ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА КОМП'ЮТЕРНИХ НАУК

Кваліфікаційна робота

на тему: **«HOTEL BOOKING WEB-APPLICATION ВИКОРИСТОВУЮЧИ NEXTJS,EXPRESSJS,MONGODB»**

на здобуття освітнього ступеня бакалавра
зі спеціальності __122 Комп'ютерні науки_____
(код, найменування спеціальності)
освітньо-професійної програми __Комп'ютерні науки _____
(назва)

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

_____ ___Олександр Мочульський_____
(підпис) *Ім'я, ПРІЗВИЩЕ здобувача*

Виконав: здобувач вищої освіти гр. __КНД-42

___Олександр МОЧУЛЬСЬКИЙ___
(Ім'я, ПРІЗВИЩЕ)

Керівник:____Віктор ВИШНІВСЬКИЙ__
Науковий ступінь, вчене звання *(Ім'я, ПРІЗВИЩЕ)*

Рецензент: _____
Науковий ступінь, вчене звання *(Ім'я, ПРІЗВИЩЕ)*

Київ – 2025

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Комп'ютерних наук

Ступінь вищої освіти «Бакалавр»

Спеціальність 122 «Комп'ютерні науки»

Освітньо-професійна програма «Комп'ютерні науки»

ЗАТВЕРДЖУЮ
Завідувач кафедрою Комп'ютерних наук

_____ Віктор ВИШНІВСЬКИЙ
«_____» _____ 2025 р.

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

Мочульський Олександр Олександрович
(прізвище, ім'я, по батькові здобувача)

1. Тема кваліфікаційної роботи: Hotel web-application використовуючи NextJS,ExpressJS,MongoDB

керівник кваліфікаційної роботи Віктор ВИШНІВСЬКИЙ д.т.н, професор
(Ім'я, ПРІЗВИЩЕ науковий ступінь, вчене звання)

затверджені наказом Державного університету інформаційно-комунікаційних технологій від «24» лютого 2025 р. №56

2. Строк подання кваліфікаційної роботи «30» травня 2025 р.
3. Вихідні дані до кваліфікаційної роботи: Express.js documentation, MDN Web Docs, NextJS documentation, MongoDB Documentation, Visual Studio Code, Stack Overflow, GitHub
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
    1. Аналіз сучасних технологій веб-розробки
    2. Аналіз вибраних технологій для проекту
    3. Практична реалізацію веб додатку
    4. Тестування та оптимізація веб-додатку

5. Перелік графічного матеріалу:

1. Мета роботи, об’єкт та предмет дослідження
2. Налаштування серверу
3. Опис основного функціоналу
4. Середовище програмування
5. Первинні налаштування
6. Використання NextJS для фронтенд
7. Інтегрування MongoDB і бекенду
8. Адаптивне маштабування
9. Тестування клієнтської частини
10. Кінцеві правки перед релізом
11. Висновки 6.

Дата видачі завдання «25» лютого 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Літературний огляд сучасних технологій веб-розробки	25.02-22.03.25	Виконано
2	Формулювання проблеми та мети дослідження	23.03-30.03.25	Виконано
3	Аналіз інструментів та фреймворків для створення динамічних веб-інтерфейсів	31.03-09.04.25	Виконано
4	Розробка дизайну та проектування інтерфейсу користувача	10.04-14.04.25	Виконано
5	Реалізація фронтенд частини з використанням NextJS	15.04-29.04.25	Виконано
6	Інтеграція фронтенду з бекендом на Express.js і MongoDB	30.04-05.05.25	Виконано
7	Тестування користувацького інтерфейсу	06.05-11.05.25	Виконано
8	Оптимізація швидкості завантаження сторінок	12.05-15.05.25	Виконано
9	Написання тексту дипломної роботи	16.05-20.05.25	Виконано
10	Редагування та оформлення дипломної роботи	21.05-25.05.25	Виконано

Здобувач вищої освіти _____ Олександр МОЧУЛЬСЬКИЙ

(підпис) *(Ім’я, ПРІЗВИЩЕ)*

Керівник
кваліфікаційної роботи _____ Віктор ВИШНІВСЬКИЙ

(підпис) *(Ім’я, ПРІЗВИЩЕ)*

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 61 сторінок, 28 рисунків, 21 джерел.

Мета роботи – розробка веб-додатку для бронювання готельних номерів з використанням Next.js, Express.js та MongoDB, спрямованого на автоматизацію процесів резервування, керування номерами та забезпечення зручного інтерфейсу для гостей і адміністраторів.

Об'єкт дослідження – процеси створення, інтеграції та функціонування веб-додатків у сфері готельного бізнесу.

Предмет дослідження – інструменти та фреймворки для розробки вебдодатків, архітектура клієнт-серверної взаємодії, системи управління даними, питання безпеки та адаптивного інтерфейсу користувача.

Короткий зміст роботи:

у кваліфікаційній роботі проаналізовано сучасні підходи до створення онлайн-систем бронювання. Проведено аналіз функціональних і нефункціональних вимог, вибрано стек технологій (Next.js, Express.js, MongoDB), спроєктовано архітектуру системи, реалізовано веб-додаток із розподілом ролей користувача (гість, клієнт, адміністратор), впроваджено базовий захист даних та протестовано ключові функції додатку. Результатом стала масштабована система для бронювання готельних номерів із підтримкою адаптивного дизайну та сучасних вимог безпеки.

КЛЮЧОВІ СЛОВА**:** Next.js, Express.js, MongoDB, бронювання готелів, веб-додаток, REST API, адаптивний дизайн, авторизація, безпека даних, UI/UX.

ЗМІСТ

ВСТУП

Дана кваліфікаційна робота є кроком у напрямку розвитку сучасних веб-додатків для готельної індустрії, які забезпечують зручність, швидкість та надійність бронювання в умовах підвищеного попиту на цифрові сервіси.

Мета кваліфікаційної роботи:

Розробка веб-додатку для бронювання готельних номерів з використанням NextJS, ExpressJS та MongoDB, спрямованого на оптимізацію процесів пошуку, резервування та управління номерами для користувачів і адміністраторів.

Завдання дослідження:

- Вивчення сучасних підходів до створення систем бронювання, аналіз функціональних вимог до готельних веб-додатків.
- Дослідження існуючих рішень та технологій у сфері веб-розробки (NextJS, ExpressJS, MongoDB). Аналіз платформ для бронювання з метою визначення оптимального стеку технологій для реалізації проекту.
- Розробка архітектури додатку: формування вимог до гостьової частини, особистого кабінету користувача та панелі адміністратора.
- Реалізація функціонального прототипу, включаючи систему бронювання, управління номерами, аутентифікацію та взаємодію з базою даних.
- Тестування та оптимізація роботи додатку. Перевірка коректності бронювань, безпеки даних та юзабіліті інтерфейсів.
- Оцінка результату: аналіз ефективності розробленого рішення для задоволення потреб користувачів і готелів.

У процесі дослідження було вирішено поставлені завдання, що дозволило створити інтуїтивний та продуктивний веб-додаток для бронювання номерів, спрямований на автоматизацію та покращення якості послуг у готельному бізнесі.

Об’єкт дослідження – процеси розробки та функціонування веб-додатків для онлайн-бронювання готельних номерів, зокрема організація взаємодії між клієнтами, адміністраторами та системою керування даними.

Предмет дослідження – проектування та реалізація hotel booking web-application з використанням NextJS (фронтенд), ExpressJS (бекенд) і MongoDB (база даних). Дослідження фокусується на оптимізації таких ключових аспектів:

- автоматизація бронювань і перевірки доступності номерів,
- забезпечення зручності інтерфейсів для різних типів користувачів (гості, адміністратори),
- інтеграція сучасних веб-технологій для швидкості, безпеки та масштабованості.

Наукова новизна роботи полягає в наступному:

Комбінація сучасних технологій для готельного бізнесу. Використання NextJS (SSR, маршрутизація), ExpressJS (REST API) та MongoDB (гнучкість схеми даних) дозволяє створити продуктивний додаток, адаптований до динамічних потреб ринку.

Оптимізація процесу бронювання. Запропоновано рішення для миттєвої перевірки доступності номерів через інтеграцію календарних сервісів і реальні оновлення бази даних, що зменшує ймовірність конфліктів бронювань.

Специфіка українського ринку. Враховано особливості локальних готелів (наприклад, підтримка мультимовності, інтеграція з українськими платіжними системами).

Розширена адмін-панель. Вдосконалений інструментарій для управління номерами, бронюваннями та користувачами, що спрощує роботу персоналу.

Ці аспекти роблять розроблений додаток інноваційним рішенням для готельної галузі, яке поєднує технологічну ефективність із практичними вимогами бізнесу.

Ключові переваги запропонованого рішення:

Крос-платформеність: веб-додаток доступний з будь-якого пристрою (ПК, смартфон, планшет) без необхідності встановлення додаткового ПО.

Інтеграційні можливості: система може бути доповнена зовнішніми сервісами (наприклад, Google Maps для локації готелів, SendGrid для email-

сповіщень).

Масштабованість: архітектура на основі MongoDB та ExpressJS дозволяє збільшувати базу даних і функціонал без критичних змін коду.

Таким чином, результати дослідження сприятимуть розвитку цифрових рішень для готельної індустрії, пропонуючи інтуїтивний, швидкий і надійний інструмент для бронювань.

Тому було поставлено актуальне завдання розробити веб-додаток для бронювання готельних номерів з використанням NextJS, ExpressJS та MongoDB. Цей проект спрямований на оптимізацію роботи готелів, покращення взаємодії з клієнтами та автоматизацію ключових процесів.

Зручність бронювання для клієнтів – система надає зрозумілий інтерфейс для пошуку, перевірки доступності номерів та миттєвого оформлення замовлення, що скорочує час на резервацію.

Ефективне управління для готелів – адмін-панель дозволяє легко додавати/редагувати номери, переглядати бронювання та аналізувати завантаженість готелю.

Зниження операційних витрат – автоматизація процесів (наприклад, підтвердження бронювань через email, відміна резервацій) зменшує навантаження на персонал.

Практична значимість роботи полягає в наступному:

- Підвищення конкурентоспроможності готелю – сучасний веб-додаток із зручним інтерфейсом залучає більше клієнтів та покращує їхній досвід.
- Економія часу та ресурсів – автоматизована система усуває необхідність ручного керування бронюваннями (телефонні дзвінки, таблиці Excel тощо).
- Масштабованість – архітектура на основі MongoDB та ExpressJS дозволяє додавати нові функції (наприклад, інтеграція з платіжними системами, системами лояльності).
- Аналітика та планування – збір даних про бронювання допомагає

адміністраторам оптимізувати ціноутворення та маркетингові стратегії.

- Доступність 24/7 – клієнти можуть бронювати номери в будь-який час без участі персоналу.

Ці переваги демонструють, що розроблений веб-додаток не лише вирішує поточні проблеми готельного бізнесу, але й відкриває нові можливості для його розвитку в умовах цифровізації.

1 ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ СИСТЕМ БРОНЮВАННЯ ГОТЕЛЬНИХ НОМЕРІВ

1.1 Актуальність розробки веб-додатків для онлайн-бронювання номерів

У сучасному цифровому суспільстві веб-технології активно проникають у всі сфери діяльності, включаючи готельно-ресторанний бізнес. Зі зростанням популярності онлайн-сервісів і збільшенням кількості користувачів, які віддають перевагу самостійному плануванню подорожей, зростає потреба у зручних, швидких і надійних засобах для онлайн-бронювання готельних номерів. Розробка веб-додатків, орієнтованих на такі послуги, набуває особливої актуальності в умовах глобалізації, цифровізації та високої конкуренції на ринку туристичних послуг.

Системи онлайн-бронювання дозволяють готелям автоматизувати ключові бізнес-процеси, зокрема управління номерами, прийом бронювань, комунікацію з клієнтами та обробку платежів. У свою чергу, користувачі отримують доступ до інформації про доступні номери, можливість перевірити наявність місць на потрібні дати, оформити бронювання в реальному часі та керувати своїми замовленнями у зручному інтерфейсі.

Актуальність створення подібних веб-додатків зумовлена також стрімким розвитком мобільних технологій, що вимагає адаптивного дизайну, високої продуктивності та безпеки обробки персональних даних. Зважаючи на підвищені вимоги до надійності таких систем, особливу увагу слід приділяти архітектурі програмного забезпечення, правильному проєктуванню бази даних та ефективному розподілу функціоналу між фронтендом і бекендом.

У результаті, розробка сучасного веб-додатку для бронювання готельних номерів є актуальним завданням, яке поєднує у собі технічну складність, потребу в

зручності для користувача та дотримання вимог безпеки і надійності.

Крім того, зростаюча популярність онлайн-бронювання обумовлюється зміною поведінки споживачів, які прагнуть мінімізувати час і зусилля, необхідні для організації поїздок. Користувачі очікують, що веб-додатки будуть інтуїтивно зрозумілими, швидкими у використанні та надаватимуть максимум інформації для прийняття рішення. У цьому контексті важливою перевагою є можливість переглядати фото номерів, читати відгуки інших гостей, а також здійснювати фільтрацію за різними критеріями — ціна, зручності, розташування тощо.

З боку власників готелів, наявність власного веб-додатку для бронювання дає змогу зменшити залежність від сторонніх платформ (OTA — Online Travel Agencies), знизити комісійні витрати та підвищити контроль над взаємодією з клієнтами. Також, завдяки прямому бронюванню через власну систему, зростає лояльність клієнтів та ефективність маркетингових кампаній.

Окремо слід відзначити роль персоналізованих функцій, таких як особистий кабінет користувача, історія бронювань, можливість скасування чи зміни замовлення, а також інструменти адміністрування, що дозволяють керувати номерним фондом, переглядати статистику завантаженості та контролювати базу користувачів. Реалізація таких функцій потребує ретельного проєктування взаємодії між користувачем і системою, а також дотримання стандартів безпеки, особливо при роботі з особистими даними та платіжною інформацією.

Таким чином, розробка веб-додатку для онлайн-бронювання готельних номерів є не лише сучасною потребою, а й інструментом підвищення ефективності готельного бізнесу, зручності для клієнтів та конкурентоспроможності на ринку туристичних послуг. Актуальність цієї теми обумовлена як технологічними трендами, так і зростаючими очікуваннями з боку кінцевих користувачів.

1.2 Аналіз існуючих рішень для бронювання готелів: переваги та недоліки

У На сучасному ринку цифрових послуг представлено велику кількість систем для онлайн-бронювання готельних номерів. Найбільш відомими серед них є Booking.com, Airbnb, Expedia, Agoda, а також спеціалізовані веб-додатки окремих готельних мереж. Ці сервіси забезпечують мільйонам користувачів швидкий доступ до інформації про житло, його бронювання, оплату та відгуки — фактично, створюючи універсальні платформи для планування подорожей - рис. 1.1.

Рисунок 1.1 – Online Travel Agencies

Серед основних переваг таких рішень варто відзначити:

- Зручність для користувача — інтуїтивно зрозумілі інтерфейси, велика кількість фільтрів і опцій пошуку.
- Доступ до великої бази пропозицій — користувач може швидко порівняти сотні варіантів за різними параметрами.

- Інтеграція з платіжними системами — забезпечує можливість миттєвого бронювання та оплати.
- Підтримка багатомовності та мультиплатформенності — сервіси доступні на різних мовах і працюють як на ПК, так і на мобільних пристроях.
- Система рейтингів і відгуків — надає користувачам змогу приймати рішення на основі реального досвіду інших гостей.

Проте, попри значну кількість переваг, існують і певні недоліки в роботі подібних платформ:

- Високі комісії для власників готелів — платформи, як правило, утримують значну частку прибутку, що знижує рентабельність.
- Обмежений контроль над клієнтською базою — готелі не мають прямого доступу до особистих даних клієнтів для подальшої роботи з ними (наприклад, програми лояльності).
- Низька індивідуалізація — шаблонна структура інтерфейсу не враховує унікальність окремого готелю чи його бренду.
- Можливість виникнення конфліктів — через дублювання інформації або затримки у синхронізації на стороні агрегатора.
- Залежність від сторонньої платформи — у разі технічних збоїв чи змін у політиці роботи агрегатора готель втрачає повноцінний доступ до клієнтів.

Аналізуючи ці чинники, можна дійти висновку, що створення власного веб-додатку для бронювання дозволяє готелям не лише зменшити операційні витрати, а й підвищити рівень персоналізації обслуговування, забезпечити кращий контроль над даними та впровадити гнучкі функціональні рішення, адаптовані до конкретної бізнес-моделі.

Booking.com — одна з найвідоміших і найбільш використовуваних платформ для бронювання готелів у всьому світі. Вона пропонує як класичне готельне житло, так і апартаменти, хостели та вілли.

Переваги:

- Найбільша база пропозицій у готельному сегменті — доступ до сотень тисяч варіантів у понад 200 країнах.
- Простий і зручний інтерфейс для користувачів з широкими можливостями фільтрації.
- Відсутність передоплати у більшості пропозицій, гнучкі умови скасування.
- Велика кількість відгуків і рейтингових систем, що допомагає приймати обґрунтовані рішення.
- Потужна служба підтримки та локалізація багатьма мовами.

Недоліки:

- Високі комісії для готелів — до 15% з кожного бронювання.
- Залежність готелю від алгоритмів ранжування платформи, що може зменшити видимість об’єкта без додаткової оплати.
- Мала можливість персоналізації взаємодії з клієнтом — відсутність доступу до контактів клієнтів у повному обсязі.
- Перенасиченість пропозиціями, що ускладнює виділення індивідуального готелю без додаткових витрат.

Airbnb зосереджується переважно на подобовій оренді квартир, будинків, кімнат і нестандартного житла. Ця платформа стала популярною завдяки моделі "від людини до людини" (P2P).

Переваги:

- Унікальний контент — користувачі можуть орендувати не тільки квартири, а й екзотичне житло, наприклад, яхти чи будинки на деревах.
- Сильне відчуття персоналізації — система комунікації між гостем і хостом забезпечує індивідуальний підхід.
- Наявність програм захисту хостів та відгукової системи.
- Простий процес реєстрації і додавання об’єкта для власників житла.

Недоліки:

- Менш зручна платформа для готелів, оскільки більшість функцій заточені під приватних осіб.
- Високі комісії.
- Відсутність стандартів якості — оскільки більшість об'єктів не сертифіковані, є ризик отримати житло, яке не відповідає опису.
- Правові та податкові обмеження в окремих країнах або містах.

