	ідентифікатор кімнати
benefit_type_id	int	-	+	ідентифікатор типу переваг

Таблиця Room_benefit_types уточнює специфічні переваги, доступні для кімнат (табл. 2.12), і прив'язує їх до відповідного типу переваг.

Таблиця 2.12

Таблиця Room_benefit_types

Назва поля	Тип даних	РК	FK	Примітка
id	int	+	-	унікальний ідентифікатор
name	varchar(255)	-	+	назва переваги
benefit_type_id	int	-	+	ідентифікатор типу переваг

Таблиця Hotel_benefits виконує аналогічну функцію для готелів (табл. 2.13) – зв'язує кожен готель із наявними зручностями на основі типу переваг.

Таблиця 2.13

Таблиця Hotel_benefits

Назва поля	Тип даних	РК	FK	Примітка
id	int	+	-	унікальний ідентифікатор
hotel_id	int	-	+	ідентифікатор готелю
benefit_type_id	int	-	+	ідентифікатор типу переваг

2.3. Реалізація функціоналу роботи вебдодатку для онлайн-бронювання з функціями порівняння

Після визначення варіантів використання, формування вимог, проєктування бази даних та аналізу ключових алгоритмів роботи додатку було розпочато етап безпосередньої реалізації. У цьому розділі детально розглядається процес створення основних функціональних модулів системи.

Для обробки зображень у системі бронювання реалізовано окремий сервіс FileService, що відповідає за завантаження, отримання URL-адреси, видалення та побудову шляху до зображень, збережених у хмарному сховищі MinIO. Сховище налаштоване як S3-сумісне середовище, розгорнуте у Docker-контейнері локальної мережі. Робота здійснюється через SDK Amazon S3, що забезпечує сумісність з MinIO.

Основний клас сервісу має наступну структуру:

```
class FileService
{
    private S3Client $s3Client;
```

```

private $storage;
private string $bucket;

public function __construct()
{
    $this->initializeS3Client();
    $this->storage = Storage::disk('minio');
    $this->bucket = config('filesystems.disks.minio.bucket');
}
...
}

```

У конструкторі класу ініціалізується S3-клієнт, який конфігурується через дані, збережені у файлі конфігурації Laravel (config/filesystems.php). Метод initializeS3Client встановлює параметри з'єднання з MinIO, включаючи ключ доступу, секретний ключ, регіон, версію та кастомний endpoint:

```

private function initializeS3Client(): void
{
    $this->s3Client = new S3Client([
        'credentials' => [
            'key' => config('filesystems.disks.minio.key'),
            'secret' => config('filesystems.disks.minio.secret'),
        ],
        'region' => config('filesystems.disks.minio.region'),
        'version' => 'latest',
        'endpoint' => config('filesystems.disks.minio.endpoint'),
        'use_path_style_endpoint' => true
    ]);
}

```

Перед завантаженням будь-якого файлу сервіс перевіряє наявність відповідного бакету в MinIO та створює його за потреби:

```

private function ensureBucketExists(): void
{
    if (!$this->s3Client->doesBucketExist($this->bucket)) {
        $this->s3Client->createBucket(['Bucket' => $this->bucket]);
    }
}

```

Метод uploadFile реалізує логіку збереження завантажених зображень у підкаталозі photos, генерує унікальну назву файлу за хешем і повертає результат операції у вигляді асоціативного масиву:

```

public function uploadFile(UploadedFile $file, string $directory = 'photos'): array
{
    $this->ensureBucketExists();
    $path = $this->storage->putFileAs($directory, $file, $file->hashName());

    if ($path === false) {

```

```

        return [...]; // обробка помилки
    }

    return [
        'success' => true,
        'path' => $path
    ];
}

```

Окрему увагу приділено генерації публічного посилання на файл, що реалізується у методі `generateFileUrl`. Якщо встановлено час дії посилання, генерується тимчасовий URL за допомогою `createPresignedRequest`, інакше використовується `getObjectUrl`.

Для видалення файлів реалізовано метод `deleteFile`, який спочатку перевіряє наявність файлу у сховищі, а потім виконує операцію видалення або повертає відповідну помилку:

```

public function deleteFile(string $path): array
{
    if (!$this->storage->exists($path)) {
        return [...]; // обробка випадку, коли файл не знайдено
    }

    $deleted = $this->storage->delete($path);

    if (!$deleted) {
        return [...]; // обробка помилки видалення
    }

    return [
        'success' => true,
        'message' => 'File successfully deleted'
    ];
}

```

Таким чином, сервіс `FileService` є універсальним інтерфейсом для взаємодії з хмарним файловим сховищем, повністю забезпечуючи потреби системи у завантаженні, доступі до зображень та їх подальшому видаленні.

Аутентифікація користувачів у системі реалізована через окремий контролер `AuthController`, який забезпечує створення нових облікових записів, вхід до системи, отримання інформації про користувача та вихід із системи.

- Реєстрація користувача. Метод `register` створює нового користувача із роллю "user" на основі валідованого запиту `StoreUserRequest`. Після створення

облікового запису генерується JWT-токен із вбудованим claim про роль користувача:

```
public function register(StoreUserRequest $request): JsonResponse
{
    $validatedData = $request->validated();

    $user = User::create([
        'id' => Str::uuid()->toString(),
        'fullname' => $validatedData['fullname'],
        'email' => $validatedData['email'],
        'phone_number' => $validatedData['phone_number'],
        'password' => Hash::make($validatedData['password']),
        'role' => 'user',
        'is_active' => 1,
    ]);

    $token = JWTAuth::claims(['role' => $user->role])->fromUser($user);

    return response()->json([
        'user' => $user,
        'token' => $token
    ], 201);
}
```

- Реєстрація власника об'єкта розміщення. Метод registerOwner реалізує створення користувача із роллю "owner" одночасно з додаванням нового готелю в систему. Запит повинен містити два набори даних: для користувача та для готелю. Обидва масиви перевіряються вручну через стандартну валідацію Laravel, після чого здійснюється створення записів у таблицях users та hotels.

```
public function registerOwner(Request $request): JsonResponse
{
    $userData = $request->get('user', []);
    $hotelData = $request->get('hotel', []);

    try {
        $this->validate(new Request($userData), (new StoreUserRequest())->rules());
        $this->validate(new Request($hotelData), (new StoreHotelRequest())->rules());
    } catch (ValidationException $e) {
        return response()->json(['errors' => $e->errors()], 422);
    }

    $user = User::create(...); // створення власника
    $hotelData['owner_user_id'] = $user->id;
    Hotel::create($hotelData);

    $token = JWTAuth::claims(['role' => $user->role])->fromUser($user);
    return response()->json(['user' => $user, 'token' => $token], 201);
}
```

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						38
Змн.	Арк.	№ докум.	Підпис	Дата		

- Вхід до системи. Для входу в систему реалізовано метод login, який перевіряє облікові дані користувача (електронну пошту та пароль). У разі успішної перевірки генерується JWT-токен із вбудованою інформацією про роль. Якщо облікові дані недійсні, або виникає помилка з JWT, повертається відповідний код помилки:

```
public function login(Request $request)
{
    $credentials = $request->only('email', 'password');

    try {
        if (! $token = JWTAuth::attempt($credentials)) {
            return response()->json(['error' => 'Invalid credentials'],
401);
        }

        $user = auth()->user();
        $token = JWTAuth::claims(['role' => $user->role])-
>fromUser($user);

        return response()->json(compact('token'));
    } catch (JWTException $e) {
        return response()->json(['error' => 'Could not create token'],
500);
    }
}
```

- Отримання інформації про користувача. Метод getUser дозволяє отримати дані аутентифікованого користувача на основі токена, надісланого у заголовку запиту. У разі недійсного або простроченого токена повертається повідомлення про помилку:

```
public function getUser()
{
    try {
        if (! $user = JWTAuth::parseToken()->authenticate()) {
            return response()->json(['error' => 'User not found'], 404);
        }
    } catch (JWTException $e) {
        return response()->json(['error' => 'Invalid token'], 400);
    }

    return response()->json(compact('user'));
}
```

Щодо реалізації логіки управління бронюваннями, то вона здійснюється за допомогою BookingController. Контролер відповідає за обробку CRUD-операцій

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						39
Змн.	Арк.	№ докум.	Підпис	Дата		

(створення, перегляд, видалення), перевірку прав доступу користувачів та автоматичне обчислення вартості бронювання.

Контролер підтримує аутентифікацію через токени, використовуючи рольову модель доступу, що базується на трьох типах користувачів: адміністратор, власник та звичайний користувач.

Метод `index` повертає список бронювань залежно від ролі аутентифікованого користувача:

- адміністратор отримує всі бронювання системи;
- звичайний користувач бачить лише власні бронювання;
- власник має доступ до бронювань, пов'язаних із його готелями.

```
public function index(): JsonResponse
{
    $user = auth()->user();
    switch ($user->role) {
        case Role::ADMIN:
            $bookings = Booking::with('room.hotel')->get();
            break;
        case Role::USER:
            $bookings = Booking::where('user_id', $user->id)-
                >with('room.hotel')->get();
            break;
        case Role::OWNER:
            $bookings = Booking::whereHas('room.hotel', function ($query)
                use ($user) {
                    $query->where('owner_user_id', $user->id);
                })->with('room.hotel')->get();
            break;
        default:
            return response()->json(['message' => 'Unauthorized'], 403);
    }

    return response()->json(BookingResource::collection($bookings),
        Response::HTTP_OK);
}
```

Метод `store` використовується звичайними користувачами для створення нового бронювання. Перед створенням виконується перевірка вхідних даних та розрахунок повної вартості на основі кількості днів проживання та ціни за добу.

```
public function store(StoreBookingRequest $request)
{
    $data = $request->validated();
    $data['user_id'] = auth()->user()->id;
```

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						40
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        $room = Room::find($data['room_id']);
        $checkIn = new DateTime($data['check_in']);
        $checkOut = new DateTime($data['check_out']);
        $daysCount = $checkIn->diff($checkOut)->days;
        $data['total_price'] = $daysCount * $room->price_per_night;
        $data['booking_status'] = "Pending";

        $booking = Booking::create($data);
        event(new BookingCreated($booking));

        return response()->json($booking, Response::HTTP_CREATED);
    }

```

Метод show реалізує логіку перегляду конкретного бронювання. Перед поверненням відповіді здійснюється авторизація через Gate-механізм Laravel, що перевіряє права доступу до відповідного запису:

```

    public function show(Booking $booking): JsonResponse
    {
        try {
            $this->authorize('view', $booking);
            return response()->json($booking, Response::HTTP_OK);
        } catch (AuthorizationException $e) {
            return response()->json(['message' => 'Unauthorized to view this
booking.'], Response::HTTP_FORBIDDEN);
        }
    }

```

Метод destroy дозволяє видалити бронювання лише за умови проходження перевірки авторизації. Якщо користувач не має відповідних прав, повертається помилка 403:

```

    public function destroy(Booking $booking): JsonResponse
    {
        try {
            $this->authorize('delete', $booking);
            $booking->delete();
            return response()->json([], Response::HTTP_NO_CONTENT);
        } catch (AuthorizationException $e) {
            return response()->json(['message' => 'Unauthorized to delete this
booking.'], Response::HTTP_FORBIDDEN);
        }
    }

```

Контролер BookingController реалізує повний цикл обробки бронювань із урахуванням ролей користувачів, дотриманням принципів безпеки та обмеженням доступу до чужих ресурсів.

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		41

Висновки до другого розділу

У другому розділі було здійснено комплексне проєктування вебдодатку для онлайн-бронювання з функцією порівняння, що включало визначення варіантів використання системи, структурування її логіки та розробку моделі даних. Проведена робота дозволила сформулювати чітке уявлення про функціональні можливості системи, ролі користувачів та їх взаємодію з вебсайтом.

Визначення варіантів використання стало відправною точкою для опису поведінки системи з точки зору кінцевих користувачів. Завдяки детальному аналізу було виділено основні сценарії для трьох ключових ролей – користувача, власника об'єкта розміщення та адміністратора. Кожній ролі було присвоєно набір функцій, що відповідають її призначенню та обсягу повноважень у системі. Такий підхід дозволив не лише систематизувати функціонал, але й закласти основу для реалізації механізмів контролю доступу, що є критично важливим для забезпечення безпеки даних та стабільності роботи платформи.

Важливою складовою проєктування стала розробка структури бази даних. Було обґрунтовано вибір системи керування базами даних MySQL як оптимального інструменту для реалізації проєкту завдяки її продуктивності, надійності, масштабованості та широкій підтримці серед розробників. Створена модель бази даних охоплює всі необхідні сутності для роботи системи, включаючи зберігання інформації про користувачів, готелі, номери, бронювання, платежі, відгуки та порівняння. Кожна таблиця має чітко визначені поля, первинні та зовнішні ключі, що забезпечує цілісність даних і підтримує логічні зв'язки між об'єктами системи.

Окрім опису логічної структури, у розділі було представлено реалізацію ключових функціональних модулів системи. Зокрема, описано сервіс роботи з файлами FileService, що відповідає за взаємодію із хмарним сховищем MinIO. Цей сервіс реалізує завантаження, отримання посилань, видалення зображень і є важливою складовою для забезпечення візуальної привабливості платформи.

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						42
Змн.	Арк.	№ докум.	Підпис	Дата		

Іншим важливим компонентом є контролер AuthController, що реалізує процеси реєстрації, аутентифікації, генерації JWT-токенів та керування сесіями користувачів, забезпечуючи безпечний доступ до системи.

Особливу увагу було приділено розробці контролера BookingController, який забезпечує управління бронюваннями, включаючи їх створення, перегляд, видалення та обробку з урахуванням ролей користувачів.

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						43
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ВЕБДОДАТКОМ ДЛЯ ОНЛАЙН-БРОНЮВАННЯ З ФУНКЦІЯМИ ПОРІВНЯННЯ

3.1. Порядок встановлення та налаштування системи

Встановлення та налаштування вебдодатку для онлайн-бронювання з функціями порівняння реалізується у відповідності до принципів контейнеризованого розгортання. Було застосовано Docker з метою досягнення ізолюваності сервісів, зручності та простоти налаштувань. Docker дозволяє запускати усі необхідні компоненти системи у вигляді окремих, взаємопов'язаних контейнерів.

Процес встановлення охоплює декілька покрокових дій:

1. Клонування проєкту та ініціалізація середовища – отримання вихідного коду системи, що містить директорії з бекендом та фронтом, конфігураційні файли, серед яких основну роль займає docker-compose.

2. Налаштування змінних середовища – створення файлу .env, де задаються такі параметри, як підключення до бази даних.

3. Побудова та запуск контейнерів – запускаються усі необхідні сервіси, виконуються міграції, формується структура бази даних.

4. Ініціалізація фронту.

Файл docker-compose.yml (додаток А) є комплексним сценарієм розгортання системи бронювання, який повністю автоматизує запуск усіх її компонентів, забезпечуючи узгоджену безперебійну роботу сервісів у єдиному контейнерному середовищі. Це значно спрощує та покращує повторюваність розгортання.

Згідно з конфігурацією, фронтенд-застосунок, розроблений на основі бібліотеки React, збирається з відповідного каталогу за допомогою окремого Dockerfile, розташованого в директорії docker/frontend.

Бекендова частина, представлена фреймворком Laravel, також збирається з окремого контексту та Dockerfile, розташованого у папці docker/api. Laravel-

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						44
Змн.	Арк.	№ докум.	Підпис	Дата		

застосунок отримує доступ до вихідного коду через монтування директорії `./ari` у відповідну точку контейнера.

Сервер `nginx` підключається до конфігураційних файлів, що розміщуються в окремій директорії `./configs/nginx`, і працює з образом `nginx:1.17`. Для забезпечення зовнішнього доступу він слухає порт, визначений змінною середовища `${NGINX_PORT}`.

Для роботи з базою даних використовується образ `MySQL` версії 5.7. Для зручного адміністрування бази даних передбачено запуск `phpMyAdmin`. Сервіс залежить від `mysql` і автоматично підключається до нього через внутрішню мережу, використовуючи змінні `PMA_HOST` і `PMA_PORT`.

Підсистема роботи з електронною поштою реалізована на основі контейнера `mailpit`. Збереження медіафайлів забезпечується через `MinIO`. Усі сервіси об'єднані в єдину логічну мережу з драйвером типу `bridge`, що дозволяє їм взаємодіяти між собою за внутрішніми `DNS`-іменами, не відкриваючи надлишкові порти назовні.

3.2. Структура інтерфейсу вебдодатку для онлайн-бронювання та порядок роботи

Інтерфейс користувача вебдодатку для онлайн-бронювання відіграє важливу роль у досягненні ключової мети – зручної та ефективної взаємодії між кінцевими користувачами: мандрівники, власники об'єктів розміщення та адміністратори системи. Саме візуальна частина вебдодатку формує перше враження та розуміння системи, а також визначає швидкість, з якою користувачі здатні знайти житло, забронювати номер, додати готель чи керувати бронюваннями.

Особливу увагу у процесі проєктування інтерфейсу приділено логіці навігації, доступності ключових елементів та відповідності принципам сучасного `UX/UI`-дизайну. У цьому підпункті проаналізовано основні компоненти інтерфейсу, зокрема головну сторінку, сторінки пошуку, порівняння

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						45
Змн.	Арк.	№ докум.	Підпис	Дата		

номерів, бронювання, управління об’єктами розміщення, перегляду історії операцій та залишення відгуків. Детальний опис інтерфейсних рішень дозволить з’ясувати, яким чином побудовано користувацьку взаємодію в системі.

На головній сторінці (рис. 3.1) користувач отримує для доступу головне меню з основними розділами вебсайту, необхідні посилання для подальшої роботи з вебдодатком (авторизація, бронювання чи перегляд обраного готелю).

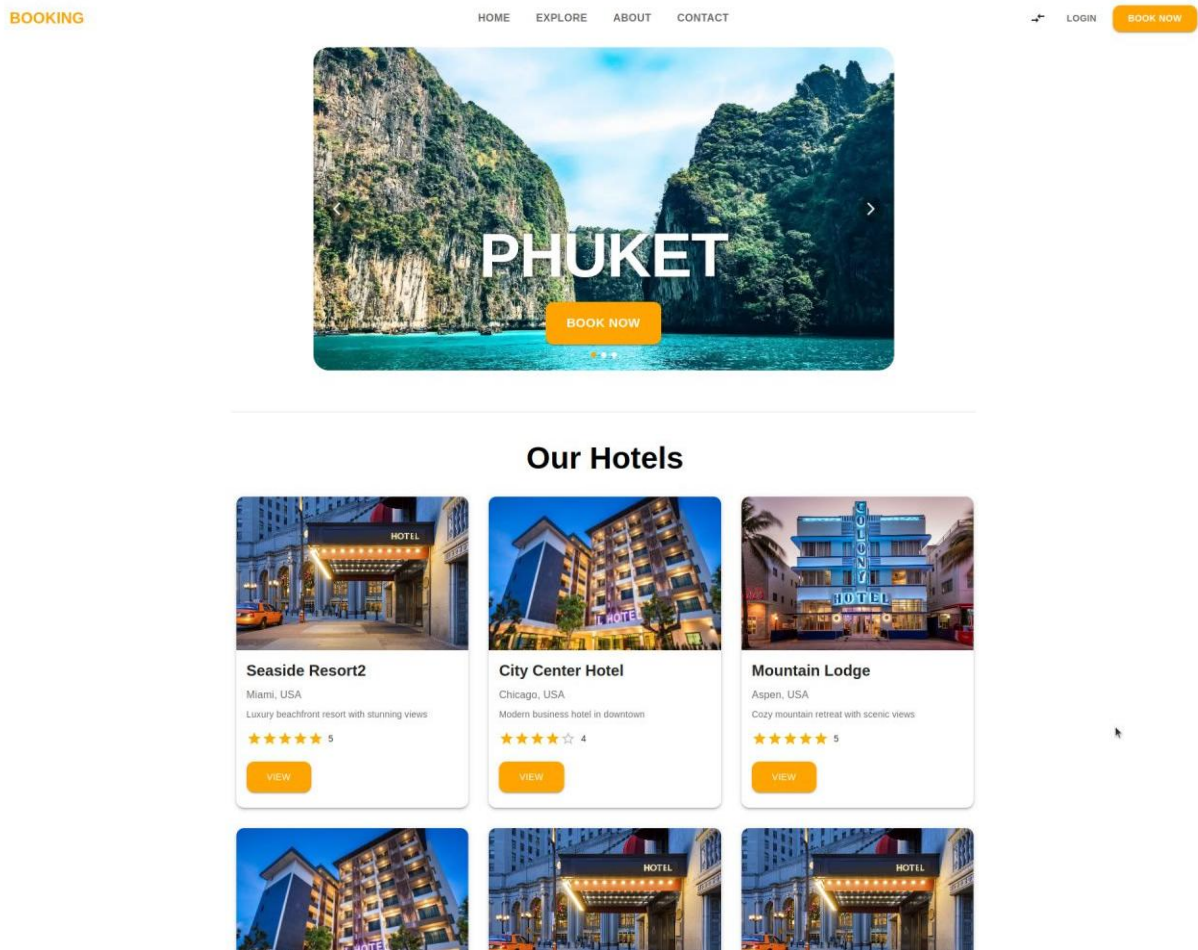


Рисунок 3.1 – Головна сторінка

З головної сторінки можна перейти до сторінки пошуку готелю (рис. 3.2). Ця сторінка дозволяє відфільтрувати готелі за такими параметрами, як назва готелю, місто, країна, дати заселення та виселення, тип готелю, ціна, рейтинг та інше. Зліва є відповідна форма з полями для введення відповідних даних. А також, на цій сторінці користувач бачить мінімальну інформацію про готелі: назву, місце розташування, головне фото, рейтинг та короткий опис.

Explore Accommodations

Filters

Search

Location

City

Country

Check-in/Check-out

