МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра інформаційних технологій

«До захисту допущено»
Т.в.о. завідувача кафедри

_____ Світлана ВАЩЕНКО

____ _____ 2023 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня магістр

зі спеціальності 122 «Комп'ютерні науки»

освітньо-наукової програми «Інформаційні технології проектування»

на тему: Web-орієнтована система замовлення послуг готельно-ресторанного комплексу «Шафран»

Здобувача (ки) групи ІТ.м-22. Станкевича Данііла Володимировича
(шифр групи) (прізвище, ім'я, по батькові)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

_____ Станкевич Данііл Володимирович _
(підпис) (Ім'я та ПРІЗВИЩЕ здобувача)

Керівник доцент кафедри інформаційних технологій,
доцент, к.т.н. Чибіряк Я.І _____
(посада, науковий ступінь, вчене звання, Ім'я та ПРІЗВИЩЕ) (підпис)

Суми – 2023

Сумський державний університет

Факультет електроніки та інформаційних технологій
Кафедра інформаційних технологій
Спеціальність 122 «Комп'ютерні науки»
Освітньо-професійна програма «Інформаційні технології проектування»

ЗАТВЕРДЖУЮ
В.о. зав. кафедри ІТ

_____ С. М. Ващенко
«___» _____ 2023 р.

ЗАВДАННЯ

на кваліфікаційну роботу магістра студентові

Станкевич Даніїл Володимирович
(прізвище, ім'я, по батькові)

1 Тема проекту Web-орієнтована система замовлення послуг готельно-ресторанного комплексу «Шафран»
затверджена наказом по університету від «05» вересня 2023 р. № 0465-VI

2 Термін здачі студентом закінченого проекту «7» грудня 2023 р.

3 Вхідні дані до проекту результати роботи web-орієнтованої технології

4 Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити) аналіз предметної області, постановка задачі, методи дослідження, проектування інформаційної технології підтримки користувачів в соціальній мережі, проектування та розробка соцільної мережі

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) актуальність проблеми, мета проетку, постановка задачі, функціональні вимоги до проекту, контекстна діаграма IDEF0, декомпозиція IDEF0, схема інформаційної технології, діаграма варіантів використання, фізична модель даних, засоби реалізації, практична реалізація, висновки

6. Консультанти випускної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

Дата видачі завдання _____.

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів випускної проекту	Термін виконання етапів проекту	Примітка
1	Огляд технологій для підтримки процесу бронювання номерів	до 23.09.2023	
2	Постановка задачі та формування завдань дослідження	до 04.10.2023	
3	Структурно-функціональне моделювання	до 25.11.2023	
4	Реалізація додатку з використанням React.js, MongoDB, Express.js та Node.js	до 01.12.2023	
5	Оформлення пояснювальної записки до кваліфікаційної роботи	до 07.12.2023	

Магістрант _____ Станкевич Д.В.

Керівник роботи _____ к.т.н., доц. Чибіряк Я.І.

АНОТАЦІЯ

Записка: 85 сторінок, 58 рисунків, 4 додатки, 30 використаних джерел та літератури.

Обґрунтування актуальності теми роботи: У контексті стрімкого розвитку готельно-ресторанного бізнесу та зростання конкуренції, необхідність оптимізації та автоматизації процесів управління готельно-ресторанним комплексом є критичною. Впровадження сучасних інформаційних технологій для підтримки замовлення послуг може значно полегшити роботу персоналу, забезпечити зручність для клієнтів та підвищити конкурентоспроможність комплексу «Шафран».

Мета роботи: розробка web-орієнтованої системи замовлення послуг для оптимізації управління готельно-ресторанним комплексом «Шафран».

Методи дослідження: використання та аналіз сучасних веб-технологій, дослідження кращих практик у сфері управління готельно-ресторанним бізнесом.

Результат: Розроблена система замовлення послуг, спрямована на оптимізацію процесів управління готельно-ресторанним комплексом. Це дозволить автоматизувати та полегшити процеси прийому та виконання замовлень, покращить взаємодію з клієнтами, а також сприятиме підвищенню рівня обслуговування та ефективності комплексу «Шафран».

Ключові слова: інформаційна технологія, захист даних, MERN, MongoDB SocketIO, React.js, готельний бізнес.

ЗМІСТ

ВСТУП **6**
1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ **8**
1.1 Огляд останніх досліджень та публікацій **8**
1.2 Аналіз програмних продуктів - аналогів **10**
1.3 Постановка задачі **18**
2 МОДЕЛЮВАННЯ WEB-ДОДАТКУ **24**
2.1 Структурно-функціональне моделювання **24**
2.2 Моделювання варіантів використання **26**
2.3 Проектування бази даних **28**
3 ПРАКТИЧНА РЕАЛІЗАЦІЯ WEB-ОРІЄНТОВАНОЇ СИСТЕМИ **33**
3.1 Архітектура web-орієнтованої системи **33**
3.2 Реалізація серверної частини **35**
3.3 Настанови з використання **42**
3.4 Тестування та аналіз хостинг систем **51**
ВИСНОВОК **55**
СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ **56**
ДОДАТОК А **60**

ВСТУП

Сучасний цифровий світ потребує інструментів і технологій, які допомагають бізнесу розвиватися в глобальному масштабі. Дійсно, веб-додаток є невід'ємною частиною будь-якої установи. Від продажу товарів/послуг онлайн до обміну важливою інформацією про компанію, присутність в Інтернеті відіграє важливу роль. Оскільки бізнес починає працювати в Інтернеті, веб-додаток - це те, що потрібно, щоб вийти на глобальні платформи.

Хоча мобільні додатки витісняють звичні веб-сайти, професійний веб-додаток є ефективним інструментом для досягнення галузевих стандартів. Завдяки персоналізованому дизайну та спеціальному контенту веб-сайти є багатоплатформними додатками, які адаптуються до будь-якого пристрою. Розробка веб-додатків для бізнесу дозволяє розробникам створювати та ділитися інформацією без обмежень.

За останні роки веб-сервіси трансформувалися, і завдяки значним покращенням у сфері безпеки та технологій традиційні програмні додатки та системи можна легко вдосконалити, мігрувавши їх у веб-сервіси. Інформаційні технології вимагають складної підготовки і наукомісткої техніки. Їх введення повинно починатися зі створення математичного забезпечення, формування інформаційних потоків у системах підготовки фахівців.

Системи розробки веб-додатків встановлюються лише на сервері, висуваючи мінімальні вимоги до робочої станції кінцевого користувача. Обслуговувати та оновлювати систему набагато простіше, будь-які клієнтські оновлення можна легко розгортати через веб-сервер.

Метою дослідження є створення web-орієнтованої системи замовлення послуг готельно-ресторанного комплексу «Шафран». Для досягнення мети потрібно вирішити наступні задачі:

- виконати аналіз програмних продуктів-аналогів;
- визначити вимоги до створюваної web-системи;

- проаналізувати та обрати технології для реалізації
- виконати етап проектування web-орієнтованої системи
- розробити архітектуру веб-додатку;
- виконати розробку бази даних;
- виконати практичну реалізацію web-системи;

Актуальність створення web-орієнтованої системи полягає в підвищенні конкурентноспроможності готельно-ресторанного комплексу, автоматизації частини бізнес-процесів замовлення готелю за рахунок впровадження інформаційних технологій в діяльність системи.

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд останніх досліджень та публікацій

Веб-сервіси Інтернету використовуються для побудови електронних бізнес-додатків, щоб забезпечити їх взаємозв'язок і гнучкість. Коли користувачеві потрібна більш ніж одна послуга одночасно, здійснюється компіляція доступних послуг, щоб задовольнити запит користувача на послугу. Коли доступний широкий спектр інтернет-послуг, ми вимагаємо належної процедури компонування послуг, заснованої на аспектах, що впливають на якість послуг. Сьогодні для одного веб-сервісу існує декілька сервісів з однаковими функціональними можливостями [1].

Для туристів IT-додатки також принесли набагато більше цікавих вражень та подорожей під час кожної подорожі. Адже перед тим, як вирушити до будь-якого місця, одним клацанням миші вони можуть дізнатися всю необхідну інформацію, вибрати нове місце для відкриття, яке відповідає їхнім інтересам, і скласти відповідний план витрат. Використовуючи методи синтезу в поєднанні зі спостереженнями, автори хотіли б припустити, що туристичним компаніям необхідно скористатися перевагами веб-сайтів та соціальних мереж в Інтернеті, таких як Facebook, Twitter, Zalo, Youtube... а також таких сайтів, як Agoda.com, Booking.com, Traveloka.com... Крім того, необхідна політика підтримки для створення кращих умов, що допомагають людям, які займаються туристичним бізнесом, а також громадські ролі [2].

Стратегічний розвиток веб-систем для надання послуг бронювання готелю надає більш інноваційні алгоритми взаємодії користувача з сервером. Якщо десять років назад користувачу необхідно було телефонувати на рецепшн та витрачати час на прослуховування консультацій менеджера готелю, то зараз всього за декілька хвилин користувач може обрати готель з всіма потребами для комфортного проведення часу [3].

Пандемія відіграла вирішальну роль у формуванні звичок споживачів щодо

онлайн замовлень. Тому, незважаючи на те що активна фаза пандемії майже завершилася, можна побачити статистику, яка показує, що в травні 2022 року служби бронювання готелів зросли на 8 відсотків у річному обчисленні.

Клієнтів підштовхують до того, щоб зробити перше замовлення в системі резервації та використовувати програмне забезпечення для онлайн-замовлень. За даними Bloomberg, 50 відсотків споживачів коли-небудь робили замовлення хоча б через один з сервісів резервації, порівняно з 44 відсотками рік тому [4].

Веб-орієнтовані системи є чудовою альтернативою нативним додаткам, підвищуючи швидкість реагування, покращуючи клієнтський досвід та продуктивність співробітників. Завдяки низьким витратам на розробку та швидкому часу на створення, веб-системи можуть суттєво вплинути на продажі та прибутки готельно-ресторанного бізнесу, що є прямими перевагами використання веб-систем для ресторанів та готелів.

Готельний та ресторанний бізнес надає широкий спектр послуг для задоволення потреб клієнтів, які подорожують або просто шукають комфортне місце для проживання та харчування. Такі підприємства можуть зосереджуватися на різних видах послуг, беручи до уваги різні потреби та вимоги своїх клієнтів [5].

Потреба в послугах виникає в різних ситуаціях, наприклад, під час подорожей, ділових зустрічей, розваг та інших заходів. Порядок надання цих послуг може визначатися відповідно до індивідуальних потреб клієнта, його уподобань, бюджету та мети візиту.

Розроблена система зможе фокусуватися на певних видах послуг залежно від унікальності готелю та його цільової аудиторії. Наприклад, якщо готель розташований у туристичній зоні, система фільтрації надасть змогу зосередитися на туристичних пропозиціях. Якщо вибір користувача орієнтований на ділових гостей, то система рекомендації надасть пропозиції які відносяться до даної потреби. Розуміння потреб цільової аудиторії є ключовим для визначення стратегії та фокусу певного виду послуг у готельному бізнесі. Загалом, система надає наступні можливості:

- вибір номеру в готелі за власними критеріями;

- додавання номерів до обраного;
- перегляд відгуків користувачів щодо користування послугами сервісу;
- перегляд власних замовлень;
- додавання додаткових опцій до замовлення (вибір типу кімнати, можливість бути з домашніми улюбленцями).

1.2 Аналіз програмних продуктів - аналогів

Для аналізу аналогів було обрано три web-орієнтовані системи, які схожі за тематикою і призначенням:

- https://www.booking.com;
- https://tripmydream.ua/hotels;
- https://hotelmix.com.ua/;

booking.com

Дана web-орієнтована система є однією з найпопулярніших онлайн-платформ для бронювання готелю, гостьових будинків, апартаментів по всьому світу Серед переваг можна відзначити великий вибір різних видів помешкань що також надає широке географічне охоплення. Чіткий та зрозумілий для користувача інтерфейс допомагає зберегти час на пошуки необхідної послуги. Система відгуків та рейтингів від інших користувачів допомагає у виборі потрібного готелю, враховуючи власні вподобання.

Незважаючи на всі переваги web системи, недоліки також присутні. У деяких випадках у користувача можуть виникати проблеми з відміненням бронювання, а також можлива зміна цін, яка змінюється в залежності від попиту [6].

На рисунках 1.1 – 1.3 представлено вигляд магазину booking.com .

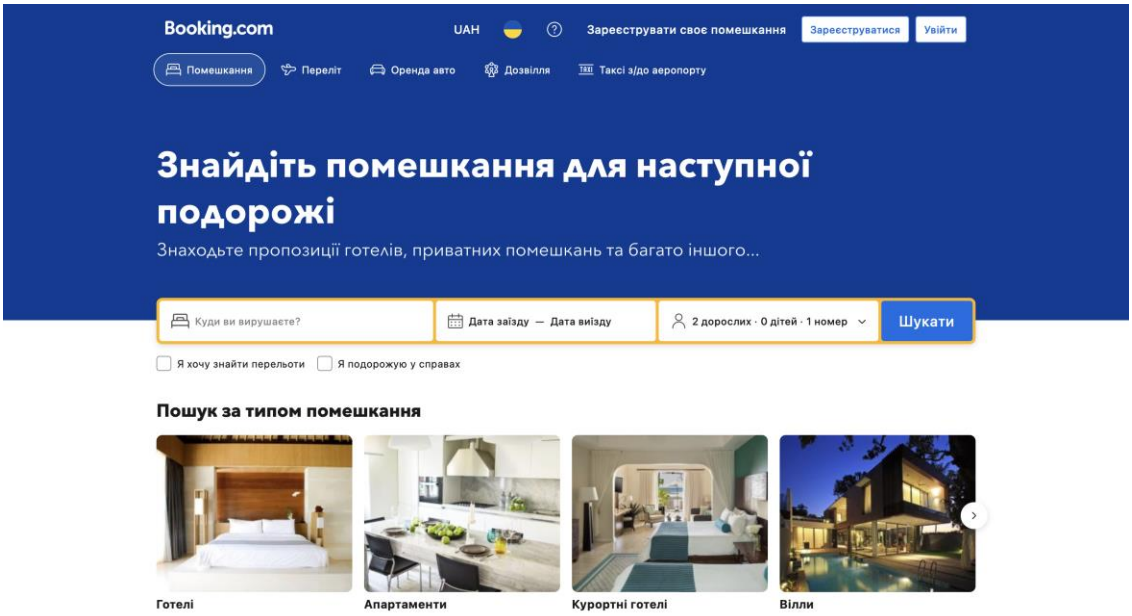

Рисунок 1.1 – Вигляд головної сторінки сайту booking.com

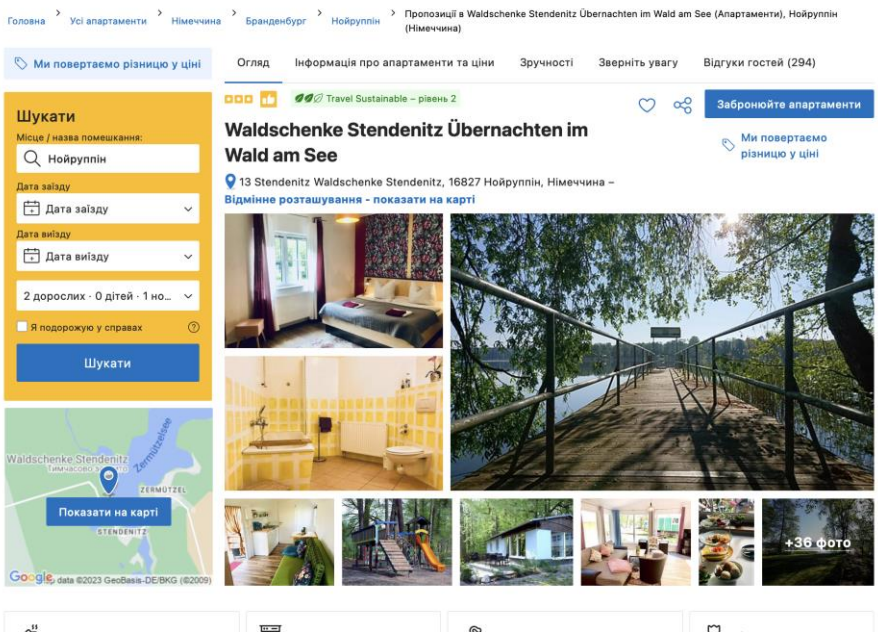

Рисунок 1.2 – Вигляд сторінки готелю сайту booking.com

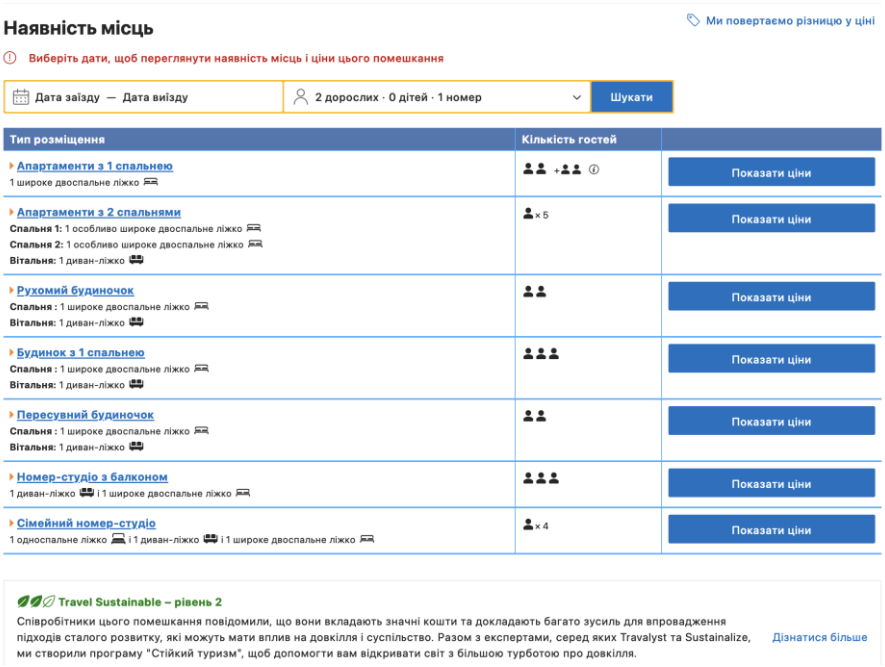

Рисунок 1.3 – Сторінка бронювання готелю сайту booking.com

gohotels.com.ua

GoToHotels.ua є українською онлайн-платформою, яка спеціалізується на бронюванні готелів, а також пакетних турів та подорожей. Серед головних переваг сайту є спеціалізація на українському ринку, що включає в себе акційні пропозиції та вигідні угоди для жителів України. Також система надає пакетні послуги та акційні пропозиції для тих, хто шукає комплексні варіанти подорожей. В порівнянні з іншими аналогічними системами, GoToHotels.ua має менший спектр географічного охоплення, що в свою чергу не надає користувачу обрати готель в будь-якій точці світу. Обмежена кількість відгуків та рейтингів не надає чітку інформацію користувачу для вибору місця відпочинку [7]. На рисунках 1.4 – 1.6 представлено вигляд сайту gohotels.com.ua

Рисунок 1.4 – Вигляд головної сторінки сайту gohotels.com.ua

Рисунок 1.5 – Вигляд сторінки обраного міста сайту gohotels.com.ua

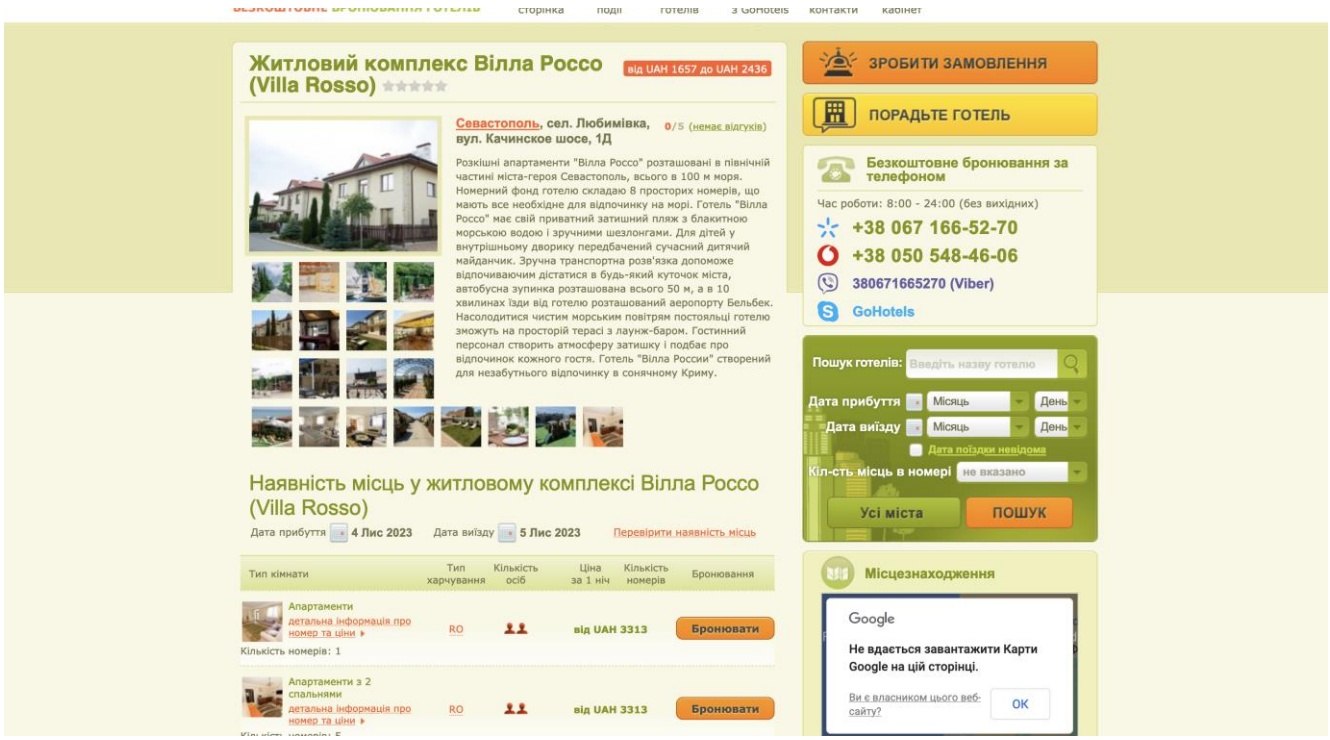

Рисунок 1.6 – Вигляд сторінки бронювання сайту gohotels.com.ua

hotelmix.com.ua

HotelMix є онлайн-платформою, яка дозволяє знаходити та порівнювати ціни на готелі у різних системах бронювання. Ця платформа надає можливість порівняти ціни з різних джерел, що може допомогти знайти найбільш вигідну пропозицію. Також дана платформа надає широкий вибір різних систем бронювання. Загалом, HotelMix може бути корисним для тих, хто шукає найкращі ціни на готелі та готовий витратити час на порівняння різних пропозицій [8]. На рисунку 1.7 – 1.9 представлено вигляд сайту hotelmix.com.ua .