Expedia — глобальний онлайн-гігант у сфері бронювання, який охоплює не лише готелі, але й авіаквитки, оренду авто, тури тощо. Його основний ринок — Північна Америка.

Переваги:

- Комплексні пакети подорожей (готель + авіа + авто), що зручно для туристів.
- Велика база готелів і тісна співпраця з великими мережами.
- Можливість участі у програмі Expedia Group Partner Central, що надає аналітику та інструменти управління готелем.
- Стабільна репутація та надійна підтримка клієнтів.

Недоліки:

- Складніша реєстрація для власників житла — потребує підтверджень і узгоджень.
- Невигідні умови для невеликих готелів і приватних власників — домінування мережевих брендів.
- Висока конкуренція всередині платформи, що ускладнює просування без додаткових інвестицій.
- Модель з передоплатою, яка не завжди зручна для користувача.

Agoda — сервіс, який особливо популярний у країнах Азіатсько-Тихоокеанського регіону. Спеціалізується переважно на бронюванні готелів.

Переваги:

- Вигідні ціни для кінцевих користувачів завдяки знижкам і "таємним пропозиціям".
- Зручний мобільний додаток, оптимізований під швидкі бронювання.
- Глибока локалізація для азійських ринків, підтримка місцевих валют і мов.
- Промо-інструменти для готелів, що дозволяють керувати знижками та рейтингами.

Недоліки:

- Обмежене географічне охоплення поза межами Азії — менш популярна у Європі та Америці.
- Менше функцій для прямої комунікації з клієнтом.
- Менш прозора система комісій і тарифів — складна структура формування вартості.
- Залежність готелів від знижкової моделі, що іноді шкодить прибутковості.

Усі описані сервіси мають свою нішу та специфіку. Booking.com домінує серед класичних готелів, Airbnb — у сегменті приватної оренди, Expedia зручна для комплексного планування подорожей, а Agoda орієнтована на азійський ринок. Незважаючи на очевидні переваги для користувачів, кожна з платформ має суттєві обмеження для готельного бізнесу, особливо у частині контролю над клієнтами, прибутковістю та гнучкістю інтерфейсу.

Саме тому створення власного веб-додатку для бронювання номерів є логічною відповіддю на потребу в незалежності, індивідуальності та економічній ефективності, особливо для малих і середніх готелів або авторських проєктів.

Таким чином, розробка індивідуального веб-додатку є доцільною альтернативою масовим платформам, особливо для малих і середніх готельних бізнесів, які прагнуть до прямої взаємодії з клієнтами, а також для великих

готельних мереж, які мають потребу у власному корпоративному рішенні з повним контролем над функціональністю.

1.3 Основні вимоги до сучасних систем онлайн-бронювання

У сучасних умовах цифрової трансформації індустрії гостинності онлайн-системи бронювання номерів є не лише додатковим інструментом для залучення клієнтів, а й невід'ємною частиною бізнес-процесів готелів, хостелів та апартаментів. Вимоги до таких систем визначаються як очікуваннями користувачів, так і технологічними стандартами, що забезпечують ефективність, надійність і безпеку роботи - рис. 1.2.

Рисунок 1.2 – Пріорітетні направлення функціоналу

Функціональні вимоги визначають безпосередні можливості системи, які реалізуються на рівні інтерфейсу користувача та бекенд-логіки. До основних

функціональних вимог належать:

- Перегляд списку готельних номерів з короткою інформацією, фотографіями, цінами та фільтрацією за параметрами (ціна, тип номера, кількість місць тощо).
- Перехід до детальної інформації про номер, включаючи опис, зручності, політику скасування та наявність на конкретні дати.
- Перевірка доступності номера у вибраний період з динамічним відображенням результату.
- Реєстрація та авторизація користувача, включно з валідацією даних і збереженням сесії.
- Можливість створення бронювання з подальшим його збереженням у базі даних.
- Персональний кабінет користувача, де він може переглядати свої бронювання, скасовувати їх та редагувати особисті дані.
- Форма зворотного зв'язку або контактів для зв’язку з адміністрацією готелю.
- Додавання/редагування/видалення номерів;
- Перегляд списку бронювань;

Нефункціональні вимоги стосуються якості реалізації системи, її продуктивності, стабільності та зручності використання:

- Зручність і зрозумілість інтерфейсу (Usability) — сайт повинен бути інтуїтивно зрозумілим як для користувачів, так і для адміністратора.
- Швидкість завантаження сторінок і мінімальна затримка при запитах до серверу.
- Сумісність із мобільними пристроями — адаптивна верстка.
- Безпечність обробки персональних даних користувачів та захист від несанкціонованого доступу.
- Надійність і стійкість до збоїв — система повинна коректно обробляти помилки та забезпечувати резервне збереження даних.

- Масштабованість — можливість легкої адаптації під більшу кількість користувачів або готелів у разі розширення проєкту.

Технологічні вимоги

- З технічної точки зору, сучасна система онлайн-бронювання повинна відповідати таким критеріям:
- Використання сучасних фреймворків для frontend та backend розробки (Next.js для клієнтської частини та Express.js для серверної логіки).
- Підключення до надійної бази даних — у даному проєкті обрано MongoDB як гнучке та масштабоване рішення для зберігання інформації про номери, бронювання та користувачів.
- REST API або GraphQL API для забезпечення взаємодії між клієнтською та серверною частинами.
- Використання екосистеми Node.js, що дозволяє створювати високопродуктивні веб-додатки..

1.4 **Загрози безпеці користувацьких даних у подібних системах та способи їх усунення.**

Система У процесі розробки та експлуатації систем онлайн-бронювання готелів однією з ключових проблем залишається питання захисту персональних даних користувачів. Такі системи обробляють чутливу інформацію — імена, контактні дані, електронні адреси, іноді навіть платіжну інформацію, що робить їх потенційною ціллю для зловмисників. Загрози безпеці користувацьких даних можуть проявлятися у різних формах: від несанкціонованого доступу до бази даних до перехоплення трафіку під час передавання даних між клієнтською та серверною частинами застосунку.

Однією з найбільш поширених проблем є зберігання паролів у незашифрованому вигляді. Це створює серйозні ризики, адже в разі витоку бази зловмисники можуть безперешкодно отримати доступ до акаунтів користувачів. Для уникнення подібного сценарію у сучасних системах застосовуються

криптографічні алгоритми, зокрема хешування паролів із використанням солі. Таким чином, навіть у разі компрометації бази даних реальні паролі залишаються захищеними.

Ще однією потенційною загрозою є міжсайтовий скриптинг (XSS), коли через вразливості у формах введення зловмисник може вставити шкідливий JavaScript-код, який згодом виконається у браузері інших користувачів. Для протидії такій загрозі застосовується перевірка й екранування вхідних даних, що надходять від користувача, а також встановлення відповідних HTTP-заголовків безпеки.

Важливу роль у забезпеченні конфіденційності даних відіграє також шифрування передаваних даних за допомогою протоколу HTTPS. Усі запити між клієнтською частиною (браузером) і сервером повинні здійснюватися виключно через захищене з’єднання, що унеможливлює перехоплення інформації третіми особами.

Окрім технічних засобів захисту, варто враховувати й людський фактор. Недостатньо складні паролі, повторне використання одного й того ж пароля для кількох сервісів, або відкритий доступ до облікового запису адміністратора можуть призвести до серйозних витоків. Тому доцільним є впровадження політик автентифікації — наприклад, обмеження кількості спроб входу, використання CAPTCHA, двофакторної автентифікації, а також розмежування прав доступу між звичайними користувачами й адміністраторами.

Окрему увагу в системах, що функціонують у мережі Інтернет, слід приділити безпеці доменних імен та запитів до DNS-серверів. Незважаючи на те, що DNS-запити здаються другорядною частиною веб-взаємодії, саме вони часто є ціллю для атак типу DNS spoofing або cache poisoning, коли користувача навмисно перенаправляють на фальшивий ресурс, зовні схожий на справжній веб-додаток для бронювання.

Щоб захистити користувачів від таких атак, застосовуються сучасні технології, зокрема DNSSEC (Domain Name System Security Extensions). Ця технологія дозволяє перевірити справжність відповіді DNS-сервера,

використовуючи цифрові підписи. У разі підміни або спроби маніпулювати запитом браузер або інша клієнтська програма виявляє, що підпис недійсний, і припиняє з'єднання. DNSSEC не шифрує сам трафік, але гарантує достовірність відповіді, що вже значно знижує ризики переадресації на шахрайські сайти - рис. 1.3.

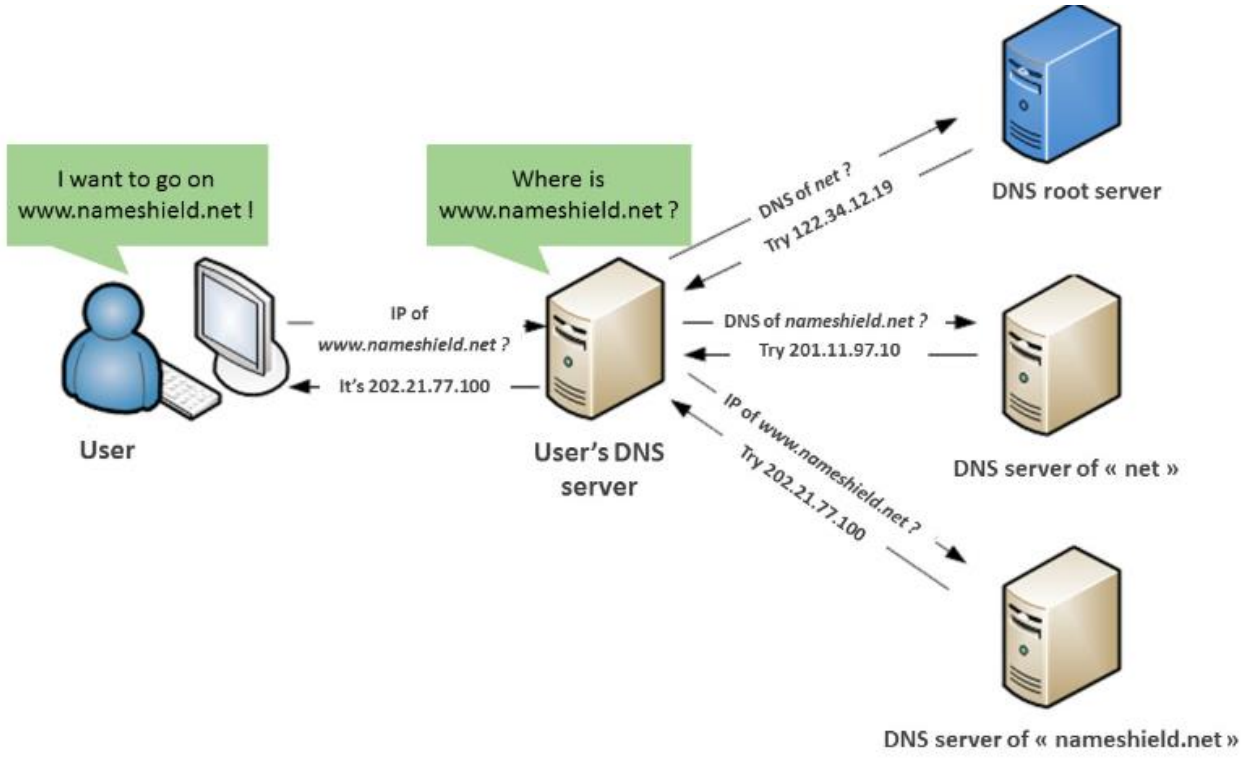


Рисунок 1.3 – Domain Name System Security

Ще один важливий захід — DNS over TLS (DoT), який забезпечує шифрування DNS-запитів на рівні транспортного протоколу. Це дозволяє уникнути того, що запити користувача до DNS (наприклад, під час переходу на сайт бронювання) будуть прочитані або змінені третіми особами. Завдяки DoT інформація про те, які саме ресурси відвідує користувач, не доступна для потенційного зловмисника або провайдера - рис. 1.4.

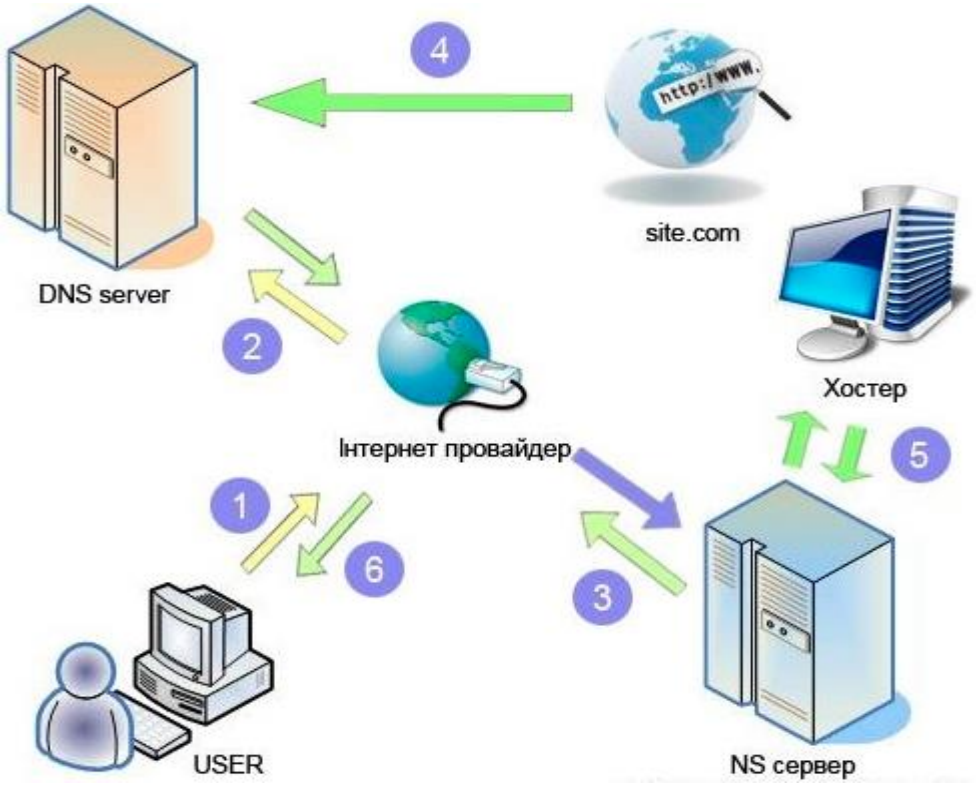


Рисунок 1.4 – DoT

Таким чином, забезпечення безпеки користувацьких даних у системах онлайн-бронювання вимагає комплексного підходу, який включає технічні, програмні та організаційні заходи. Лише за умови їх гармонійного поєднання можна гарантувати захищеність персональної інформації користувачів і забезпечити довіру до сервісу з боку цільової аудиторії.

1.5 **Інструменти та технології, які використовуються в сучасних рішеннях для готельного бронювання**

У сучасних системах онлайн-бронювання готелів застосовується широкий спектр інструментів і технологій, що забезпечують як зручність користувача, так і надійність, масштабованість та безпеку самого сервісу. Вибір конкретних засобів залежить від функціональних вимог, цільової аудиторії, навантаження на систему та потреб у подальшому розширенні.

На рівні клієнтської частини найбільш поширеним підходом є використання JavaScript-фреймворків і бібліотек, таких як React, Vue або Next.js. Зокрема, Next.js

надає гнучкість для побудови серверно-рендерених сторінок, що позитивно впливає на SEO-оптимізацію, швидкість завантаження сторінки та користувацький досвід. Вебінтерфейси розробляються з урахуванням принципів адаптивності, що дозволяє працювати з додатком як з комп’ютера, так і з мобільних пристроїв.

Для побудови серверної логіки активно застосовується Node.js з фреймворком Express.js, який забезпечує ефективну обробку HTTP-запитів, реалізацію REST API та інтеграцію з базами даних. Завдяки своїй подієво-орієнтованій моделі Express.js дозволяє створювати високонавантажені сервіси з низькою затримкою відповіді.

У якості системи керування базами даних зазвичай використовується MongoDB — документо-орієнтована база даних, що добре підходить для зберігання гнучких структур даних, таких як інформація про номери, бронювання чи користувачів. Вона легко масштабується та забезпечує високу швидкодію, що особливо важливо для динамічних систем бронювання.

На рівні безпеки активно застосовуються технології автентифікації та авторизації користувачів, зокрема, механізми на основі сесій або токенів (наприклад, JWT), хоча у навчальних або локальних проєктах можлива реалізація спрощеної моделі автентифікації. Також широко використовуються HTTPS-з'єднання, хешування паролів із використанням бібліотек типу bcrypt, а для захисту API-ендпоінтів — middleware-рішення для перевірки прав доступу.

Рішеннях для онлайн-бронювання номерів готелів інструменти й технології не обмежуються лише створенням інтерфейсів та обробкою запитів. Повноцінна система передбачає інтеграцію низки технологічних компонентів, які забезпечують її повсякденне функціонування, безперебійну роботу, масштабування та зручність як для кінцевого користувача, так і для адміністраторів.

Важливу роль у розвитку таких платформ відіграють хмарні технології. Завдяки сервісам на зразок Vercel, Render, Heroku, AWS, Google Cloud або Microsoft Azure, розробники отримують змогу автоматизовано розгортати додатки, масштабувати серверні ресурси в разі збільшення навантаження, та забезпечувати високу доступність системи. Це дозволяє невеликим командам реалізовувати проєкти з якістю, що відповідає сучасним корпоративним стандартам.

Ще одним важливим компонентом є інтеграція платіжних систем, таких як Stripe, PayPal, Fondy, які дозволяють користувачам здійснювати оплату бронювань у зручний спосіб. Впровадження таких сервісів потребує не лише технічної реалізації, а й дотримання стандартів безпеки (наприклад, PCI DSS), а також використання SSL-сертифікатів і безпечної передачі даних.

Окрему роль відіграють технології повідомлень та сповіщень. У багатьох системах для бронювання впроваджується механізм надсилання листів підтвердження електронною поштою, нагадувань про наближення дати заїзду чи повідомлень про зміни в статусі бронювання. Для цього використовуються служби типу SMTP-серверів, API сервісів на зразок SendGrid, Mailgun або Firebase Cloud Messaging.