Check-in mm/dd/yyyy

Check-out mm/dd/yyyy

Hotel Type

Type

Price Range

Min Price

Max Price

Rating Range

Min Rating

Max Rating

Sleeping Places



Seaside Resort2
Miami, USA
★ 5
Luxury beachfront resort with stunning views

VIEW DETAILS



City Center Hotel
Chicago, USA
★ 4
Modern business hotel in downtown

VIEW DETAILS



Mountain Lodge
Aspen, USA
★ 5
Cozy mountain retreat with scenic views

VIEW DETAILS



Palm Beach Resort
Honolulu, USA
★ 4



Metropolitan Hotel
Boston, USA
★ 4



Heritage Inn
Charleston, USA
★ 5

Рисунок 3.2 – Сторінка пошуку готелю

Кожен користувач має змогу підібрати собі певний готель, враховуючи найважливіші аспекти саме для нього. Приклад результату пошуку готелів за вказаними параметрами зображено на рисунку 3.3.

BOOKING

HOME EXPLORE ABOUT CONTACT

Explore Accommodations

Filters

Search

Seaside

Location

City

Country

Check-in/Check-out

Check-in mm/dd/yyyy

Check-out mm/dd/yyyy

Hotel Type

Type



Seaside Resort2
Miami, USA
★ 5
Luxury beachfront resort with stunning views

VIEW DETAILS

Рисунок 3.3 – Сторінка пошуку готелю з активними фільтрами

Після відбору списку готелів, які відповідають необхідним критеріям, користувач має змогу переглянути детальну інформацію про той чи інший готель. Для цього після короткого опису у кожного помешкання є кнопка «View details».

Перехід на сторінку перегляду готелю (рис. 3.4) відкриває доступ користувачу до більш розширеної інформації: тут представлено повний опис готелю, зазначено його переваги, рейтинг, фотогалерею, відгуки інших користувачів, а також контактні дані.

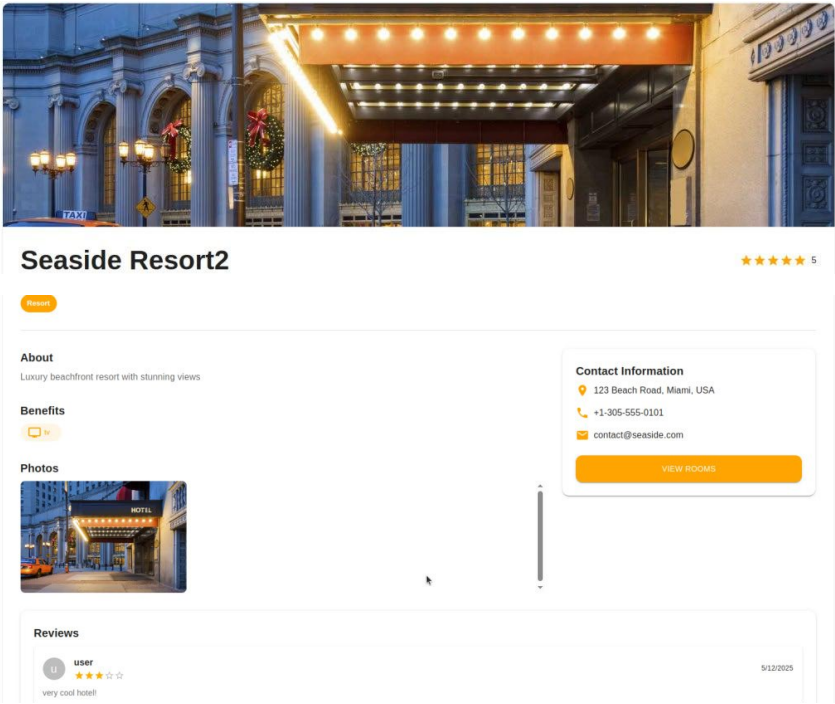


Рисунок 3.4 – Сторінка перегляду інформації про готель

Після перегляду розширеної інформації про готель користувач має можливість перейти до ознайомлення з доступними для бронювання кімнатами. Дана сторінка (рис. 3.5) побудована за аналогічним принципом, що й список готелів: представлений перелік номерів, кожен з яких містить стислу, але ключову інформацію – фотографію, вартість за ніч, максимальну кількість осіб для проживання та площу кімнати.

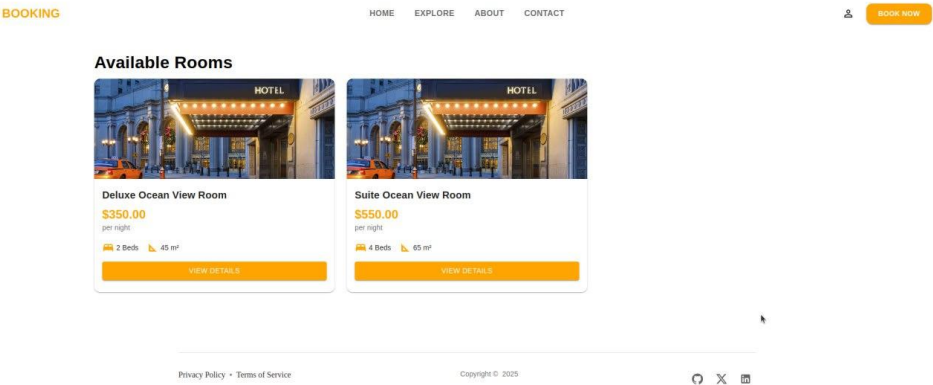


Рисунок 3.5 – Сторінка перегляду доступних номерів обраного готелю

Після вибору відповідної кімнати користувач має змогу натиснути кнопку «View details», яка здійснює перехід на сторінку з розгорнутою інформацією про обраний номер (рис. 3.6). На цій сторінці подано детальний опис номеру,

включаючи характеристики, такі як розмір, кількість спальних місць, наявність ванної кімнати, час заселення та виселення, рік останнього ремонту, перелік доступних зручностей тощо. Також відображаються фотографії інтер'єру та інформація про ціну за ніч.

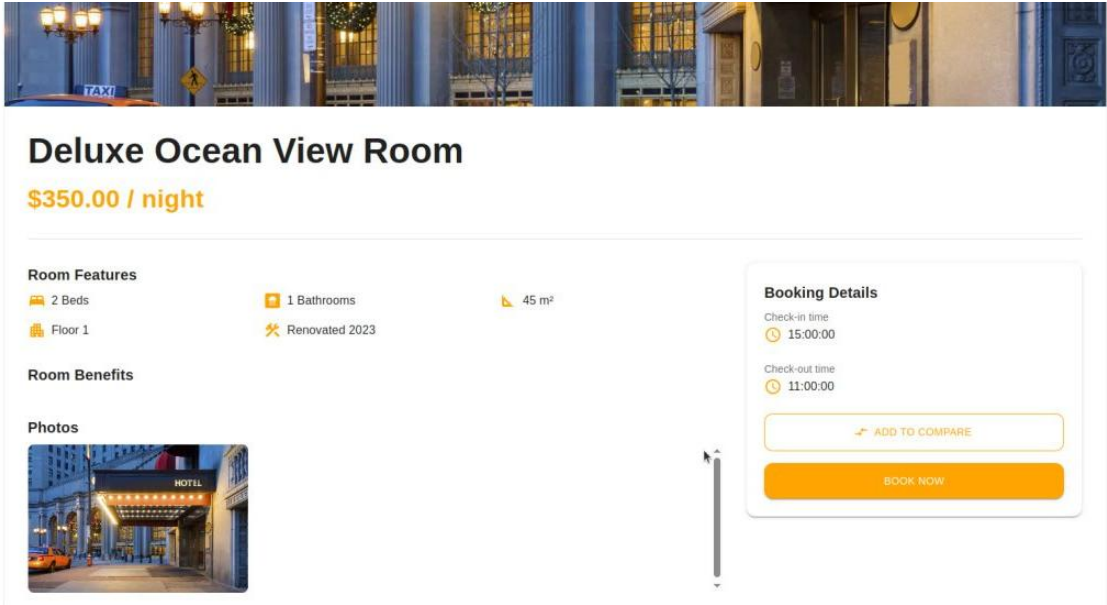


Рисунок 3.6 – Сторінка перегляду інформації про номер

Користувач має змогу не лише ознайомитися з детальною інформацією про номер, а й скористатися інтерактивною функцією порівняння кімнат. За допомогою спеціальної кнопки номер можна додати до списку для порівняння, а надалі – сформувати перелік кількох варіантів, які його зацікавили. Це особливо зручно у випадках, коли користувач прагне здійснити виважений вибір, порівнюючи кілька подібних за ціною, але різних за зручностями або характеристиками пропозицій.

Після додавання двох номерів, система надає доступ до спеціальної сторінки порівняння (рис. 3.7), де всі обрані кімнати представлені в табличному вигляді з розбиттям параметрів на категорії – площа, кількість спальних місць, тип ванної кімнати, набір зручностей, час заселення тощо. Для полегшення аналізу, система візуально підсвічує кращі значення серед представлених варіантів: наприклад, більша площа або нижча ціна виділяються зеленим кольором.

Room Comparison

+ ADD MORE ROOMS

RESORT (2)	
Features	Seaside Resort2 Miami, USA
Hotel Information	
Hotel Type	Resort
Hotel Rating	5 Stars
Location	123 Beach Road
City	Miami
Country	USA
Room Details	
Room Type	Deluxe Ocean View
Price per Night	\$350.00
Room Size	45 m²
Sleeping Places	2
Bathrooms	1
Floor Number	1
Last Renovated	2023
Booking Information	
Check-in Time	15:00:00
Check-out Time	11:00:00
Availability	Available

Рисунок 3.7 – Сторінка порівняння номерів

Такий підхід дозволяє швидко оцінити, яка з кімнат є вигіднішою за сукупністю параметрів, не вдаючись до детального повторного перегляду кожної окремої позиції.

Якщо ж користувач буде готовий забронювати обраний варіант, то спершу йому потрібно бути авторизованим у системі. Для входу у систему на сторінці авторизації (рис. 3.8) потрібно ввести електронну пошту та пароль, вказані при реєстрації.

Welcome Back

Please sign in to continue

Email Address *

Password *

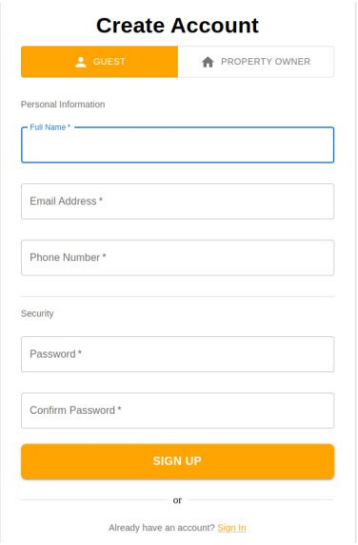
SIGN IN

or

Don't have an account? [Sign Up](#)

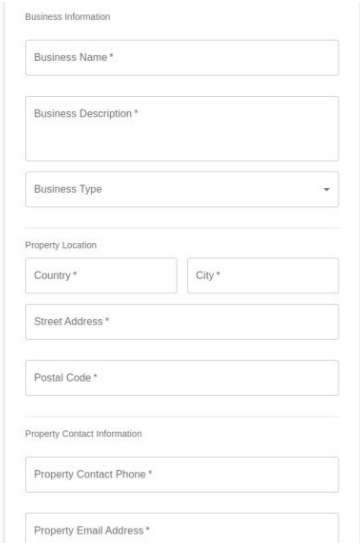
Рисунок 3.8 – Сторінка авторизації в систему

Якщо користувач ще не має облікового запису в системі, він може створити його, перейшовши на сторінку реєстрації. Залежно від обраної ролі, система надає можливість зареєструватися як гість (рис. 3.9) або як власник помешкання (рис. 3.10). Для кожного типу реєстрації передбачені окремі форми з відповідними полями: гостю необхідно ввести основні персональні дані, такі як повне ім'я, номер телефону, електронну адресу та пароль, тоді як для власника передбачено розширену форму, яка додатково включає інформацію про готель або інше помешкання, що буде розміщене на платформі.



The 'Create Account' form for a guest features a title 'Create Account' and two tabs: 'GUEST' (selected) and 'PROPERTY OWNER'. Under 'Personal Information', there are input fields for 'Full Name *', 'Email Address *', and 'Phone Number *'. Under 'Security', there are fields for 'Password *' and 'Confirm Password *'. A large orange 'SIGN UP' button is at the bottom, with a link 'Already have an account? Sign In' below it.

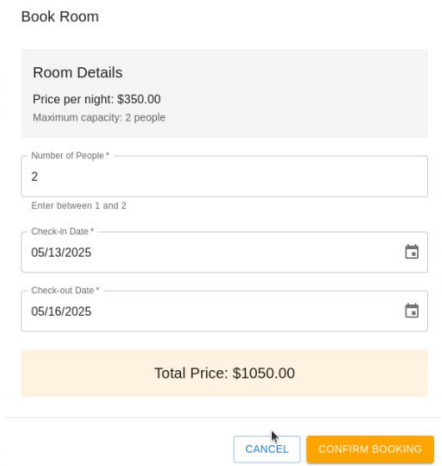
Рисунок 3.9 – Форма реєстрації для гостя



The 'Create Account' form for a property owner has a title 'Create Account' and two tabs: 'GUEST' and 'PROPERTY OWNER' (selected). It is divided into three sections: 'Business Information' with fields for 'Business Name *', 'Business Description *', and a 'Business Type' dropdown; 'Property Location' with fields for 'Country *', 'City *', 'Street Address *', and 'Postal Code *'; and 'Property Contact Information' with fields for 'Property Contact Phone *' and 'Property Email Address *'.

Рисунок 3.10 – Форма реєстрації для власника готелю

Після авторизації в системі користувач може перейти до бронювання обраного номеру (рис. 3.11), де потрібно вказати кількість осіб, дати заселення та виселення.



The 'Book Room' form displays 'Room Details' with 'Price per night: \$350.00' and 'Maximum capacity: 2 people'. It includes a 'Number of People *' field with the value '2' and a note 'Enter between 1 and 2'. There are date pickers for 'Check-in Date *' (05/13/2025) and 'Check-out Date *' (05/16/2025). A summary box shows 'Total Price: \$1050.00'. At the bottom are 'CANCEL' and 'CONFIRM BOOKING' buttons.

Рисунок 3.11 – Форма бронювання номеру

Користувачі, що зареєстровані як власники об'єктів розміщення, отримують розширений доступ до функціональних можливостей системи, зокрема можуть керувати своїми готелями та номерами. Після входу в обліковий запис, власник потрапляє на сторінку, де відображається перелік його помешкань (рис. 3.12).

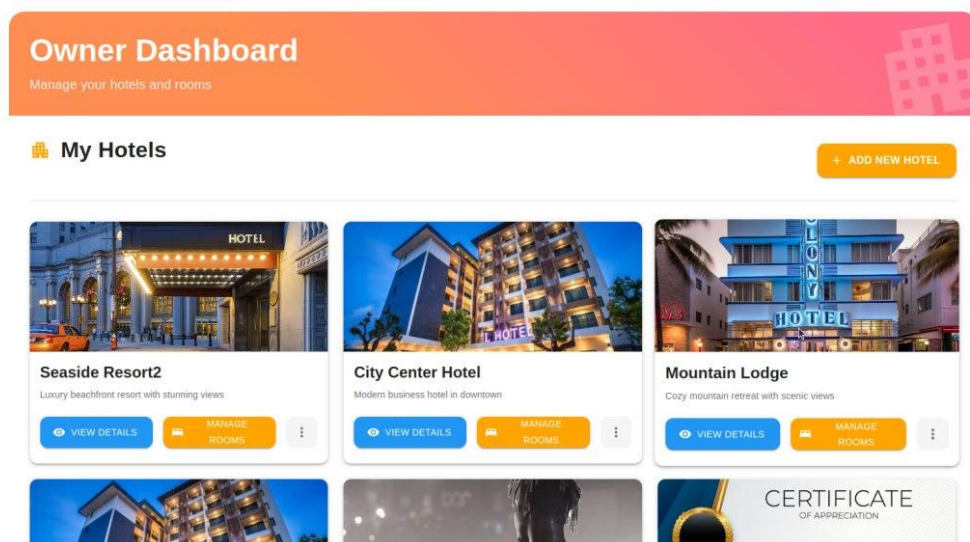


Рисунок 3.12 – Панель керування об'єктами розміщення

З цієї сторінки власник має змогу здійснювати повне адміністрування інформації про власні об'єкти розміщення. Зокрема, він може оновлювати загальні відомості про готелі, додавати актуальні фотографії, редагувати опис, вказувати точну адресу, характеристики та перелік доступних зручностей – усе, що може вплинути на рішення потенційного гостя.