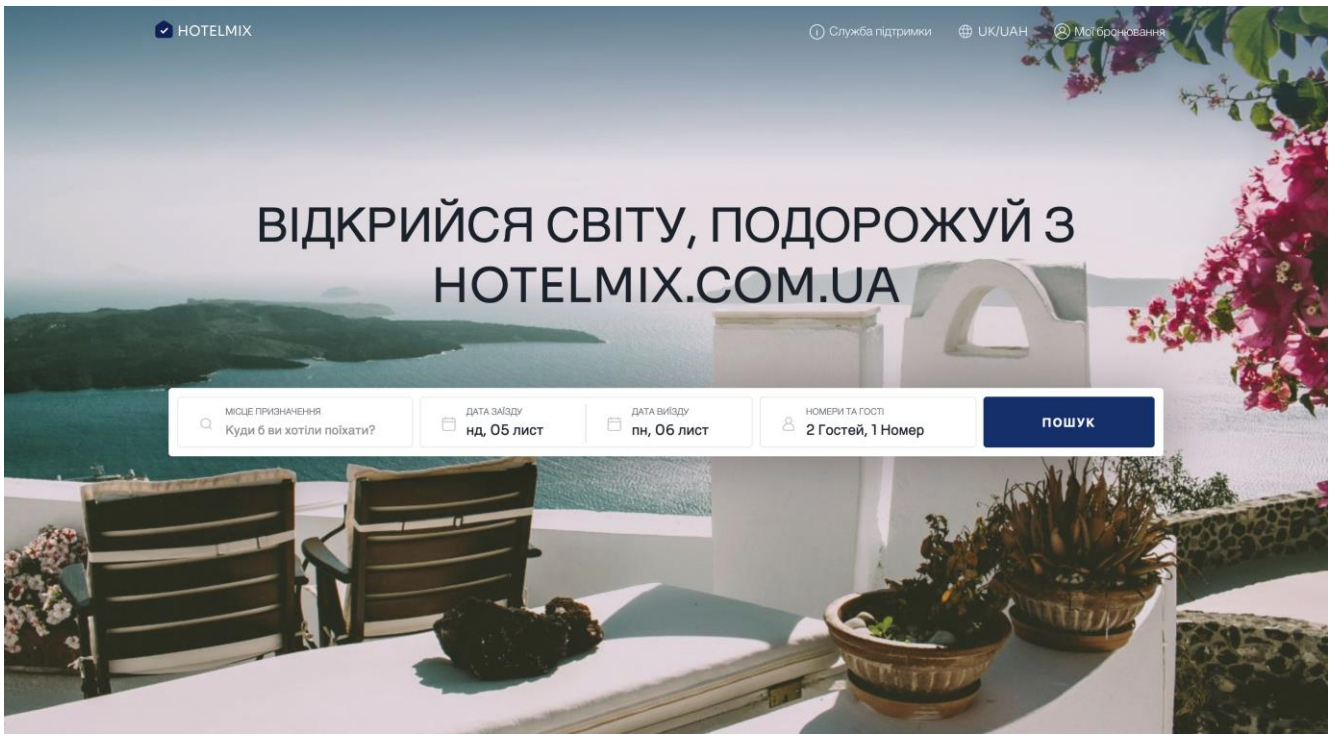

Рисунок 1.7 – Вигляд головної сторінки сайту hotelmix.com.ua

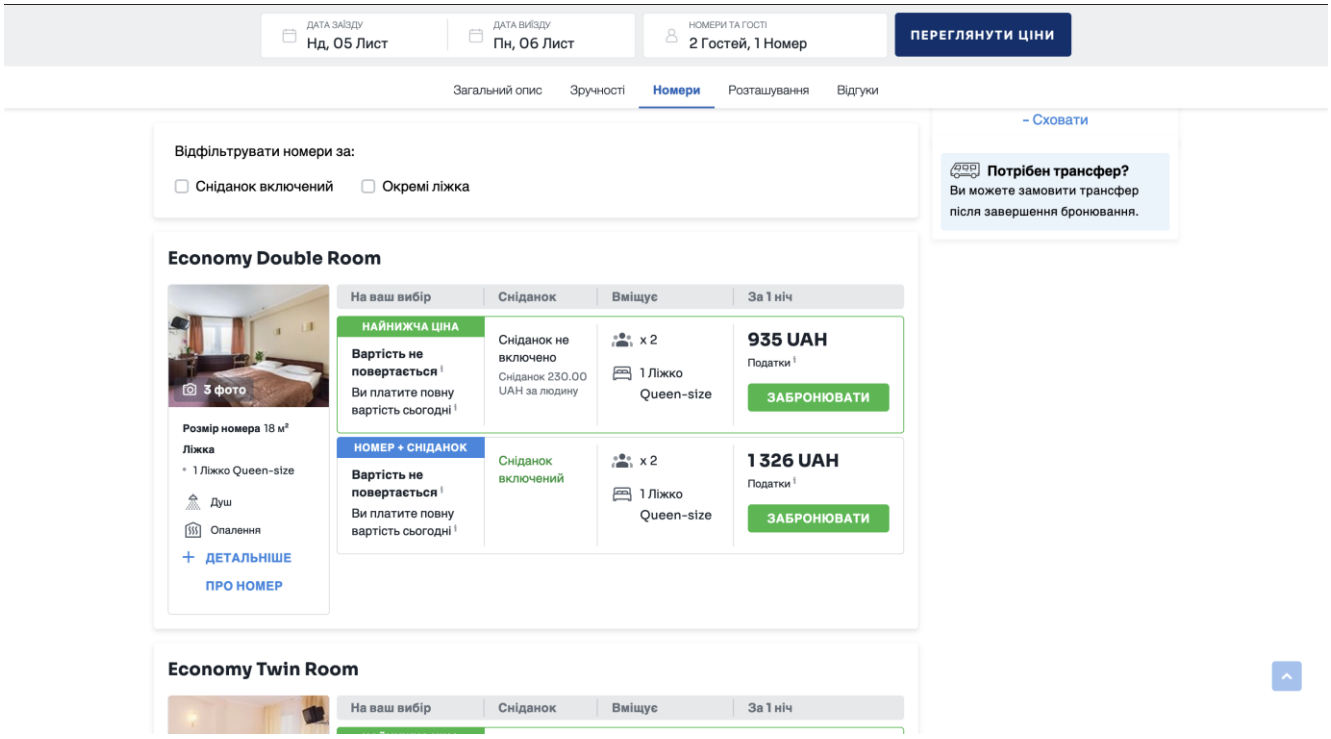

Рисунок 1.8 – Вигляд сторінки обраного готелю hotelmix.com.ua

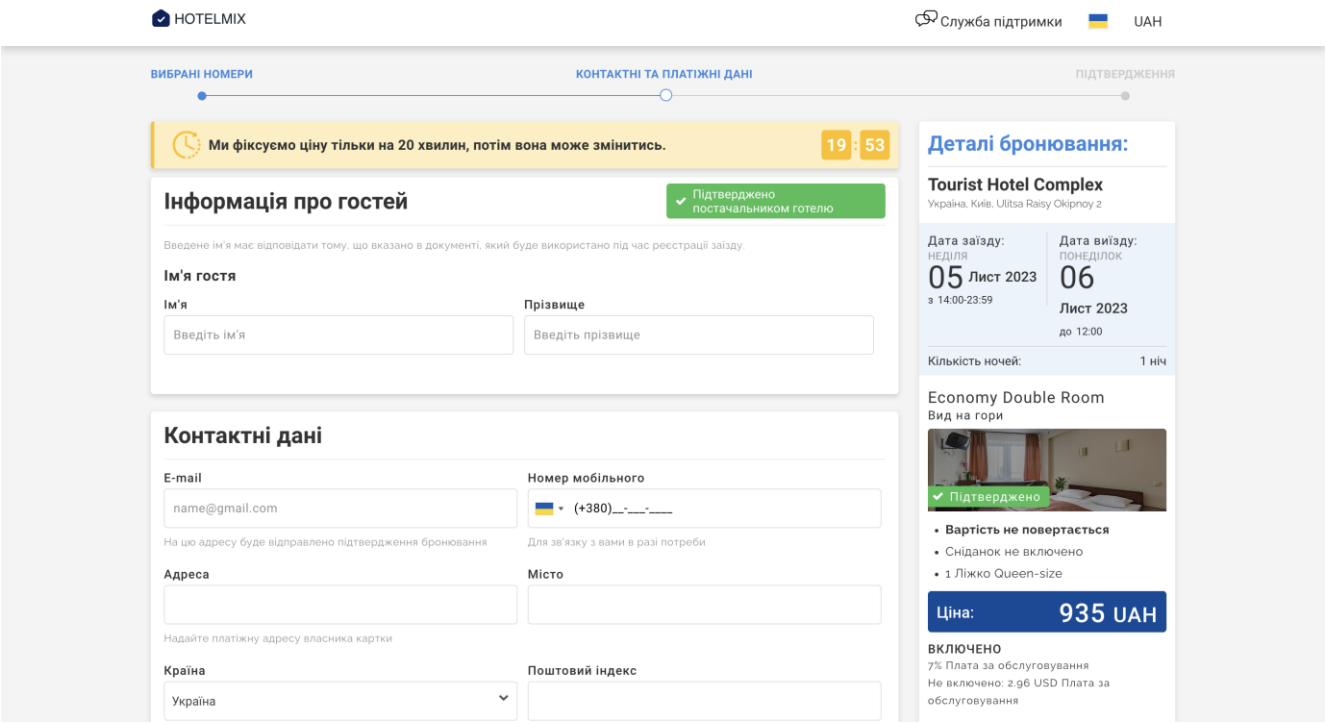

Рисунок 1.9 – Вигляд сторінки бронювання готелю hotelmix.com.ua

Виконавши аналіз аналогів web-систем для бронювання готелю була створена таблиця порівняння(табл 1.1), для того щоб визначити головні моменти для реалізації проекту.

Серед основних недоліків існуючих аналогів слід відзначити найбільш загальні:

- недостатній набір функцій;
- оптимізація сайту;
- присутність реклами.

До переваг додатків-аналогів можна віднести:

- оптимізація під мобільні пристрої;
- наявність відгуків.

Таблиця 1.1 – Аналіз розглянутих сервісів-аналогів

Критерії оцінювання	Сайти – аналоги		
	booking.com	gohotels.com.ua	hotelmix.com.ua
Наявність адаптивної верстки	-	+	+
Система хешування особистих даних	+	+	+
Інтуїтивно зрозумілий інтерфейс	+	+	+
Наявність системи фільтрації даних	-	+	+
Відсутність реклами	+	+	+
Наявність актуальних цін	-	+	-

При розробці проекту необхідно враховувати вище перераховані результати аналізу, та розробити додаток, який б усував явні недоліки.

Розроблена система має надавати покращену технологію збереження та хешування даних шляхом використання технології Socket.IO. Також для більш чіткого вибору за критеріями користувач отримуватиме можливість використовувати систему фільтрації та опції для додавання послуг за бажанням. За допомогою інтуїтивно зрозумілого та адаптивного інтерфейсу, користувач знайде необхідні для нього послуги.

1.3 Постановка задачі

Аналіз призначений для того, щоб розробити систему, за допомогою якого користувач міг виконати замовлення послуг готельно-ресторанного комплексу «Шафран» в задані дні місяця. Даний сервіс повинен бути реалізований як web-додаток, який буде доступний в мережі Інтернет для всіх доступних пристроїв.

Web-система повинен надавати наступні можливості:

- авторизація;
- перегляд всіх доступних номерів готелю;
- пошук номера за вказаними критеріями за допомогою системи фільтрації;
- бронювання послуг щодо номерів готелю;
- наявність зворотнього зв’язку;
- адміністративна панель.

Web-додаток має містити користувацьку та адміністративну частини, які мають бути реалізовані за допомогою клієнт-серверної архітектури.

Адміністративна панель має надавати авторизованому користувачу з привілеями можливість виконувати операції додаванням, оновленням та видаленням інформації про наявні готелі, а також має можливість редагувати окремі дані, головна задача яких надавати користувачу основну інформацію про діяльність web-орієнтованої системи.

Для користувача необхідно реалізувати зрозумілий інтерфейс, який швидко буде реагувати на запити користувача.

Для успішного виконання завдання кваліфікаційної роботи магістра необхідно вирішити наступні завдання:

- провести огляд популярних аналогів, таких як Booking.com, Airbnb, або інші системи, які надають подібні послуги та визначити переваги та недоліки;
- визначити функціональні вимоги, такі як можливості пошуку, бронювання, обслуговування користувачів;
- реалізувати клієнт-серверну архітектуру;
- реалізувати структуру, функціонал та дизайн web-системи;
- створити логічну модель бази даних, використовуючи ER-діаграми для визначення сутностей та їх зв'язків, а також обрати систему управління базами даних та реалізувати створену модель;

1.4 Аналіз технологій та засобів реалізації web-сервісу

Для аналізу технологій та засобів реалізації web-сервісу було обрано 3 види архітектур:

- клієнт-серверна архітектура;
- хмарна атхітектура;
- об`єктно-орієнтована архітектура;

На таблиці 1.2 представлено порівняльну характеристику кожної з архітектур

Критерій	Клієнт-серверні архітектура	Хмарна архітектура	Об'єктно-орієнтована архітектура
Розділення клієнтської та серверної частини	Зручна технологія, але збільшує обмін даними	Наявна технологія яка полегшує масштабування	Розділення завдяки об'єктам та класам
Масштабованість розробки	Залежить від серверної інфраструктури	Легке масштабування, гнучкість	Залежить від ефективного використання об'єктів
Безпека та конфіденційність збереження даних	Залежить від налаштувань серверу	Недостатній захист	Використання інкапсуляції та

Критерій	Клієнт-серверні архітектура	Хмарна архітектура	Об'єктно-орієнтована архітектура
			доступу до об'єктів
Швидкодія	Залежить від реалізації сервера	Залежить від хмарної інфраструктури	Залежить від ефективного використання об'єктів
Зручність розробки та підтримки	Залежить від вибору технологій	Висока	Зручна завдяки об'єктно-орієнтованому підходу

Джерело: побудовано автором

Для даної системи було обрано клієнт – серверну архітектуру, яка містить наступні переваги:

1. Сервер містить усі необхідні дані. Як наслідок, легко захистити дані і запропонувати автентифікацію та авторизацію.
2. Не обов'язково, щоб сервер знаходився близько до клієнтів. До інформації все одно можна отримати ефективний доступ.
3. Клієнти мають обмежене значення в цій архітектурі і потребують менше управління на сервері адміністрування рівня.
4. Дана артіхітектура пропонує інтеграцію сервісів, що дозволяє всім клієнтам отримати доступ до корпоративних даних за допомогою своїх терміналів і усуває всі права на вхід.
5. Відновлення даних за допомогою клієнт-серверної моделі простіше, ніж за допомогою децентралізованих серверних моделей.

Оскільки метою розробки є розробка web-орієнтованої системи замовлення послуг готельно – ресторанного комплексу «Шафран» з застосуванням клієнт-серверної архітектури з інтерактивними елементами, то для цього буде

використовуватися технологія MERN, яка відноситься до клієнт-серверної технології [9].

Стек MERN - це популярний фреймворк JavaScript, який використовується для створення веб-додатків. Це абревіатура від MongoDB, Express.js, React та Node.js. Кожен компонент стеку відіграє певну роль у процесі розробки. Стек MERN, відомий своєю універсальністю та ефективністю, дозволяє розробникам створювати надійні та масштабовані додатки за допомогою JavaScript.

Стек MERN вирізняється своєю універсальністю, ефективністю та наскрізною реалізацією JavaScript при створенні веб-додатків. Він дозволяє розробникам використовувати єдину мову JavaScript у всьому стеку, що дає змогу повторно використовувати код і спрощує процес розробки. Ось коротке пояснення чотирьох компонентів стеку:

MongoDB - це база даних NoSQL, яка використовується як компонент для зберігання даних у стеку MERN. Відома своєю гнучкістю та масштабованістю, MongoDB є чудовим вибором для обробки великих обсягів даних. MongoDB зберігає дані в JSON-подібних документах, що дозволяє легко інтегруватися з додатками на основі JavaScript. Мова запитів і можливості маніпулювання даними роблять її потужним інструментом для керування та пошуку даних у MERN-додатках [10].

Express.js - це фреймворк для web-додатків, який працює на сервері Node.js. За допомогою його функцій та утилі для створення веб-додатків та інтерфейсів прикладного програмування (API) можна спростити обробку HTTP-запитів, визначення маршрутів та управління проміжним програмним забезпеченням. Це дозволить швидко та ефективно створити надійний серверних додаток. Головна перевага Express.js заключається в легкості інтеграції з іншими компонентами стеку MERN і забезпеченні ефективної комунікації між інтерфейсом і бекендом [11].

React - це інтерфейсна частина стеку MERN, яка буде використана для розробки користувацького інтерфейсу (UI). Архітектура, на якій базується React, є компонентно-орієнтованою. Це дозволить створити та підтримувати багаторазові

компоненти інтерфейсу системи. React забезпечить віртуальну об'єктну модель документа (DOM), яка оптимізує продуктивність рендерингу, ефективно оновлюючи лише необхідні частини інтерфейсу. Декларативний синтаксис та ефективний рендеринг роблять React потужним інструментом для створення інтерактивних та динамічних веб-додатків [12].

Node.js - це поширене середовище JavaScript. Зокрема, це середовище виконання, що дозволить використовувати код JavaScript в різних місцях, в тому числі за межами веб-браузера. Він слугує бекенд-компонентом у стеку MERN. Node.js забезпечує керовану подіями, неблокуючу модель вводу/виводу, яка є дуже масштабованою та ефективною. Даний компонент стеку дозволить створювати серверні додатки за допомогою JavaScript, забезпечуючи спільне використання коду між інтерфейсом і бекендом. Node.js має широку екосистему модулів і пакетів, що дозволить легко інтегруватися з іншими технологіями та бібліотеками [13].

Серверний рівень - це другий рівень стеку MERN, нижче рівня React, і складається з Express.js та Node.js. Саме тут відбувається перенаправлення URL-адрес та HTTP-запити. Клієнтський рівень і рівень бази даних з'єднані цим рівнем, створюючи плавний перехід зверху вниз.

Останній рівень стеку MERN, рівень бази даних, зберігає дані вашого додатку за допомогою MongoDB. Цей рівень ефективно зберігає інформацію до моменту її отримання.

В цілому, стек MERN пропонує всеосяжну та згуртовану платформу для створення повноцінних JavaScript-додатків. Він дозволяє розробникам створювати сучасні, адаптивні та ефективні веб-додатки. Компоненти працюють гармонійно, поєднуючи свої сильні сторони для створення високопродуктивних додатків, здатних обробляти складні функціональні можливості [14].

Для створення проекту та перевірки кінцевого результату також необхідно використовувати наступні програмні продукти:

- Visual Studio Code для написання коду;
- Firefox для перевірки створених елементів та адміністрування БД;
- Insomniac для тестування JSON-запитів;

2 МОДЕЛЮВАННЯ WEB-ДОДАТКУ

2.1 Структурно-функціональне моделювання

Складна структура та швидка розробка веб-сервісів може призвести до появи вразливостей та недоліків, які можуть скомпрометувати конфіденційні дані. Виявлення таких вразливостей на ранніх стадіях циклу розробки гарантує, що їх буде легше і дешевше виправити. Структурне моделювання веб-додатків стає все більш важливим для виявлення та подальшого усунення таких недоліків [15].

Метод функціонального моделювання IDEF0 призначений для моделювання рішень, дій та діяльності організації або системи.

IDEF0 включає в себе як визначення мови графічного моделювання (синтаксис і семантика), так і опис комплексної методології розробки моделей.

IDEF0 використовується для відображення потоку даних, управління системою та функціонального потоку процесів життєвого циклу. IDEF0 здатна графічно представляти широкий спектр ділових, виробничих та інших видів операцій підприємства з будь-яким рівнем деталізації. Вона забезпечує строгий і точний опис, а також сприяє узгодженості використання та інтерпретації [16].

Контекстна діаграма процесу бронювання готелю через web-орієнтований сервіс представлена на рисунку 2.1.

Для контекстної діаграми було визначено такі дані:

- Вхідні дані: запит на бронювання послуг готелю.
- Вихідні дані: заброньований номер, оновлена історія замовлень
- Управління: список доступних номерів, правила бронювання, спектр послуг.
- Механізми: апаратне забезпечення, web-сервіс, технічне забезпечення, користувач.

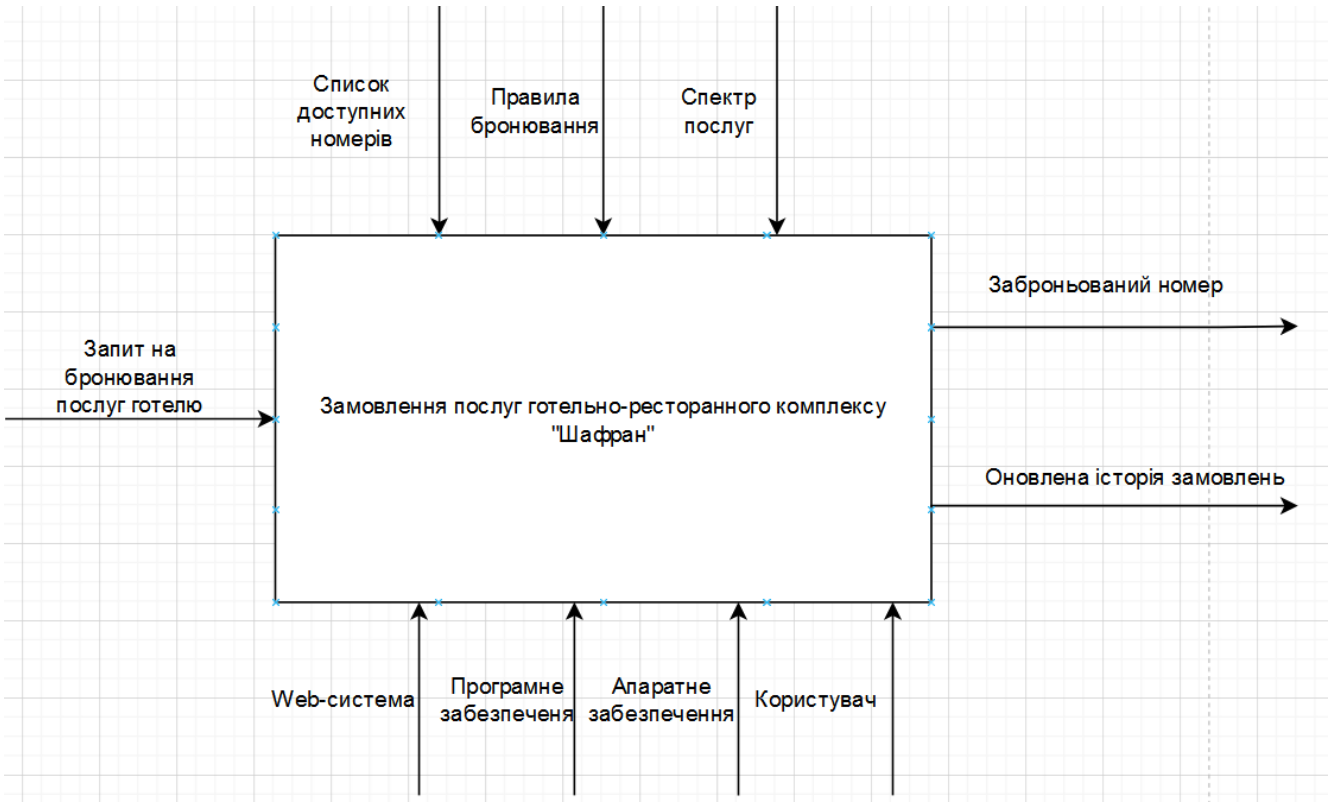

Рисунок 2.1 – Функціональна діаграма

Декомпозиція IDEF0 для деталізації внутрішніх процесів зображена на рисунку 2.2.