Для забезпечення цілісності даних та захисту від їх втрати важливою є реалізація механізмів резервного копіювання. Більшість сучасних систем впроваджують автоматичне або ручне створення резервних копій бази даних і файлів. MongoDB, наприклад, підтримує такі механізми через MongoDB Atlas або за допомогою cron-скриптів для збереження даних у хмарному сховищі.

Для глибшої аналітики та прийняття рішень на основі даних у системи бронювання також вбудовуються модулі збору статистики, які дозволяють відстежувати завантаженість номерів, популярність дат, середній час перебування тощо. Для цього використовуються як сторонні сервіси (Google Analytics, Mixpanel), так і кастомні рішення на основі власних логів і агрегованих даних.

Не менш важливою складовою сучасних систем є інтеграція з картографічними сервісами (наприклад, Google Maps), яка дозволяє користувачу легко оцінити розташування готелю та його віддаленість від ключових об’єктів. Це не лише покращує зручність, а й підвищує довіру до платформи.

Також набирає популярності впровадження елементів штучного інтелекту, які використовуються для персоналізованих рекомендацій номерів, аналізу відгуків користувачів або автоматичної підтримки через чат-боти. Для цього можуть застосовуватись бібліотеки Python (наприклад, Scikit-learn, TensorFlow) або готові AI-платформи.

Усі ці компоненти інтегруються в єдину систему через API або мікросервіси, що дозволяє забезпечити модульність, гнучкість та розширюваність системи. Таким чином, технологічний стек сучасної системи онлайн-бронювання значно ширший, ніж традиційні засоби веброзробки, й охоплює цілу екосистему інструментів для підтримки повноцінного функціонування сервісу на практиці.

Таким чином, сучасні рішення для онлайн-бронювання номерів будуються на поєднанні перевірених вебтехнологій, гнучких фреймворків і засобів забезпечення безпеки, що в сукупності формують надійну та масштабовану архітектуру вебзастосунку.

2 ТЕОРЕТИЧНИЙ АНАЛІЗ РОЗРОБКИ HOTEL BOOKING WEB-APPLICATION

2.1 Аналіз функціональних і нефункціональних вимог до системи

Під час розробки програмного забезпечення одним з ключових етапів є визначення вимог до системи. Цей процес дозволяє сформулювати чітке розуміння того, що саме має виконувати система (функціональні вимоги), а також якими характеристиками вона повинна володіти для ефективної, стабільної та безпечної роботи (нефункціональні вимоги).

Функціональні вимоги описують конкретні дії, які має виконувати система, реакції на ті чи інші запити користувача, обробку даних та бізнес-логіку. Вони визначають, які сервіси буде надано, хто може ними користуватись, і як відбувається взаємодія з користувачем та адміністратором. Іншими словами, це вимоги до поведінки системи.

Нефункціональні вимоги, у свою чергу, стосуються якості функціонування системи. Вони не описують конкретні функції, але регламентують параметри надійності, продуктивності, зручності використання, безпеки, масштабованості, часу відгуку тощо.

Функціональні вимоги:

1.1. Гостьова частина

- Система має надавати можливість перегляду списку готельних номерів з основними параметрами (тип номера, ціна, фото).
- Користувач може переглядати детальну інформацію про номер (опис, зручності, фотогалерея, відгуки).
- Система має надавати функціонал перевірки доступності номерів на конкретні дати (календар бронювань).
- Користувач може надсилати повідомлення через форму контактів (збереження у БД).

- Система повинна мати функції реєстрації (email, пароль, ім’я) та входу (автентифікація через JWT або сесії).
- Після входу користувач може створювати бронювання (вибір дат, підтвердження, оплата або резервація без оплати).

1.2. Користувач (Особистий кабінет)

- Користувач може переглядати список своїх активних бронювань.
- Користувач може скасовувати бронювання.
- Користувач може редагувати свій профіль (зміна email, пароля, контактних даних).

1.3. Адміністратор (Панель управління)

- Адміністратор може додавати, редагувати та видаляти готельні номери (фото, опис, ціна, доступність).
- Адміністратор може переглядати всі бронювання.

2. Нефункціональні вимоги:

2.1. Продуктивність

- Час завантаження сторінок має бути ≤ 2 сек (оптимізація NextJS, кешування).
- Система повинна витримувати ≥ 1000 одночасних користувачів (масштабування бекенду).

2.2. Безпека

- Всі дані користувачів (паролі, бронювання) мають бути захищені (HTTPS, хешування паролів).
- Доступ до адмін-панелі лише через авторизацію з рівнем доступу "admin".

2.3. Юзабіліті (Зручність використання)

- Інтерфейс має бути адаптивним (коректне відображення на ПК, планшетах, смартфонах).

2.4. Надійність

- Система має автоматично резервувати номери під час бронювання (уникнення дублювань).

- Регулярне резервне копіювання бази даних (на випадок збоїв).

2.5. Сумісність

- Підтримка останніх версій браузерів (Chrome, Firefox, Safari, Edge).
- Можливість інтеграції з платіжними системами (Stripe, LiqPay).

2.6. Масштабованість

- Можливість додавання нових функцій (наприклад, система лояльності, відгуки).

2.2 Візуалізація функцій та взаємодії між компонентами системи

Перед початком робіт, потрібно провести візуалізацію функціональних елементів. Також описати їх взаємодію між користувачами на основі блок схем.

Спочатку потрібно створити UML-діаграму взаємодії. UML це стандартизована мова моделювання, яка використовується для візуального опису, проектування та документування програмних систем. Вона допомагає розробникам, аналітикам та іншим зацікавленим сторонам зрозуміти структуру, поведінку та взаємодію компонентів системи на різних рівнях абстракції.

Вона бла розроблений з метою уніфікації різних методів об’єктно-орієнтованого моделювання, що існували раніше, і наразі є одним з найпоширеніших засобів для моделювання програмних продуктів. Він містить набір різноманітних діаграм, кожна з яких виконує свою специфічну роль у процесі розробки: від опису структури даних до відображення послідовності операцій або взаємодії між користувачами та системою.

В контексті розробки веб-додатка для онлайн-бронювання готельних номерів UML-діаграми дозволяють наочно продемонструвати основні функції системи, ролі користувачів, їхню взаємодію з різними компонентами, а також бізнес-логіку, що стоїть за кожною операцією. Такий підхід сприяє кращому розумінню вимог, покращує комунікацію між членами команди розробників і допомагає уникнути помилок на ранніх стадіях проєкту.

Зокрема, діаграми прецедентів (Use Case Diagrams) використовуються для ілюстрації взаємодії між акторами (користувачами або іншими системами) та функціями системи. Вони чітко показують, які дії доступні для кожного типу користувача, що полегшує аналіз та планування подальшої розробки - рис. 2.1.

Отже, застосування UML-діаграм у даній роботі дозволяє створити структурований та наочний опис архітектури веб-застосунку, забезпечуючи основу для його успішної реалізації та подальшої підтримки.

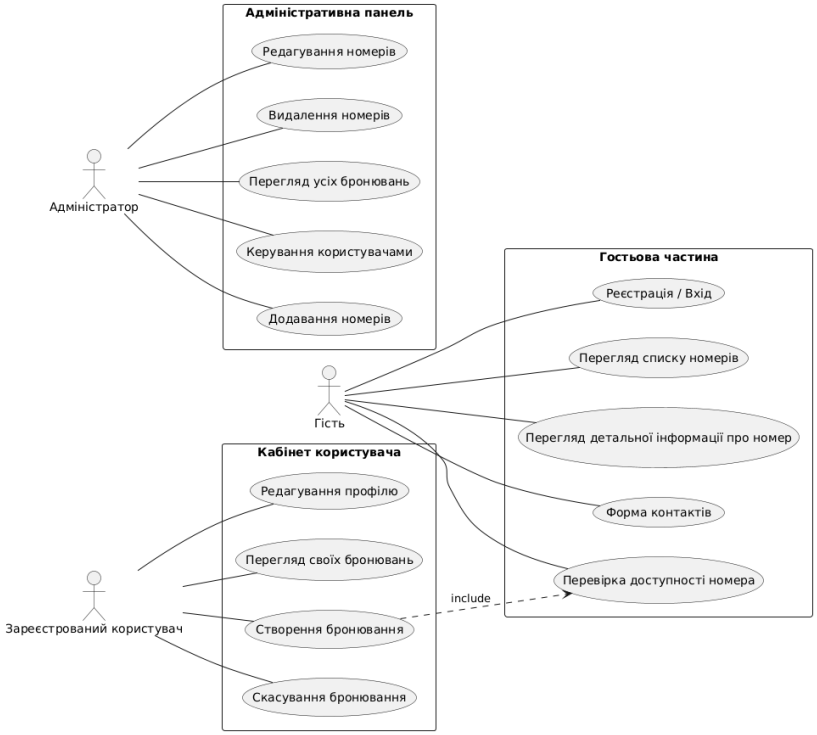


Рисунок 2.1 – UML діаграма

Ця діаграма показує три основні ролі — гість, зареєстрований користувач і адміністратор — та їх основні дії на сайті. Вона також показує, що створення бронювання включає перевірку доступності номера.

Архітектурна блок-схема відображає ключові компоненти веб-додатку для бронювання готельних номерів та їх взаємодію. Система складається з кількох основних модулів: клієнтської частини, серверної частини, бази даних і зовнішніх сервісів. Клієнтська частина реалізована на Next.js і відповідає за інтерфейс користувача, включно з гостьовою частиною, кабінетом користувача та адміністративною панеллю. Серверна частина на Express.js обробляє запити

клієнтів, виконує бізнес-логіку, а також взаємодіє з базою даних MongoDB для збереження інформації про користувачів, номери і бронювання. Крім того, система може інтегруватися з зовнішніми сервісами, наприклад, платіжними шлюзами для оплати бронювань або службами повідомлень для інформування користувачів - рис. 2.2.

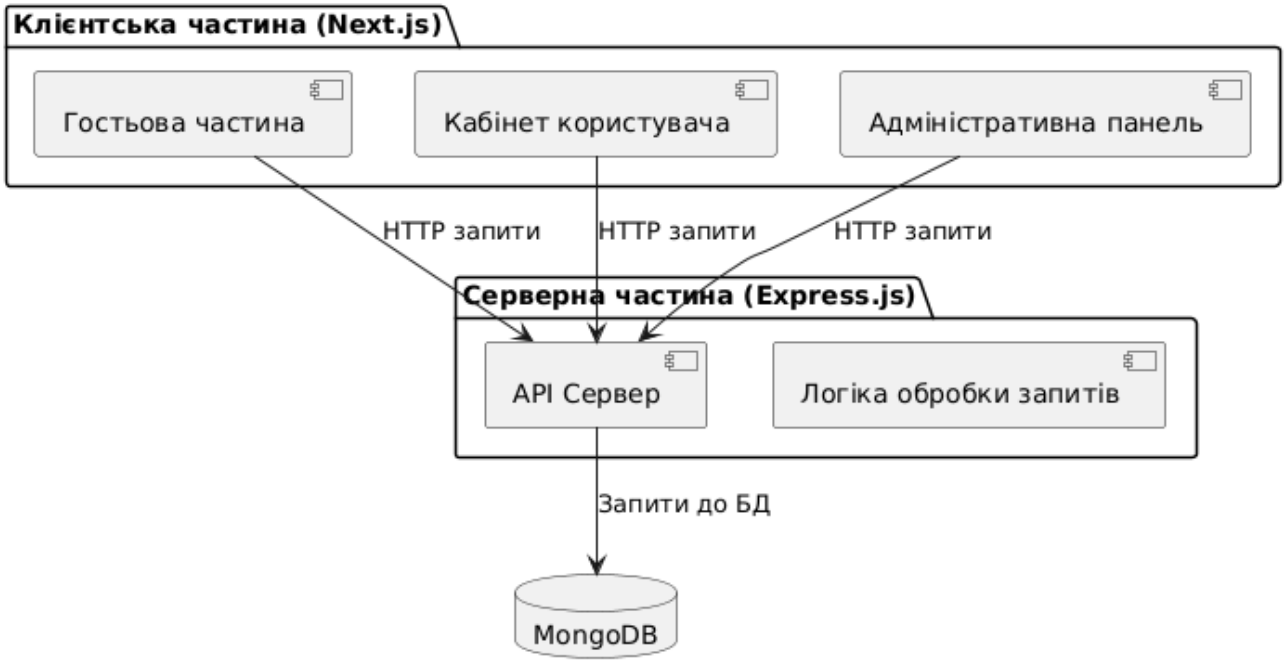


Рисунок 2.2 – Архітектурна додатку

Діаграма активності, подана нижче, відображає повну послідовність дій, які виконує звичайний користувач у веб-додатку для онлайн-бронювання готелів. Вона охоплює типовий життєвий цикл взаємодії з системою: від першого відвідування сайту до перегляду та редагування персональної інформації в особистому кабінеті.

Процес розпочинається з того, що користувач відкриває головну сторінку сайту, переглядає доступні номери та вивчає детальну інформацію про них. Далі він вибирає дати для бронювання, після чого система перевіряє, чи користувач авторизований. Якщо так — перевіряється доступність номера, і в разі позитивного результату створюється бронювання. Якщо номер зайнятий — система інформує користувача про недоступність. Якщо ж користувач не авторизований, система пропонує йому пройти реєстрацію або вхід.

Після успішної авторизації користувач може завершити бронювання, а також перейти до особистого кабінету, де він має змогу переглядати свої попередні замовлення, а за потреби — редагувати персональні дані. Таким чином, ця діаграма дозволяє візуально зрозуміти ключові сценарії використання додатку, логіку обробки запитів та умови переходів між різними станами взаємодії - рис. 2.3.

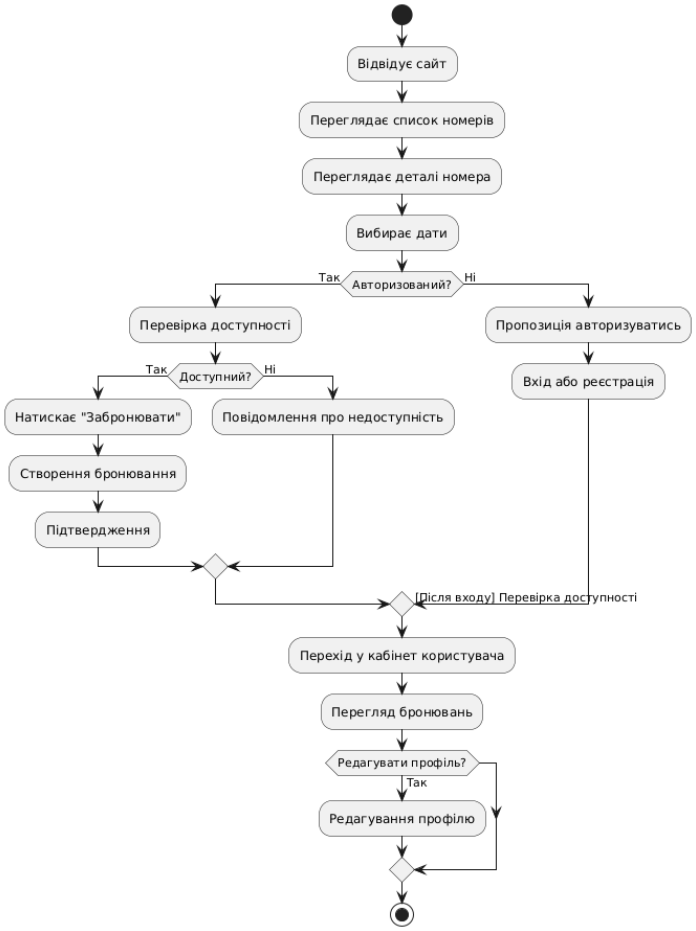


Рисунок 2.3 – Алгоритм роботи з сервісом

Блок-схема процесу бронювання номера демонструє послідовність кроків, які виконує система при оформленні замовлення користувачем. Цей процес починається з вибору номера, перевірки його доступності на вказані дати, а також авторизації користувача. Якщо всі умови виконано, система створює бронювання та зберігає його у базі даних, після чого користувачу відображається підтвердження. У випадку помилок або невідповідностей користувач отримає відповідне повідомлення. Така блок-схема дозволяє краще зрозуміти логіку взаємодії між клієнтом, сервером і базою даних.Блок-схема процесу бронювання

номера демонструє послідовність кроків, які виконує система при оформленні замовлення користувачем.

Цей процес починається з вибору номера, перевірки його доступності на вказані дати, а також авторизації користувача. Якщо всі умови виконано, система створює бронювання та зберігає його у базі даних, після чого користувачу відображається підтвердження. У випадку помилок або невідповідностей користувач отримає відповідне повідомлення. Така блок-схема дозволяє краще зрозуміти логіку взаємодії між клієнтом, сервером і базою даних - рис. 2.4.

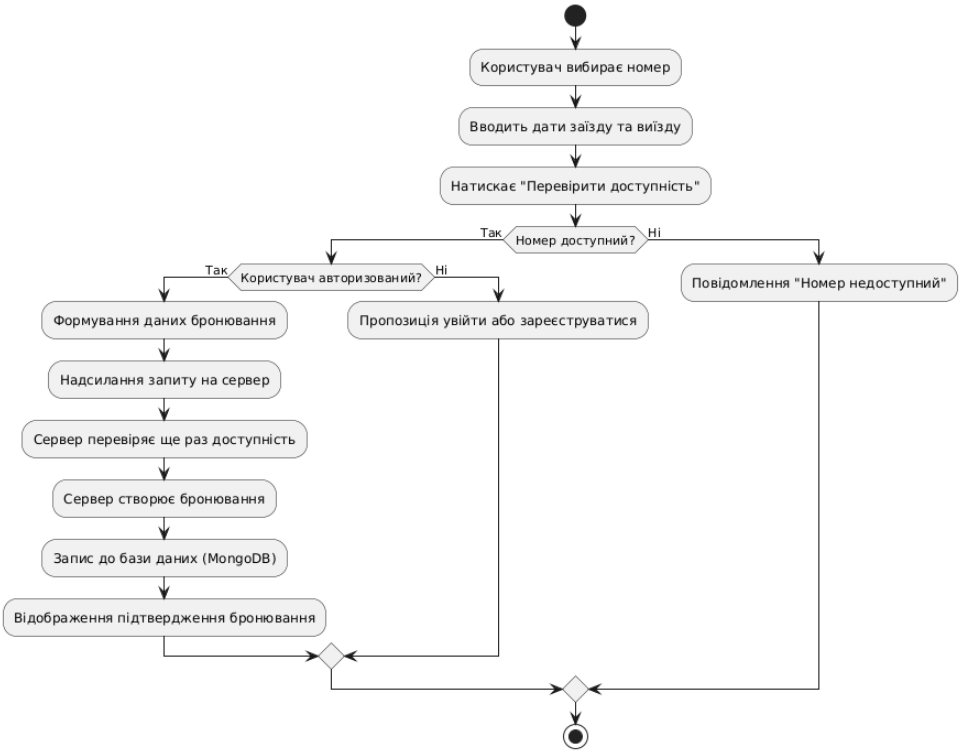


Рисунок 2.4 – Бронювання номеру

Завдяки створеним UML-діаграмам і блок-схемам було візуалізовано архітектуру системи, логіку дій користувача та внутрішні процеси, зокрема — бронювання номерів. Це дозволяє не лише краще зрозуміти загальну структуру і взаємозв’язки компонентів системи, а й забезпечити чітке планування її подальшої розробки.