Окрім цього, система дозволяє перейти до панелі керування номерним фондом (рис. 3.13). На цій сторінці власник має можливість створювати нові кімнати, задавати їхні основні параметри (кількість спальних місць, площу, ціну за добу), вказувати час заїзду та виїзду, а також додавати або змінювати перелік переваг і зручностей.

У разі необхідності власник може також швидко видалити застарілу або неактуальну інформацію, підтримуючи достовірність даних на платформі. Такий рівень функціональності сприяє ефективному управлінню об'єктами проживання та підвищує якість взаємодії з користувачами.

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		52

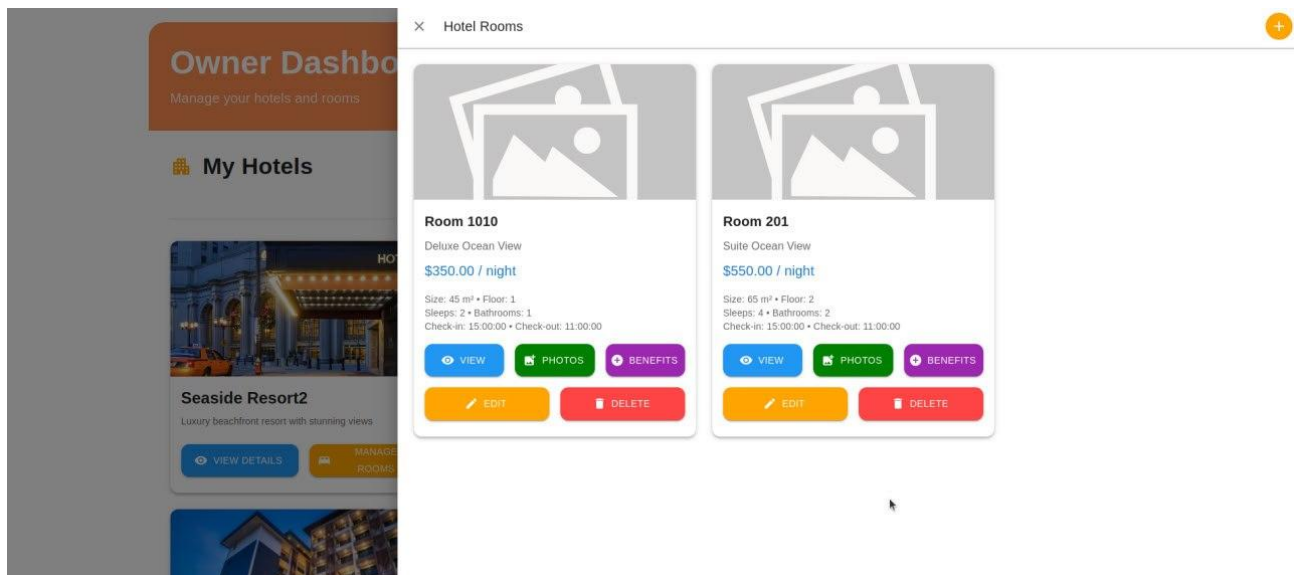


Рисунок 3.13 – Панель керування номерами готелю

Після активації відповідних кнопок користувачеві відкривається модальне вікно, яке дозволяє швидко та зручно внести зміни або доповнити інформацію про обраний номер. Завдяки такому підходу інтерфейс сторінки перегляду номерів стає більш динамічним і насиченим змістом, що, у свою чергу, підвищує її інформативність для потенційних гостей. Модальні вікна створюють зручне середовище для редагування – без потреби переходити на окремі сторінки, що економить час і робить процес управління контентом інтуїтивно зрозумілим.

Рисунок 3.14 – Форма для редагування інформації про номер

Рисунок 3.15 – Форма для додавання фотографій

Для користувачів із роллю «Адміністратор» у системі передбачено розширені функціональні можливості, які охоплюють управління ключовими складовими вебдодатку. Зокрема, адміністратор отримує доступ до спеціальних розділів, призначених для адміністрування системних довідників (рис. 3.16).

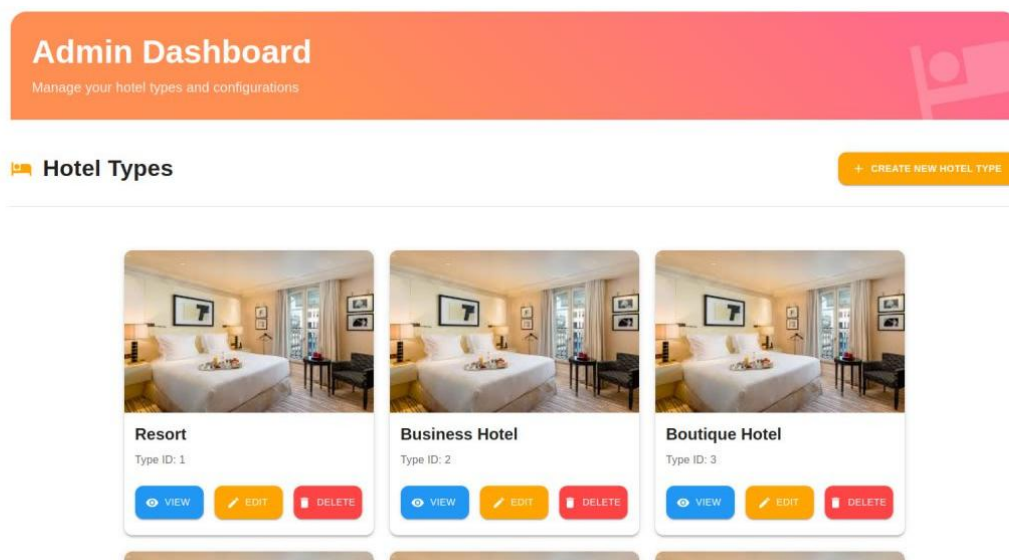


Рисунок 3.16 – Панель керування типами готелів

Через інтерфейс керування він може створювати, редагувати або видаляти записи, що стосуються типів готелів, категорій зручностей, а також переваг, доступних як для готелів, так і для окремих кімнат (рис. 3.17).

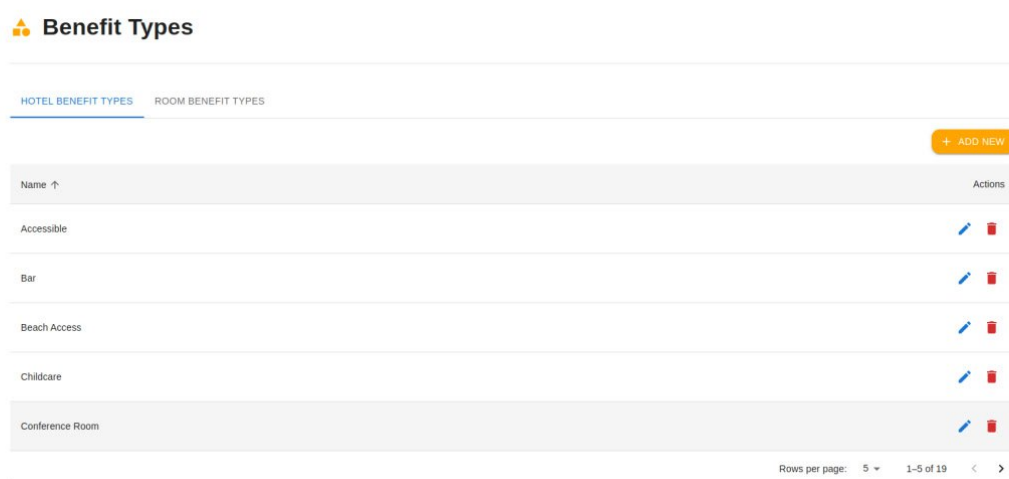


Рисунок 3.17. – Інтерфейс керування категоріями переваг для готелів та кімнат

Ці дії дозволяють підтримувати актуальність і логічну структуру даних у системі, сприяючи точному відображенню доступних опцій для власників помешкань і користувачів.

3.3. Тестування роботи вебдодатку

Процес тестування відіграє ключову роль у розробці програмних продуктів, адже саме на цьому етапі здійснюється перевірка відповідності

реалізованої системи встановленим функціональним і нефункціональним вимогам. У рамках розробки веб-додатку для онлайн-бронювання з можливістю порівняння варіантів проживання тестування дає змогу виявити технічні помилки у роботі основних модулів, проаналізувати надійність взаємодії між окремими складовими системи, а також оцінити логічність і зручність інтерфейсу з позиції кінцевого користувача.

Для перевірки працездатності та стабільності розробленого вебдодатку було проведено мануальне тестування ключових сценаріїв з акцентом на обробку виняткових ситуацій [20]. Зокрема, тестування охоплювало перевірку коректності відображення повідомлень про помилки у випадках порушення умов доступу, неправильного введення даних або звернення до неіснуючих ресурсів.

Під час цього етапу було навмисно змодельовано помилки, які можуть траплятися під час користування вебдодатком.

1. Спроби доступу до неіснуючого маршруту або ресурсу, наприклад, при переході за помилковим URL або зверненні до вже видаленої сторінки готелю чи кімнати. У процесі тестування здійснювалися навмисні спроби звернення до неіснуючих маршрутів, після чого система коректно відображала сторінку з кодом помилки 404 (рис. 3.18). Дизайн відображення помилки передбачає зручне повідомлення про те, що запитаний ресурс не знайдено, а також надає користувачеві можливість швидко повернутися до головних розділів сайту.

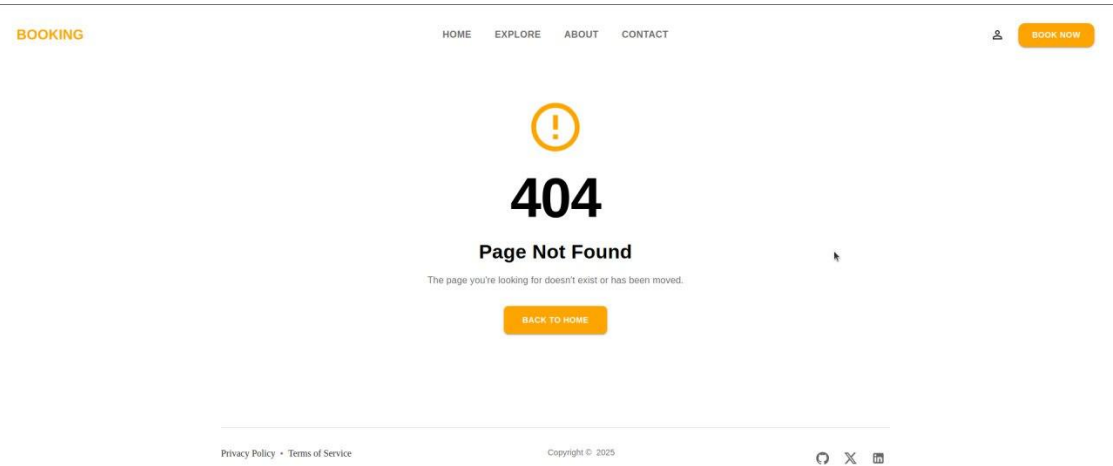


Рисунок 3.18 – Сторінка помилки 404

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						55
Змн.	Арк.	№ докум.	Підпис	Дата		

2. Автентифікований користувач намагається отримати доступ до ресурсу або виконати дію, яка йому не дозволена відповідно до наданих прав. Наприклад, якщо користувач намагається видалити бронювання, яке йому не належить, або власник намагається переглянути об'єкт, що не прив'язаний до його профілю. Під час тестування було перевірено, що система перехоплює такі запити та перенаправляє користувача на окрему сторінку з повідомленням про відмову в доступі (рис. 3.19), яке супроводжується рекомендацією повернутися на домашню сторінку.

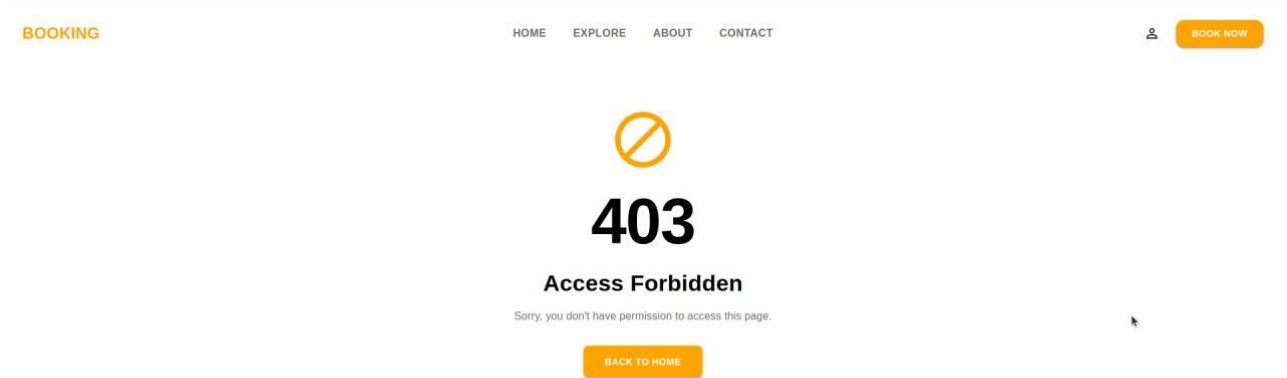


Рисунок 3.19 – Сторінка помилки 403

3. Критичні помилки на рівні серверної логіки. Було змодельовано ситуації, що можуть викликати внутрішню помилку сервера. У результаті користувачу відображалась сторінка з повідомленням про внутрішню помилку (рис. 3.20), що дозволяє уникнути відображення сирих технічних повідомлень та забезпечити базовий рівень зручності навіть у критичних випадках.

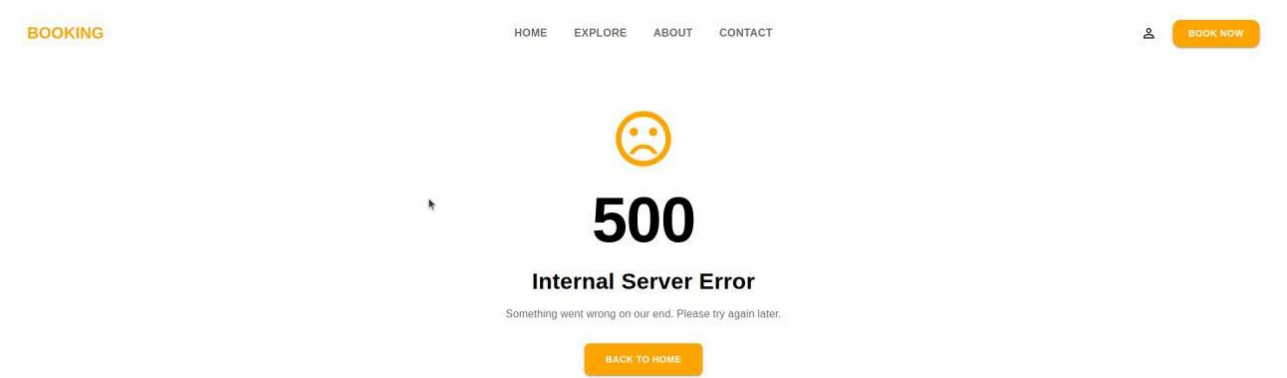


Рисунок 3.20 – Сторінка помилки 500

У межах тестування вебдодатку для онлайн-бронювання з функціями порівняння було також проведено перевірку надійності та коректності реалізації механізмів валідації вхідних даних під час процесу реєстрації нового користувача. Основна увага була приділена перевірці того, як система реагує на неправильне або неповне заповнення обов'язкових полів, а також наскільки інформативними є повідомлення про помилки (рис. 3.21).

The screenshot shows a 'Create Account' form with two tabs: 'GUEST' (selected) and 'PROPERTY OWNER'. The form is divided into three sections: 'Personal Information', 'Security', and a final 'SIGN UP' button. In the 'Personal Information' section, the 'Full Name' field contains 'User'. The 'Email Address' field contains '123127' and shows a red border with the error message 'Invalid email format'. The 'Phone Number' field contains 'kdhasdk' and shows a red border with the error message 'Invalid phone number format'. In the 'Security' section, the 'Password' field contains '****' and shows a red border with the error message 'Password must be at least 6 characters'. The 'Confirm Password' field contains '*****'. At the bottom of the form, there is a red error message: 'Registration failed. Please try again.'

Рисунок 3.21 – Перевірка валідації вхідних даних

Зокрема, було змодельовано кілька ситуацій із навмисним порушенням вимог до форматів:

1. Неправильний формат електронної пошти. Під час введення адреси без символу “@” або з некоректним доменом система автоматично блокувала відправку форми та виводила чітке повідомлення про неправильний формат email-адреси.
2. Невалідний номер телефону. Було перевірено обробку введення неповного номера, алфавітних символів або номерів із некоректною кількістю цифр. Система у всіх випадках спрацьовувала правильно, виводячи під відповідним полем повідомлення про вимогу ввести дійсний номер телефону.
3. Недостатня довжина пароля. Для перевірки безпеки користувача було введено пароль, що не відповідав встановленому мінімуму символів.

Результатом стало інформативне повідомлення про необхідність створити більш складний пароль згідно з вимогами.

Загалом, результати тестування засвідчили, що валідація форми реєстрації реалізована відповідно до стандартів сучасної веброзробки: вона є клієнт-серверною, ефективно перехоплює некоректні значення ще до їх обробки сервером, а також формує зрозумілі та локалізовані повідомлення для користувача.

Висновки до третього розділу

У третьому розділі було детально розглянуто ключові етапи впровадження, структуру інтерфейсу та результати тестування вебдодатку для онлайн-бронювання з функціями порівняння. Систематизовано порядок розгортання вебдодатку у середовищі контейнеризації Docker, що дало змогу досягти ізольованості компонентів, стандартизації процесу інсталяції та забезпечення повторюваності розгортання незалежно від середовища. Завдяки використанню docker-compose реалізовано централізовану конфігурацію і запуск усіх необхідних сервісів – бекенду на Laravel, фронтенду на React, MySQL-сервера, файлового сховища MinIO, phpMyAdmin, SMTP-сервісу та веб-сервера Nginx.