Декомпозиція діаграми оформлення замовлення товару представлена наступними підпроцесами:

- авторизація користувача;
- вибір номеру для бронювання;
- заповнення форми для бронювання;
- перегляд замовлення в особистому кабінеті;

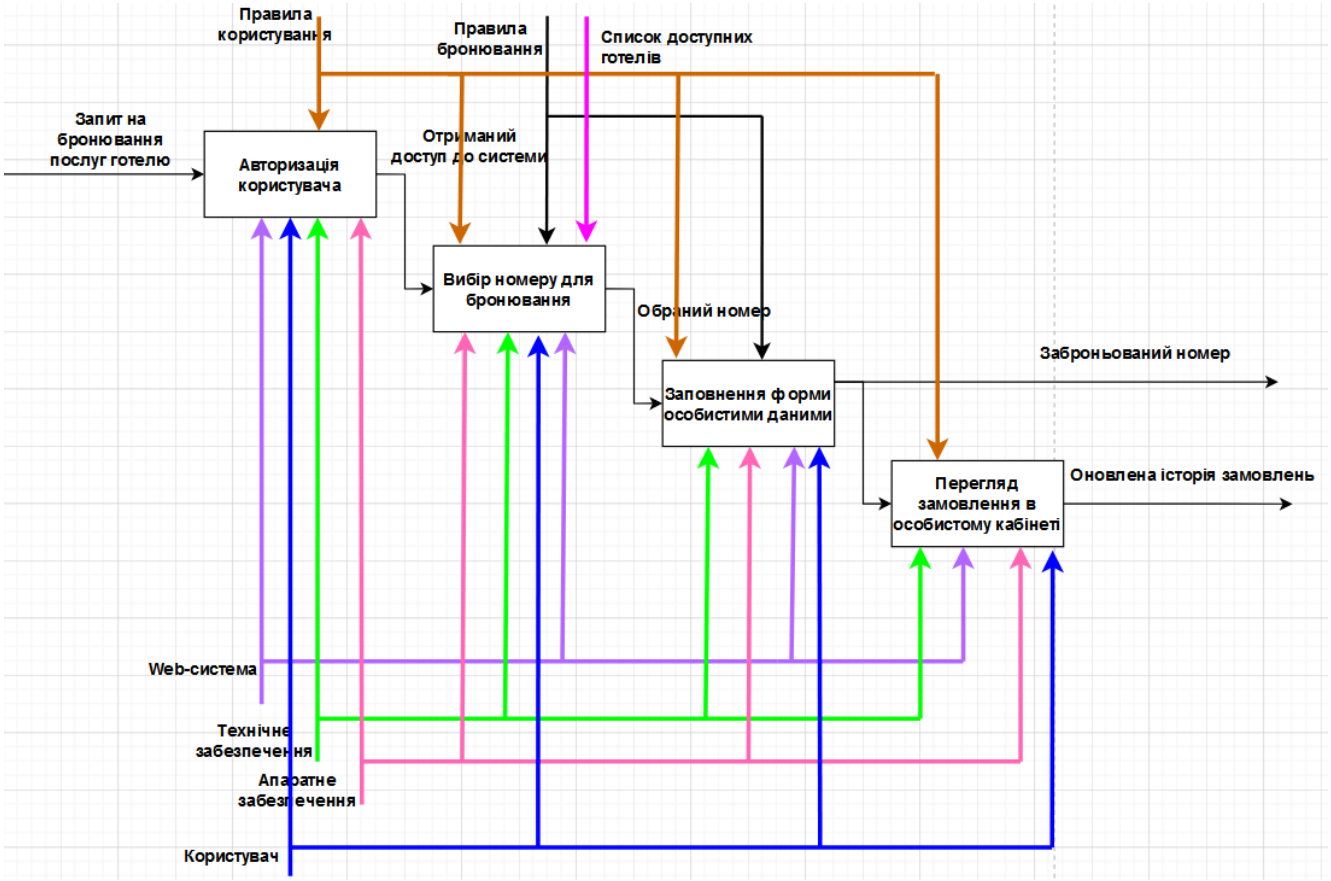

Рисунок 2.2 – Діаграма декомпозиції оформлення замовлення

2.2 Моделювання варіантів використання

При моделюванні варіантів використання, діаграма класів - це тип статичної структурної діаграми, яка описує структуру системи, показуючи класи системи, їхні атрибути, операції (або методи) та взаємозв'язки між об'єктами. Діаграма класів є основним будівельним блоком об'єктно-орієнтованого моделювання [17].

Однією з основних відмінностей між діаграмами класів та іншими типами діаграм є використання мови візуального моделювання. Діаграми класів використовуються для представлення статичних уявлень в додатках, в той час як UML в першу чергу використовується для візуального моделювання. Діаграма варіантів використання, діаграма послідовності та діаграма класів - це також три діаграми, з якими ви повинні бути знайомі. Діаграма UML представляє різні об'єкти, атрибути, операції та зв'язки системи. Подібно до діаграм класів, діаграми UML показують динамічний стан системи [18].

Для web-сервісу підтримки замовлення послуг готельно-ресторанного бізнесу варіанти використання є наступними:

- перегляд інформації;
- перегляд доступних готелів;
- відправка зворотньої форми;
- оформлення бронювання;
- tracking бронювання;
- додавання, редагування та видалення інформації.

Акторами на use-case діаграмі є адміністратор, користувач, сервер та база даних. Діаграма варіантів використання web-орієнтованої системи замовлення послуг готельно-ресторанного бізнесу представлена на рисунку 2.3.

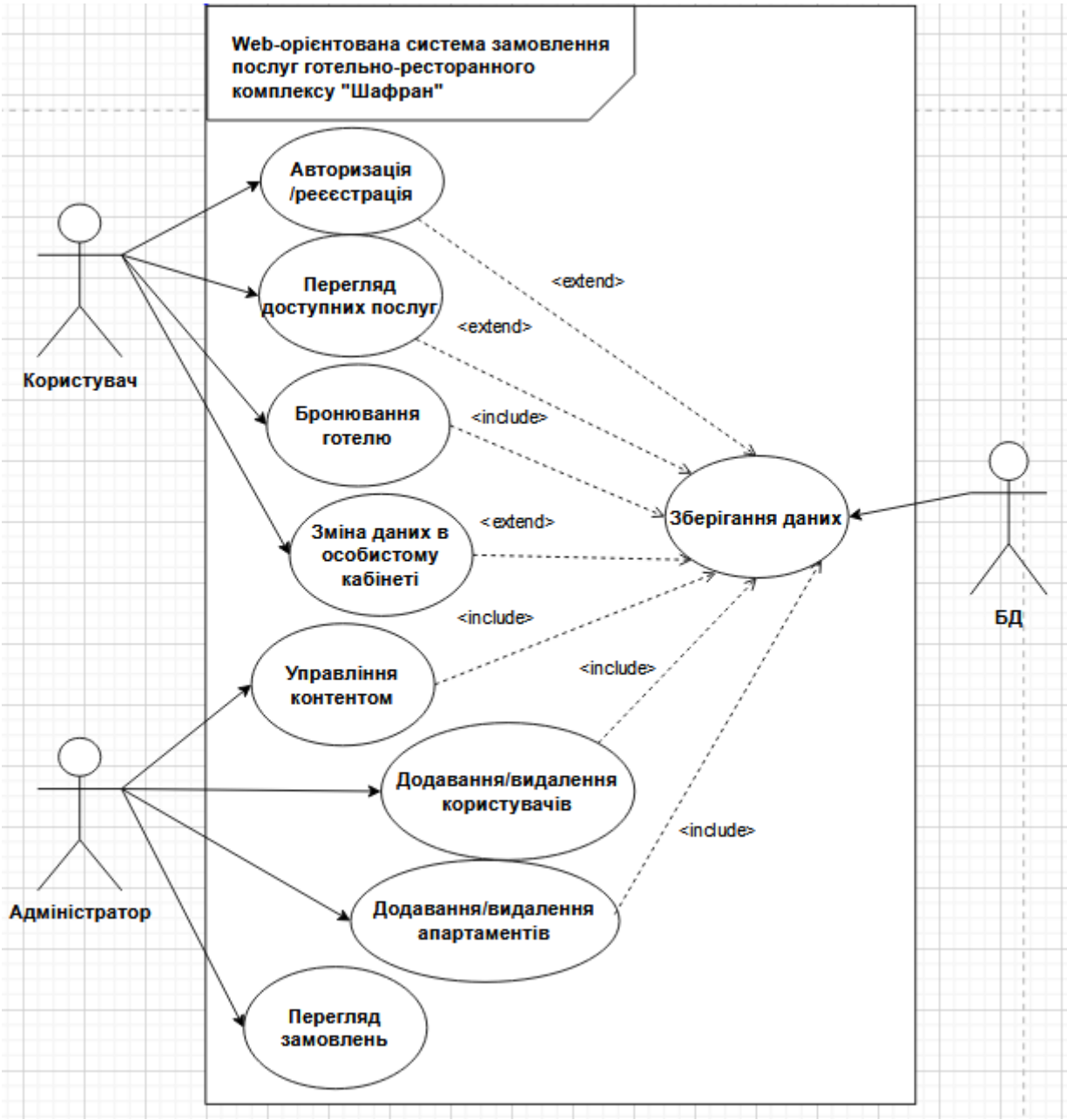

Рисунок 2.4 – Діаграма варіантів використання

2.3 Проектування бази даних

MongoDB - це документно-орієнтована база даних, яка зберігає дані у гнучких документах у форматі JSON, а не в рядках і стовпцях, як традиційні бази даних SQL.

В порівнянні з іншими БД, гнучкість схеми даних і горизонтальна масштабованість MongoDB роблять її ефективною для проектів, що мають справу

з різноманітними структурами даних і великими обсягами інформації. Вбудована реплікація та швидкий доступ до даних роблять її привабливим вибором для сучасної веб-розробки та інших великомасштабних додатків [19].

Ось деякі ключові переваги використання MongoDB:

1. Гнучка модель даних - поля можуть змінюватися від документа до документа, що дозволяє зберігати різні типи даних в одній колекції без дорогої міграції схем.
2. Висока продуктивність - MongoDB була розроблена для забезпечення високої доступності та масштабованості. Вона використовує індексування для підтримки швидких запитів і шардинг для розподілу даних між серверами.
3. Розширені запити - підтримує гнучкі, потужні конструкції запитів для пошуку та агрегування даних. Запити можуть повертати певні поля, а також включати функції JavaScript.
4. Висока доступність - Вбудована реплікація забезпечує автоматичне обхід відмов і резервування. Дані реплікуються на декілька серверів.
5. Масштабованість - Горизонтальне масштабування дозволяє легко розширювати ємність бази даних шляхом додавання нових серверів.

В документо-орієнтованій системі MongoDB центральним поняттям є документ, який зберігає інформацію. В документі опису номеру надано атрибути та типи збереження даних з унікальним номером.

```
{"_id":{"$oid":"65400ad0467abf200c072e18"},
"name":"Room1",
"type":"single_room",
"city":"ukraine",
"address":"somewhere",
"distance":"500",
"photos":[],
```

"title":"First Numb of Room",

"desc":"Roomdesc",

"rooms":[],

"cheapestPrice":{"$numberInt":"100"},

"featured":false,"__v":{"$numberInt":"0"}}

MongoDB використовує гнучку структуру документів у форматі BSON (двійковий JSON). На відміну від традиційних реляційних баз даних, де схема даних фіксована і строго визначена, MongoDB дозволяє документам у колекції мати різні поля. Це дозволяє зберігати дані без необхідності створювати структури таблиць або визначати зв'язки [20].

Особливість БД даної системи заключається в можливості додавання нових атрибутів та даних до існуючих. В документі про послуги для номерів описано наступним чином:

{
  "_id": {"$oid": "65401a6bff18d48a3c1ab8ab"},
  "serviceName": "Spa Package",
  "description": "Relax and rejuvenate with our
exclusive spa package",
  "price": {"$numberInt": "50"},
  "createdAt": {"$date": {"$numberLong":
"1698701103000"}},
  "updatedAt": {"$date": {"$numberLong":
"1698701103000"}},
  "__v": {"$numberInt": "0"}
}

Також, дану базу даних можна легко розширювати по горизонталі. Це означає, що дані можуть бути розподілені між декількома серверами для забезпечення високої доступності та підтримки великих обсягів даних. Систему можна розширювати, додаючи нові сервери без значного впливу на продуктивність.

Кожен документ являє собою набір ключ-значення. В даному документі _id це ключ а ObjectId – значення.

{"_id":{"$oid":"65401729ff18d48a3c1ab8a7"},
"title":"2 bedroomroom",
"price":{"$numberInt":"250"},
"maxPeople":{"$numberInt":"3"},
"desc":"King size bed, 1 bathroom, 1 wifi",
"roomNumbers":[{"number":{"$numberInt":"2"},
"unavailableDates":[],
"_id":{"$oid":"65401729ff18d48a3c1ab8a8"}},
{"number":{"$numberInt":"4"},
"unavailableDates":[],
"_id":{"$oid":"65401729ff18d48a3c1ab8a9"}}],
"createdAt":{"$date":{"$numberLong":"1698699049997"}},
"updatedAt":{"$date":{"$numberLong":"1698699049997"}},
"__v":{"$numberInt":"0"}}

Рисунок 2.6 – Документ збереження інформації про апартаменти

Також для зберігання інформації про користувачів було створено документ «users» який представлено нижче.

{"_id":{"$oid":"653edf5f616d2abf82680dbe"},
"username":"Stankevich",
"email":"stankevich1@gmail.com",
"password":"$2a$10$mocon4Q14CSCzKccmo6.DA.NiohtcUj/8Ji
J.gvUa/fQhV.ooC2",
"isAdmin":true,

"createdAt":{"$date":{"$numberLong":"1698619231656"}},
"updatedAt":{"$date":{"$numberLong":"1698619231656"}},
"__v":{"$numberInt":"0"}}

MongoDB пропонує різноманітні варіанти формулювання запитів. На додаток до стандартних операцій пошуку, критерії пошуку можуть бути визначені за допомогою JavaScript-подібних виразів. Це дозволяє легко маніпулювати даними і витягувати потрібну інформацію з бази даних.

Також MongoDB підтримує індекси для підвищення продуктивності пошукових операцій. Для різних полів документа можна створювати різні індекси, що полегшує оптимізацію запитів і підвищує продуктивність системи.

Гнучкість бази даних MongoDB досягається завдяки її документно-орієнтованому підходу та здатності зберігати дані різних структур в одній колекції.

Щоб додати нову послугу (наприклад, SPA-пакет), необхідно створити новий документ у відповідній колекції, де зберігається інформація про послугу. Новий документ може містити назву послуги, опис, ціну та інші необхідні атрибути. Важливо зазначити, що структура інших документів у колекції не може бути змінена.

Щоб видалити послугу, необхідно видалити відповідний документ у колекції послуг. MongoDB надає корисні оператори, такі як db.collection.deleteOne() і db.collection.deleteMany().

Також MongoDB дозволяє легко адаптувати схему даних до подібних систем. Кожен документ може мати власну структуру. Наприклад, якщо різні номери готелю мають різні специфікації, кожен документ може містити різні поля. Можливість динамічного додавання нових полів до документа дозволяє легко адаптувати схему до потреб кожного готелю та системи [21].

Загалом, MongoDB може допомогти задовольнити потреби у зберіганні та управлінні даними сучасних веб- та мобільних додатків, створених за допомогою гнучких методологій.

3 ПРАКТИЧНА РЕАЛІЗАЦІЯ WEB-ОРІЄНТОВАНОЇ СИСТЕМИ

3.1 Архітектура web-орієнтованої системи

Архітектура MERN дозволяє легко побудувати 3-рівневу архітектуру (фронтенд, бекенд, база даних) повністю за допомогою JavaScript та JSON [22]. На рисунку 3.1 представлено вигляд архітектури web-орієнтованої системи.

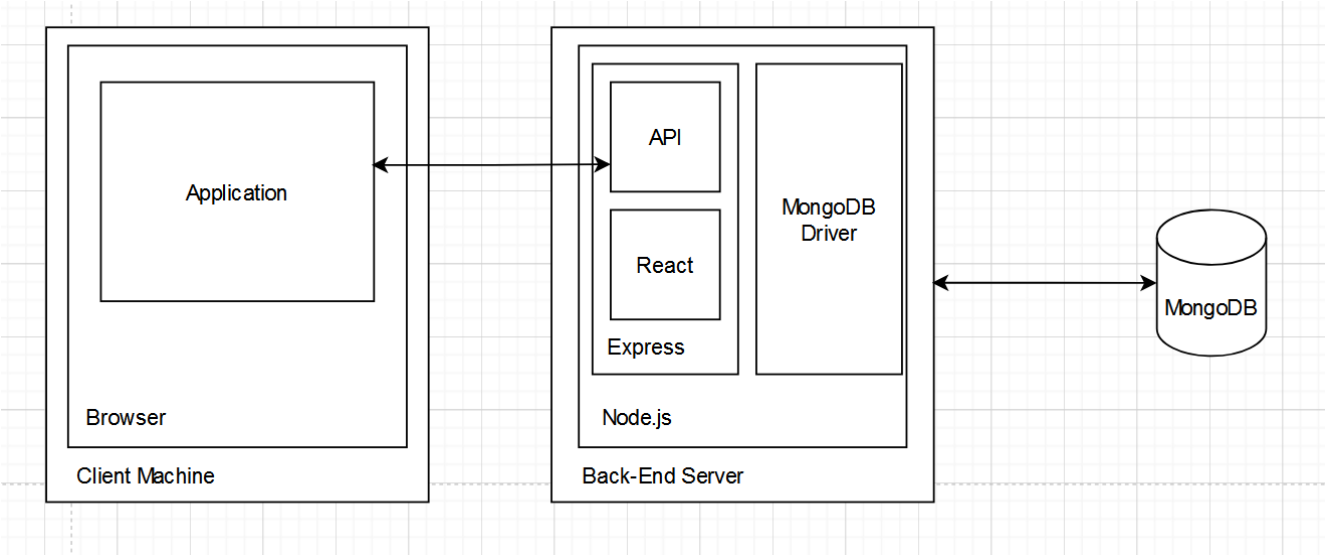

Рисунок 3.1 – Вигляд архітектури web-орієнтованої системи

Архітектура web-орієнтованої системи базується на типовій моделі MVC. Клієнтський рівень (View) буде створений за допомогою Javascript, HTML та CSS з використанням ReactJS як фреймворку. Цей рівень архітектури - це те, з чим користувач буде взаємодіяти, щоб отримати доступ до функцій web-орієнтованої системи.

Рівень бізнес-логіки (контролер) буде написаний з використанням NodeJs та ExpressJS, і цей рівень представляє собою сервер додатків, який буде виступати в якості моста зв'язку для клієнтського рівня та рівня бази даних [23]. Цей рівень буде обслуговувати HTML-сторінки на пристрої користувача, приймати HTTP-запити від користувача та надавати відповідну відповідь (рис 3.2).

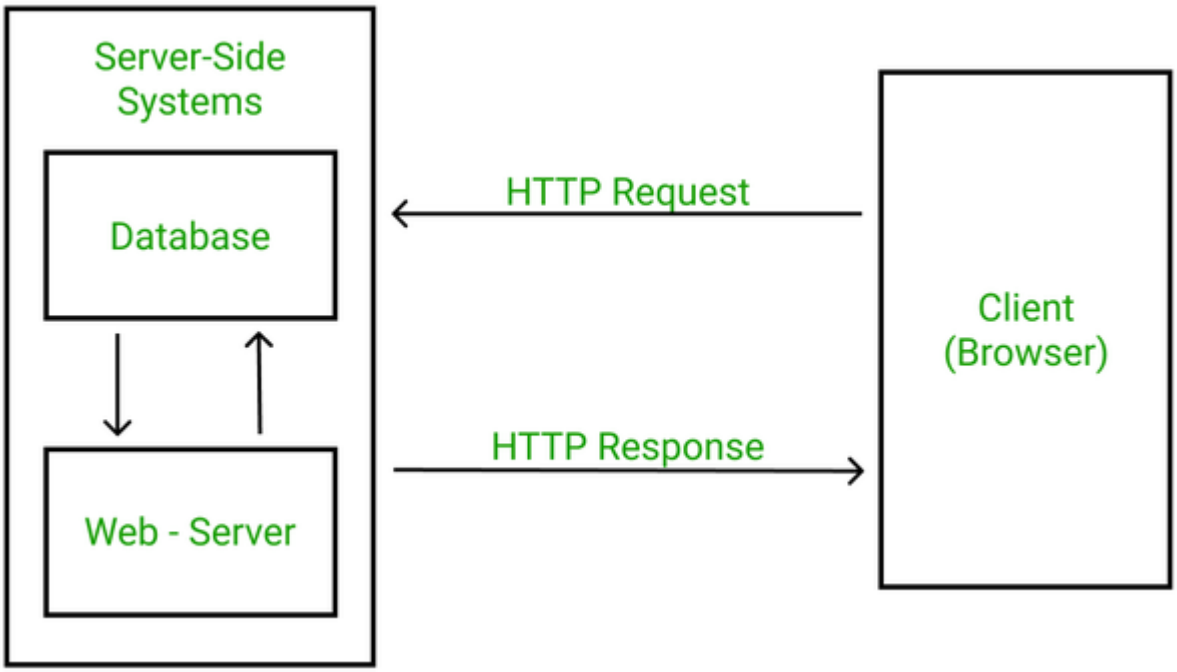

Рисунок 3.2 – Приклад виконання HTTP запитів

Рівень бази даних (модель) буде хостити MongoDB . Тут зберігаються всі важливі дані, необхідні для роботи нашого додатку.

Для використання MongoDB необхідно скористатися хмарним середовищем Cloud MongoDB. При створенні бази даних користувач отримує унікальний ключ, який необхідно використати при підключенні до серверної частини.

На рисунку 3.3 представлено результат підключення бази даних до серверної частини.