Отже, виконана підготовча робота створює міцне підґрунтя для реалізації ефективного веб-додатку для бронювання готельних номерів, що відповідає сучасним вимогам як щодо функціональності, так і щодо безпеки.

2.3 Обґрунтування вибору засобів розробки: Next.js, Express.js, MongoDB

У процесі розробки веб-додатку для бронювання готельних номерів було ретельно проаналізовано сучасні технології та обрано оптимальний стек: Next.js для фронтенду, Express.js для бекенду та MongoDB як систему керування базами даних. Цей вибір ґрунтується на конкретних вимогах проекту та перевагах кожної з технологій.

Next.js був обраний як основний фреймворк для фронтенду через його здатність забезпечувати швидке завантаження сторінок завдяки серверному рендерингу (SSR) та статичній генерації. Це особливо важливо для додатку бронювань, де користувачі очікують миттєвого відгуку системи. Вбудована підтримка API-маршрутів дозволяє ефективно обробляти запити на перевірку доступності номерів, а оптимізована робота з зображеннями забезпечує гарне відображення фотографій номерів на будь-яких пристроях. Порівняно з чистим React, Next.js пропонує кращі можливості для SEO, що є критичним для гостинних додатків. Переваги:

- SSR (Server-Side Rendering) та статична генерація
- Забезпечує швидке завантаження сторінок (SEO-оптимізація), що критично для бронювань, де клієнти очікують миттєвого відгуку.
- Дозволяє попередньо завантажувати дані (наприклад, список номерів) для зменшення часу очікування.
- Вбудований API-маршрутизатор
- Спрощує створення ендпоінтів (наприклад, для перевірки доступності номерів) без додаткового бекенду.
- Оптимізація зображень (Image Component)
- Автоматичне налаштування розмірів фото номерів для різних пристроїв, що підвищує юзабіліті.

- Легка інтеграція з бекендом
- Next.js добре працює з Express.js через API-роути, що спрощує передачу даних (наприклад, бронювань).

Для бекенд-розробки було обрано Express.js - мінімалістичний та гнучкий фреймворк на Node.js. Він ідеально підходить для створення REST API, яке обслуговуватиме всі основні операції додатку: обробку бронювань, управління користувачами та роботу з номерами. Express.js дозволяє легко інтегрувати middleware для забезпечення безпеки, включаючи аутентифікацію за допомогою JWT та валідацію вхідних даних. Його архітектура дозволяє ефективно обробляти велику кількість одночасних запитів, що важливо під час сезону активних бронювань. Переваги:

- Мінімалістичний фреймворк дозволяє швидко створювати REST API для обробки бронювань, користувачів та номерів.
- Масштабованість
- Дозволяє обробляти сотні запитів одночасно.
- Легка інтеграція з MongoDB
- Завдяки драйверам (наприклад, Mongoose) можна ефективно працювати з даними.
- Middleware для безпеки
- Валідація даних, JWT-аутентифікація та обмеження запитів (rate limiting) запобігають атакам.

В якості основної бази даних обрано MongoDB - документоорієнтовану NoSQL систему. Вона ідеально підходить для гостинних додатків завдяки своїй гнучкості та відсутності жорсткої схеми даних. Це дозволяє легко додавати нові поля та характеристики до опису номерів без необхідності зміни структури бази. MongoDB ефективно працює з даними у форматі JSON, що ідеально поєднується з JavaScript-стеком додатку. Система показує високу продуктивність при операціях читання та запису, що критично важливо для часто оновлюваних даних про бронювання. Можливість горизонтального масштабування дозволяє системі

зростати разом із бізнесом, обслуговуючи все більшу кількість бронювань. Переваги:

- Немає жорсткої структури, що дозволяє легко додавати нові поля (наприклад, зручності номерів).
- Швидкість роботи з даними
- Оптимізована для операцій читання/запису, що важливо для часто оновлюваних бронювань.
- JSON-подібний формат
- Ідеально підходить для JavaScript-стеку (Next.js + Express.js).
- Підтримує горизонтальне масштабування для великої кількості бронювань.

Вибір цього технологічного стеку дозволяє створити сучасний, продуктивний та масштабований додаток для бронювання готельних номерів. Використання JavaScript на всіх рівнях додатку (фронтенд, бекенд, база даних) значно спрощує процес розробки та підтримки. Next.js забезпечує швидкий та зручний інтерфейс для користувачів, Express.js обробляє бізнес-логіку та API, а MongoDB надійно зберігає всі необхідні дані - рис. 2.5.

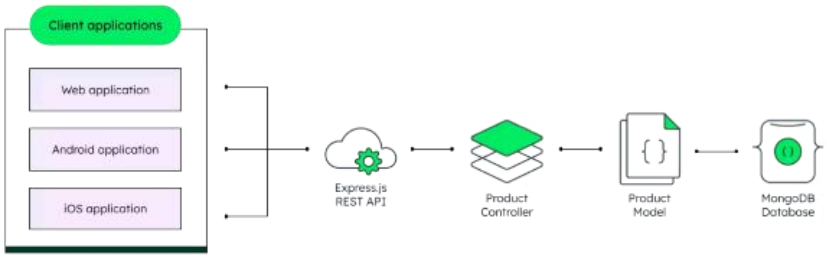


Рисунок 2.5 – Логічна модель додатку

Така комбінація технологій оптимально підходить для реалізації всіх функціональних вимог проекту, забезпечуючи високу продуктивність, безпеку та зручність використання.

2.4 Проєктування маршрутизації та логіки між фронтендом і бекендом

Проєктування маршрутизації та логіки взаємодії між фронтендом і бекендом є ключовим етапом у розробці будь-якого веб-застосунку. У рамках даного дипломного проєкту використано сучасний стек технологій: Next.js — для побудови фронтенду, Express.js — як серверну частину, та MongoDB — для збереження даних.

Маршрутизація у даній системі побудована за принципом розділення відповідальності між клієнтською (frontend) та серверною (backend) частинами. Фронтенд відповідає за відображення інтерфейсу, взаємодію з користувачем, відправку HTTP-запитів та отримання відповіді від серверу. Бекенд обробляє запити, працює з базою даних, здійснює валідацію та реалізує бізнес-логіку додатку.

З боку фронтенду (Next.js) було реалізовано наступні маршрути:

- / – головна сторінка з відображенням списку номерів;
- /room/[id] – сторінка перегляду деталей конкретного номера;
- /login та /register – маршрути для аутентифікації користувачів;
- /account – особистий кабінет користувача;
- /admin – інтерфейс адміністратора (з обмеженим доступом).

На бекенді (Express.js) реалізовано REST API з такими основними маршрутами:

- GET /api/rooms — отримання списку всіх номерів;
- GET /api/rooms/:id — отримання інформації про конкретний номер;
- POST /api/bookings — створення нового бронювання;
- GET /api/bookings/user/:id — отримання бронювань певного користувача;
- DELETE /api/bookings/:id — скасування бронювання;
- POST /api/users/login та POST /api/users/register — аутентифікація;
- PUT /api/users/:id — оновлення профілю користувача;

- POST /api/rooms та DELETE /api/rooms/:id — маршрути для додавання та видалення номерів (доступні лише адміністратору).

Комунікація між клієнтом і сервером реалізується через HTTP-запити, які відправляються з фронтенду за допомогою fetch або бібліотеки axios. У відповідь сервер надсилає дані у форматі JSON. Для збереження стану користувача використовується локальне сховище (localStorage), де зберігається ID поточного користувача - рис. 2.6.

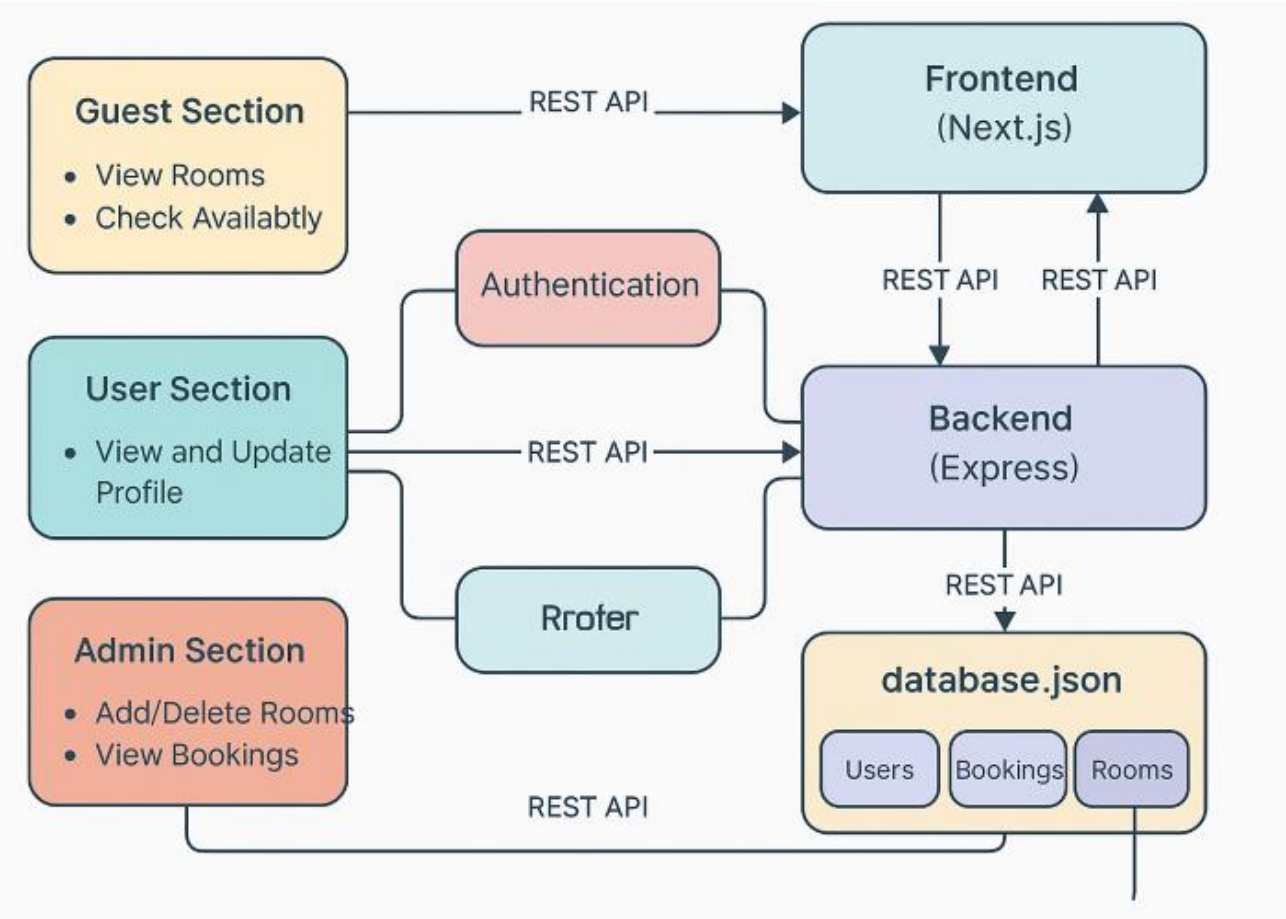


Рисунок 2.6 – Алгоритм роботи маршрутизації

Завдяки чіткому розділенню обов’язків між клієнтською та серверною логікою, система легко масштабовується, забезпечує хорошу модульність коду і спрощує процес тестування. Такий підхід також дозволяє реалізувати додаткові функції без суттєвого переписування вже існуючого функціоналу, що є перевагою для подальшого розвитку системи.

2.5 Обґрунтування архітектури даних і безпеки користувачів.

Проєктування архітектури даних є критично важливим етапом розробки будь-якого сучасного веб-додатку. У випадку системи онлайн-бронювання готелів структура зберігання, обробки та доступу до даних безпосередньо впливає на стабільність роботи, масштабованість та рівень безпеки проєкту. Основними об'єктами, що потребують надійної організації, є дані про користувачів, номери, бронювання, а також історія взаємодій із системою.

Для реалізації цього веб-додатку було обрано базу даних MongoDB, яка є документо-орієнтованою та надає гнучкий механізм зберігання JSON-подібних документів. Її структура дозволяє швидко масштабувати проєкт та адаптуватися до змін у структурі даних без потреби модифікувати всю базу. Дані зберігаються у трьох основних колекціях: users, rooms і bookings, що забезпечує логічне розділення та полегшує обробку запитів.

З архітектурної точки зору, розподіл на Frontend (Next.js) та Backend (Express.js) дозволяє організувати чітке розмежування обов'язків між візуальним інтерфейсом та логікою обробки запитів, що спрощує підтримку та розвиток проєкту. Вся взаємодія між клієнтською частиною та сервером відбувається через REST API, що дозволяє централізовано контролювати авторизацію, перевірку прав доступу та обробку даних.

Особлива увага приділяється захисту персональних даних користувачів. Для авторизації реалізовано просту систему логіну з валідацією паролів, що зберігаються у хешованому вигляді за допомогою бібліотеки bcrypt. Ідентифікація користувача в межах сесії забезпечується через збереження його ідентифікатора у localStorage або sessionStorage, що дозволяє реалізувати базову автентифікацію без використання складніших механізмів, таких як JWT.

Додатково передбачені такі заходи безпеки:

- Розмежування доступу до функціоналу залежно від ролі користувача (гість, зареєстрований користувач, адміністратор);
- Валідація даних як на фронтенді, так і на бекенді для запобігання SQL-ін’єкціям, XSS-атакам та передачі некоректних даних;
- Шифрування чутливих даних, таких як паролі користувачів;

Окрім внутрішньої безпеки, система враховує загрози, пов’язані з передачею даних мережею. Для цього можна передбачити використання HTTPS як основного протоколу передачі, а також впровадження захисту DNS-запитів через механізми DNS over TLS (DoT) або DNSSEC, що знижує ризик атаки типу «людина посередині» при розв’язанні доменів.

Таким чином, архітектура даних у розробленій системі дозволяє досягти балансу між функціональністю, масштабованістю та безпекою, що є ключовим для ефективної роботи онлайн-платформи з бронювання готельних номерів - рис. 2.7.

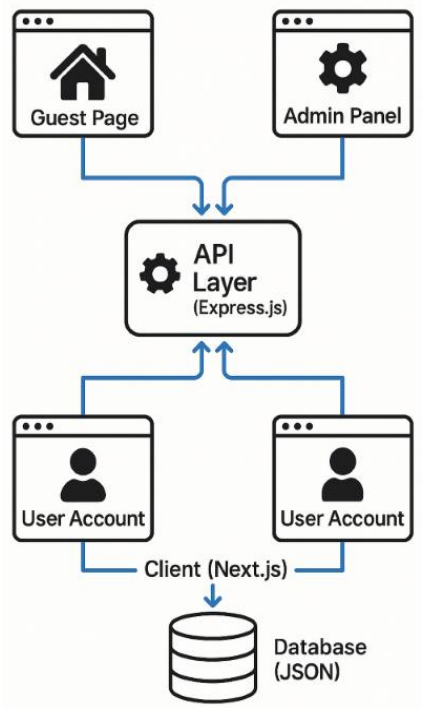


Рисунок 2.7 – Архітектурні одиниці доступу

3 ПРАКТИЧНА РЕАЛІЗАЦІЯ HOTEL BOOKING WEB-APPLICATION

3.1 Розробка користувацького інтерфейсу

Розробка користувацького інтерфейсу (UI) є одним з ключових етапів створення сучасного веб-додатку. Інтерфейс не лише визначає зручність взаємодії користувача із системою, але й значною мірою впливає на загальне враження від використання додатку.

Інтерфейс розділено на кілька основних логічних частин відповідно до ролей користувачів:

Гостьова частина (Public Pages) — доступна всім відвідувачам сайту без авторизації. Включає:

- Головну сторінку з переліком доступних номерів;
- Сторінку з деталями окремого номера;
- Перевірки доступності за датами;
- Сторінки реєстрації та авторизації.

Особистий кабінет користувача — доступний після входу в систему. Реалізовано функції:

- Перегляд особистих бронювань;
- Можливість скасування або зміни бронювання;
- Редагування профілю (ім’я, пароль).

Адміністративна панель — доступна лише користувачам з правами адміністратора - рис. 3.1. Дозволяє:

- Додавати, редагувати або видаляти готельні номери;
- Переглядати список усіх бронювань;
- Переглядати список зареєстрованих користувачів.

```css
@import "tailwindcss";

:root {
  --background: #ffffff;
  --foreground: #171717;
}

@theme inline {
  --color-background: var(--background);
  --color-foreground: var(--foreground);
  --font-sans: var(--font-geist-sans);
  --font-mono: var(--font-geist-mono);
}

@media (prefers-color-scheme: dark) {
  :root {
    --background: #0a0a0a;
    --foreground: #ededed;
  }
}

body {
  background: var(--background);
  color: var(--foreground);
  font-family: Arial, Helvetica, sans-serif;
}

html, body, #__next {
  height: 100%;
  margin: 0;
  padding: 0;
}
```

Рисунок 3.1 – Налаштування базових кольрів інтерфейсу

Для реалізації інтерфейсу було обрано Next.js — фреймворк на основі React, який забезпечує серверний рендеринг, високу продуктивність і підтримку SEO. Це дозволяє створювати масштабовані й ефективні веб-застосунки зі швидкою реакцією на дії користувача.