Аналіз структури інтерфейсу вебдодатку засвідчив високий рівень організації навігаційної логіки та відповідність сучасним принципам UX/UI-дизайну. Описані інтерфейсні рішення забезпечують логічний та інтуїтивно зрозумілий шлях користувача – від пошуку готелю до оформлення бронювання, включаючи можливість реєстрації, перегляду номерів, додавання варіантів до порівняння та оцінювання інформації на основі зручних візуальних підказок. Реалізація функціоналу порівняння номерів надає користувачеві інструмент для прийняття зважених рішень, що вигідно вирізняє платформу серед існуючих аналогів. Рольова модель доступу, зокрема поділ між гостями, власниками об'єктів і адміністраторами, дозволяє ефективно керувати різними аспектами платформи з урахуванням специфіки взаємодії кожного типу користувача.

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						58
Змн.	Арк.	№ докум.	Підпис	Дата		

Особливу увагу в межах розділу було приділено перевірці працездатності розробленого додатку через процедури ручного тестування. Було змодельовано типові помилки та виняткові ситуації, серед яких: порушення прав доступу, звернення до неіснуючих маршрутів та внутрішні помилки сервера. Реалізовані сторінки обробки помилок з кодами 403, 404 та 500 забезпечують не лише технічну коректність реакції системи, а й належний рівень користувацького досвіду, навіть у ситуаціях виняткової поведінки. Окремо перевірено функціонування механізмів валідації вхідних даних у формах реєстрації, що довело здатність системи коректно виявляти помилки на етапі введення та формувати релевантні повідомлення.

Технічна реалізація, інтерфейсна складова та результат тестування підтверджують відповідність розробленого рішення сучасним вимогам до вебсервісів у сфері туризму та онлайн-бронювання.

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						59
Змн.	Арк.	№ докум.	Підпис	Дата		

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи було комплексно вирішено завдання розробки вебдодатку для онлайн-бронювання з функціями порівняння, що відповідає сучасним потребам туристичного ринку. Проведений аналіз тенденцій цифровізації туристичної галузі дозволив чітко сформулювати актуальні вимоги до таких систем, визначивши ключові фактори, які забезпечують зручність взаємодії користувачів і сприяють розвитку малого готельного бізнесу. Особлива увага була приділена вирішенню проблем, притаманних популярним платформам, таким як недостатні інструменти порівняння, складна комунікація та фінансові бар'єри для малих об'єктів розміщення.

Здійснений аналіз аналогів підтвердив необхідність створення додатку, який поєднує глибоку персоналізацію пошуку з розширеним функціоналом порівняння варіантів проживання. Використання обґрунтованого архітектурного підходу з чітким розподілом на фронтенд та бекенд, який взаємодіє через RESTful API, дозволило побудувати гнучку та масштабовану систему, готову до подальших інтеграцій.

Процес проєктування забезпечив створення чіткої логічної структури вебдодатку, що включає розроблену модель бази даних із детальною специфікацією таблиць та взаємозв'язків між ними, а також ідентифікацію основних сценаріїв використання. Такий підхід заклав фундамент для реалізації ефективного контролю доступу, стабільності роботи та безпеки зберігання даних.

Розроблене рішення побудоване на сучасних технологіях (Docker, Laravel, React, MySQL, MinIO), що забезпечило не лише функціональну завершеність проєкту, але й його надійність, зручність супроводу та адаптивність до змін середовища. Запровадження контейнеризації за допомогою Docker дозволило спростити процедуру розгортання і забезпечити прогнозовану роботу системи в будь-яких умовах експлуатації.

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						60
Змн.	Арк.	№ докум.	Підпис	Дата		

Детально опрацьований інтерфейс веб-додатку відповідає сучасним стандартам UX/UI-дизайну, забезпечуючи інтуїтивно зрозумілий шлях користувача – від пошуку та порівняння номерів до бронювання й адміністрування власних об’єктів розміщення. Реалізація механізмів порівняння дозволяє користувачам здійснювати усвідомлений та ефективний вибір житла, що створює конкурентну перевагу розробленого продукту на ринку.

Проведене ручне тестування підтвердило стабільність роботи розробленої платформи у типових сценаріях, включаючи реагування на помилки різних категорій. Валідація полів реєстрації продемонструвала високий рівень захисту системи від некоректних даних, що додатково підвищує її надійність та зручність у практичному застосуванні.

Таким чином, отримані результати свідчать про успішну реалізацію поставленої мети кваліфікаційної роботи та відповідність створеного продукту поставленим вимогам.

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						61
Змн.	Арк.	№ докум.	Підпис	Дата		

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Airbnb [Електронний ресурс] – Режим доступу до ресурсу: <https://www.airbnb.com/>
2. Beaulieu A. Learning SQL: Generate, Manipulate, and Retrieve Data / O'Reilly Media, 2020. – 377 p.
3. Booking.com [Електронний ресурс] – Режим доступу до ресурсу: <https://www.booking.com/>
4. Expedia.com [Електронний ресурс] – Режим доступу до ресурсу: <https://www.expedia.com/>
5. Forta B. SQL in 10 Minutes, Sams Teach Yourself / Sams Publishing, 2012. – 288 p.
6. Laravel Tutorials [Електронний ресурс] – Режим доступу до ресурсу: <https://laravel-news.com/category/tutorials>
7. MySQL. Документація [Електронний ресурс] – Режим доступу до ресурсу: <https://dev.mysql.com/doc/>
8. Nickoloff J. Docker in Action / Manning, 2019. – 350 p.
9. Reese G. The REST API Design Handbook / George Reese, Christian Reilly // Amazon Digital Services, 2012. – 81 p.
10. RESTful API [Електронний ресурс] – Режим доступу до ресурсу: <https://www.techtarget.com/searchapparchitecture/definition/RESTful-API>
11. Stauffer M. Laravel: Up & Running: A Framework for Building Modern PHP Apps 3rd Edition / O'Reilly Media, 2023. – 534 p.
12. Turnbull J. The Docker Book: Containerization Is the New Virtualization / United States: James Turnbull, 2014. – 344 p.
13. Use Case Diagram - Unified Modeling Language (UML) – Режим доступу до ресурсу: <https://www.geeksforgeeks.org/use-case-diagram/>
14. UML models and diagrams [Електронний ресурс] – Режим доступу до ресурсу: <https://www.ibm.com/docs/en/rational-soft-arch/9.7.0?topic=diagrams-uml-models>

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						62
Змн.	Арк.	№ докум.	Підпис	Дата		

15. What is a RESTful API? [Електронний ресурс] – Режим доступу до ресурсу: <https://aws.amazon.com/what-is/restful-api/>

16. What's the Difference Between Monolithic and Microservices Architecture? [Електронний ресурс] – Режим доступу до ресурсу: <https://aws.amazon.com/compare/the-difference-between-monolithic-and-microservices-architecture/>

17. Документація MinIO [Електронний ресурс] – Режим доступу до ресурсу: <https://min.io/docs/minio/container/index.html>

18. Документація React [Електронний ресурс] – Режим доступу до ресурсу: <https://react.dev/learn>

19. Офіційна документація Docker [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.docker.com/>

20. Тестування веб-проектів: основні етапи та поради [Електронний ресурс] – Режим доступу до ресурсу: <https://qalight.ua/baza-znaniy/testuvannya-veb-proektiv-osnovni-etapi-ta-poradi/>

21. Що таке нефункціональні вимоги: приклади, визначення, повне керівництво [Електронний ресурс] – Режим доступу до ресурсу: <https://visuresolutions.com/uk/blog/non-functional-requirements/>

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						63
Змн.	Арк.	№ докум.	Підпис	Дата		

ДОДАТКИ

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						64
Змн.	Арк.	№ докум.	Підпис	Дата		

Файл docker-compose.yaml