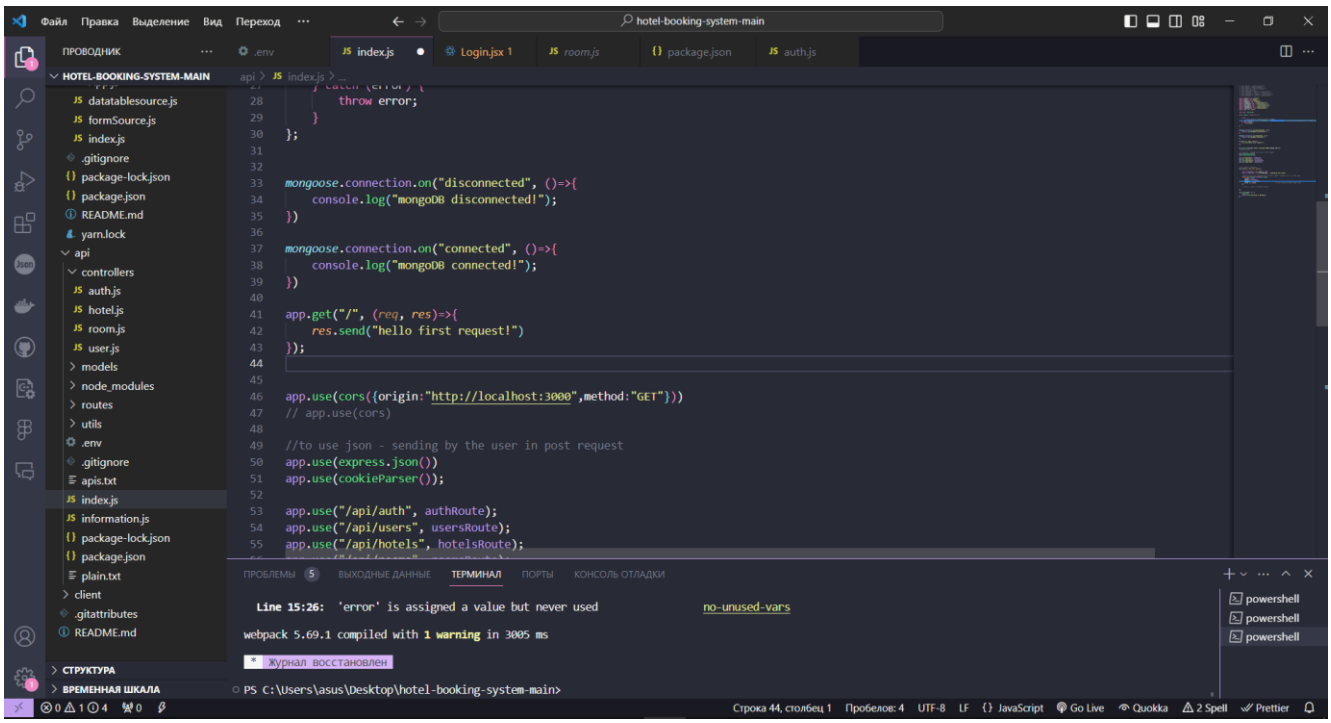

Рисунок 3.3 – Результат підключення бази даних до серверної частини

Після успішного підключення до серверної частини, через термінал програмного додатку Visual Studio Code сервер надає повідомлення (рис. 3.4).

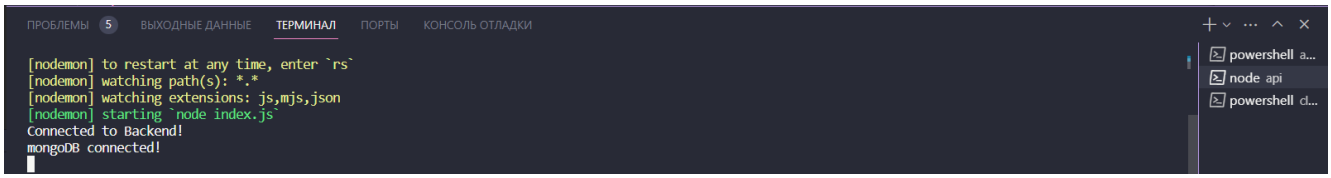

Рисунок 3.4 – Повідомлення про успішне підключення

3.2 Реалізація серверної частини

За допомогою стеку технологій MERN необхідно використовувати Express та Node.js, які також написані на JavaScript, для створення бек-енду. Вони виконують такі завдання, як обробка даних, обробка запитів користувачів та забезпечення безперебійної роботи.

Для початку необхідно встановити Node.js. Для цього необхідно перейти за посиланням https://nodejs.org/en/ і завантажити версію LTS або поточну версію.

Після цього необхідно створити проект за допомогою Node.js. Для цього необхідно відкрити термінал та створити новий каталог для проекту Node.js, виконавши наступну команду:

```
mkdir myProject && cd myProject
```

Усередині каталогу "myProject" необхідно створити ще один каталог під назвою "server" і перейти до нього за допомогою наступної команди:

```
mkdir server && cd server
```

Тепер необхідно ініціалізувати новий проект Node.js у каталозі "server", виконавши наступну команду:

```
npm init -y
```

В результаті буде створено файл package.json, який зберігає інформацію про додаток та його залежності.

Після цього необхідно встановити необхідні залежності для проекту Node.js за допомогою наступної команди:

```
npm install express cors dotenv
```

Express: Встановлює веб-фреймворк для Node.js, який спрощує розробку.

Cors: Дозволяє розподіляти ресурси між різними джерелами, дозволяючи вашому серверу обробляти запити з різних доменів.

Dotenv: Допомагає завантажувати змінні оточення з файлу .env, зберігаючи конфігурацію окремо від коду.

Після цього необхідно створити файл з назвою server.js і написати наступний код.

```
// server.js
const express = require('express');
const bodyParser = require('body-parser');
const cors = require('cors');
```

```
const app = express();
const port = 3000;

app.use(bodyParser.json());
app.use(cors());

app.listen(port, () => {
  console.log(`Server is running on port ${port}`);
});
```

Сервер починає прослуховування на вказаному порту (змінна оточення PORT або порт 5000).

Функція dbo.connectToServer встановлює з'єднання з базою даних MongoDB під час запуску сервера.

Повідомлення про те, що сервер запущено і він слухає на вказаному порту, виводиться в консоль.

Загалом, цей код встановлює базовий сервер Express.js, вмикає CORS, розбір JSON і підключення до бази даних MongoDB. Тепер сервер надає можливість почати визначати маршрути та обробляти різні HTTP-запити у файлі record.js, щоб створити свій серверний додаток.

Після того, як було налаштовано сервер, необхідно створити моделі, для збереження інформації в базі даних. На рисунках 3.5 – 3.7 представлено результат створення моделей бази даних MongoDB.

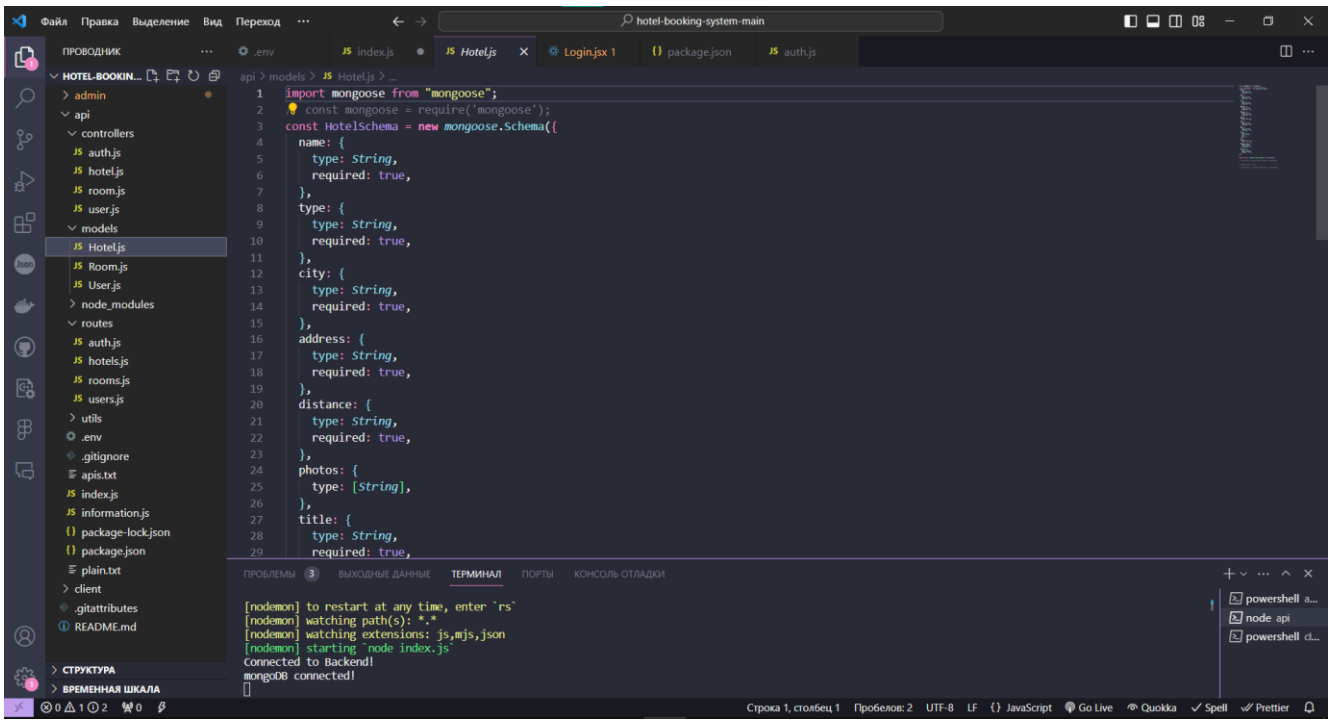

Рисунок 3.5 – Результат створення моделі «Hotel.js»

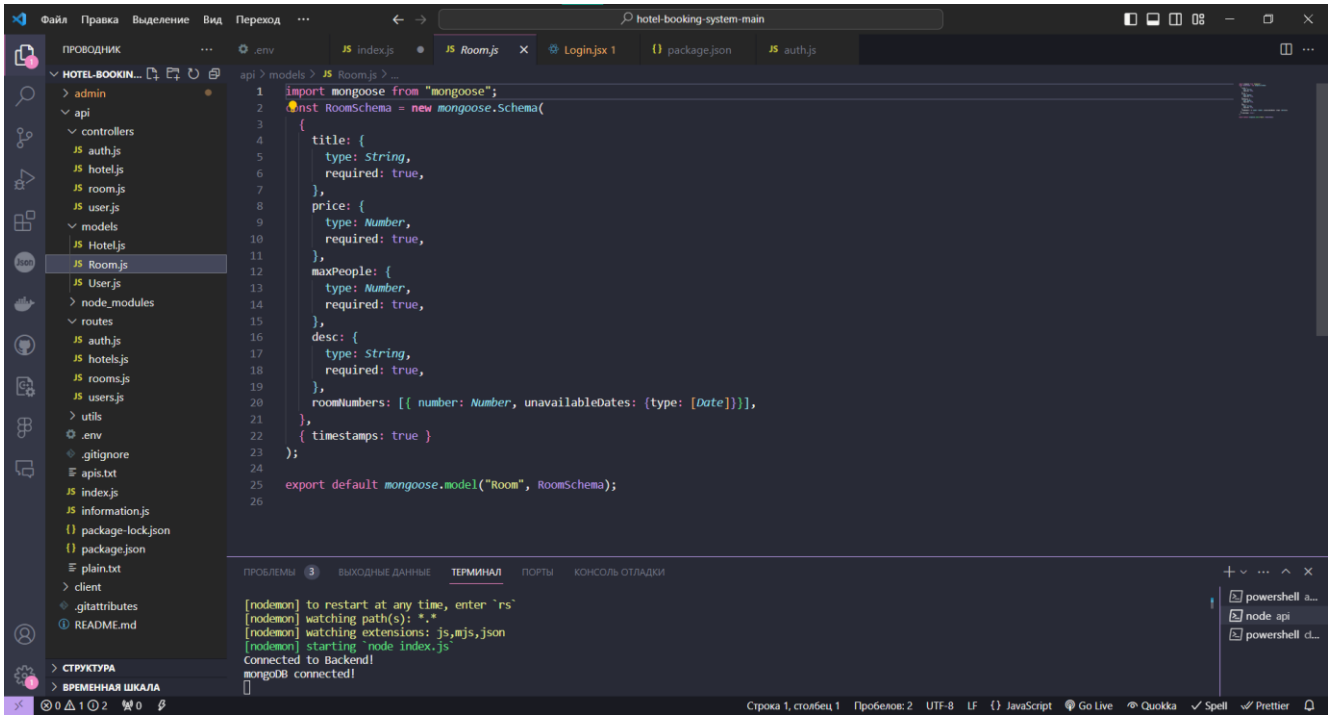

Рисунок 3.6 – Результат створення моделі «Room.js»

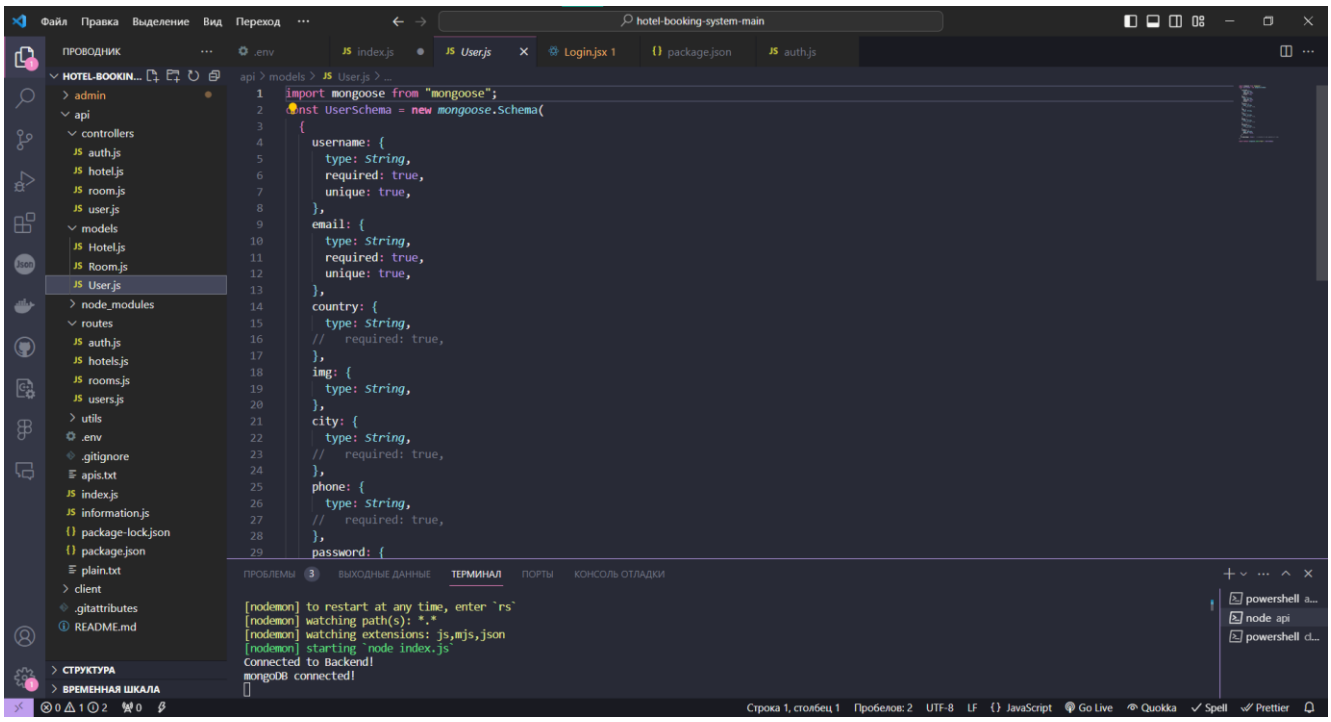

Рисунок 3.7 – Результат створення моделі «User.js»

Після цього необхідно створити контролери. Контролери - це те, що безпосередньо відповідає на кожен HTTP-запит, який надходить до додатку, тому кожен веб-запит призводить до створення (якщо він маршрутизований) нового екземпляра контролера (класу). На рисунку 3.8 представлено створений контролер серверної сторони про користувачів. Повний лістинг коду user.js представлено в додатку Б.

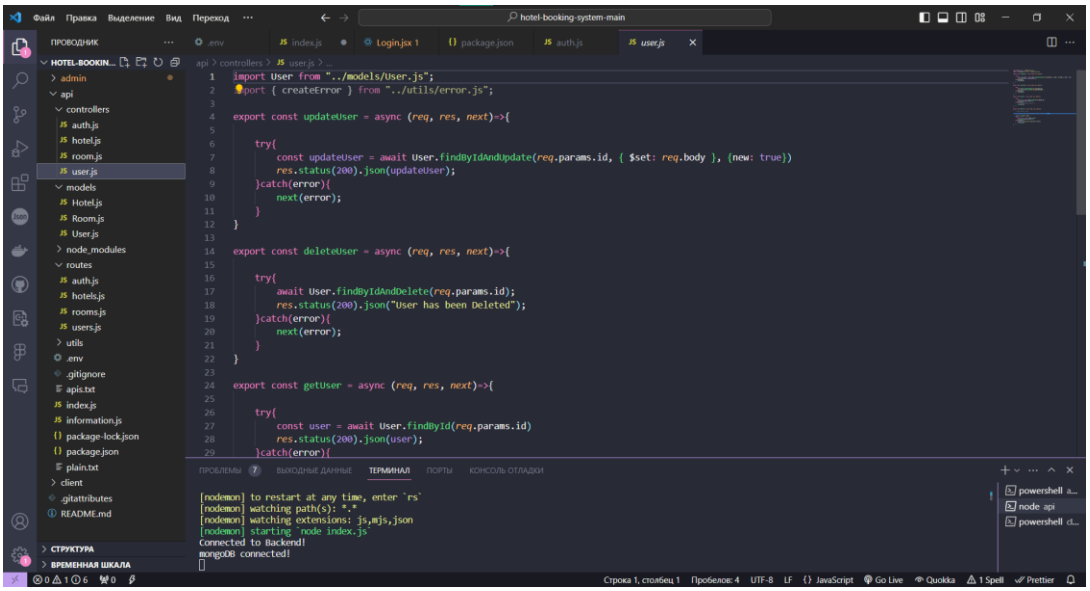

Рисунок 3.8 – Створений контролер користувачів

Також для даного контролеру було створено роутер для встановлення URL адреси. На рисунку 3.9 представлено результат створення роутеру для контролеру «user.js»

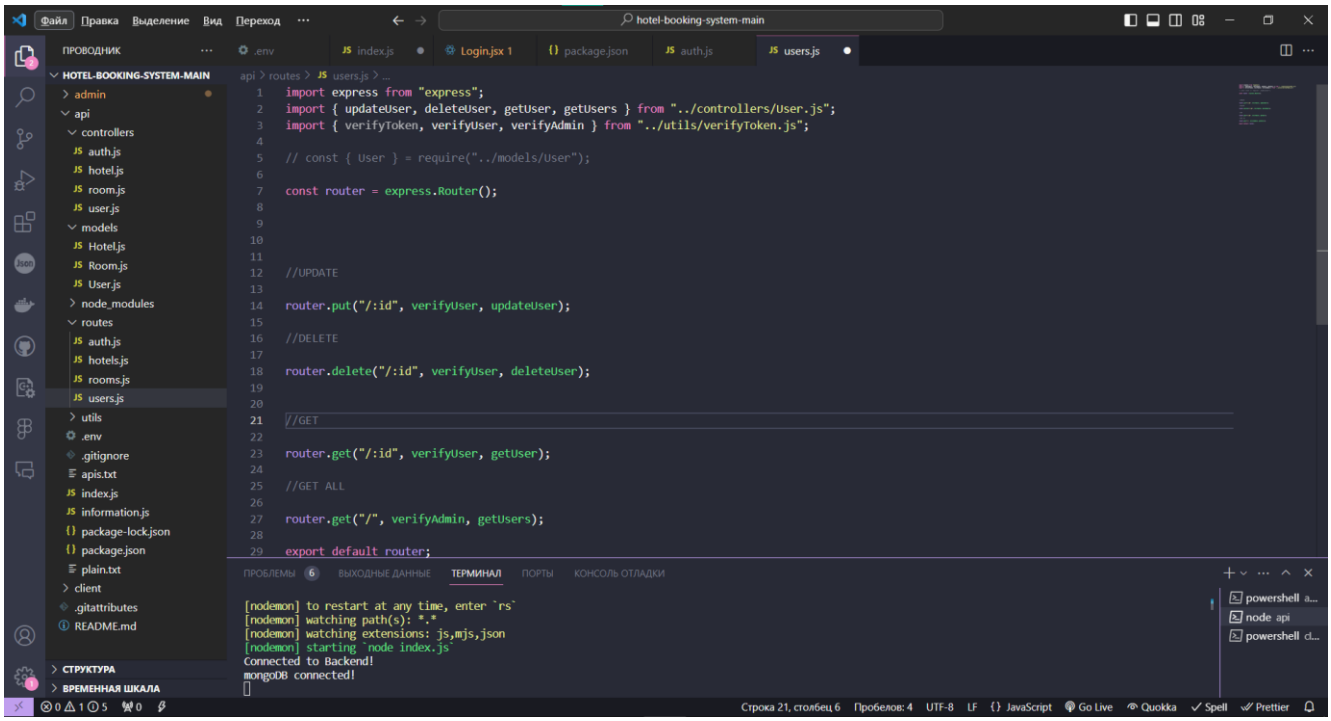

Рисунок 3.9 – Результат створення роутеру для контролеру «user.js»

Також було встановлено всі залежності за допомогою пакетного менеджеру npm. Результат представлено на рисунку 3.10.

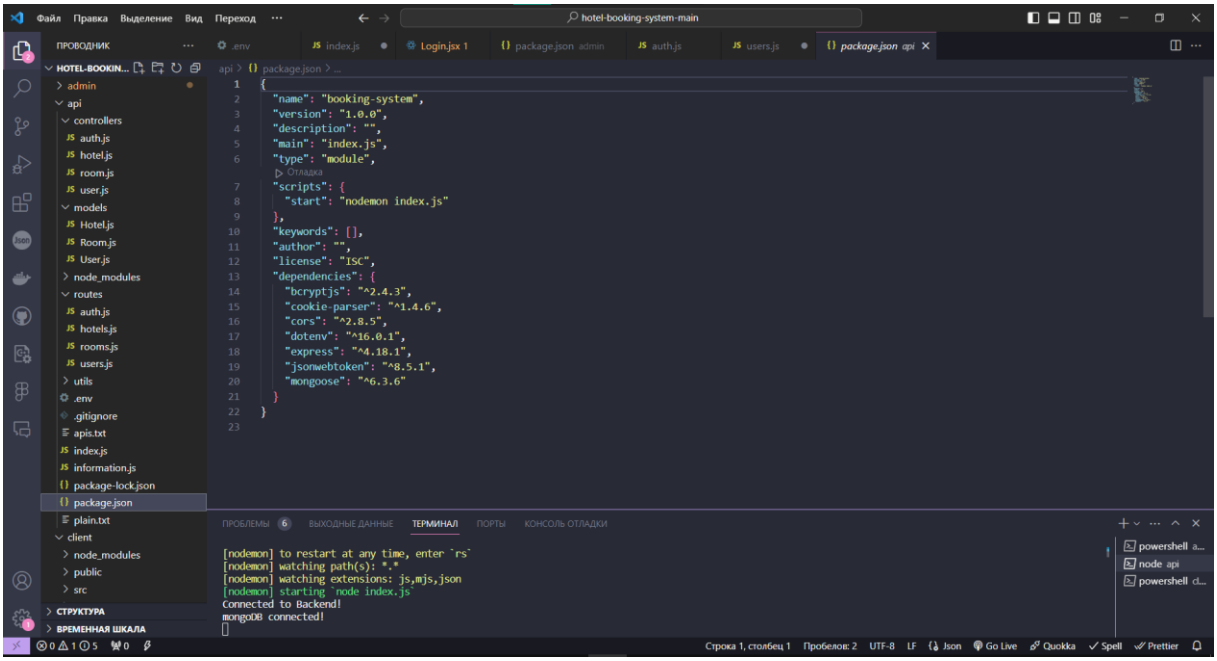

Рисунок 3.10 – Результат встановлення всіх залежностей

За аналогічним сценарієм було створено клієнтську частину web-орієнтованої системи. Для відображення навігаційного меню було створено шапку та підвал web-системи. На рисунках 3.11 – 3.12 представлено результат створення «Header.jsx» та «Footer.jsx». Повний лістинг коду «Header.jsx» та «Footer.jsx» наведено в додатку В.