Усі компоненти інтерфейсу побудовані із дотриманням принципів адаптивного дизайну, що гарантує коректну роботу на різних типах пристроїв — від комп’ютерів до мобільних телефонів.

У випадку веб-застосунку для бронювання готельних номерів важливо створити інтуїтивно зрозуміле, естетично привабливе та функціональне середовище - рис. 3.2.

```
import Link from 'next/link';
import { useEffect, useState } from 'react';
import 'bootstrap/dist/css/bootstrap.min.css';

export default function Home() {
  const [rooms, setRooms] = useState([]);
  const [user, setUser] = useState(null);

  useEffect(() => {
    fetch('/api/rooms')
      .then(res => res.json())
      .then(data => setRooms(data));

    const stored = localStorage.getItem('currentUser');
    if (stored) {
      setUser(JSON.parse(stored));
    }
  }, []);

  const handleLogout = () => {
    localStorage.removeItem('currentUser');
    window.location.reload();
  };

```

Рисунок 3.2 – Точка входа index.js

Яке дозволяє користувачеві швидко знайти потрібну інформацію, здійснити бронювання, переглянути власні дані та керувати своїм профілем - рис. 3.3.

```
  return (
<div className="d-flex flex-column min-vh-100">
      {/* Хедер */}
      <header>
        <nav className="navbar navbar-expand-lg navbar-dark bg-dark shadow-sm">
          <div className="container">
            <Link href="/" className="navbar-brand fs-4 fw-bold">
              <span className="text-primary">Hotel</span>App
            </Link>
            <button
              className="navbar-toggler"
              type="button"
              data-bs-toggle="collapse"
              data-bs-target="#navbarNav"
              aria-controls="navbarNav"
              aria-expanded="false"
              aria-label="Toggle navigation"
            >
              <span className="navbar-toggler-icon"></span>
            </button>
            <div className="navbar-collapse" id="navbarNav">
              <ul className="navbar-nav ms-auto">
                {!user ? (
                  <>
                    <li className="nav-item">
                      <Link href="/register" className="nav-link">
                        Реєстрація
                      </Link>
                    </li>
                    <li className="nav-item">
                      <Link href="/login" className="nav-link btn btn-primary text-white ms-lg-2">
                        Вхід
                      </Link>
                    </li>
                  </>
                ) : (
                  <>
                    <li className="nav-item">
                      <Link href="/my-bookings" className="nav-link">
                        Мої бронювання
                      </Link>
                    </li>
                    <li className="nav-item">
                      <Link href="/edit-profile" className="nav-link">
                        Редагувати профіль
                      </Link>
                    </li>
                    <li className="nav-item">
                      <button
                        className="btn btn-outline-secondary nav-link ms-lg-2"
```

Рисунок 3.3 – Верстка головної сторінки

3.2 Реалізація гостьової частини

Гостьова частина веб-додатку є першим рівнем взаємодії користувача із системою бронювання. Вона призначена для надання загальнодоступної інформації про готельні номери, а також забезпечує базову навігацію, реєстрацію нових користувачів і вхід у систему - рис. 3.4.

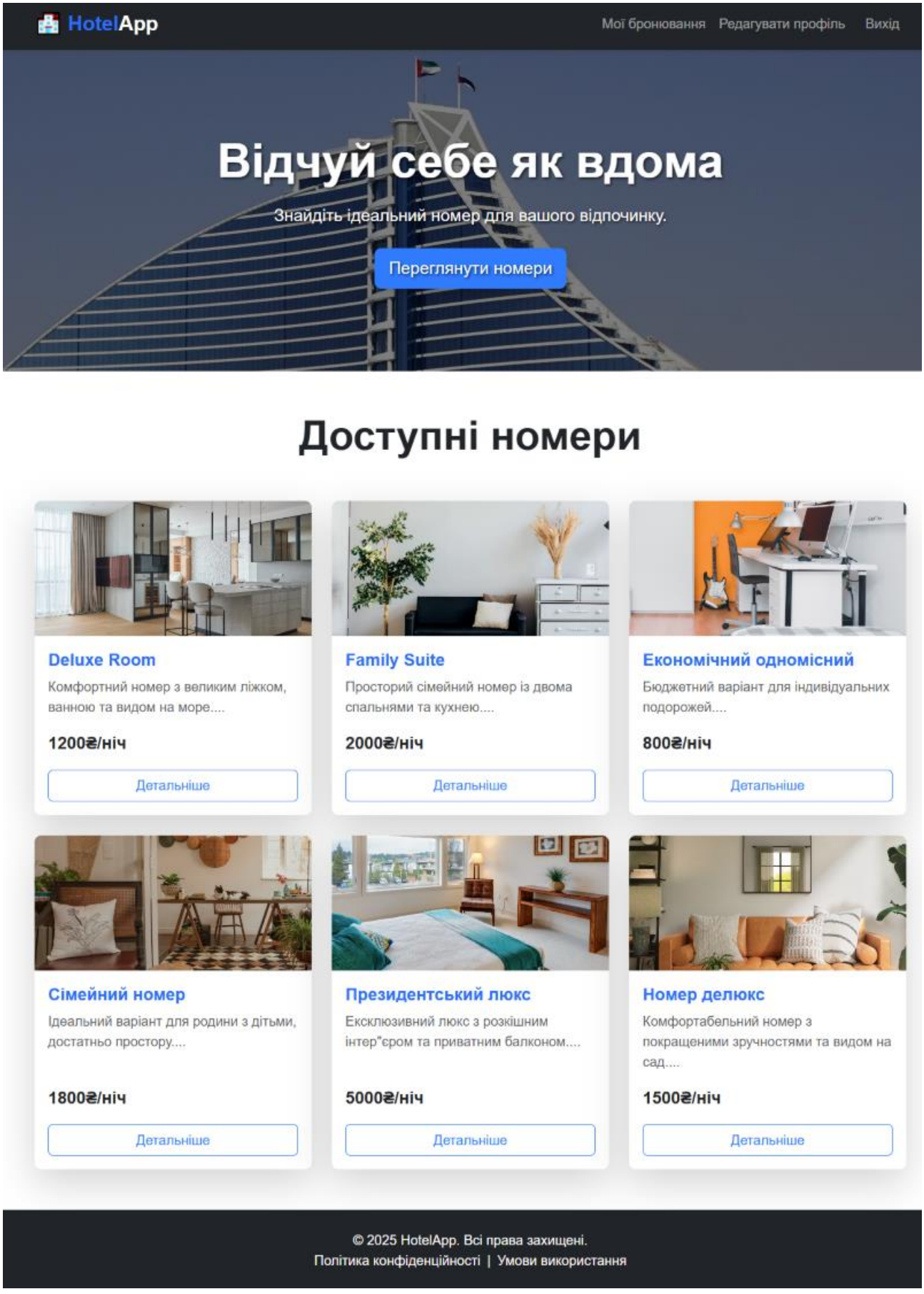


Рисунок 3.4 – Головна сторінка

На головній сторінці реалізовано відображення списку доступних номерів із фотографіями, коротким описом, ціною та кнопкою "Детальніше" - рис. 3.5.

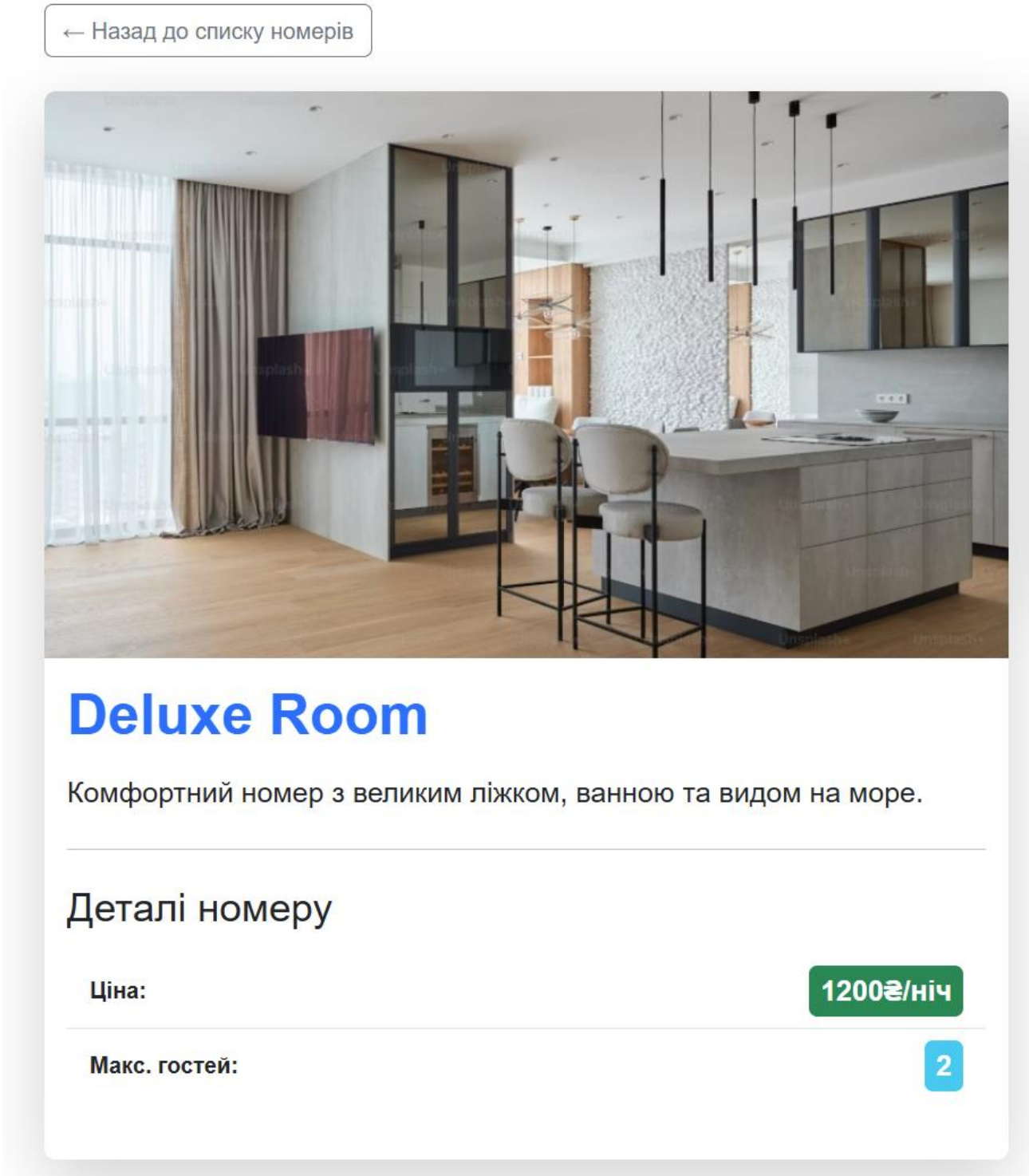


Рисунок 3.5 – Сторінка з інформацією про номер

Кожен номер має свою окрему сторінку з повною інформацією, де представлено характеристики кімнати (тип, місткість), фото, а також календар для вибору дат і перевірки доступності.

Система дозволяє здійснити перевірку доступності номера на конкретні дати, використовуючи інтерактивну форму - рис. 3.6.

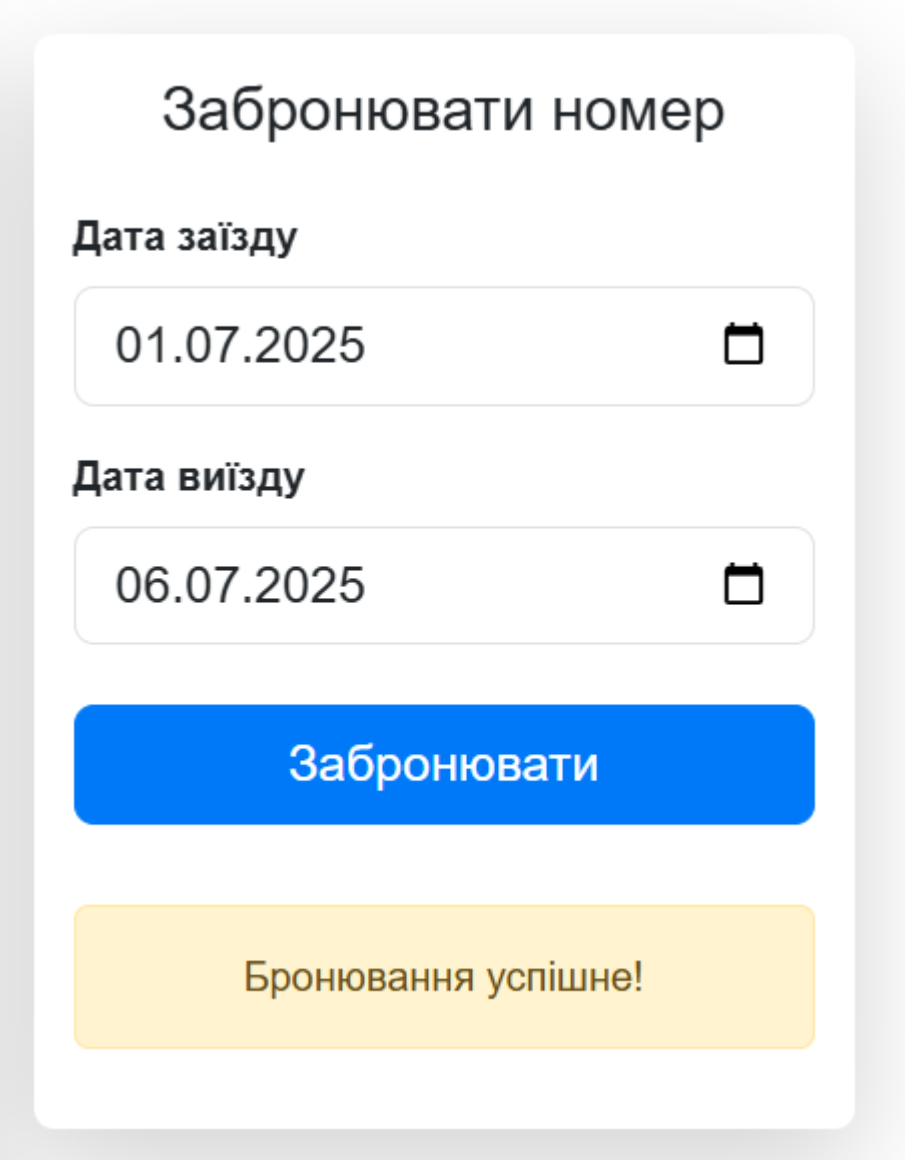


Рисунок 3.6 – Форма бронювання і перевірки

При спробі бронювання, якщо користувач не авторизований, система автоматично перенаправляє його на сторінку входу.

3.3 Реалізація функціоналу користувача

Після успішної реєстрації та входу користувач отримує доступ до персонального кабінету, який містить розділи для керування бронюваннями та

профілем - рис. 3.7. Це дозволяє забезпечити повноцінну взаємодію користувача із системою - рис. 3.8.

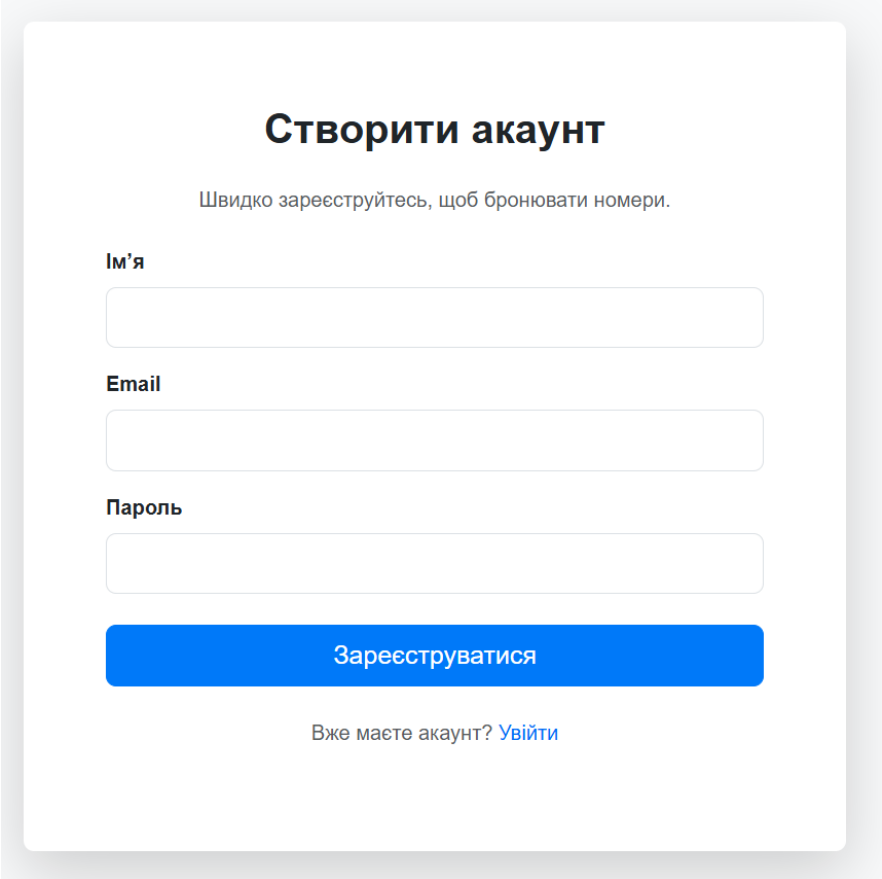


Рисунок 3.7 – Форма реєстрації

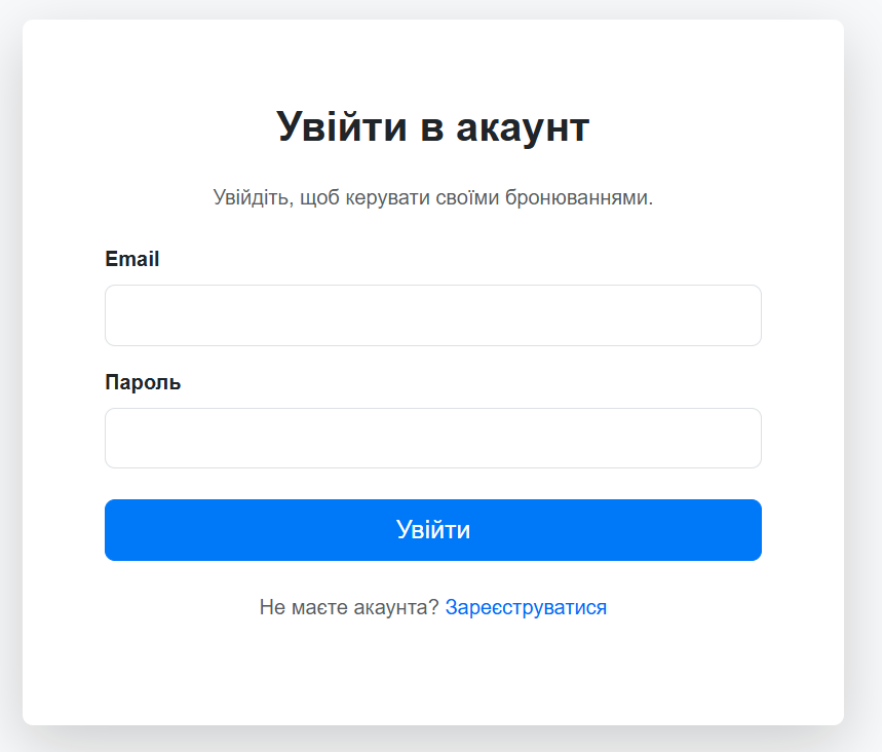


Рисунок 3.8 – Форма авторизації

У розділі "Мої бронювання" користувач бачить список усіх здійснених бронювань: номер кімнати, дати заїзду і виїзду, статус бронювання. Якщо дата ще не настала, користувач може скасувати бронювання, після чого статус оновлюється або запис видаляється - рис. 3.9.