```

services:
  frontend:
    build:
      context: ./frontend
      dockerfile: ../docker/frontend/Dockerfile
    command: ["sh", "-c", "npm install && npm run dev -- --host 0.0.0.0"]
    volumes:
      - ./frontend:/app
    networks:
      - network

  api:
    build:
      context: ./api
      dockerfile: ../docker/api/Dockerfile
    volumes:
      - ./api:/var/www/website/api
    networks:
      - network

  nginx:
    image: nginx:1.17
    volumes:
      - ./configs/nginx:/etc/nginx/conf.d
    ports:
      - ${NGINX_PORT}:80
    networks:
      - network

  mysql:
    image: mysql:5.7
    environment:
      MYSQL_ROOT_PASSWORD: ${MYSQL_ROOT_PASSWORD}
      MYSQL_DATABASE: ${MYSQL_DATABASE}
      MYSQL_USER: ${MYSQL_USER}
      MYSQL_PASSWORD: ${MYSQL_PASSWORD}
    ports:
      - "3306:3306"
    volumes:
      - mysql_data:/var/lib/mysql
    networks:
      - network

  phpmyadmin:
    image: phpmyadmin/phpmyadmin
    restart: unless-stopped
    ports:
      - 8080:80
    depends_on:
      - mysql
    environment:

```

					ІПЗ.КР.Б – 121 – 25 – ІЗ	Арк.
						65
Змн.	Арк.	№ докум.	Підпис	Дата		

```
PMA_HOST: mysql
PMA_PORT: 3306
networks:
- network
```

```
mailpit:
  image: axllent/mailpit
  ports:
    - "1025:1025" # SMTP
    - "8025:8025" # Web UI
  networks:
    - network
```

```
minio:
  image: quay.io/minio/minio
  container_name: minio
  restart: always
  ports:
    - "9000:9000" # API
    - "9090:9090" # Web UI
  environment:
    MINIO_ROOT_USER: admin
    MINIO_ROOT_PASSWORD: admin1234
  volumes:
    - minio_data:/data
  networks:
    - network
  command: server /data --console-address ":9090"
```

```
networks:
  network:
    driver: bridge
```

```
volumes:
  mysql_data:
  minio_data:
```

					ІПЗ.КР.Б – 121 – 25 – ІЗ	Арк.
						66
Змн.	Арк.	№ докум.	Підпис	Дата		

BookingController.php

```

<?php

namespace App\Http\Controllers;

use App\Constants\Role;
use App\Events\BookingCreated;
use App\Http\Requests\StoreBookingRequest;
use App\Http\Resources\BookingResource;
use App\Models\Booking;
use App\Models\Room;
use DateMalformedStringException;
use DateTime;
use Illuminate\Auth\Access\AuthorizationException;
use Illuminate\Http\JsonResponse;
use Symfony\Component\HttpFoundation\Response;
use Vyuldashev\LaravelOpenApi\Attributes as OpenApi;