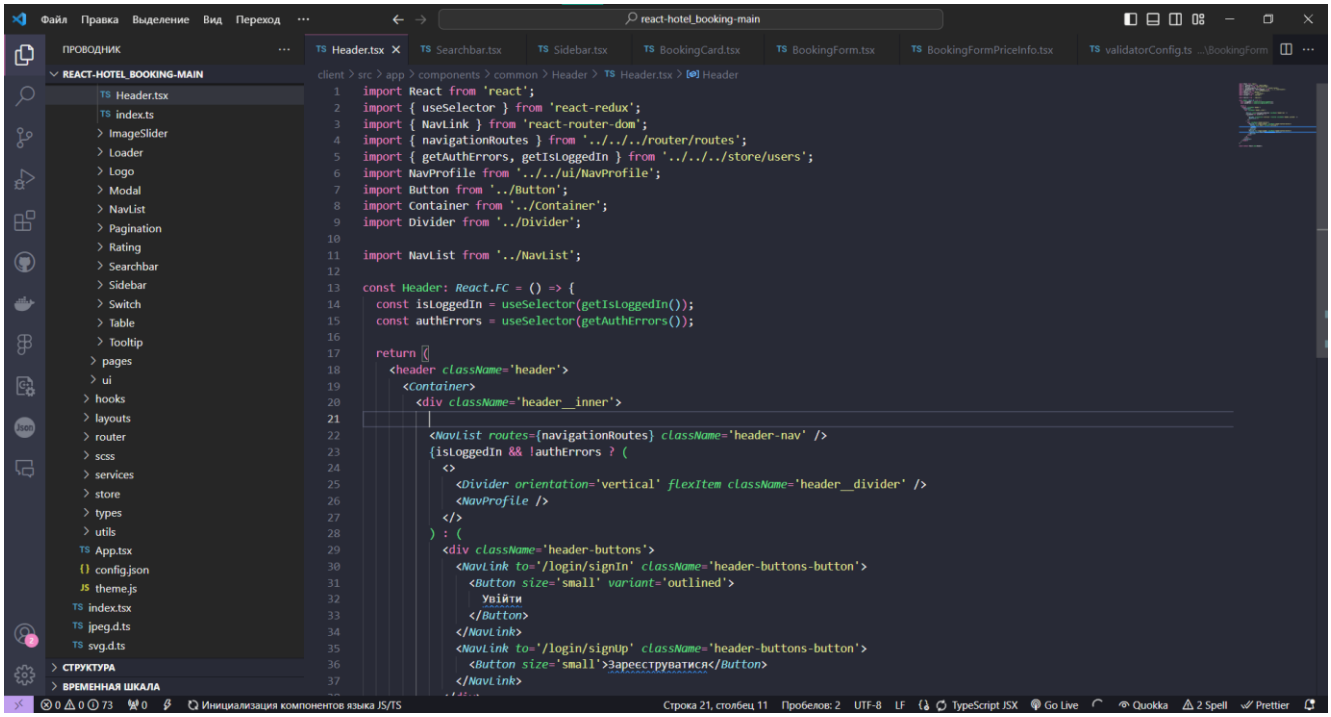

Рисунок 3.11 – Результат створення «Header.jsx»

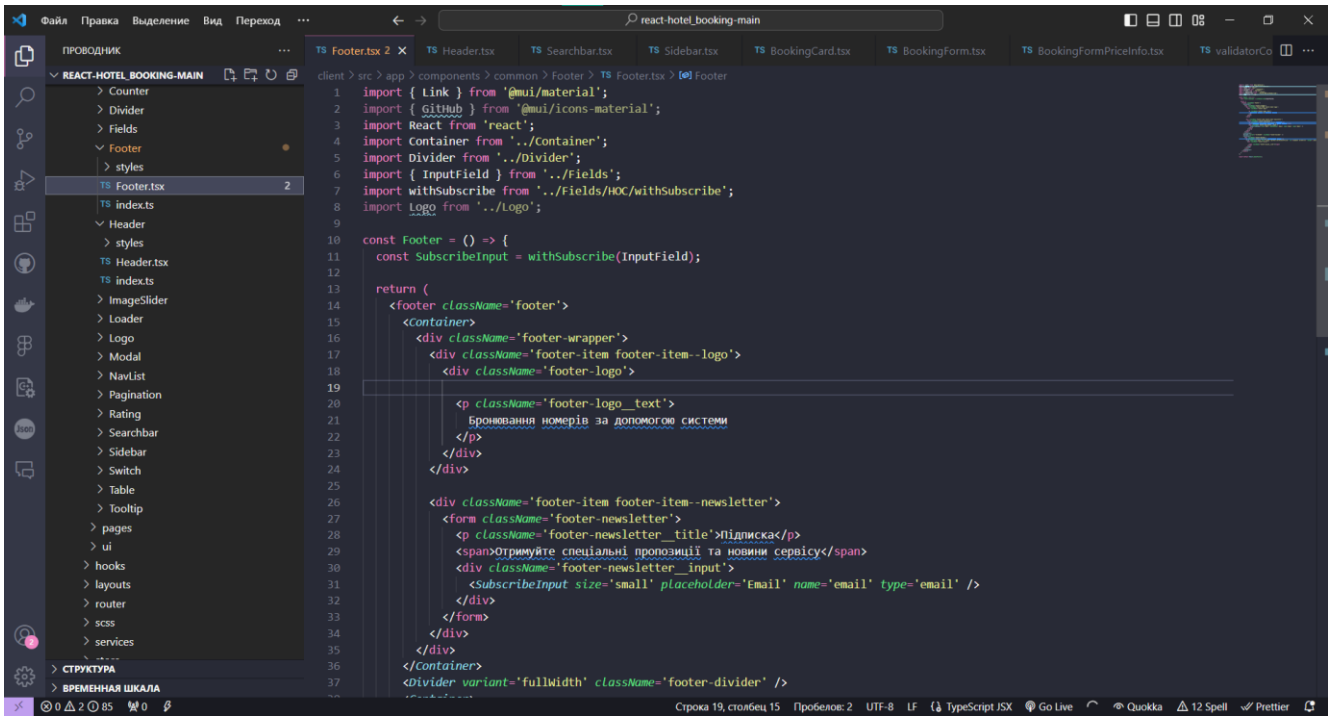

Рисунок 3.12 – Результат створення «Footer.jsx»

3.3 Настанови з використання

Для того щоб отримати доступ до системи бронювання готельно – ресторанних послуг, необхідно виконати операцію реєстрації. На рисунках 3.13 – 3.14 представлено вигляд сторінок реєстрації та авторизації. Кожного користувача база даних зберігає та надає роль «user».

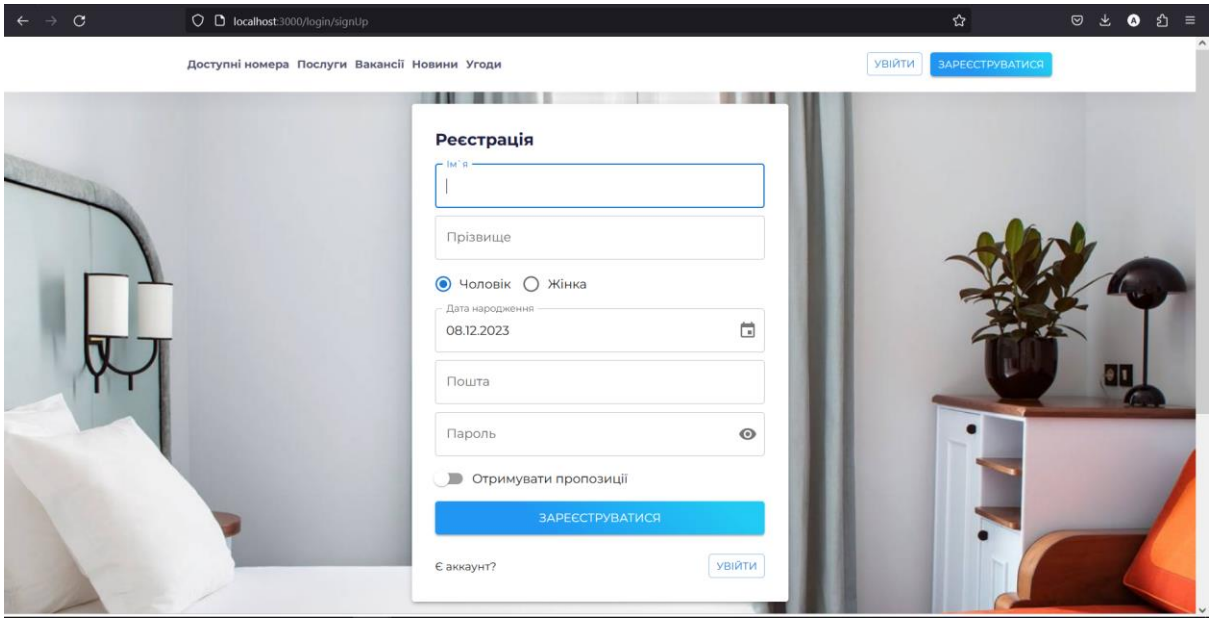

Рисунок 3.13 – Вигляд сторінки реєстрації

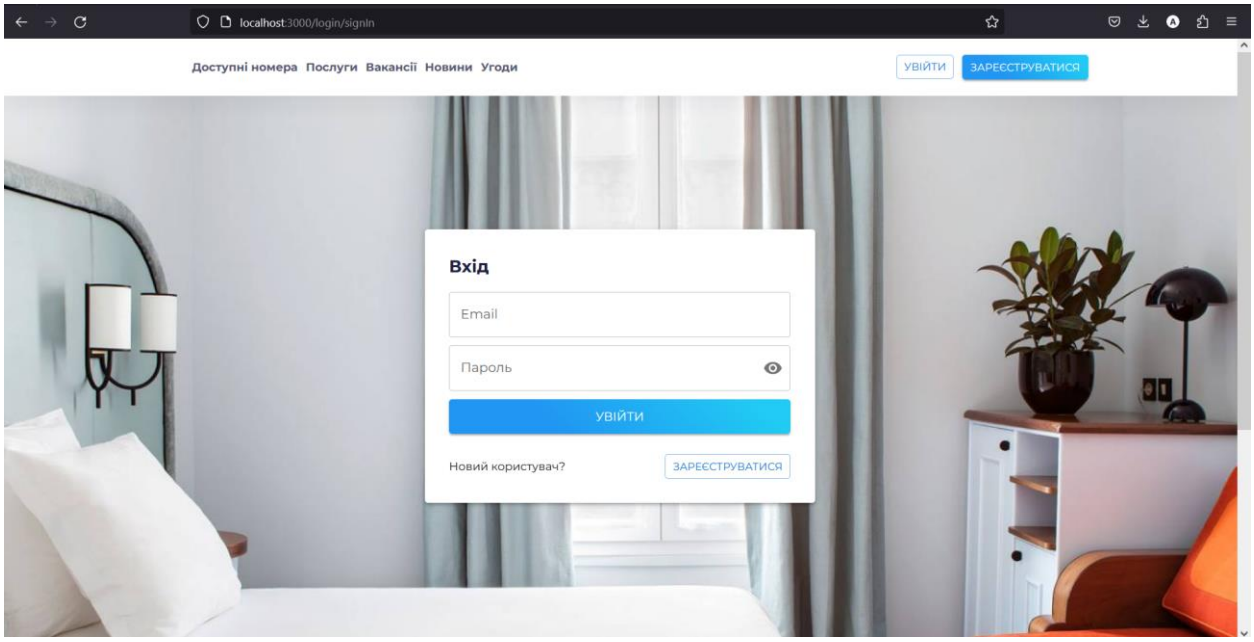

Рисунок 3.14 – Вигляд сторінки авторизації

Неавторизований користувач має право тільки на перегляд доступних послуг web-орієнтованої системи. Після успішної авторизації користувача буде успішно переадресовано на головну сторінку web системи. На рисунку 3.15 представлено вигляд головної сторінки web-системи.

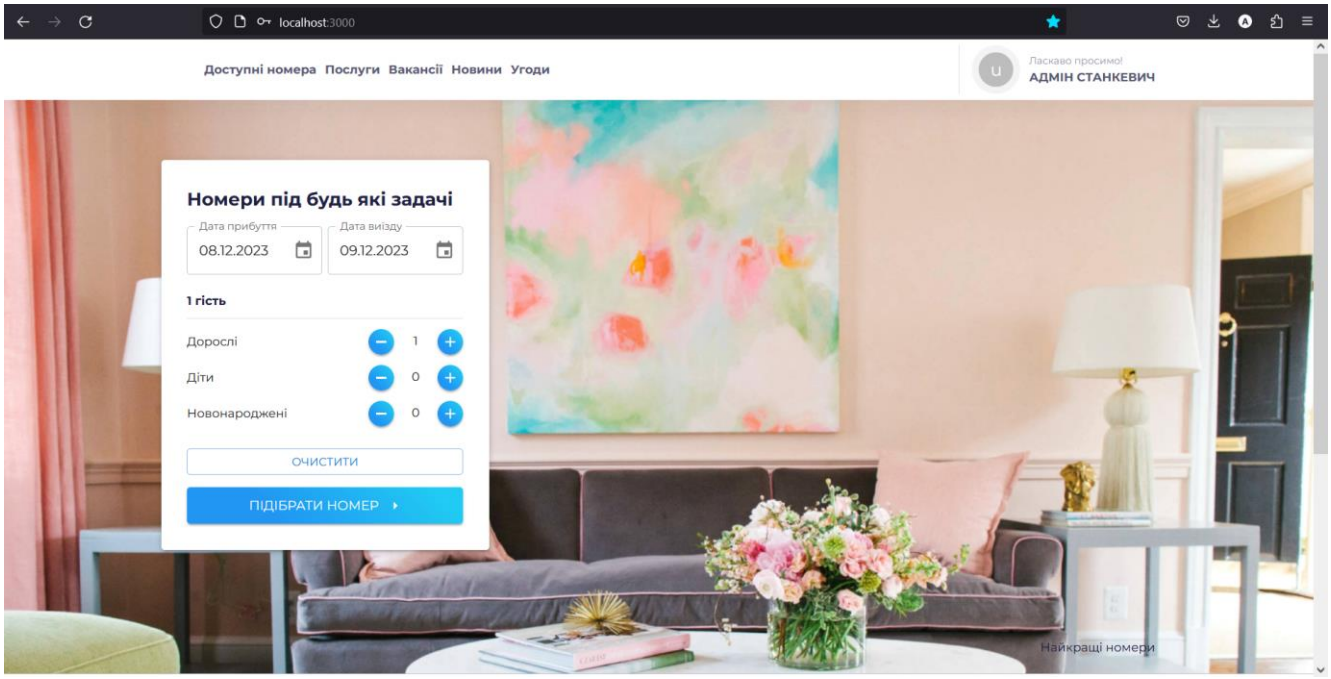

Рисунок 3.15 – Вигляд головної сторінки

На головній сторінці користувач може використати систему підбору номерів готелю за допомогою системи фільтрації. Після вибору даних, користувач повинен натиснути на кнопку «Підібрати номер» і після цього система надасть всі доступні готелі, які потрапляють під критерій користувача. На рисунку 3.16 представлено результат генерації доступних готелів на основі запиту користувача.

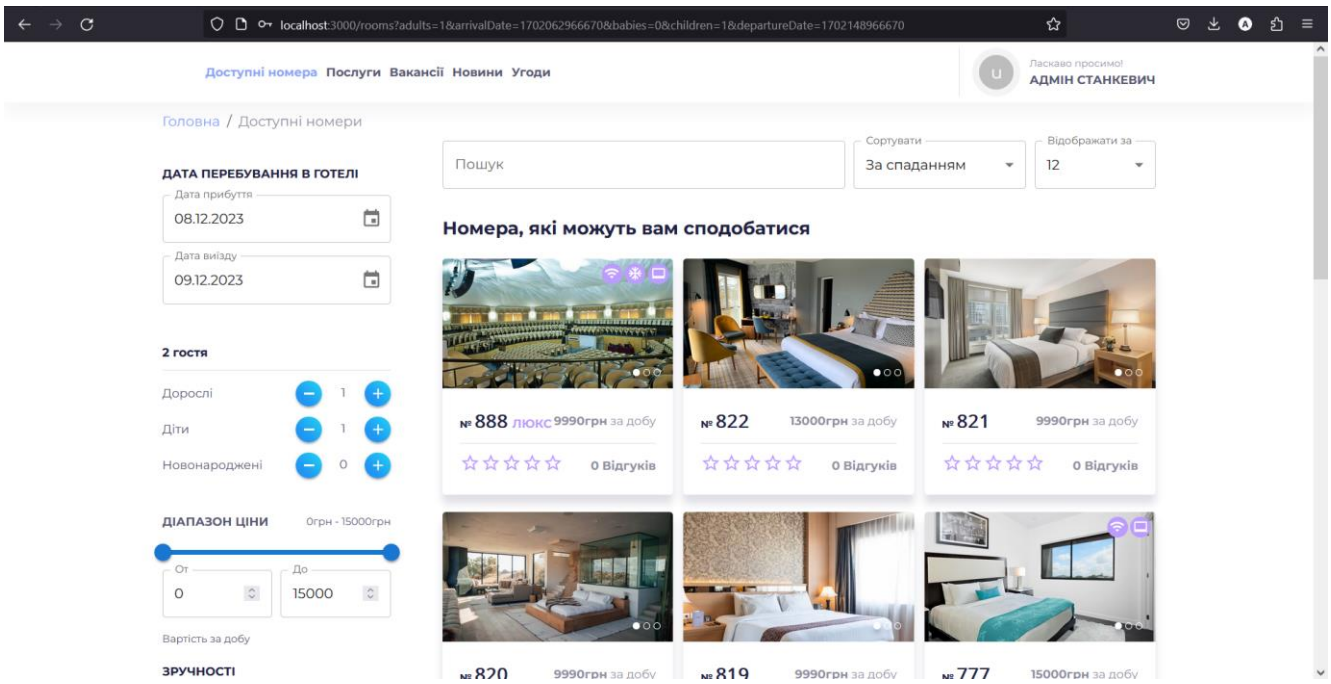

Рисунок 3.16 – Результат генерації доступних готелів на основі запиту користувача

Створена система фільтрації доступних номерів та апартаментів надає можливість по наданим критеріям формувати запит для результату, який задовольнить користувача. На рисунку 3.17 представлено вигляд системи фільтрації для підбору номеру та апартаментів.

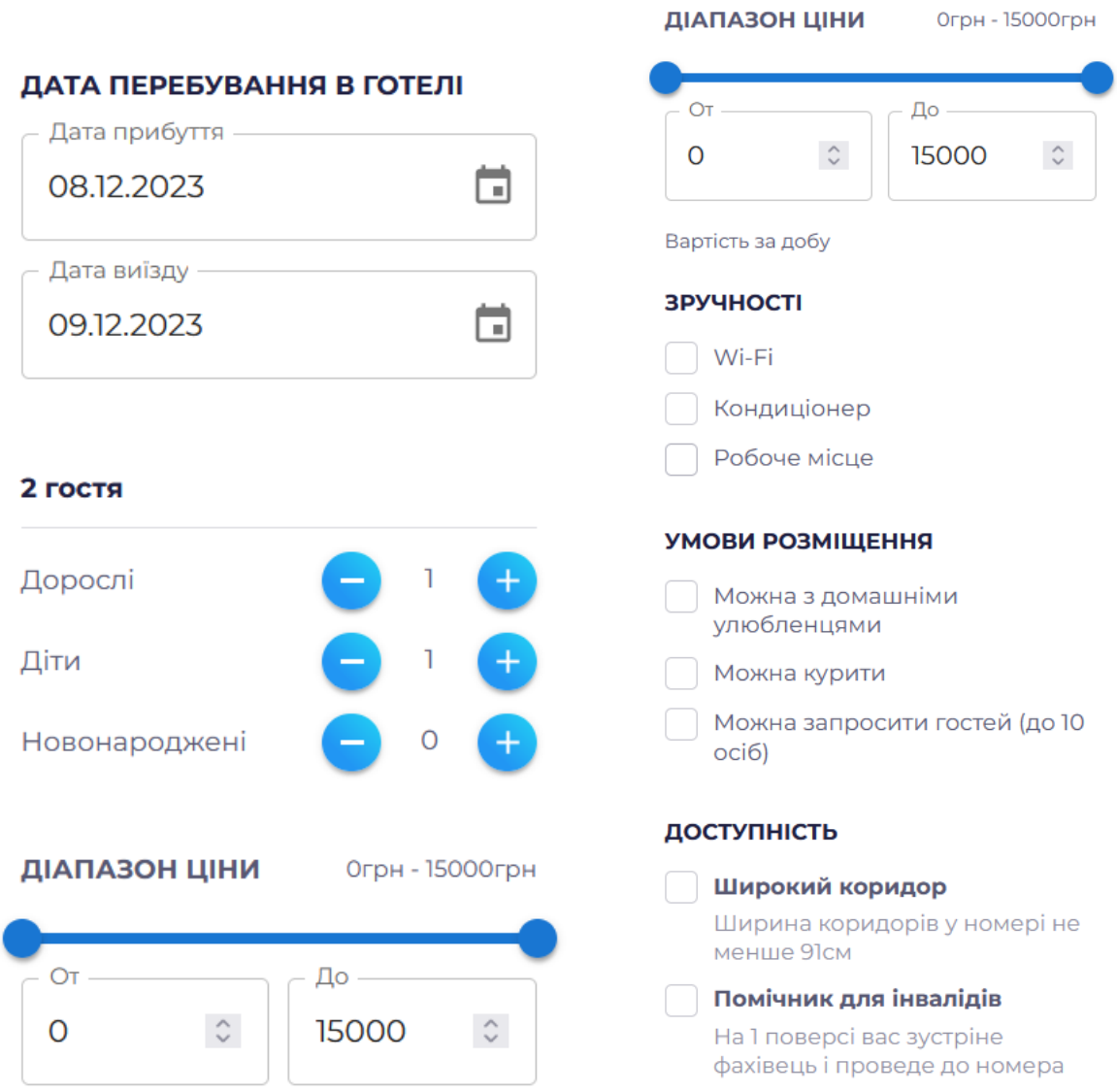

Рисунок 3.17 – Вигляд системи фільтрації

Після підбору номерів та апартаментів, користувач може перейти на сторінку обраного номеру. Наприклад, користувач хоче замовити актову залу для власного концерту. При переході на сторінку обраного номеру, користувач може ознайомитись з доступною інформацією про цей номер. На рисунку 3.18 представлено вигляд сторінки обраного номеру.