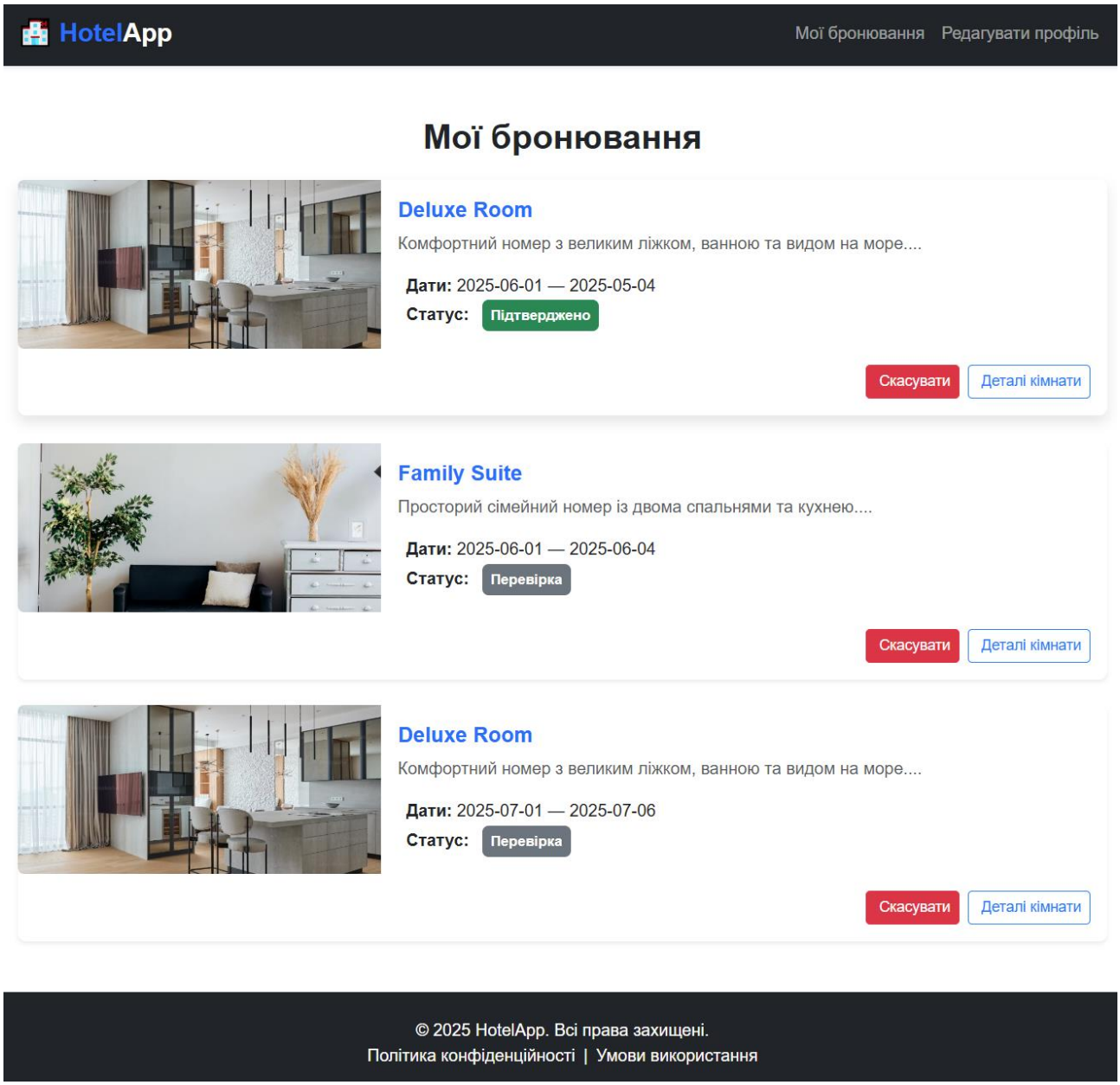


Рисунок 3.9 – Сторінка всіх бронювань

Реалізовано редагування особистого профілю: зміну імені та паролю. Дані валідуються на фронтенді, а потім оновлюються у базі даних MongoDB через запит до бекенду рис. 3.10.

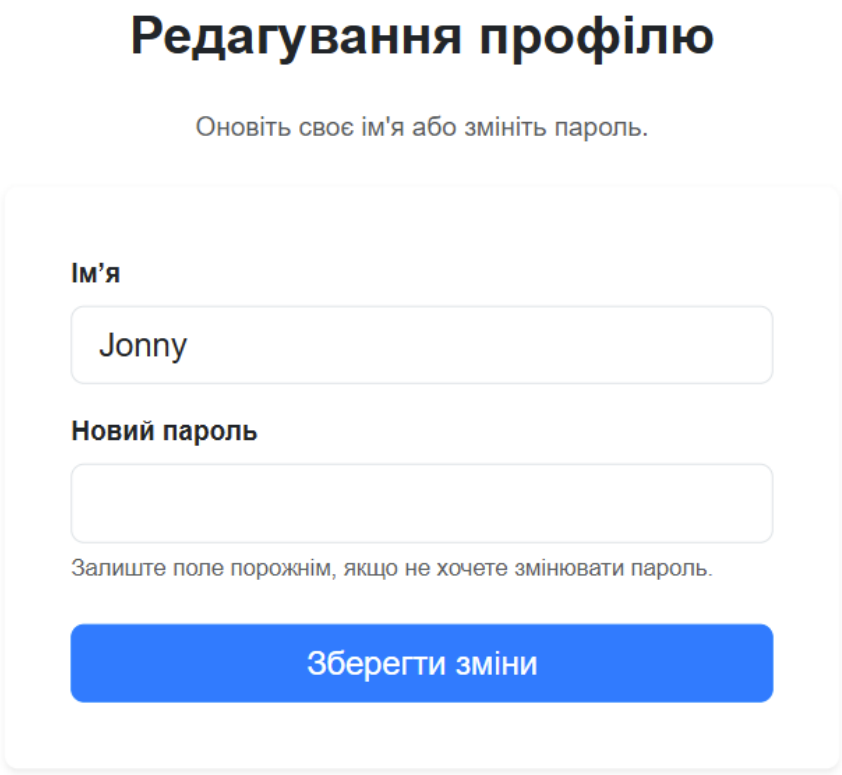


Рисунок 3.10 – Форма редагування профіля

Усі функції користувача захищені: прямий доступ до сторінок кабінету без авторизації неможливий. Для збереження сесії використовується захищене cookie або токен, залежно від реалізації (у базовій версії – сесії або localStorage).

3.4 Реалізація функціоналу адміністратора

На Адміністративна частина доступна лише користувачам із правами адміністратора. Після входу адміністратор перенаправляється у панель керування/

Список кімнат — тут можна переглянути весь список кімнат і видалити не активні пропозиції - рис. 3.11.

Усі кімнати

Керуйте наявними номерами готелю: переглядайте, редагуйте або видаляйте.

Deluxe Room

Комфортний номер з великим ліжком, ванною та видом на море.

Ціна: ₴1200 / ніч
Макс. гостей: 2

Видалити

Family Suite

Просторий сімейний номер із двома спальнями та кухнею.

Ціна: ₴2000 / ніч
Макс. гостей: 4

Видалити

Економічний одномісний

Бюджетний варіант для індивідуальних подорожей.

Ціна: ₴800 / ніч
Макс. гостей: 1

Видалити

Сімейний номер

Ідеальний варіант для родини з дітьми, достатньо простору.

Ціна: ₴1800 / ніч
Макс. гостей: 6

Видалити

Рисунок 3.11 – Список всіх кімнат

Додати новий номер — тут реалізовано створення номерів. Кожна кімната має атрибути:

- назва
- опис
- ціна
- фото
- кількість місць

Бронювання — адміністратор бачить список усіх бронювань, включаючи інформацію про користувача, дати, статус - рис. 3.12. Це дозволяє контролювати

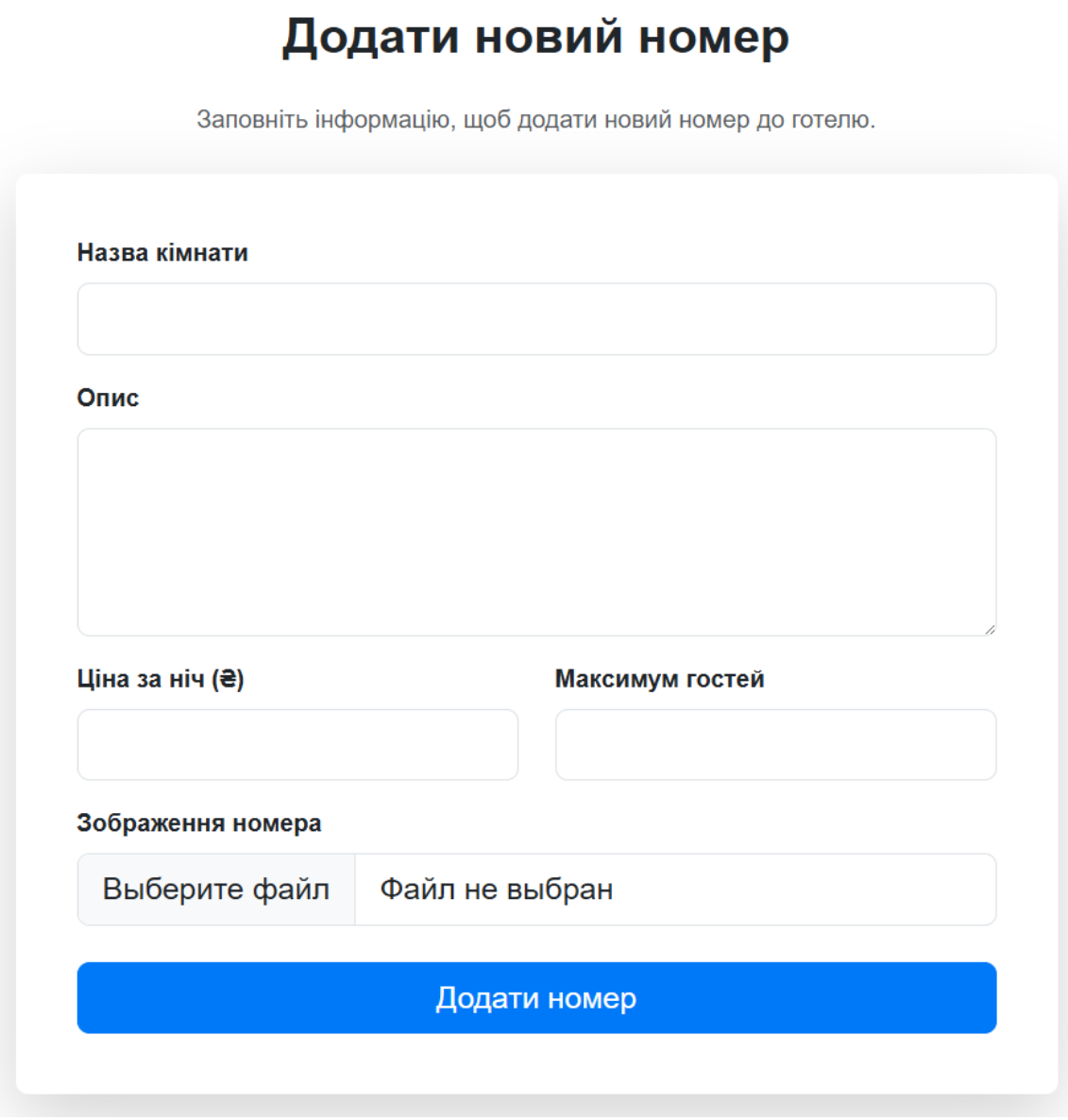


Рисунок 3.12 – Форма додавання даних

заповненість і переглядати історію рис. 3.13.

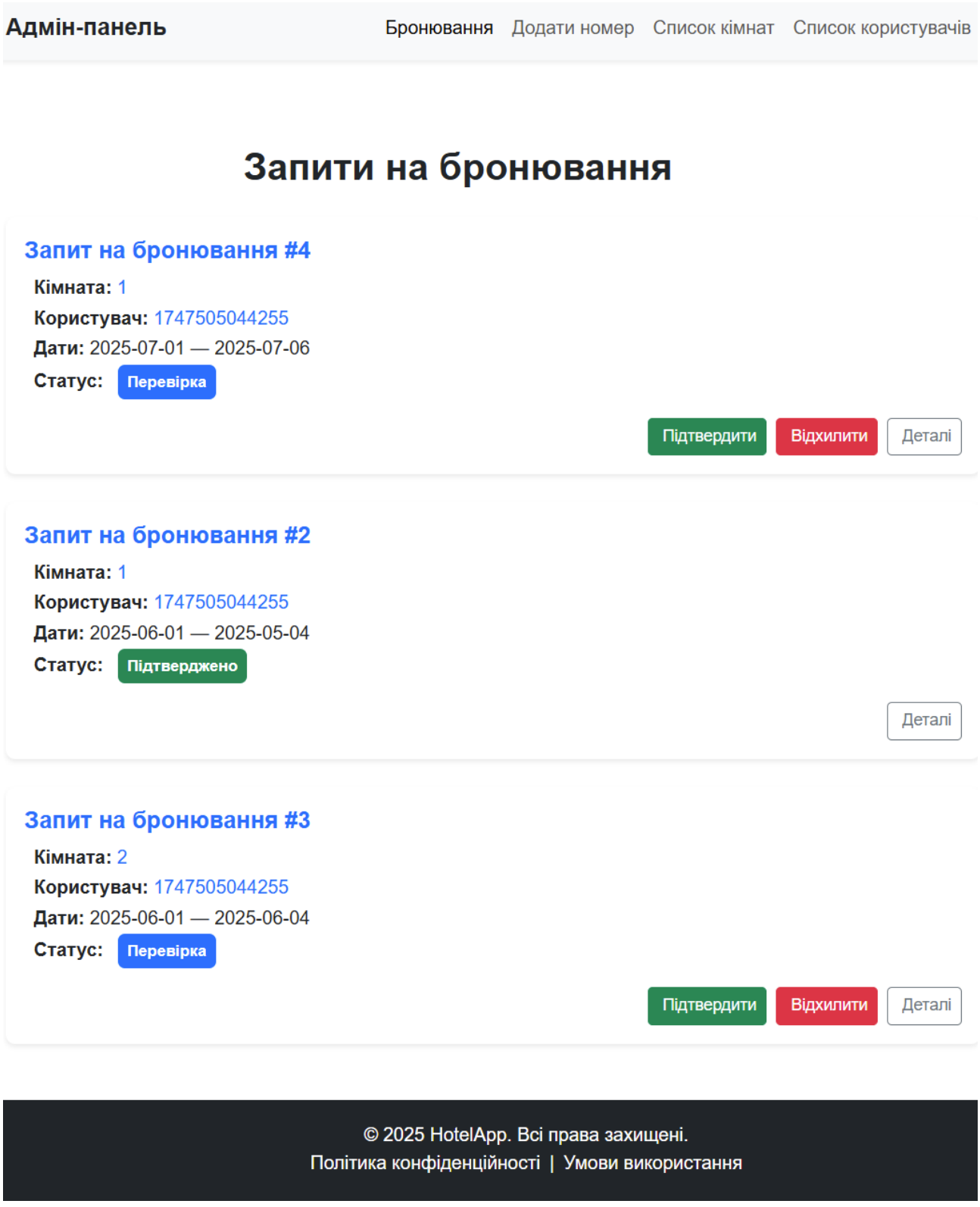


Рисунок 3.13 – Список запитів на бронювання

Користувачі — у цьому розділі адміністратор може переглядати зареєстрованих користувачів, їхні email-адреси, дати реєстрації. При потребі можна заблокувати або видалити обліковий запис рис. 3.14.