#[OpenApi\PathItem]
class BookingController extends Controller
{
    /**
     * Display a listing our bookings.
     */
    #[OpenApi\Operation(tags: ['Booking'], method: 'GET')]
    public function index(): JsonResponse
    {
        $user = auth()->user();
        if(!$user){
            return response()->json(['error' => 'Unauthorized'],
Response::HTTP_UNAUTHORIZED);
        }
        switch ($user->role) {
            case Role::ADMIN:
                $bookings = Booking::with('room.hotel')->get();
                break;
            case Role::USER:
                $bookings = Booking::where('user_id', $user->id)
                    ->with('room.hotel')
                    ->get();
                break;
            case Role::OWNER:
                $bookings = Booking::whereHas('room.hotel', function
($query) use ($user) {
                    $query->where('owner_user_id', $user->id);
                })
                    ->with('room.hotel')
                    ->get();
                break;
            default:
                return response()->json(['message' => 'Unauthorized'],
403);
        }
    }
}

```

					ІІЗ.КР.Б – 121 – 25 – ІІЗ	Арк.
						67
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    }

    return response()->json(BookingResource::collection($bookings),
Response::HTTP_OK);
}

/**
 * Create a new booking (USER ONLY)
 * @throws DateMalformedStringException
 */
#[OpenApi\Operation(tags: ['Booking'], method: 'POST')]
public function store(StoreBookingRequest $request)
{
    $data = $request->validated();
    $user_id = auth()->user()->id;
    $data['user_id'] = $user_id;

    $room = Room::find($data['room_id']);

    $checkIn = new DateTime($data['check_in']);
    $checkOut = new DateTime($data['check_out']);
    $interval = $checkIn->diff($checkOut);
    $daysCount = $interval->days;
    $data['total_price'] = $daysCount * $room->price_per_night;
    $data['booking_status'] = "Pending";

    $booking = Booking::create($data);

    event(new BookingCreated($booking));

    return response()->json($booking, Response::HTTP_CREATED);
}

public function create(StoreBookingRequest $request)
{
    return response()->json(null, Response::HTTP_METHOD_NOT_ALLOWED);
}

/**
 * Display a single booking.
 *
 * @param Booking $booking
 * @return JsonResponse
 */
#[OpenApi\Operation(tags: ['Booking'], method: 'GET')]
public function show(Booking $booking): JsonResponse
{
    try {
        $this->authorize('view', $booking);
        return response()->json($booking, Response::HTTP_OK);
    } catch (AuthorizationException $e) {
        return response()->json([
            'message' => 'Unauthorized to view this booking.',
        ], Response::HTTP_FORBIDDEN);
    }
}

```

					ІІЗ.КР.Б – 121 – 25 – ІІЗ	Арк.
						68
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    }

    /**
     * Delete Booking
     *
     * @param Booking $booking
     * @return JsonResponse
     */
    #[OpenApi\Operation(tags: ['Booking'], method: 'DELETE')]
    public function destroy(Booking $booking): JsonResponse
    {
        try {
            $this->authorize('delete', $booking);
            $booking->delete();
            return response()->json([], Response::HTTP_NO_CONTENT);
        } catch (AuthorizationException $e) {
            return response()->json([
                'message' => 'Unauthorized to delete this booking.',
            ], Response::HTTP_FORBIDDEN);
        }
    }
}

```

					ІПЗ.КР.Б – 121 – 25 – ІЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		69

AuthController.php

```

<?php

namespace App\Http\Controllers;

use App\Http\Requests\StoreHotelRequest;
use App\Http\Requests\StoreUserRequest;
use App\Models\Hotel;
use Illuminate\Http\JsonResponse;
use Illuminate\Http\Request;
use App\Models\User;
use Illuminate\Support\Facades\Hash;
use Illuminate\Support\Str;
use Illuminate\Validation\ValidationException;
use Tymon\JWTAuth\Facades\JWTAuth;
use Tymon\JWTAuth\Exceptions\JWTException;
use Vyuldashev\LaravelOpenApi\Attributes as OpenApi;

#[OpenApi\PathItem]
class AuthController extends Controller
{
    /**
     * Create new user.
     *
     * Creates new user or returns already existing user by email.
     */
    #[OpenApi\Operation(tags: ['Auth'], method: 'POST')]
    public function register(StoreUserRequest $request): JsonResponse
    {
        $validatedData = $request->validated();

        $user = User::create([
            'id' => Str::uuid()->toString(),
            'fullname' => $validatedData['fullname'],
            'email' => $validatedData['email'],
            'phone_number' => $validatedData['phone_number'],
            'password' => Hash::make($validatedData['password']),
            'role' => 'user',
            'is_active' => 1,
        ]);

        $token = JWTAuth::claims(['role' => $user->role])-
            >fromUser($user);

        return response()->json([
            'user' => $user,
            'token' => $token
        ], 201);
    }

    /**

```

					ІПЗ.КР.Б – 121 – 25 – ІЗ	Арк.
						70
Змн.	Арк.	№ докум.	Підпис	Дата		

```

* Create new owner user.
*
* Creates new owner user or returns already existing user by email.
*/
#[OpenApi\Operation(tags: ['Auth'], method: 'POST')]
public function registerOwner(Request $request): JsonResponse
{
    $userData = $request->get('user', []);
    $hotelData = $request->get('hotel', []);

    try {
        $this->validate(new Request($userData), (new
StoreUserRequest())->rules());
        $this->validate(new Request($hotelData), (new
StoreHotelRequest())->rules());
    } catch (ValidationException $e) {
        return response()->json([
            'errors' => $e->errors(),
        ], 422);
    }

    $user = User::create([
        'id' => Str::uuid()->toString(),
        'fullname' => $userData['fullname'],
        'email' => $userData['email'],
        'phone_number' => $userData['phone_number'],
        'password' => Hash::make($userData['password']),
        'role' => 'owner',
        'is_active' => 1,
    ]);
    $hotelData['owner_user_id'] = $user->id;

    Hotel::create($hotelData);

    $token = JWTAuth::claims(['role' => $user->role])-
>fromUser($user);
    return response()->json([
        'user' => $user,
        'token' => $token
    ], 201);
}

/**
* Login user.
*
* Login user.
*/
#[OpenApi\Operation(tags: ['Auth'], method: 'POST')]
public function login(Request $request)
{
    $credentials = $request->only('email', 'password');

    try {
        if (! $token = JWTAuth::attempt($credentials)) {

```

					ІІЗ.КР.Б – 121 – 25 – ІІЗ	Арк.
						71
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        return response()->json(['error' => 'Invalid
credentials'], 401);
    }

    // Get the authenticated user.
    $user = auth()->user();

    // (optional) Attach the role to the token.
    $token = JWTAuth::claims(['role' => $user->role])-
>fromUser($user);

    return response()->json(compact('token'));
} catch (JWTException $e) {
    return response()->json(['error' => 'Could not create token'],
500);
}
}

/**
 * Get user info.
 *
 * Get user info.
 */
#[OpenApi\Operation(tags: ['Auth'], method: 'GET')]
public function getUser()
{
    try {
        if (! $user = JWTAuth::parseToken()->authenticate()) {
            return response()->json(['error' => 'User not found'],
404);
        }
    } catch (JWTException $e) {
        return response()->json(['error' => 'Invalid token'], 400);
    }

    return response()->json(compact('user'));
}

// User logout
public function logout()
{
    JWTAuth::invalidate(JWTAuth::getToken());

    return response()->json(['message' => 'Successfully logged out']);
}
}

```

CompaarisonPage.jsx

```

import React from 'react';
import { useState, useEffect } from 'react';
import { Link as RouterLink, useNavigate } from 'react-router-dom';
import { Container, Paper, Typography, Box, CircularProgress, IconButton,
Button, Tabs, Tab } from '@mui/material';
import Table from '@mui/material/Table';
import TableBody from '@mui/material/TableBody';
import TableCell from '@mui/material/TableCell';
import TableContainer from '@mui/material/TableContainer';
import TableHead from '@mui/material/TableHead';
import TableRow from '@mui/material/TableRow';
import DeleteIcon from '@mui/icons-material/Delete';
import CompareArrowsIcon from '@mui/icons-material/CompareArrows';
import AddIcon from '@mui/icons-material/Add';
import LocationOnIcon from '@mui/icons-material/LocationOn';
import { useSnackbar } from 'notistack';
import { getComparisons, deleteComparison } from
'../services/room.service';
import { useAuth } from "../contexts/AuthContext";

const ComparisonPage = () => {
  const [comparisons, setComparisons] = useState([]);
  const [groupedComparisons, setGroupedComparisons] = useState({});
  const [currentHotelType, setCurrentHotelType] = useState('');
  const [loading, setLoading] = useState(true);
  const { enqueueSnackbar } = useSnackbar();
  const navigate = useNavigate();
  const { isAuthenticated } = useAuth();

  const fetchComparisons = async () => {
    try {
      const rooms = await getComparisons();
      setComparisons(rooms);

      const grouped = rooms.reduce((acc, comparison) => {
        const hotelType = comparison.room.hotel.hotel_type.type;
        if (!acc[hotelType]) {
          acc[hotelType] = [];
        }
        acc[hotelType].push(comparison);
        return acc;
      }, {});

      setGroupedComparisons(grouped);
      if (Object.keys(grouped).length > 0) {
        setCurrentHotelType(Object.keys(grouped)[0]);
      }
    } catch (error) {
      console.error('Error fetching comparisons:', error);
      enqueueSnackbar('Failed to load comparisons', { variant:
'error' });
    }
  };

```

					ІПЗ.КР.Б – 121 – 25 – ІЗ	Арк.
						73
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        } finally {
            setLoading(false);
        }
    };

    const handleDelete = async (id) => {
        try {
            await deleteComparison(id);
            enqueueSnackbar('Room removed from comparison', { variant:
'success' });
            await fetchComparisons();
        } catch (error) {
            console.error('Error removing comparison:', error);
            enqueueSnackbar('Failed to remove room', { variant: 'error' });
        }
    };

    const getComparisonStyle = (key, value, comparisons) => {
        if (value === null || value === undefined || comparisons.length <
2) return {};

        const values = comparisons.map(c => {
            const val = key.includes('.')
                ? key.split('.').reduce((obj, k) => obj?.[k], c.room)
                : c.room[key];
            return typeof val === 'number' ? val : parseFloat(val);
        }).filter(val => !isNaN(val));

        if (values.length < 2) return {};

        const max = Math.max(...values);
        const min = Math.min(...values);
        const current = typeof value === 'number' ? value :
parseFloat(value);

        if (isNaN(current)) return {};

        const getBestStyle = () => ({
            bgcolor: 'rgba(76, 175, 80, 0.1)',
            color: '#2e7d32',
            fontWeight: 'bold'
        });

        const getWorstStyle = () => ({
            bgcolor: 'rgba(244, 67, 54, 0.1)',
            color: '#d32f2f'
        });

        switch (key) {
            case 'price_per_night':
                if (current === min && current > 0) return getBestStyle();
                if (current === max) return getWorstStyle();
                break;
            case 'size_sqm':
            case 'sleeping_places_count':

```

					ІІЗ.КР.Б – 121 – 25 – ІІЗ	Арк.
						74
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        case 'bathroom_count':
        case 'hotel.rating':
        case 'last_renovation_year':
            if (current === max && current > 0) return getBestStyle();
            if ((current === min || current === 0) && max > 0) return
getWorstStyle();
            break;
        }

        return {};
    };

    useEffect(() => {
        if (isAuthenticated) {
            fetchComparisons();
        } else {
            setLoading(false);
        }
    }, [isAuthenticated]);

    if (loading) {
        return (
            <Box display="flex" justifyContent="center"
alignItems="center" minHeight="80vh">
                <CircularProgress sx={{ color: 'orange' }} />
            </Box>
        );
    }

    if (!isAuthenticated) {
        return (
            <Container maxWidth="xl" sx={{ py: 4 }}>
                <Box sx={{
                    display: 'flex',
                    flexDirection: 'column',
                    alignItems: 'center',
                    gap: 3,
                    p: 4,
                    borderRadius: 2,
                    bgcolor: 'rgba(255, 165, 0, 0.05)',
                    border: '1px dashed orange'
                }}>
                    <CompareArrowsIcon sx={{ fontSize: 48, color: 'orange'
}} />
                    <Typography variant="h5" textAlign="center"
color="text.secondary">
                        Please log in to use the comparison feature
                    </Typography>
                    <Button
                        variant="contained"
                        component={RouterLink}

to={` /login?returnTo=${encodeURIComponent('/compare')}` }
                        sx={{
                            bgcolor: 'orange',