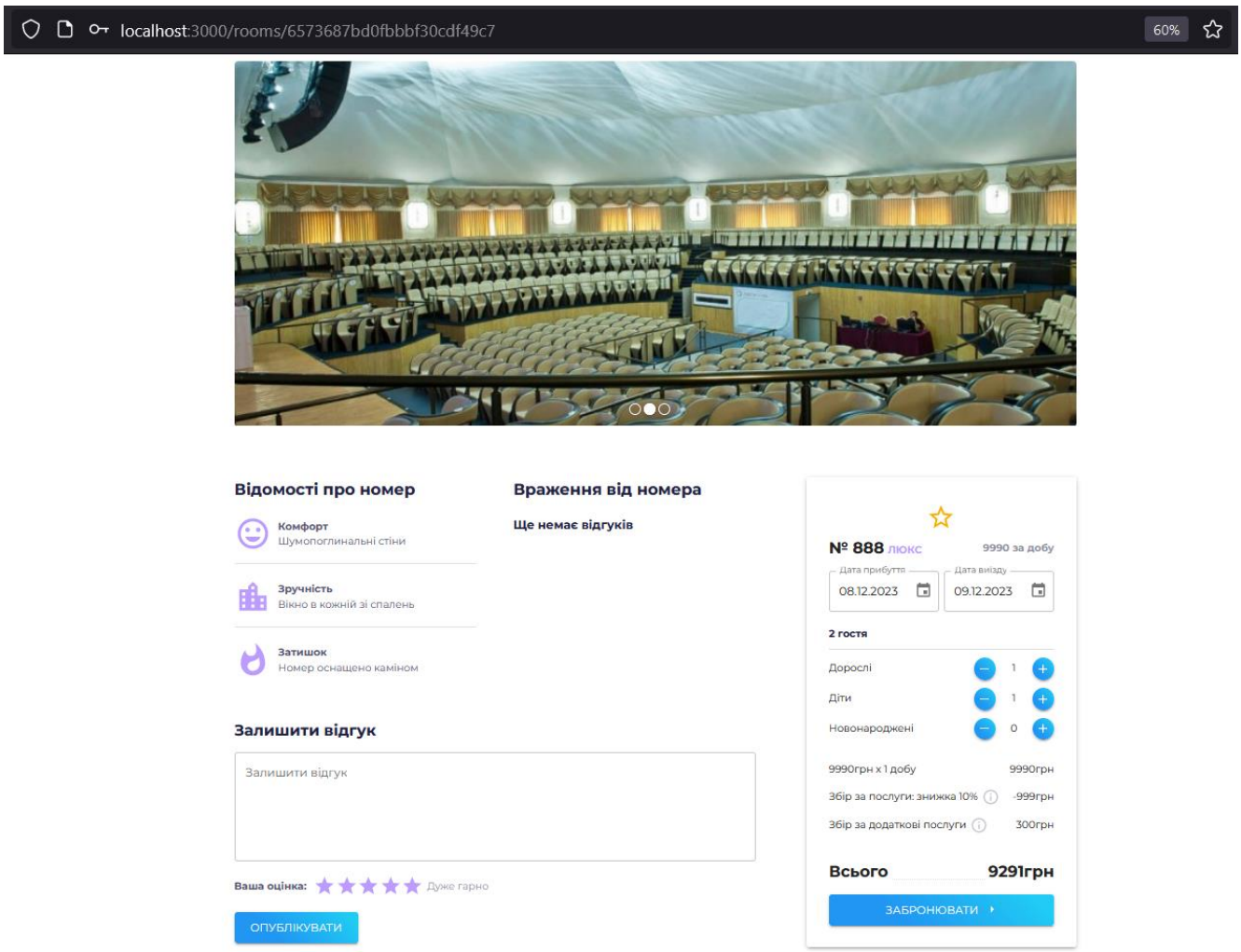

Рисунок 3.18 – Вигляд сторінки обраного номеру

При необхідності, користувач може додати номер до обраних, переглянути ціну за оренду номеру за добу а також залишити власний відгук з власною оцінкою. На рисунку 3.19 представлено результат публікації відгуку з оцінкою.

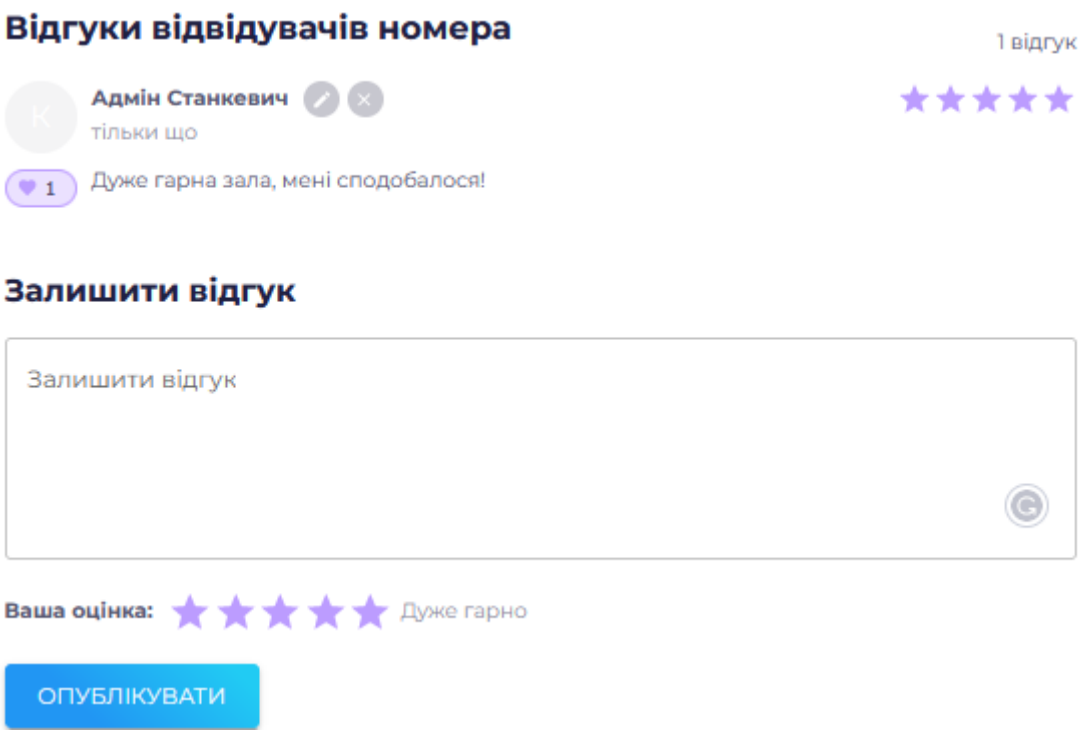

Рисунок 3.19 – Результат додавання відгуку

Після перегляду інформації про обраний номер, користувач може виконати бронювання. На рисунку 3.20 представлено блок з бронюванням номеру. Користувач може обрати дату та кількість людей при заселенні, а також переглянути фінальну ціну.

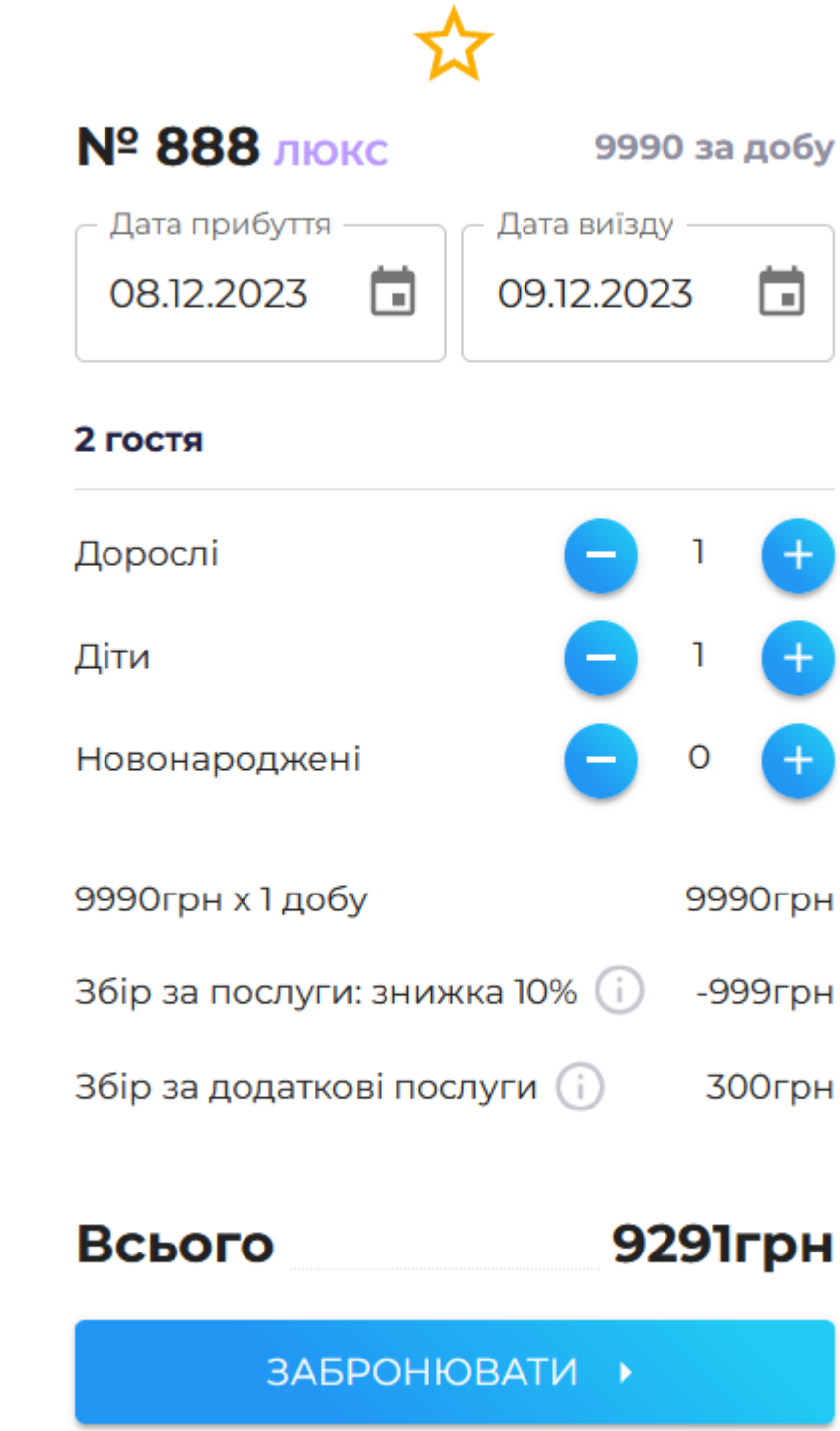

Рисунок 3.20 – Блок для бронювання обраного номеру

Після того, як користувач натиснув на кнопку «Забронювати», він отримує повідомлення про успішне бронювання (рис 3.21).

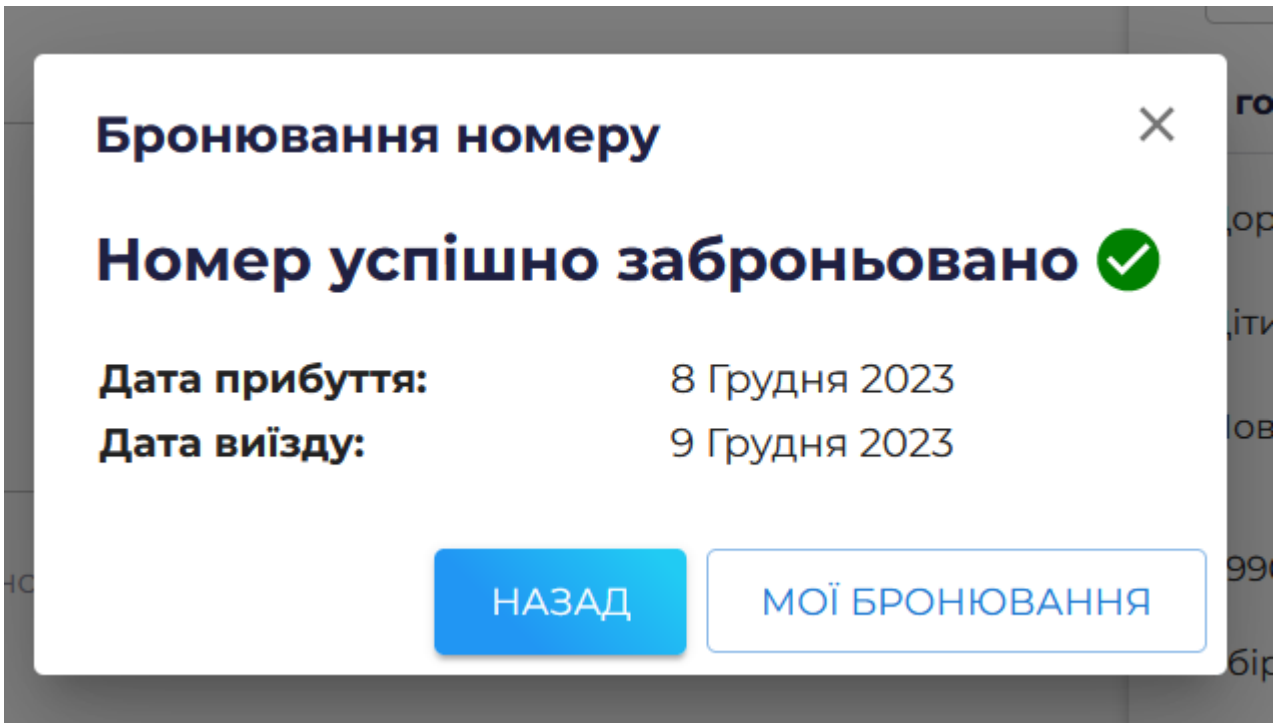

Рисунок 3.21 – Результат про успішне бронювання

При виборі дати бронювання на яку вже виконав замовлення інший користувач система надасть повідомлення про неможливість виконати бронюваня (рис 3.22).

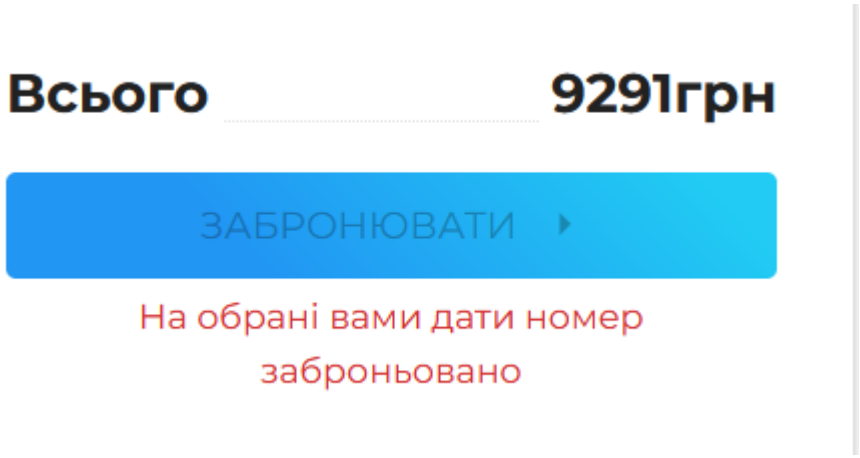

Рисунок 3.22 – Повідомлення про відмову можливості бронювання

Після цього користувач може переглянути власні бронювання в особистому кабінеті. На рисунках 3.23 – 3.24 представлено вигляд особистого кабінету користувача та сторінку перегляду власних бронювань.

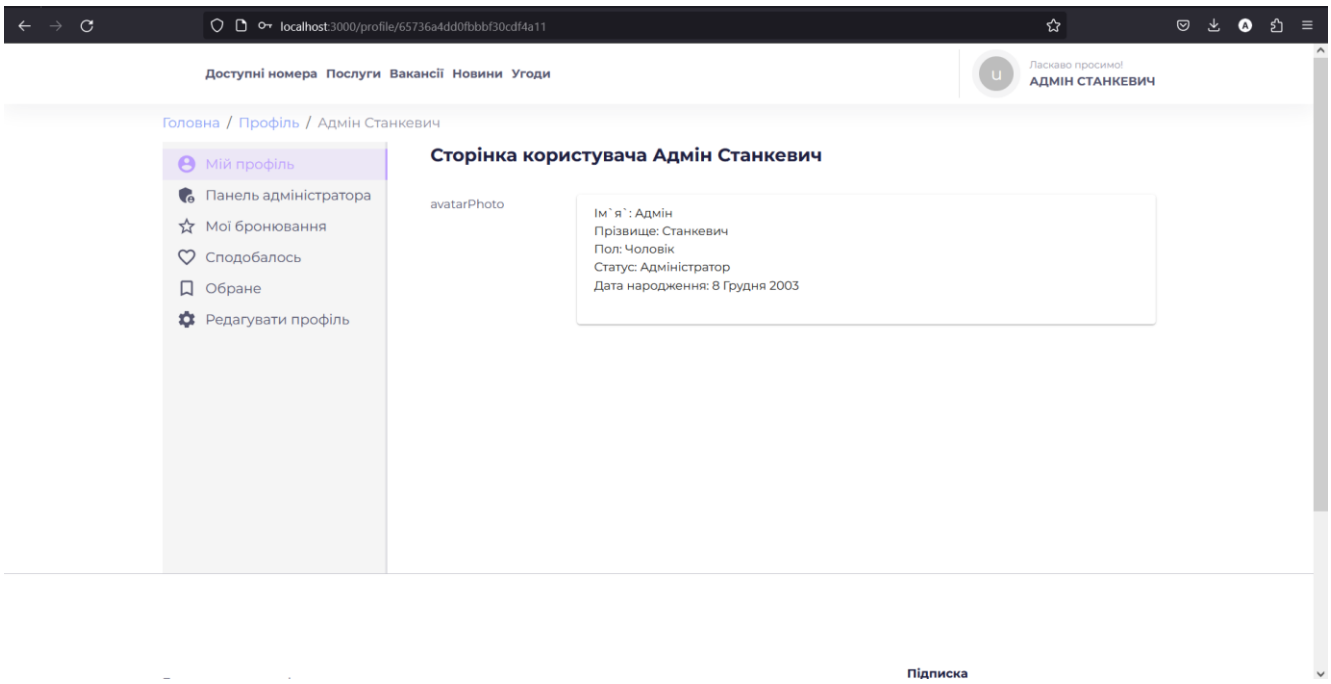

Рисунок 3.23 – Вигляд особистого кабінету користувача

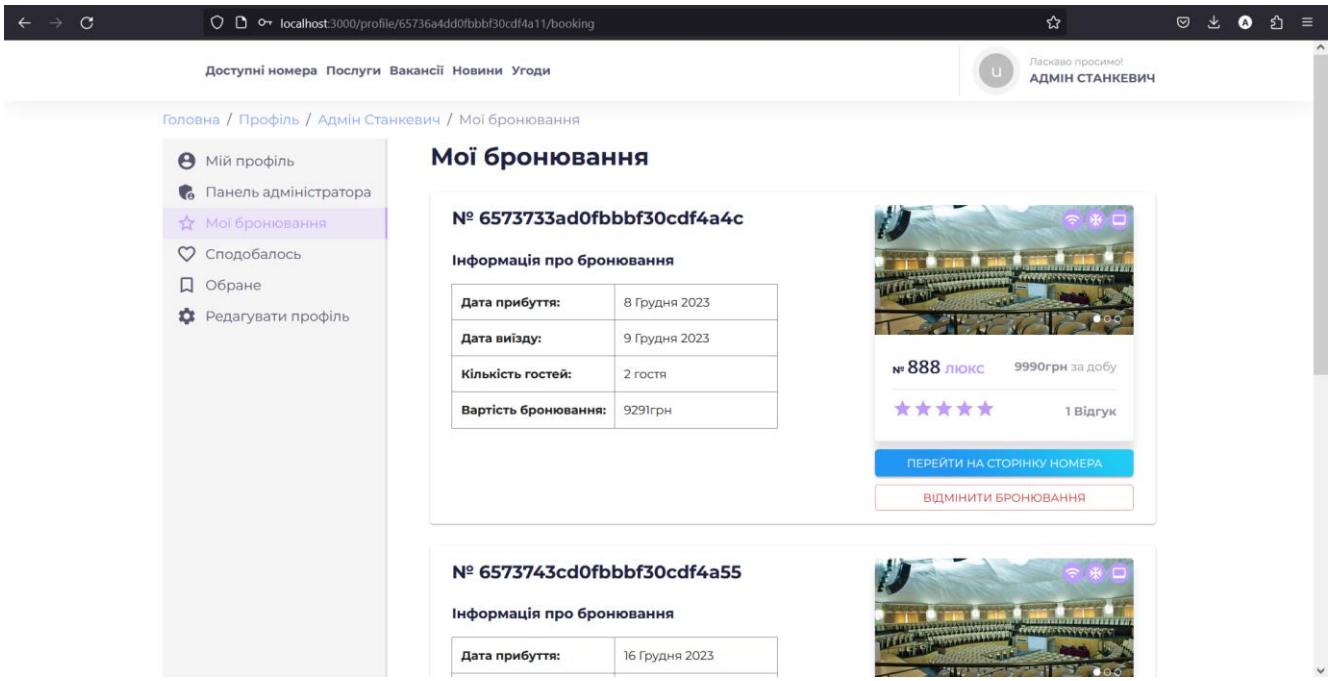

Рисунок 3.24 – Вигляд сторінки заброньованих номерів

Також для користувачів є роль «Admin», яка надає можливість користуватись панеллю адміністратора та переглядати бронювання інших користувачів з можливістю видалення та зміни номерів та апартаментів. На рисунку 3.25 представлено вигляд адміністративної панелі.