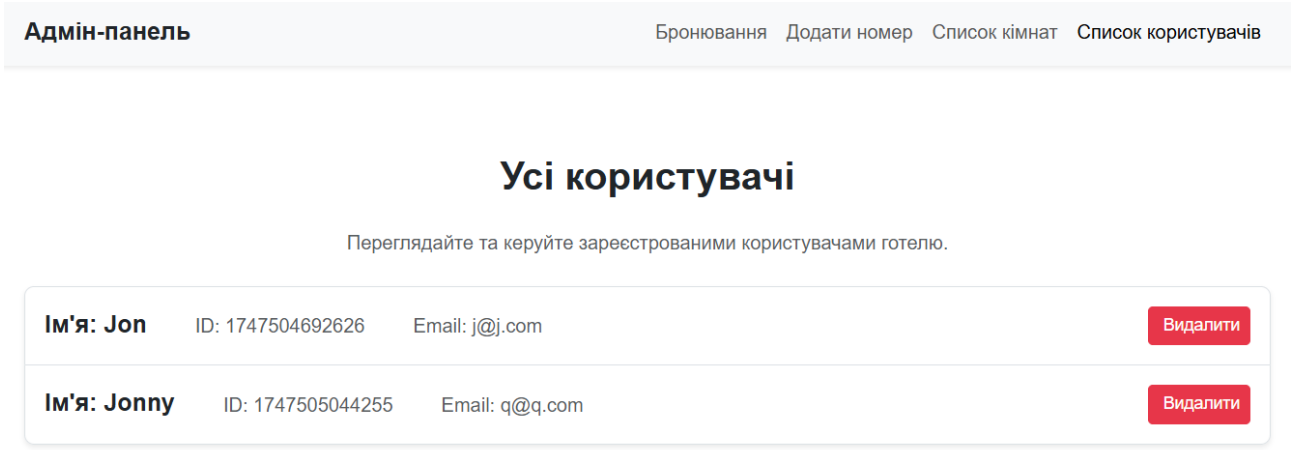


Рисунок 3.14 – Список користувачів

Реалізація функціоналу сторінок виконана на next.js - рис. 3.15.

```
import { useEffect, useState } from 'react';
import Link from 'next/link';

export default function AdminUsersPage() {
  const [users, setUsers] = useState([]);
  const [message, setMessage] = useState('');
  const [loading, setLoading] = useState(true);

  useEffect(() => {
    const fetchUsers = async () => {
      try {
        setLoading(true);
        const res = await fetch('/api/admin/get-users');
        if (!res.ok) {
          throw new Error(`HTTP error! status: ${res.status}`);
        }
        const data = await res.json();
        setUsers(data);
      } catch (error) {
        console.error("Помилка завантаження користувачів:", error);
        setMessage('Не вдалося завантажити користувачів. Спробуйте пізніше.');
      } finally {
        setLoading(false);
      }
    };
    fetchUsers();
  }, []);
```

Рисунок 3.15 – Список користувачів у адміністративній панелі

Уся логіка обробки запитів з боку адміністратора реалізована через Express.js. Завдяки MongoDB забезпечується ефективне зберігання даних і швидкий пошук - рис. 3.16.

```
import { useState } from 'react';
import Link from 'next/link';

export default function AddRoom() {
  const [title, setTitle] = useState('');
  const [description, setDescription] = useState('');
  const [pricePerNight, setPricePerNight] = useState('');
  const [maxGuests, setMaxGuests] = useState('');
  const [fakeImage, setFakeImage] = useState(null);
  const [message, setMessage] = useState('');
  const [loading, setLoading] = useState(false);

  const handleAdd = async () => {
    if (!title || !description || !pricePerNight || !maxGuests) {
      setMessage('Будь ласка, заповніть усі поля');
      return;
    }

    const res = await fetch('/api/admin/add-room', {
      method: 'POST',
      headers: { 'Content-Type': 'application/json' },
      body: JSON.stringify({
        title,
        description,
        pricePerNight: Number(pricePerNight),
        maxGuests: Number(maxGuests),
      }),
    });

    const data = await res.json();
    if (data.success) {
      setMessage('Номер успішно додано!');
      setTitle('');
      setDescription('');
      setPricePerNight('');
      setMaxGuests('');
      setFakeImage(null);
    } else {
      setMessage('Помилка при додаванні номера');
    }
  };

```

Рисунок 3.16 – Додавання нового номеру в базу даних

3.5 Тестування функціоналу веб-додатку

Після завершення розробки основного функціоналу веб-додатку для онлайн-бронювання номерів було проведено комплексне тестування з метою перевірки коректності роботи всіх компонентів системи, виявлення помилок і підтвердження відповідності заявленим функціональним вимогам.

Тестування здійснювалося у декілька етапів:

- Модульне тестування (Unit Testing) — перевірка окремих функцій бекенду функції створення бронювання, перевірки доступності, додавання номера до бази даних.
- Інтеграційне тестування (Integration Testing) — перевірка взаємодії між фронтендом і бекендом (через API), тестування процесу логіну, реєстрації, створення та видалення бронювання.
- Системне тестування (System Testing) — повна перевірка системи на цілісність та працездатність у реальному середовищі.
- Тестування безпеки — спроби несанкціонованого доступу до закритих частин додатку (наприклад, доступ до панелі адміністратора без прав).

Перевірка роботи сервера - рис. 3.17.

```
○ Compiled /api/rooms in 77ms (148 modules)
GET /api/rooms 304 in 100ms
GET /api/rooms 304 in 2ms
GET /api/rooms 304 in 2ms
GET /api/rooms 304 in 1ms
○ Compiled /_error in 98ms (442 modules)
GET /.well-known/appspecific/com.chrome.devtools.json 404 in 358ms
○ Compiled /admin/users in 168ms (384 modules)
○ Compiled /api/admin/get-users in 27ms (108 modules)
GET /api/admin/get-users 200 in 40ms
GET /api/admin/get-users 200 in 40ms
○ Compiled /admin/bookings in 66ms (401 modules)
○ Compiled /api/admin/get-bookings in 75ms (110 modules)
GET /api/admin/get-bookings 304 in 82ms
GET /api/admin/get-bookings 304 in 2ms
○ Compiled /api/rooms in 22ms (112 modules)
GET /api/rooms 304 in 25ms
GET /api/rooms 304 in 1ms
GET /api/admin/get-users 304 in 2ms
GET /api/admin/get-users 304 in 1ms
GET /api/admin/get-bookings 304 in 2ms
GET /api/admin/get-bookings 304 in 1ms
○ Compiled /api/admin/confirm-booking in 76ms (106 modules)
PUT /api/admin/confirm-booking?id=4 200 in 83ms
○ Compiled in 134ms (289 modules)
○ Compiled / in 180ms (384 modules)
GET / 200 in 449ms
○ Compiled /api/rooms in 27ms (108 modules)
GET /api/rooms 304 in 38ms
```

Рисунок 3.17 – Перевірка роботи сервера

У ході тестування використовувались ручні сценарії, так і автоматизовані тести за допомогою таких інструментів, Jest для модульних тестів у Node.js - рис. 3.18.

```
const request = require('supertest');
const app = require('../server'); // Express app
const mongoose = require('mongoose');

describe('POST /api/bookings', () => {
  it('створює нове бронювання', async () => {
    const response = await request(app)
      .post('/api/bookings')
      .send({
        userId: '665c1a1c0f1a0a1a0a1a0a1a',
        roomId: '665c1b1b0f2b0b2b0b2b0b2b',
        checkIn: '2025-06-01',
        checkOut: '2025-06-05'
      });

    expect(response.statusCode).toBe(201);
    expect(response.body).toHaveProperty('bookingId');
    expect(response.body.success).toBe(true);
  });
});
```

Рисунок 3.18 – Автоматизований тест Jest

Сценарію ручного тесту

- Сценарій: користувач реєструється, логіниться, обирає вільний номер і створює бронювання.
- Очікуваний результат: після підтвердження дати бронювання, новий запис з’являється у персональному кабінеті користувача. На бекенді створюється новий документ у колекції bookings, а статус номера на вибрані дати стає недоступним.
- Результат: усі кроки виконані успішно. Дані записані в базу. Повідомлення "Бронювання успішно створено!" з’являється на екрані.

Виявлені помилки у процесі тестування:

- Некоректна обробка дат у часовому поясі UTC+3 (виправлено через нормалізацію дати на сервері).

- Звичайні користувачі мали змогу перейти на сторінку адміністрування за прямим посиланням (усунуто додатковою перевіркою прав доступу на рівні бекенду).

Таким чином, у результаті тестування вдалося підтвердити працездатність усіх ключових сценаріїв роботи веб-додатку, забезпечити стабільну роботу з MongoDB, коректну взаємодію між фронтендом і бекендом через REST API, а також реалізувати основні заходи безпеки для захисту даних і доступу.

ВИСНОВОК

Під час дослідження проблематики розробки сучасних систем бронювання було ретельно проаналізовано актуальні виклики, пов'язані з автоматизацією гостинних послуг. Основна увага була приділена створенню ефективного веб-рішення з використанням технологій NextJS, ExpressJS та MongoDB, яке забезпечує зручний інтерфейс для клієнтів та ефективне управління для адміністраторів готелів.

Аналіз ринку готельних послуг виявив ключові вимоги до сучасних систем бронювання: швидкість обробки запитів, зручність інтерфейсу, надійність зберігання даних та гнучкість у керуванні номерами. Це стало основою для обґрунтування вибору технологічного стеку, де NextJS забезпечує швидкий рендеринг інтерфейсу, ExpressJS відповідає за бізнес-логіку, а MongoDB гарантує надійне зберігання динамічних даних.

На етапі проектування було розроблено архітектуру додатку, яка включає три основні модулі: гостьовий інтерфейс для бронювання, особистий кабінет користувача та адміністративну панель. Особливу увагу приділено системі перевірки доступності номерів у реальному часі, що є критично важливим для уникнення конфліктів бронювань. Модель даних була оптимізована для швидкої обробки запитів та ефективного управління великим потоком бронювань.

Практична реалізація системи демонструє ефективність обраного підходу. NextJS забезпечив швидке завантаження сторінок та адаптивний інтерфейс, ExpressJS показав стабільну роботу при обробці одночасних запитів на бронювання, а MongoDB довела свою ефективність у роботі з динамічно змінюваними даними. Розроблений механізм бронювання включає всі необхідні етапи - від вибору дат до підтвердження резервації, з можливістю подальшого скасування.

Тестування системи в умовах, близьких до реальних, підтвердило її стабільність та ефективність. Середній час обробки запиту на бронювання не перевищує 1,5 секунди, навіть при піковому навантаженні. Система демонструє

гарні показники масштабованості, що дозволяє адаптувати її як для невеликих готельних комплексів, так і для великих мереж.

Розроблений веб-додаток представляє собою сучасне рішення для автоматизації процесів бронювання, яке поєднує високий рівень користувацького досвіду з ефективними інструментами управління. Використання сучасного JavaScript-стеку дозволяє легко інтегрувати додаткові функції, такі як система лояльності або інтеграція з зовнішніми платіжними сервісами. Система може бути успішно впроваджена в готельному бізнесі, сприяючи підвищенню якості обслуговування клієнтів та оптимізації роботи персоналу.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Tanenbaum A., Wetherall D. Computer Networks. — 6th ed. — Pearson, 2021. – – 960 p.
2. Cloudflare. What is a DNS Firewall? — URL: https://www.cloudflare.com/learning/dns/dns-firewall/ (дата звернення: 29.04.2025).
3. W3Schools (HTML/CSS). — URL: https://www.w3schools.com/ (дата звернення: 01.05.2025).
4. Flanagan D. JavaScript: The Definitive Guide. — 7th ed. — O’Reilly, 2020. — 704 p.
5. OWASP. Web Application Security Testing Guide. — URL: https://owasp.org/www-project-web-security-testing-guide/ (дата звернення: 08.04.2025).
6. Bootstrap Documentation. — URL: https://getbootstrap.com/docs/ (дата звернення: 29.04.2025).
7. Flanagan D. JavaScript: The Definitive Guide. — 7th ed. — O’Reilly, 2020. — 704 p.
8. Tanenbaum A., Wetherall D. Computer Networks. — 6th ed. — Pearson, 2021. – – 960 p.
9. MongoDB Documentation. — URL: https://www.mongodb.com/docs/manual/ (дата звернення: 19.05.2025).
10. Next.js Documentation. — URL: https://nextjs.org/docs (дата звернення: 20.05.2025).
11. Express.js Guide. — URL: https://expressjs.com/en/starter/installing.html (дата звернення: 18.05.2025).
12. MDN Web Docs (JavaScript). — URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript (дата звернення: 21.05.2025).
13. W3Schools (HTML/CSS). — URL: https://www.w3schools.com/ (дата звернення: 22.05.2025).

14. OWASP. Web Application Security Testing Guide. — URL: https://owasp.org/www-project-web-security-testing-guide/ (дата звернення: 16.05.2025).
15. Cloudflare. What is DNSSEC? — URL: https://www.cloudflare.com/learning/dns/dnssec/ (дата звернення: 15.05.2025).
16. OpenAPI Specification. — URL: https://swagger.io/specification/ (дата звернення: 13.05.2025).
17. Bootstrap Documentation. — URL: https://getbootstrap.com/docs/ (дата звернення: 14.05.2025).
18. Cypress Documentation. — URL: https://docs.cypress.io/ (дата звернення: 17.05.2025).
19. Jest – JavaScript Testing Framework. — URL: https://jestjs.io/docs/getting-started (дата звернення: 19.05.2025).
20. DigitalOcean. How to Build a REST API with Express.js — URL: https://www.digitalocean.com/community/tutorials/build-a-restful-api-using-node-and-express-4 (дата звернення: 12.05.2025).
21. React Official Documentation. — URL: https://reactjs.org/docs/getting-started.html (дата звернення: 20.05.2025).

ДОДАТОК А

Лістинг index.js

```
import Link from 'next/link';
import { useEffect, useState } from 'react';
import 'bootstrap/dist/css/bootstrap.min.css';

export default function Home() {
  const [rooms, setRooms] = useState([]);
  const [user, setUser] = useState(null);

  useEffect(() => {
    fetch('/api/rooms')
      .then(data => setRooms(data));

    const stored = localStorage.getItem('currentUser');
  }, []);

  const handleLogout = () => {
    localStorage.removeItem('currentUser');
    window.location.reload();
  };

  return (
<div className="d-flex flex-column min-vh-100">
      {/* Хедер */}
      <header>
        <nav className="navbar navbar-expand-lg navbar-dark bg-dark
shadow-sm">
          <div className="container">
            <Link href="/" className="navbar-brand fs-4 fw-bold">
              🛏 <span className="text-primary">Hotel</span>App
            </Link>
            <button
              className="navbar-toggler"
              type="button"
              data-bs-toggle="collapse"
              data-bs-target="#navbarNav"
              aria-controls="navbarNav"
              aria-expanded="false"
              aria-label="Toggle navigation"
            >
              <span className="navbar-toggler-icon"></span>
            </button>
            <div className="navbar-collapse" id="navbarNav">
              <ul className="navbar-nav ms-auto">
                {!user ? (
                  <>
                    <li className="nav-item">
                      <Link href="/register" className="nav-link">
                        Реєстрація
                      </Link>
```

```
                  </li>
                  <li className="nav-item">
                    <Link href="/login" className="nav-link btn
btn-primary text-white ms-lg-2">
                      Вхід
                    </Link>
                  </li>
                </>
              ) : (
                <>
                  <li className="nav-item">
                    <Link href="/my-bookings" className="nav-
link">
                      Мої бронювання
                    </Link>
                  </li>
                  <li className="nav-item">
                    <Link href="/edit-profile" className="nav-
link">
                      Редагувати профіль
                    </Link>
                  </li>
                  <li className="nav-item">
                    <button
                      className="btn btn-outline-secondary nav-
link ms-lg-2"
                      onClick={handleLogout}
                    >
                      Вихід
                    </button>
                  </li>
                </>
              )}
            </ul>
          </div>
        </div>
      </nav>
    </header>

    {/* Hero секція з фоновим зображенням */}
    <section className="hero-section text-white d-flex align-
items-center justify-content-center text-center">
      <div className="container">
        <h1 className="display-4 fw-bold mb-3">Відчуй себе як
вдома</h1>
        <p className="lead mb-4">
          Знайдіть ідеальний номер для вашого відпочинку.
        </p>
        <Link href="#rooms-list" className="btn btn-primary btn-lg
shadow">
          Переглянути номери
        </Link>
      </div>
```

```
      </section>

      {/* Основний контент */}
      <main className="container my-5 flex-grow-1">
        <h2 id="rooms-list" className="text-center mb-5 display-5
fw-bold text-dark">
          Доступні номери
        </h2>
        <div className="row g-4">
          {rooms.map((room) => (
            <div key={room.id} className="col-md-6 col-lg-4">
              <div className="card h-100 shadow-lg border-0 rounded-
3">
                <img
                  src={`/images/${room.images[0]}`}
                  className="card-img-top rounded-top-3 room-img"
                  alt={room.title}
                />
                <div className="card-body d-flex flex-column">
                  <h5 className="card-title text-primary fw-bold mb-
2">{room.title}</h5>
                  <p className="card-text text-muted mb-3 flex-grow-
1">
                    {room.description.slice(0, 100)}...
                  </p>
                  <p className="card-text fs-5 fw-bold text-dark mb-
3">
                    {room.pricePerNight}₴/ніч
                  </p>
                  <Link href={`/${room.id}`} className="btn btn-
outline-primary mt-auto">
                    Детальніше
                  </Link>
                </div>
              </div>
            </div>
          ))}
        </div>
      </main>
      <footer className="bg-dark text-white py-4 mt-auto">
        <div className="container text-center">
          <p className="mb-0">
            &copy; {new Date().getFullYear()} HotelApp. Всі права
захищені.
          </p>
          <p className="mb-0">
            <Link href="#" className="text-white text-decoration-
none mx-2">Політика конфіденційності</Link>
            |
            <Link href="#" className="text-white text-decoration-
none mx-2">Умови використання  </Link></p></div></footer> </div> );}
```