```

					ІІЗ.КР.Б – 121 – 25 – ІІЗ	Арк.
						75
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        '&:hover': { bgcolor: '#e68900' },
        px: 3,
        py: 1
      }}
    >
      Login
    </Button>
  </Box>
</Container>
);
}

if (Object.keys(groupedComparisons).length === 0) {
  return (
    <Container maxWidth="xl" sx={{ py: 4 }}>
      <Box sx={{
        display: 'flex',
        flexDirection: 'column',
        alignItems: 'center',
        gap: 3,
        p: 4,
        borderRadius: 2,
        bgcolor: 'rgba(255, 165, 0, 0.05)',
        border: '1px dashed orange'
      }}>
        <CompareArrowsIcon sx={{ fontSize: 48, color: 'orange'
}} />
        <Typography variant="h5" textAlign="center"
color="text.secondary">
          No rooms to compare
        </Typography>
        <Button
          variant="contained"
          startIcon={<AddIcon />}
          onClick={() => navigate('/explore')}
          sx={{
            bgcolor: 'orange',
            '&:hover': { bgcolor: '#e68900' },
            px: 3,
            py: 1
          }}
        >
          Add Rooms to Compare
        </Button>
      </Box>
    </Container>
  );
}

const compareFeatures = [
  {
    section: 'Hotel Information', features: [
      { label: 'Hotel Type', key: 'hotel.hotel_type.type' },
      { label: 'Hotel Rating', key: 'hotel.rating', suffix: '
Stars', highlight: true },

```

					ІІЗ.КР.Б – 121 – 25 – ІІЗ	Арк.
						76
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        { label: 'Location', key: 'hotel.address' },
        { label: 'City', key: 'hotel.city' },
        { label: 'Country', key: 'hotel.country' }
      ]
    },
    {
      section: 'Room Details', features: [
        { label: 'Room Type', key: 'room_type' },
        { label: 'Price per Night', key: 'price_per_night', prefix:
'$', highlight: true },
        { label: 'Room Size', key: 'size_sqm', suffix: ' m²',
highlight: true },
        { label: 'Sleeping Places', key: 'sleeping_places_count',
highlight: true },
        { label: 'Bathrooms', key: 'bathroom_count', highlight:
true },
        { label: 'Floor Number', key: 'floor_number' },
        { label: 'Last Renovated', key: 'last_renovation_year',
highlight: true }
      ]
    },
    {
      section: 'Booking Information', features: [
        { label: 'Check-in Time', key: 'check_in_time' },
        { label: 'Check-out Time', key: 'check_out_time' },
        { label: 'Availability', key: 'is_available', transform:
(value) => value ? 'Available' : 'Not Available' }
      ]
    }
  ];

  const currentComparisons = groupedComparisons[currentHotelType] || [];
  return (
    <Container maxWidth="xl" sx={{ py: 4 }}>
      <Box sx={{ display: 'flex', justifyContent: 'space-between',
alignItems: 'center', mb: 3 }}>
        <Box sx={{ display: 'flex', alignItems: 'center', gap: 2
}}>
          <CompareArrowsIcon sx={{ color: 'orange', fontSize: 32
}} />
          <Typography variant="h4" fontWeight="bold">
            Room Comparison
          </Typography>
        </Box>
        <Button
          variant="contained"
          startIcon={<AddIcon />}
          onClick={() => navigate('/explore')}
          sx={{
            bgcolor: 'orange',
            '&:hover': { bgcolor: '#e68900' },
            borderRadius: 2,
            px: 3
          }}
        >

```

					ІІЗ.КР.Б – 121 – 25 – ІІЗ	Арк.
						77
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        Add More Rooms
      </Button>
    </Box>

    <Tabs
      value={currentHotelType}
      onChange={(e, newValue) => setCurrentHotelType(newValue)}
      sx={{ mb: 2 }}
    >
      {Object.keys(groupedComparisons).map((hotelType) => (
        <Tab
          key={hotelType}
          label={` ${hotelType}
            (${groupedComparisons[hotelType].length})`}
          value={hotelType}
          sx={{
            '&.Mui-selected': {
              color: 'orange',
            }
          }}
        />
      ))}
    </Tabs>

    <TableContainer component={Paper} sx={{ borderRadius: 2,
      boxShadow: '0 4px 6px rgba(0, 0, 0, 0.1)' }}>
      <Table>
        <TableHead>
          <TableRow sx={{ bgcolor: 'rgba(255, 165, 0, 0.1)'
            >
            <TableCell><Typography
              fontWeight="bold">Features</Typography></TableCell>
            {currentComparisons.map((comparison) => (
              <TableCell key={comparison.id}>
                <Box sx={{
                  display: 'flex',
                  alignItems: 'center',
                  justifyContent: 'space-between',
                  borderBottom: '2px solid orange',
                  pb: 1
                }}>
                  <Box>
                    <Typography
                      fontWeight="bold">{comparison.room.hotel.name}</Typography>
                    <Typography variant="caption"
                      color="text.secondary" sx={{ display: 'flex', alignItems: 'center', gap: 0.5 }}>
                      <LocationOnIcon sx={{
                        fontSize: 14 }} />
                      {comparison.room.hotel.city}, {comparison.room.hotel.country}
                    </Typography>
                  </Box>
                  <IconButton
                    onClick={() =>
                      handleDelete(comparison.id)}

```

					ІІЗ.КР.Б – 121 – 25 – ІІЗ	Арк.
						78
Змн.	Арк.	№ докум.	Підпис	Дата		

```

size="small"
sx={{
  color: 'text.secondary',
  '&:hover': {
    color: 'error.main',
    bgcolor: 'error.light'
  }
}}
>
  <DeleteIcon />
</IconButton>
</Box>
</TableCell>
)}}
</TableRow>
</TableHead>
<TableBody>
  {compareFeatures.map((section) => (
    <React.Fragment key={section.section}>
      <TableRow>
        <TableCell
colSpan={currentComparisons.length + 1}>
          <Typography
            variant="subtitle1"
            sx={{
              color: 'orange',
              fontWeight: 'bold',
              py: 1
            }}
          >
            {section.section}
          </Typography>
        </TableCell>
        <TableRow>
          {section.features.map(({ label, key,
prefix = '', suffix = '', transform, highlight }) => (
            <TableRow key={key} sx={{
              '&:nth-of-type(odd)': { bgcolor:
'rgba(0, 0, 0, 0.02)' }
            }}>
              <TableCell sx={{ fontWeight: 500
}}>{label}</TableCell>
            {currentComparisons.map((comparison) => {
              const value =
key.includes('.')
?
key.split('.').reduce((obj, k) => obj?.[k], comparison.room)
: comparison.room[key];
              return (
                <TableCell
key={` ${comparison.id}-${key}` }

```

```

                                sx={highlight      ?
getComparisonStyle(key, value, currentComparisons) : {}
                                >
                                {transform          ?
transform(value) : `${prefix}${value}${suffix}`}
                                </TableCell>
                                );
                                }}}
                                </TableRow>
                                )))
                                </React.Fragment>
                                )))
                                </TableBody>
                                </Table>
                                </TableContainer>
                                </Container>
                                );
};

export default ComparisonPage;

```

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						80
Змн.	Арк.	№ докум.	Підпис	Дата		

RoomComparisonController.php

```

<?php

namespace App\Http\Controllers;

use App\Http\Requests\StoreRoomComparsionRequest;
use App\Http\Requests\UpdateRoomComparsionRequest;
use App\Http\Resources\RoomComparsionResource;
use App\Models\RoomComparsion;
use App\Models\Room;
use Illuminate\Http\JsonResponse;
use Illuminate\Http\Resources\Json\AnonymousResourceCollection;
use Symfony\Component\HttpFoundation\Response;
use Vyuldashev\LaravelOpenApi\Attributes as OpenApi;

#[OpenApi\PathItem]
class RoomComparsionController extends Controller
{
    // public function __construct()
    // {
    //     $this->authorizeResource(RoomComparsion::class, 'roomComparsion');
    // }
    /**
     * Display a listing of the resource (USER only)
     */
    #[OpenApi\Operation(tags: ['Room Comparison'], method: 'GET')]
    public function index(): JsonResponse
    {
        $comparisons = RoomComparsion::where('user_id', auth()->user()-
>id)
            ->with(['room.hotel.hotelType'])
            ->get()
            ->groupBy('room.hotel.hotelType.name');

        return response()->json([
            'data' => $comparisons
        ], Response::HTTP_OK);
    }
    /**
     * Store a newly created resource in storage (USER only)
     */
    #[OpenApi\Operation(tags: ['Room Comparison'], method: 'POST')]
    public function store(StoreRoomComparsionRequest $request):
JsonResponse
    {
        $data = $request->validated();
        $data['user_id'] = auth()->id();
        $room = Room::find($data['room_id']);
        if(!$room){
            return response()->json(['error' => 'Room not found'],
Response::HTTP_NOT_FOUND);

```

					ІІЗ.КР.Б – 121 – 25 – ІІЗ	Арк.
						81
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    }

    $data['hotel_type_id'] = $room->hotel->hotelType->id;

    $comparison = RoomComparsion::create($data);
    return response()->json(new RoomComparsionResource($comparison),
Response::HTTP_CREATED);
    }

    /**
     * Display the specified resource (USER only)
     */
    #[OpenApi\Operation(tags: ['Room Comparison'], method: 'GET')]
    public function show(RoomComparsion $roomComparsion): JsonResponse
    {
        $this->checkOwnedUser($roomComparsion);
        return response()->json(new
RoomComparsionResource($roomComparsion), Response::HTTP_OK);
    }

    /**
     * Update the specified resource in storage (USER only)
     */
    #[OpenApi\Operation(tags: ['Room Comparison'], method: 'POST')]
    public function update(UpdateRoomComparsionRequest $request,
RoomComparsion $roomComparsion): JsonResponse
    {
        $this->checkOwnedUser($roomComparsion);
        $roomComparsion->update($request->validated());
        return response()->json(new
RoomComparsionResource($roomComparsion), Response::HTTP_OK);
    }

    /**
     * Remove the specified resource from storage (USER only)
     */
    #[OpenApi\Operation(tags: ['Room Comparison'], method: 'DELETE')]
    public function destroy(RoomComparsion $roomComparsion): JsonResponse
    {
        $this->checkOwnedUser($roomComparsion);
        $roomComparsion->delete();
        return response()->json([], Response::HTTP_NO_CONTENT);
    }

    private function checkOwnedUser(RoomComparsion $roomComparsion)
    {
        if($roomComparsion->user_id != auth()->id()){
            return response()->json(['error' => 'User id not provided'],
Response::HTTP_FORBIDDEN);
        }
    }
}

```

					ІІЗ.КР.Б – 121 – 25 – ІІЗ	Арк.
						82
Змн.	Арк.	№ докум.	Підпис	Дата		