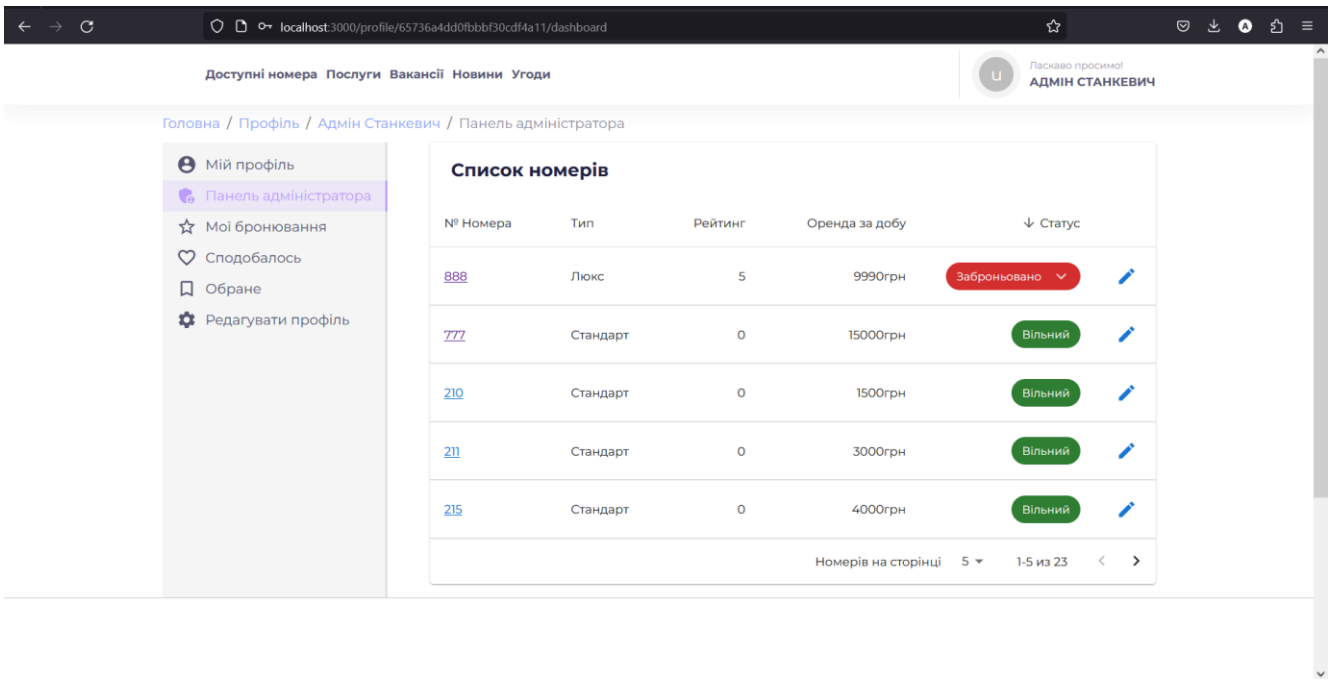

Рисунок 3.25 – Вигляд сторінки доступних номерів для редагування

3.4 Тестування та аналіз хостинг систем

Для тестування взаємодії клієнта з сервером було виконано тестування JSON-запитів. Для цього процесу було використано додаток Insomniac. На рисунку 3.26 представлено результат тестування виводу даних про номери готелю.

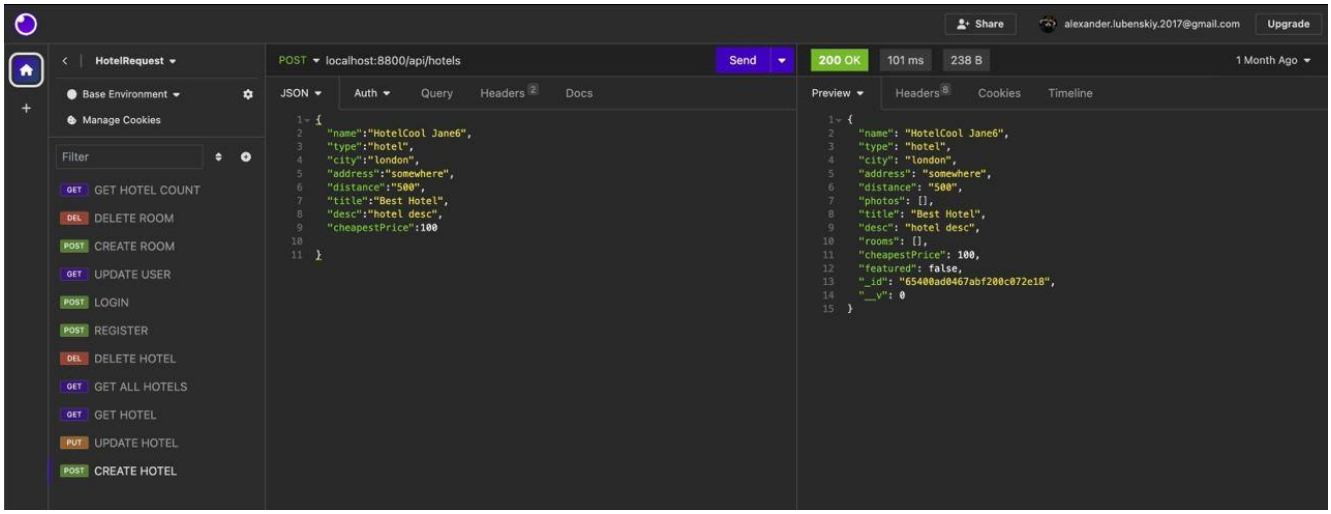

Рисунок 3.26 – Результат тестування JSON-запиту про наявні номери

Також було протестовано процес виводу інформацію даних. Результат представлено на рисунках 3.27 – 3.28.

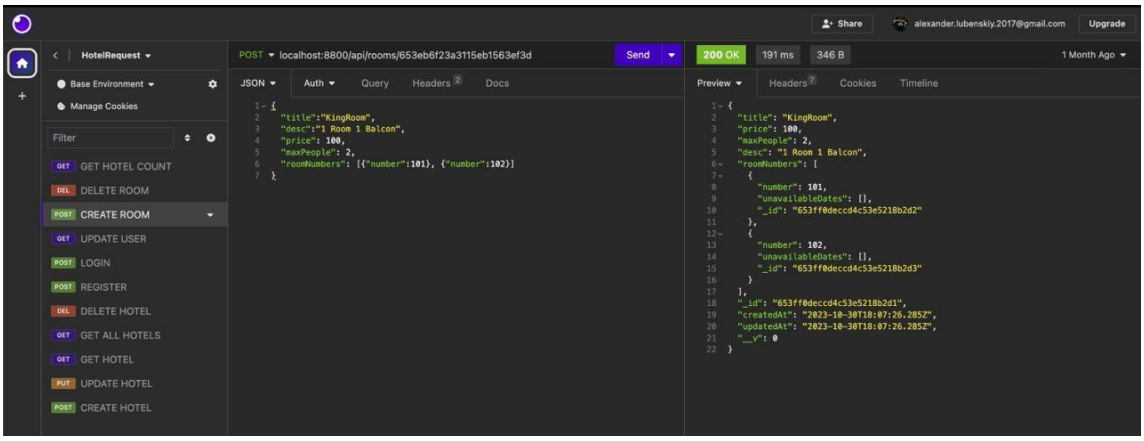

Рисунок 3.28 – Результат тестування інформації про опис кімнати

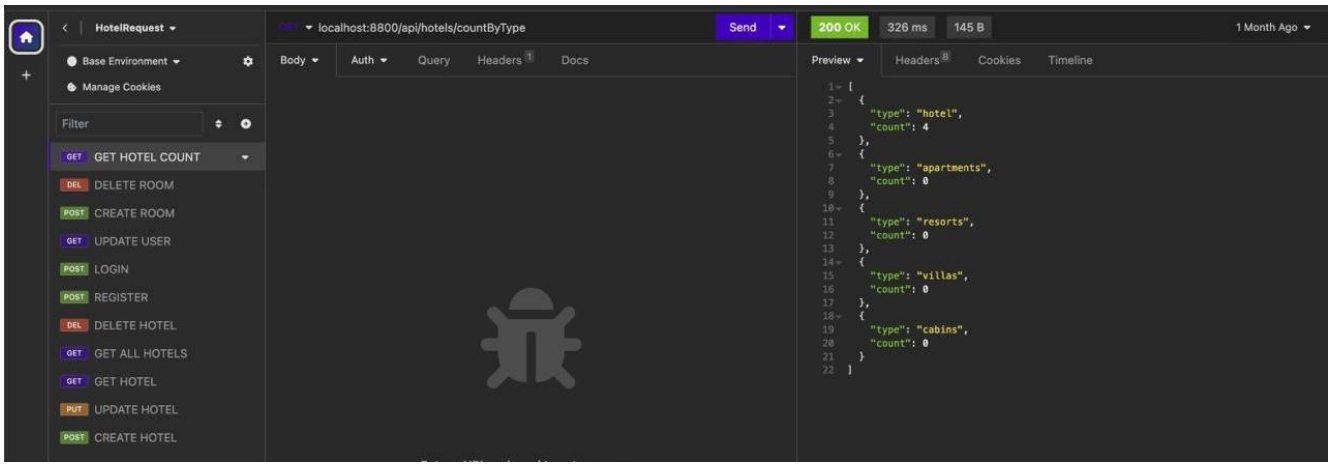

Рисунок 3.29 – Результат тестування виводу інформації про типи кімнат

Для аналізу хостинг-провайдерів було обрано наступні системи:

- S-host.com.ua;
- Hostlife.net;
- Ukraine.com.ua;

Для визначення більш зручної хостинг-системи було створено порівняльну таблицю (табл 3.1) з описаними критеріями оцінювання.

Таблиця 3.1 – Порівняльна таблиця хостинг-систем

Критерії	Хостинг-системи		
	S-host.com.ua	Hostlife.net	Ukraine.com.ua
Безкоштовний тестовий обліковий запис, днів	8	8	8
Дисковий простір, МБ	1000	500	2000
Додатково 1ГБ дискового простору, грн\рік	89	59	59
Трафік	Безліміт	Безліміт	Безліміт
Виділена IP-адреса, грн\міс	30	30	29
FTP облікові записи	1	1	5
Поштові скриньки	250	3	10
Кількість доменів	1	1	5
Кількість поштових доменів	25	1	5
БД \ користувачі БД	1/1	1/1	2/5

Після детального аналізу хостинг систем було визначено що Ukraine.com.ua -це відмінний хостинг для створення власної веб-системи. Цей хостинг-провайдер відомий своєю високою надійністю і повноцінною підтримкою, яка забезпечує безперебійну роботу ваших веб-проектів. "Ukraine.com.ua" він надає всі інструменти, необхідні для підтримки роботи web-системи, включаючи швидке розгортання, надійні сервери, ефективне управління ресурсами та зручний інтерфейс для адміністрування.

ВИСНОВОК

Результатом дослідження стала розробка веб-системи замовлення послуг для готельно-ресторанного комплексу "Шафран". Аналіз цільового домену та подібних додатків підтвердив, що розробка веб-системи замовлення послуг доречна та підходить для готельного та ресторанного сектору.

Були визначені основні особливості та вимоги до додатку, а також встановлені основні цілі та завдання проекту. Порівняння з існуючими рішеннями підтвердило необхідність таких програм для оптимізації процесів обслуговування та підвищення зручності клієнтів.

Крім того, було досліджено та впроваджено найбільш корисні інструменти та методи для реалізації веб-додатку. Інструменти та технології, досліджені та впроваджені, включають новітні технології веб-розробки, бази даних і ефективні методи управління проектами. Таким чином досягається оптимальний баланс між функціональністю системи, продуктивністю та безпекою.

Складені робочі плани та аналіз ризиків допомогли підготуватися до етапу реалізації проекту, передбачити можливі виклики та забезпечити ефективне впровадження веб-системи.

Розроблена система відкриває широкі можливості для служби замовлень готельно-ресторанного комплексу "Шафран" та сприяє підвищенню його конкурентоспроможності на ринку. Впровадження інформаційних технологій у бізнес-процеси сприяло підвищенню ефективності роботи комплексу.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. What Is Web 2.0? Definition, Impact, and Examples Автор: Will Kenton, Khadija Khartit. [Електронний ресурс] – Режим доступу до ресурсу: https://www.investopedia.com/terms/w/web-20.asp (дата звернення 04.11.2023).
2. How to Create a Travel and Tourism Website in 2023 [Електронний ресурс] – Режим доступу до ресурсу: https://www.aimprosoft.com/blog/how-to-build-a-travel-website/ (дата звернення 04.11.2023).
3. Evolution of Web Systems Автор Damiano Distante [Електронний ресурс] – Режим доступу до ресурсу: https://www.researchgate.net/publication/264824613_Evolution_of_Web_Systems (дата звернення 04.11.2023).
4. 5 Benefits of Hotel Reservation & Booking Application [Електронний ресурс] – Режим доступу до ресурсу: https://www.optisolbusiness.com/insight/5-benefits-of-hotel-reservation-and-booking-application (дата звернення 05.11.2023).
5. Top 10 benefits and advantages of a hotel management system [Електронний ресурс] – Режим доступу до ресурсу: https://www.mews.com/en/blog/advantages-of-hotel-management-system (дата звернення 04.11.2023).
6. Web-сервіс www.booking.com [Електронний ресурс] – Режим доступу до ресурсу: https://www.booking.com (дата звернення 05.11.2023).
7. Web-сервіс gohotels.com.ua [Електронний ресурс] – Режим доступу до ресурсу: gohotels.com.ua (дата звернення 12.05.2023).
8. Web-сервіс hotelmix.com.ua [Електронний ресурс] – Режим доступу до ресурсу: hotelmix.com.ua (дата звернення 12.05.2023).
9. MERN.js official web-site [Електронний ресурс] – Режим доступу до ресурсу: https://mern.js.org/ (дата звернення 14.05.2023).
10. MongoDB Official web-site [Електронний ресурс] – Режим доступу до ресурсу: https://www.mongodb.com/ (дата звернення 15.05.2023).

11. Документація з використання Express.js [Електронний ресурс] – Режим доступу до ресурсу: http://expressjs.com/uk/ (дата звернення 15.05.2023).
12. Документація з використання React.js [Електронний ресурс] – Режим доступу до ресурсу: https://react.dev/blog/2023/03/16/introducing-react-dev (дата звернення 15.05.2023).
13. Документація з використання Node.js [Електронний ресурс] – Режим доступу до ресурсу: https://nodejs.org/en (дата звернення 15.05.2023).
14. Why choose MERN stack for modern business applications in 2022 [Електронний ресурс] – Режим доступу до ресурсу: https://www.softwebsolutions.com/resources/build-apps-with-mern-stack.html (дата звернення 15.05.2023).
15. The function analysis diagram: Intended benefits and coexistence with other functional models [Електронний ресурс] – Режим доступу до ресурсу: https://www.researchgate.net/publication/259434209_The_function_analysis_diagram_Intended_benefits_and_coexistence_with_other_functional_models (дата звернення 15.05.2023).
16. What is IDEF - Definition, Methods, and Benefits [Електронний ресурс] – Режим доступу до ресурсу: https://www.edrawsoft.com/what-is-idef.html (дата звернення 15.05.2023).
17. Use Cases: the Pros and Cons Автор: Donald G. Firesmith [Електронний ресурс] – Режим доступу до ресурсу: https://www.cs.hmc.edu/~mike/courses/mike121/readings/reqsModeling/firesmith.htm (дата звернення 15.05.2023).
18. Web Application Development using UML Автор Dilip Kothamasu [PDF] – Режим доступу до ресурсу: https://www.cs.wcupa.edu/~zjiang/csc321_Dilip.pdf (дата звернення 15.05.2023).
19. Why Use MongoDB? Advantages of MongoDB Автор Abhresh Sugandhi від 29.09.2023 [Електронний ресурс] – Режим доступу до ресурсу: https://www.knowledgehut.com/blog/web-development/advantages-of-mongodb (дата звернення 15.05.2023).

20. All About MongoDB: Benefits, Features, and Tools to Use with [Електронний ресурс] – Режим доступу до ресурсу: https://javascript.plainenglish.io/all-about-mongodb-benefits-features-and-tools-to-use-with-b06eb18ddf3d (дата звернення 15.05.2023).
21. The Benefits of a Cloud Database [Електронний ресурс] – Режим доступу до ресурсу: https://www.mongodb.com/cloud-database/benefits (дата звернення 15.05.2023).
22. Understanding MVC Architecture in the MERN Stack Database [Електронний ресурс] – Режим доступу до ресурсу: https://medium.com/@ansari028amaan/understanding-mvc-architecture-in-the-mern-stack-5cc083828298 (дата звернення 15.05.2023).
23. Model-View-Controller(MVC) architecture for Node applications [Електронний ресурс] – Режим доступу до ресурсу: https://www.geeksforgeeks.org/model-view-controllermvc-architecture-for-node-applications/ (дата звернення 15.05.2023).
24. How to Use SMART Goals to Succeed in Web Development by S. Anderson [Електронний ресурс] – Режим доступу до ресурсу: https://www.nucamp.co/blog/smart-goal-setting-for-developers (дата звернення 12.05.2023).
25. What Is Work Breakdown Structure and How to Use it? Автор I.Krasteva [Електронний ресурс] – Режим доступу до ресурсу: https://businessmap.io/blog/work-breakdown-structure (дата звернення 12.05.2023).
26. Organization Breakdown Structure – OBS (#9 in the series How to Plan and Organize a Project) Автор Michael D. Taylor [Електронний ресурс] – Режим доступу до ресурсу: https://pmhut.com/organization-breakdown-structure-obs (дата звернення 12.05.2023).
27. How to Use Gantt Charts for Software Development Автори Eugene L., Tania H. [Електронний ресурс] – Режим доступу до ресурсу:

https://rubygarage.org/blog/gantt-charts-in-web-development (дата звернення 12.05.2023).

28. The Importance of Risk Management in Project Success [Електронний ресурс] – Режим доступу до ресурсу: https://gururo.com/the-importance-of-risk-management-in-project-success (дата звернення 12.05.2023).
29. 5 Steps Project Risk Management Process and It's Benefits [Електронний ресурс] – Режим доступу до ресурсу: https://www.vinsys.com/blog/project-risk-management-process (дата звернення 12.05.2023).
30. The Benefits of Risk Management Planning [Електронний ресурс] – Режим доступу до ресурсу: https://pmcenter.bellevue.edu/2016/06/19/the-benefits-of-risk-management-planning/ (дата звернення 12.05.2023).

ДОДАТОК А

Деталізація мети проекту методом SMART

SMART - це найкраща практика постановки цілей. SMART-цілі повинні бути конкретними, вимірюваними, досяжними, реалістичними та обмеженими в часі. Ставлячи мету, людина складає дорожню карту для досягнення конкретного завдання. Елементи цієї системи працюють разом, щоб створити ціль, яка є ретельно і продумано спланованою, досяжною і відстежуваною.

SMART-цілі надають численні можливості для особистого та ділового успіху і можуть допомогти в досягненні короткострокових і довгострокових цілей. Також вони допомагають розбити ширші цілі на конкретні та досяжні завдання, забезпечуючи таким чином відчуття спрямованості та фокусування на бажаному результаті. Ставлячи цілі, люди можуть встановити орієнтири та оцінити свої сильні та слабкі сторони у досягненні поставлених цілей. Після того, як цілі визначені, більшість людей захочуть наполегливо працювати, щоб їх досягти. Наприклад, компанія, яка хоче збільшити продажі, може створити програму заохочення працівників, яка стимулюватиме їх до досягнення певних цілей. SMART-цілі є досяжними, але водночас і складними.

Складні аспекти SMART-цілей автоматично змушують людей виходити із зони комфорту, щоб вони могли діяти. Дотримуючись SMART-цілей і завдань, люди витрачають менше часу на виконання несуттєвих завдань і витрачають більше часу на досягнення результатів [24]. Результати наведені у таблиці Б.1

Таблиця Б.1 – Формалізація мети за технологією SMART

Конкретизація	Розробити web-орієнтовану систему, яка дозволить клієнтам готельно-ресторанного комплексу "Шафран" замовляти послуги та ознайомлюватися з переліком товарів.
Вимірюваний	Вимірюється за кількістю клієнтів, які успішно здійснюють замовлення через систему, а також за збільшенням обсягу продажів через цей канал.

Досяжний	Проєкт реалізовується у відповідності до рівня досвіду та на основі затвердженого технічного завдання.
Значимість	Враховуючи потреби та очікування клієнтів, розробка такої системи вплине на підвищення конкурентоспроможності готельно-ресторанного комплексу "Шафран".
Обмежений в часі	Проект має бути завершений та впроваджений впродовж визначеного терміну, згідно з розробленим графіком робіт.

Work Breakdown Structure (WBS) – Ієрархічна структура робіт

WBS (work breakdown structure) - це метод, який дозволяє зробити складні проекти більш керованими. Це означає розбиття проекту на завдання та підзадачі, щоб у майбутньому не було місця для плутанини. Зазвичай це практикується менеджерами проектів, щоб спростити процес виконання завдань в рамках проекту.

Кожен проект, незалежно від того, малий він чи великий, відіграє ключову роль у визначенні довголіття бізнесу. Адже з кожним успішним проектом команда завойовує довіру та лояльність клієнтів, а це утримує бізнес в лідируючих позиціях [25]. На рисунку Б.1 представлено WBS проєкту щодо розробки web-орієнтованої систему готельно-ресторанного комплексу "Шафран".

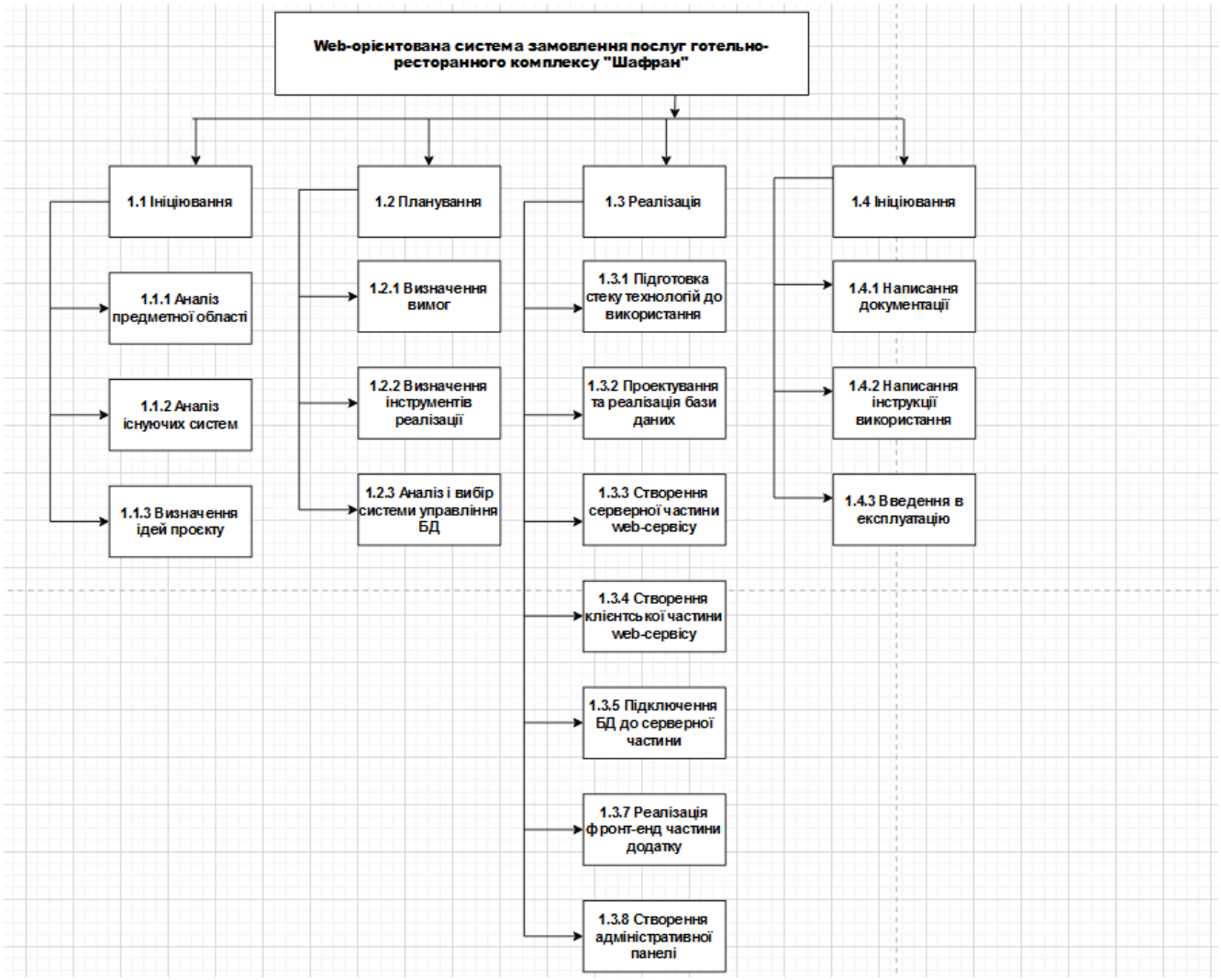

Рисунок Б.1 – WBS-структура робіт проекту

Організаційна структура - це формальні та неформальні політики і процедури, які компанії використовують для управління бізнес-операціями. Малі та домашні підприємства, як правило, не використовують організаційні структури, оскільки власник бізнесу зазвичай відповідає за виконання більшості бізнес-функцій. Великі компанії використовують організаційну структуру для ефективного управління бізнес-функціями та працівниками. Хоча тип організаційної структури зазвичай залежить від розміру компанії та її діяльності, існує кілька універсальних переваг, які вона дає [26].

На рисунку Б.2 представлено організаційну структуру планування проекту.

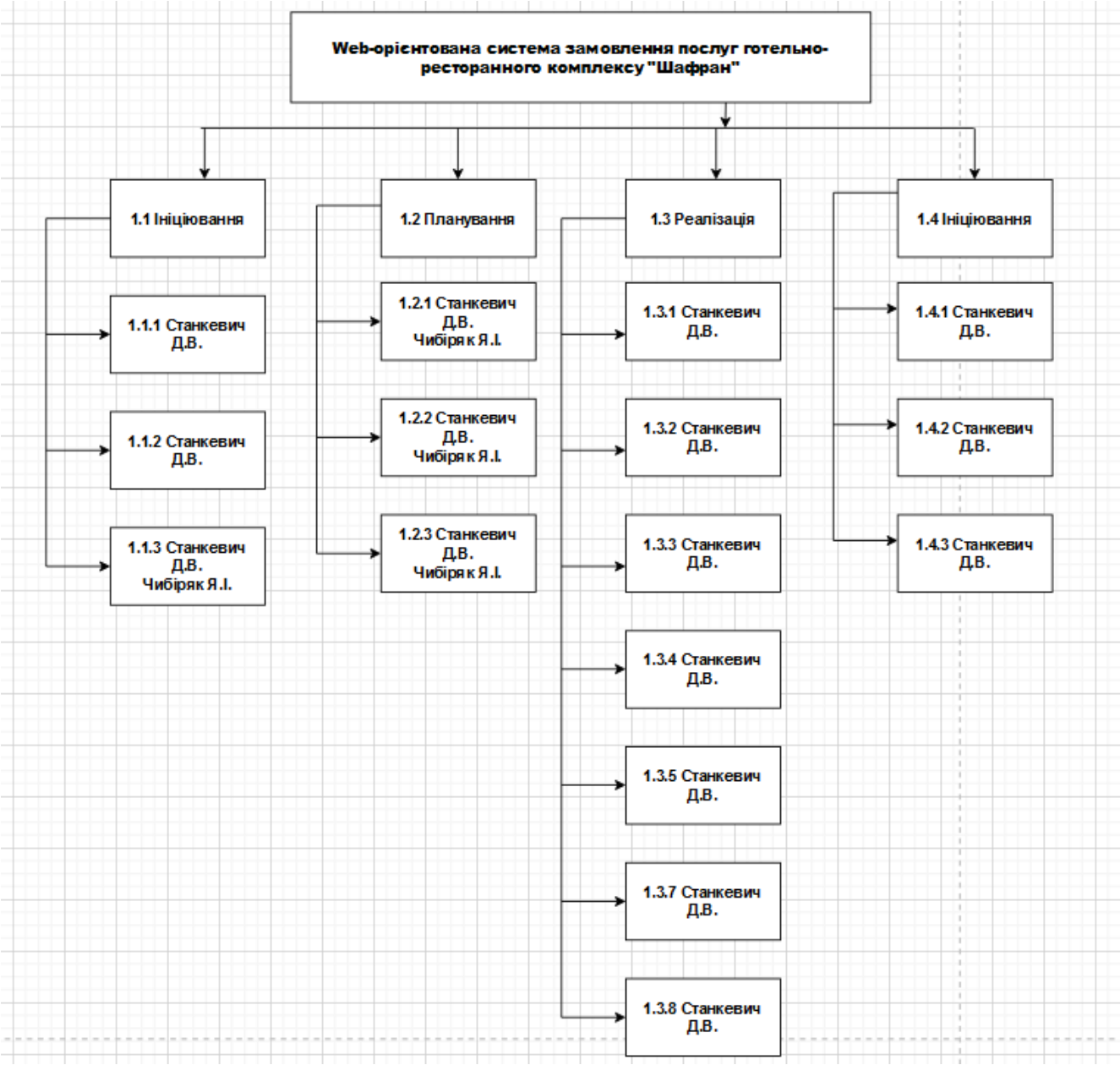

Рисунок Б.2 – OBS-структура проекту

Діаграма Ганта

Побудова календарного графіку (діаграми Ганта) є одним з важливих етапів планування проекту, що виглядає як розклад виконання робіт з реальним розподілом дат [27].

Розробка web-орієнтованої системи замовлення послуг готельно-ресторанного комплексу "Шафран"	93 дней	Вт 12.09.23	Чт 18.01.24			
Ініціалізація проекту	4 дней	Вт 12.09.23	Пт 15.09.23			
Опрацювання ТЗ	1 день	Сб 16.09.23	Сб 16.09.23		Станкевич Д.В.	
Аналіз сервісів	1 день	Вс 17.09.23	Вс 17.09.23	3	Станкевич Д.В.	$10000
Пошук контенту	1 день	Пн 18.09.23	Пн 18.09.23	4	Станкевич Д.В.	$1,116.00
Ідентифікація проекту	1 день	Вт 19.09.23	Вт 19.09.23	5	Чибіряк Я.І.	$72.00
Планування проекту	**15 дней**	Ср 20.09.23	Вт 10.10.23	6		$72.00
Структура проекту	2 дней	Ср 11.10.23	Чт 12.10.23	7	Станкевич Д.В.	$72.00
Розробка дизайну та макету	2 дней	Пт 13.10.23	Пн 16.10.23	8	Станкевич Д.В.	$900.00
Розробка календарного плану «Розробка web-орієнтованої системи замовлення послуг готельно-ресторанного комплексу "Шафран"»	5 дней	Вт 17.10.23	Пн 23.10.23	9	Станкевич Д.В.	$1,432.00
Управління ресурсами	1 день	Вт 24.10.23	Вт 24.10.23	10	Станкевич Д.В.	$72.00
Управління фінансами	1 день	Ср 25.10.23	Ср 25.10.23	11	Станкевич Д.В.	$72.00
Управління якістю	1 день	Чт 26.10.23	Чт 26.10.23	12	Станкевич Д.В.	$100.00
Оптимізація плану	1 день	Пт 27.10.23	Пт 27.10.23	13	Станкевич Д.В.	$72.00
Затвердження плану	1 день	Сб 28.10.23	Сб 28.10.23	14	Чибіряк Я.І.	$72.00
Реалізація проекту	47 дней	Вс 29.10.23	Пн 01.01.24		Станкевич Д.В.	$7,020.00
Розробка web-сервісу	25 дней	Вс 29.10.23	Чт 30.11.23		Станкевич Д.В.	$3,250.00
Реалізація дизайну та макету	9 дней	Пн 30.10.23	Чт 09.11.23		Станкевич Д.В.	$1,300.00

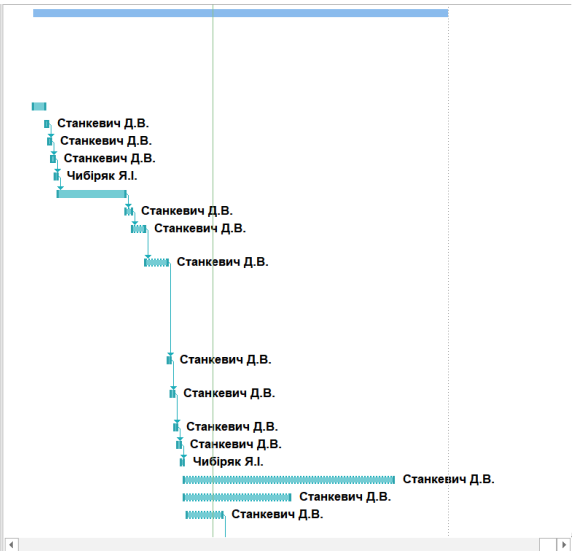

Рисунок Б.3 – Діаграма Ганта. Частина 1

Реалізація проекту	47 дней	Вс 29.10.23	Пн 01.01.24		Станкевич Д.В.	$7,020.00
Розробка web-сервісу	25 дней	Вс 29.10.23	Чт 30.11.23		Станкевич Д.В.	$3,250.00
Реалізація дизайну та макету	9 дней	Пн 30.10.23	Чт 09.11.23		Станкевич Д.В.	$1,300.00
Розробка Front-end частини	8 дней	Пт 10.11.23	Вт 21.11.23	18	Станкевич Д.В.	$1,500.00
Розробка Back-end частини	5 дней	Ср 22.11.23	Вт 28.11.23	19	Станкевич Д.В.	$350.00
Проектування БД	3 дней	Ср 29.11.23	Пт 01.12.23	20	Станкевич Д.В.	$100.00
Створення компонентів	1 день	Чт 30.11.23	Чт 30.11.23	21	Станкевич Д.В.	$100.00
Створення зв'язку між користувачем та додатком	**21 дней**	Пт 01.12.23	Пт 29.12.23		Станкевич Д.В.	$3,270.00
Розробка модулю для пошуку готелю за критеріями	1 день	Сб 02.12.23	Сб 02.12.23		Станкевич Д.В.	$270.00
Розробка модулів реєстрації та авторизації нового користувача	1 день	Пн 04.12.23	Пн 04.12.23	24	Станкевич Д.В.	$1,500.00
Розробка модулю замовлення готелю	2 дней	Вт 05.12.23	Ср 06.12.23	25	Станкевич Д.В.	$1,500.00
Реалізація зв'язку БД з додатком	4 дней	Чт 07.12.23	Вт 12.12.23	26	Станкевич Д.В.	$500.00
Тестування	2 дней	Ср 13.12.23	Чт 14.12.23	27	Станкевич Д.В.	$500.00
Завершення проекту	6 дней	Пт 15.12.23	Пт 22.12.23	28	Станкевич Д.В.	$432.00
Створення документації	1 день	Пт 15.12.23	Пт 15.12.23	29	Станкевич Д.В.	$72.00
Створення інструкції користувача	2 дней	Сб 16.12.23	Пн 18.12.23	30	Станкевич Д.В.	$72.00
Запуск і підтримка	2 дней	Вт 19.12.23	Ср 20.12.23	31	Станкевич Д.В.	$72.00
Архівація проекту	1 день	Чт 21.12.23	Чт 21.12.23	32	Станкевич Д.В.	$72.00

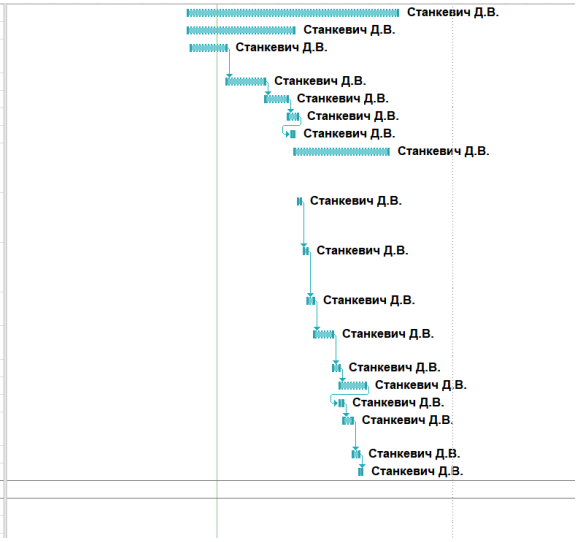

Рисунок Б.4 – Діаграма Ганта. Частина 2

Управління ризиками проєкту

Управління проектними ризиками допомагає прогнозувати та пом'якшувати ризики що допомагає уникнути будь-яких втрат або конфліктів у проекті [28].

У таблиці Б.3 представлено шкалу для класифікації ризиків за величиною впливу на проект та ймовірністю виникнення.

Таблиця Б.3 – Шкала оцінювання ризиків за ймовірністю виникнення та величиною впливу.

Оцінка	**Ймовірність виникнення**	**Вплив ризику**	**Тип ризику**
1	Низька	Низький	Прийнятні
2	Середня	Середній	Виправдані
3	Висока	Високий	Недопустимі

Для уникнення появи критичних ризиків було розроблено таблицю реагування яке на основі показників в таблиці Б.3 надає чітке розуміння стратегії реагування [29][30]. У результаті цього було отримано матрицю ймовірності. Оцінювання виконується за показниками, що описані в таблиці Б.3.

У результаті планування реагування було отримано матрицю ймовірності виникнення ризиків. Зеленим кольором на матриці позначені прийнятні ризики, жовтим – виправдані, а червоним – недопустимі.

		Probability		
1	Impact	RS_2		RS_4 RS_5
2		RS_13	RS_14 RS_9 RS_3	
3		RS_12	RS_8 RS_11 RS_15	RS_6 RS_7 RS_10 RS_1
		1	2	3

Таблиця Б.6. – Матриця ймовірності

Таблиця Б.4 – Шкала оцінювання за рівнем ризику.

№	Назва	Межі	Ризики, які входять
1	Прийнятні	1<R<2	8,11,12,13,15
2	Виправдані	3<R<4	1,2,3,6,7,9,10,14
3	Недопустимі	6<R<9	4,5

Таблиця Б.5 – Ризики та стратегії реагування

ID	Опис ризику	Ймовірність виникнення ризику	Вплив ризику	Ранг ризику	Тип стратегії реагування	Стратегія реагування
ID_1	Технічні проблеми сервера або бази даних	Низька	Середній	3	Попередження	Регулярне резервне копіювання даних та моніторинг роботи серверів. Швидке відновлення роботи в разі виникнення проблем.

Продовження таблиці Б.5 – Ризики та стратегії реагування

ID_2	Атаки на безпеку (наприклад, DDoS атаки або хакерські атаки)	Низька	Середній	4	Прийняття	Використання заходів безпеки, включаючи захист від DDoS, шифрування даних та регулярну аудиторію безпеки.
ID_3	Виток конфіденційної інформації	Середня	Середній	4	Пом’якшення	Реалізація строгих прав доступу до даних, шифрування конфіденційної інформації та навчання персоналу щодо правил безпеки.

Продовження таблиці Б.5 – Ризики та стратегії реагування

ID_4	Низька швидкість завантаження сторінок	Середня	Високий	6	Попередження	Оптимізація коду та зображень, використання кешування для прискорення завантаження.

Продовження таблиці Б.5 – Ризики та стратегії реагування

ID_5	Труднощі з інтеграцією платіжних систем	Висока	Високий	9	Пом’якшення	Ретельне тестування платіжних систем перед впровадженням та підтримка клієнтів у випадку труднощів.

Продовження таблиці Б.5 – Ризики та стратегії реагування

ID	Опис ризику	Ймовірність виникнення	Вплив ризику	Ранг ризику	Тип стратегії реагування	Стратегія реагування
ID_6	Зміни в законодавстві щодо надання послуг	Середня	Середній	4	Пом’якшення	Постійне моніторинг законодавчих змін та адаптація системи до нових вимог.

Продовження таблиці Б.5 – Ризики та стратегії реагування

ID_7	Відмова у наданні послуги хостинг-постачальником	Низька	Середній	2	Пом’якшення	Мати резервні копії даних та можливість швидко перемістити систему до іншого хостингу
ID_8	Низька ефективність рекламних кампаній	Середня	Середній	4	Пом’якшення	Аналіз та корекція рекламних стратегій, залучення професіоналів для оптимізації рекламних кампаній.

Продовження таблиці Б.5 – Ризики та стратегії реагування

ID_9	Неадекватна підтримка клієнтів	Низька	Середній	3	Попередження	Навчання персоналу, впровадження системи керування зверненнями клієнтів та швидке вирішення проблем.

Продовження таблиці Б.5 – Ризики та стратегії реагування

ID_10	Недостатня відмінність від конкурентів	Низька	Середній	2	Попередження	Аналіз конкурентів та розробка унікальних пропозицій для клієнтів..
ID_11	Технічна застарілість системи	Низька	Низький	1	Використання	Обговорити вимоги і збитки від можливих змін проекту.

ID_12	Відмова або недоцільність інтеграції з іншими системами	Середня	Низький	2	Перенесення	Враховувати потреби та можливості інтеграції з іншими системами на етапі розробки

Продовження таблиці Б.5 – Ризики та стратегії реагування

ID_14	Проблеми зі скасуванням або поверненням замовлень	Середня	Середня	4	Прийняття	Розробка чітких правил та процедур для обробки скасувань та повернень.

Продовження таблиці Б.5 – Ризики та стратегії реагування

ID_14	Неякісне введення та оновлення даних адміністратором	Низька	Середній	3	Пом’якшення	Надання адміністратору необхідних інструментів та навчання правильному веденню даних.
ID_15	Неочікувані технічні проблеми з оновленням системи	Низька	Середній	3	Прийняття	Ретельне тестування нових оновлень перед їх впровадженням та маючи можливість швидко відкатити зміни у разі проблем

ДОДАТОК Б

Лістинг коду user.js

```
import User from "../models/User.js";
import { createError } from "../utils/error.js";

export const updateUser = async (req, res, next)=>{

    try{
        const updateUser = await User.findByIdAndUpdate(req.params.id, { $set: req.body }, {new: true})
        res.status(200).json(updateUser);
    }catch(error){
        next(error);
    }
}

export const deleteUser = async (req, res, next)=>{

    try{
        await User.findByIdAndDelete(req.params.id);
        res.status(200).json("User has been Deleted");
    }catch(error){
        next(error);
    }
}

export const getUser = async (req, res, next)=>{

    try{
```

```
        const user = await User.findById(req.params.id)
        res.status(200).json(user);
    }catch(error){
        // res.status(500).json(error);
        next(error);
    }
}

export const getUsers = async (req, res, next)=>{

    // const failed = true;

    // if(failed) return next(createError(401, "You are not Auny"));

    console.log("User route")
    try{
        const users = await User.find();
        res.status(200).json(users);
    }catch(error){
        console.log("get user controller"+error);
        next(error);
    }
}
```

ДОДАТОК В

Лістинг коду «Header.jsx»

```
import React from 'react';
import { useSelector } from 'react-redux';
import { NavLink } from 'react-router-dom';
import { navigationRoutes } from
'../../../router/routes';
import { getAuthErrors, getIsLoggedIn } from
'../../../store/users';
import NavProfile from '../../ui/NavProfile';
import Button from '../Button';
import Container from '../Container';
import Divider from '../Divider';

import NavList from '../NavList';

const Header: React.FC = () => {
  const isLoggedIn = useSelector(getIsLoggedIn());
  const authErrors = useSelector(getAuthErrors());

  return (
    <header className='header'>
      <Container>
        <div className='header__inner'>

          <NavList routes={navigationRoutes}
className='header-nav' />
          {isLoggedIn && !authErrors ? (
            <>
```

```
              <Divider orientation='vertical' flexItem
classNameName='header__divider' />
              <NavProfile />
            </>
          ) : (
            <div classNameName='header-buttons'>
              <NavLink to='/login/signIn'
classNameName='header-buttons-button'>
                <Button size='small'
variant='outlined'>
                  Увійти
                </Button>
              </NavLink>
              <NavLink to='/login/signUp'
classNameName='header-buttons-button'>
                <Button
size='small'>Зареєструватися</Button>
              </NavLink>
            </div>
          )}
        </div>
      </Container>
    </header>
  );
};

export default React.memo(Header);
```

Лістинг коду «Footer.jsx»

```
import { Link } from '@mui/material';
import { GitHub } from '@mui/icons-material';
import React from 'react';
import Container from '../Container';
import Divider from '../Divider';
import { InputField } from '../Fields';
import withSubscribe from
'../Fields/HOC/withSubscribe';
import Logo from '../Logo';

const Footer = () => {
  const SubscribeInput = withSubscribe(InputField);

  return (
    <footer className='footer'>
      <Container>
        <div className='footer-wrapper'>
          <div className='footer-item footer-item--
logo'>
            <div className='footer-logo'>

              <p className='footer-logo__text'>
                Бронювання номерів за допомогою системи
              </p>
            </div>
          </div>
```

```
          <div className='footer-item footer-item--
newsletter'>
            <form className='footer-newsletter'>
              <p className='footer-
newsletter__title'>Підписка</p>
              <span>Отримуйте спеціальні пропозиції та
новини сервісу</span>
              <div className='footer-
newsletter__input'>
                <SubscribeInput size='small'
placeholder='Email' name='email' type='email' />
              </div>
            </form>
          </div>
        </div>
      </Container>
      <Divider variant='fullWidth' className='footer-
divider' />
      <Container>
        <div className='footer-bottom'>
          <p className='footer-copyright'>Магістерська
робота</p>
          <Link underline='none'
href='https://github.com/Solexofficial' rel='noopener
noreferrer' target='_blank'>
            <div className='footer-social'>

              <span className='footer-
social__link'></span>
            </div>
          </Link>
        </div>
```

```
      </Container>
    </footer>
  );
};

export default React.memo(Footer);
```

ДОДАТОК Г

Лістинг коду «user.routes.js»

```
const express = require('express');
const User = require('../models/User');
const auth = require('../middleware/auth.middleware');
const router = express.Router({ mergeParams: true });

router.get('/', async (req, res) => {
  try {
    const users = await User.find();
    res.status(200).send(users);
  } catch (error) {
    res.status(500).json({
      message: 'На сервері виникла помилка',
    });
  }
});

router.patch('/:userId', auth, async (req, res) => {
  try {
    const { userId } = req.params;

    if (userId === req.user._id) {
      const updatedUser = await
User.findByIdAndUpdate(userId, req.body, { new: true
});
      res.send(updatedUser);
    } else {
      res.status(401).json({
```

```
        message: 'Unauthorized',
      });
    }
  } catch (error) {
    res.status(500).json({
      message: 'На сервері виникла помилка',
    });
  }
});

module.exports = router;
```