ДОДАТОК Б

Лістинг файлу admin/add-room.js

```
import { useEffect, useState } from 'react';
import Link from 'next/link';

export default function AdminBookings() {
  const [bookings, setBookings] = useState([]);
  const [message, setMessage] = useState('');

  useEffect(() => {
    fetch('/api/admin/get-bookings')
      .then(data => {
        const sorted = data.sort((a, b) => new Date(b.startDate) -
new Date(a.startDate));
        setBookings(sorted);
      });
  }, []);

  const confirmBooking = async (id) => {
    const res = await fetch(`/api/admin/confirm-booking?id=${id}`, {
method: 'PUT' });
    const data = await res.json();

    if (data.success) {
      setBookings(prev =>
        prev.map(b => b.id === id ? { ...b, status: 'Підтверджено' }
: b)
      );
      setMessage('Бронювання підтверджено');
    } else {
      setMessage('Помилка при підтвердженні');
    }
  };

  return (
    <div className="d-flex flex-column min-vh-100">
      {/* Адмін-меню */}
      <nav className="navbar navbar-expand-lg navbar-light bg-light
shadow-sm mb-4">
        <div className="container">
          <span className="navbar-brand fw-bold text-dark">Адмін-
панель</span>
          <button
            className="navbar-toggler"
            type="button"
            data-bs-toggle="collapse"
            data-bs-target="#adminNavbar"
            aria-controls="adminNavbar"
            aria-expanded="false"
            aria-label="Toggle navigation"
          >
```

```
            <span className="navbar-toggler-icon"></span>
          </button>
          <div className="navbar-collapse" id="adminNavbar">
            <ul className="navbar-nav ms-auto">
              <li className="nav-item">
                <Link href="/admin/bookings" className="nav-link
active">
                  Бронювання
                </Link>
              </li>
              <li className="nav-item">
                <Link href="/admin/add-room" className="nav-link">
                  Додати номер
                </Link>
              </li>
              <li className="nav-item">
                <Link href="/admin/rooms" className="nav-link">
                  Список кімнат
                </Link>
              </li>
              <li className="nav-item">
                <Link href="/admin/users" className="nav-link">
                  Список користувачів
                </Link>
              </li>
            </ul>
          </div>
        </div>
      </nav>

      {/* Основний контент - Запити на бронювання */}
      <main className="container my-5 flex-grow-1">
        <h2 className="mb-4 text-center fw-bold text-dark">Запити на
бронювання</h2>

        {message && (
          <div className={`alert ${message.includes('підтверджено')
? 'alert-success' : message.includes('відхилено') ? 'alert-warning'
: 'alert-info'} text-center mb-4`} role="alert">
            {message}
          </div>
        )}

        {bookings.length === 0 ? (
          <div className="alert alert-info text-center py-4 shadow-
sm" role="alert">
            <h4 className="alert-heading">Наразі немає нових запитів
на бронювання.</h4>
            <p className="mb-0">Все тихо! Перевірте пізніше.</p>
          </div>
        ) : (
          <div className="row row-cols-1 g-4"> {/* Використовуємо
Bootstrap сітку для карток */}
```

```
            {bookings.map(b => {
              // Визначаємо клас для статусу бронювання
              let statusBadgeClass = 'bg-secondary';
              if (b.status === 'Підтверджено') {
                statusBadgeClass = 'bg-success';
              } else if (b.status === 'Перевірка') {
                statusBadgeClass = 'bg-primary';
              } else if (b.status === 'Відхилено') {
                statusBadgeClass = 'bg-danger';
              }

              return (
                <div key={b.id} className="col">
                  <div className="card h-100 shadow-sm border-0
rounded-3 booking-request-card">
                    <div className="card-body">
                      <h5 className="card-title text-primary fw-bold
mb-2">Запит на бронювання #{b.id}</h5>
                      <ul className="list-unstyled mb-3">
                        <li>
                          <strong><i className="bi bi-door-open me-
2"></i>Кімната:</strong>{' '}
                          <Link href={`/admin/rooms/${b.roomId}`}
className="text-decoration-none">
                            {b.roomId}
                          </Link>
                        </li>
                        <li>
                          <strong><i className="bi bi-person me-
2"></i>Користувач:</strong>{' '}
                          <Link href={`/admin/users/${b.userId}`}
className="text-decoration-none">
                            {b.userId}
                          </Link>
                        </li>
                        <li>
                          <strong><i className="bi bi-calendar-check
me-2"></i>Дати:</strong>{' '}
                          {b.startDate} &mdash; {b.endDate}
                        </li>
                        <li>
                          <strong><i className="bi bi-info-circle
me-2"></i>Статус:</strong>{' '}
                          <span className={`badge
${statusBadgeClass} ms-2 p-2`}>
                            {b.status}
                          </span>
                        </li>
                      </ul>
                      <div className="d-flex justify-content-end
align-items-center mt-3">
                        {b.status === 'Перевірка' && (
                          <>
```

```
                              <button
                                className="btn btn-success btn-sm me-
2"
                                onClick={() => confirmBooking(b.id)}
                              >
                                <i className="bi bi-check-circle me-
1"></i>Підтвердити
                              </button>
                              <button
                                className="btn btn-danger btn-sm"
                                onClick={() => rejectBooking(b.id)}
                              >
                                <i className="bi bi-x-circle me-
1"></i>Відхилити
                              </button>
                            </>
                          )}
                          {/* Можна додати кнопку "Деталі" для
перегляду повного запиту, незалежно від статусу */}
                           <Link target="_blank" href={`/${b.id}`}
className="btn btn-outline-secondary btn-sm ms-2">
                            <i className="bi bi-search me-
1"></i>Деталі
                           </Link>
                        </div>
                      </div>
                    </div>
                  </div>
                );
              })}
            </div>
          )}
        </main>
      </div>
    );
}
```

ДОДАТОК С

Лістинг package-lock.json


```
{
  "name": "booking",
  "version": "0.1.0",
  "lockfileVersion": 3,
  "requires": true,
  "packages": {
    "": {
      "name": "booking",
      "version": "0.1.0",
      "dependencies": {
        "bootstrap": "^5.3.6",
        "express": "^4.18.2",
        "next": "^15.3.2",
        "react": "^19.0.0",
        "react-dom": "^19.0.0"
      },
      "devDependencies": {
        "@tailwindcss/postcss": "^4",
        "tailwindcss": "^4"
      }
    },
    "node_modules/@alloc/quick-lru": {
      "version": "5.2.0",
      "resolved": "https://registry.npmjs.org/@alloc/quick-lru/-
/quick-lru-5.2.0.tgz",
      "integrity": "sha512-
UrcABB+4bUrFABwbluTIBErXwvbsU/V7TZWfmbgJfbkwiBuziS9gxdODUyuiecfdGQ85
jglMW6juS3+z5TsKLw==",
      "dev": true,
      "engines": {
        "node": ">=10"
      },
      "funding": {
        "url": "https://github.com/sponsors/sindresorhus"
      }
    },
    "node_modules/@ampproject/remapping": {
      "version": "2.3.0",
      "resolved":
"https://registry.npmjs.org/@ampproject/remapping/-/remapping-
2.3.0.tgz",
      "integrity": "sha512-
30iZtAPgz+LTIYoeivqYo853f02jBYSd5uGnGpkFV0M3xOt9aN73erkgYAmZU43x4Vfq
cnLxW9Kpg3R5LC4YYw==",
      "dev": true,
      "dependencies": {
        "@jridgewell/gen-mapping": "^0.3.5",
        "@jridgewell/trace-mapping": "^0.3.24"
      },
```

```
      "engines": {
        "node": ">=6.0.0"
      }
    },
    "node_modules/@emnapi/runtime": {
      "version": "1.4.3",
      "resolved": "https://registry.npmjs.org/@emnapi/runtime/-
/runtime-1.4.3.tgz",
      "integrity": "sha512-
pBPWdu6MLKROBX05wSNKcNb++m5Er+KQ9QkB+WVM+pW2Kx9hoSrVTnu3BdkI5eBLZoKu
/J6mW/B6i6bJB2ytXQ==",
      "optional": true,
      "dependencies": {
        "tslib": "^2.4.0"
      }
    },
    "node_modules/@img/sharp-darwin-arm64": {
      "version": "0.34.1",
      "resolved": "https://registry.npmjs.org/@img/sharp-darwin-
arm64/-/sharp-darwin-arm64-0.34.1.tgz",
      "integrity": "sha512-
pn44xgBtgpEbZsu+lWf2KNb6OAf70X68k+yk69Ic2Xz11zHR/w24/U49XT7AeRwJ0Px+
mhALhU5LPci1Aymk7A==",
      "cpu": [
        "arm64"
      ],
      "optional": true,
      "os": [
        "darwin"
      ],
      "engines": {
        "node": "^18.17.0 || ^20.3.0 || >=21.0.0"
      },
      "funding": {
        "url": "https://opencollective.com/libvips"
      },
      "optionalDependencies": {
        "@img/sharp-libvips-darwin-arm64": "1.1.0"
      }
    },
    "node_modules/@img/sharp-darwin-x64": {
      "version": "0.34.1",
      "resolved": "https://registry.npmjs.org/@img/sharp-darwin-
x64/-/sharp-darwin-x64-0.34.1.tgz",
      "integrity": "sha512-
VfuYgG2r8BpYiOUN+BfYeFo69nP/MIwAtSJ7/Zpxc5QF3KS22z8Pvg3FkrSFJBPNQ7mm
cUcYQFBmEQp7eu1F8Q==",
      "cpu": [
        "x64"
      ],
      "optional": true,
      "os": [
        "darwin"
```

```
    ],
    "engines": {
      "node": "^18.17.0 || ^20.3.0 || >=21.0.0"
    },
    "funding": {
      "url": "https://opencollective.com/libvips"
    },
    "optionalDependencies": {
      "@img/sharp-libvips-darwin-x64": "1.1.0"
    }
  },
```

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО–НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Спеціальність 122. Комп’ютерні науки

Hotel booking web-application використовуючи NextJS,ExpressJS,MongoDB

Виконав
студент *Олександр Мочульський*
Науковий
керівник *Віктор ВИШНІВСЬКИЙ*

Мета кваліфікаційної роботи

Розробка веб-додатку для бронювання готельних номерів з використанням NextJS, ExpressJS та MongoDB, спрямованого на оптимізацію процесів пошуку, резервування та управління номерами для користувачів і адміністраторів.

Завдання дослідження:

- Вивчення сучасних підходів до створення систем бронювання.
- Дослідження існуючих рішень та технологій у сфері веб-розробки.
- Розробка архітектури додатку.
- Реалізація функціонального прототипу, включаючи систему бронювання, управління номерами, аутентифікацію та взаємодію з базою даних.
- Тестування та оптимізація роботи додатку.

Актуальність розробки

- Системи онлайн-бронювання дозволяють готелям автоматизувати ключові бізнес-процеси, зокрема управління номерами, прийом бронювань, комунікацію з клієнтами та обробку платежів.
- У свою чергу, користувачі отримують доступ до інформації про доступні номери, можливість перевірити наявність місць на потрібні дати, оформити бронювання в реальному часі та керувати своїми замовленнями у зручному інтерфейсі.

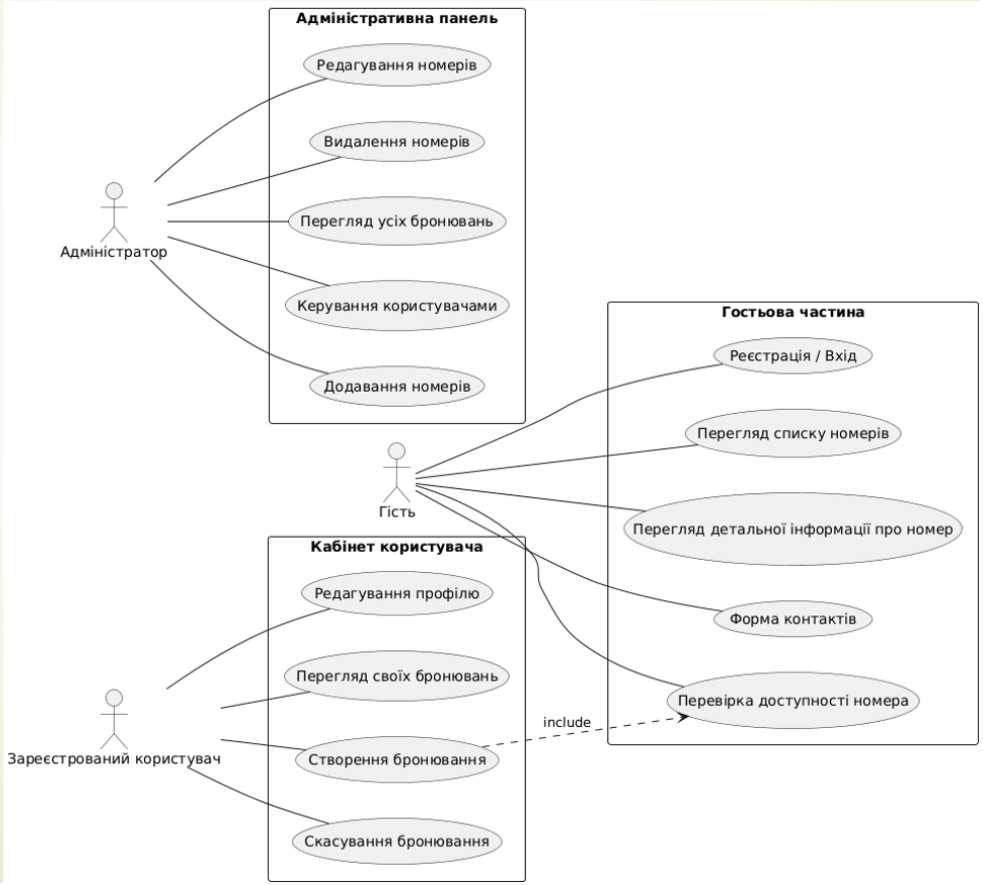
Актуальність створення подібних веб-додатків зумовлена також стрімким розвитком мобільних технологій, що вимагає адаптивного дизайну, високої продуктивності та безпеки обробки персональних даних.

Основні вимоги до сучасних систем онлайн-бронювання

Візуалізація функцій та взаємодії між компонентами системи

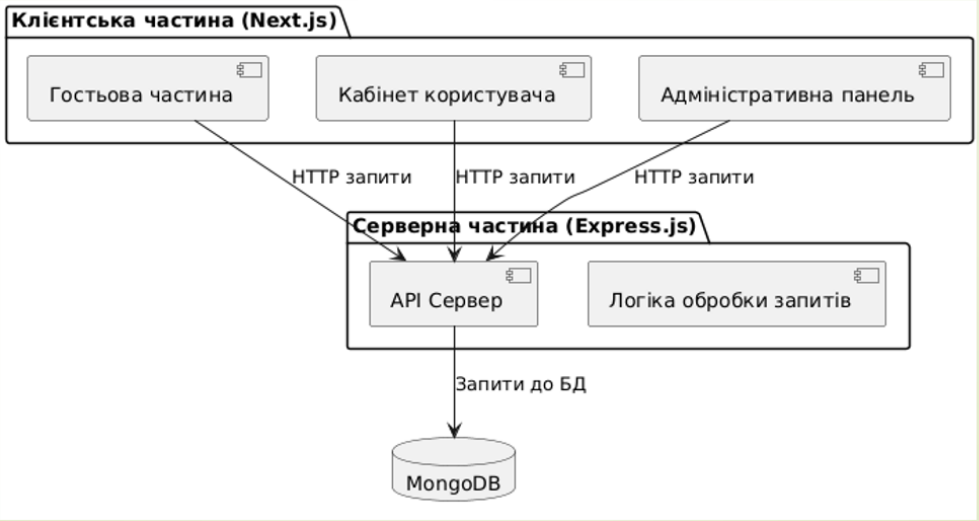
Ця діаграма показує три основні ролі — гість, зареєстрований користувач і адміністратор — та їх основні дії на сайті.

5



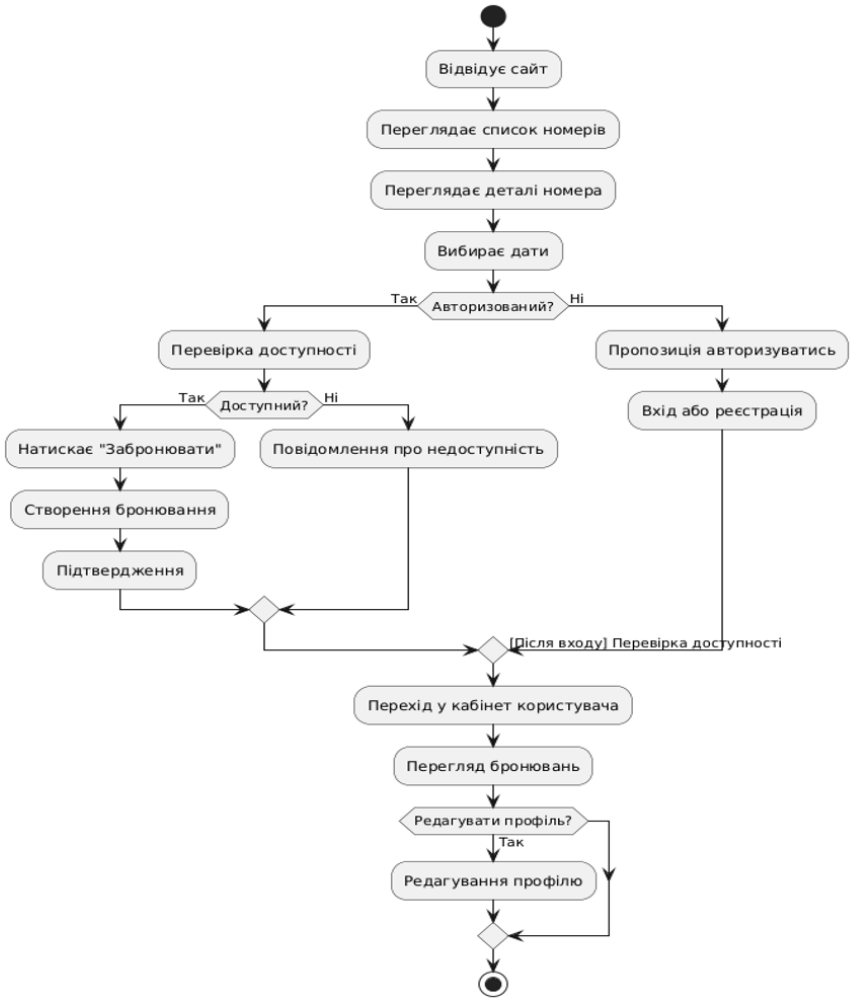
Візуалізація функцій та взаємодії між компонентами системи

Архітектурна блок-схема відображає ключові компоненти веб-додатку для бронювання готельних номерів та їх взаємодію. Система складається з кількох основних модулів: клієнтської частини, серверної частини, бази даних і зовнішніх сервісів. Клієнтська частина реалізована на Next.js і відповідає за інтерфейс користувача, включно з гостьовою частиною, кабінетом користувача та адміністративною панеллю.



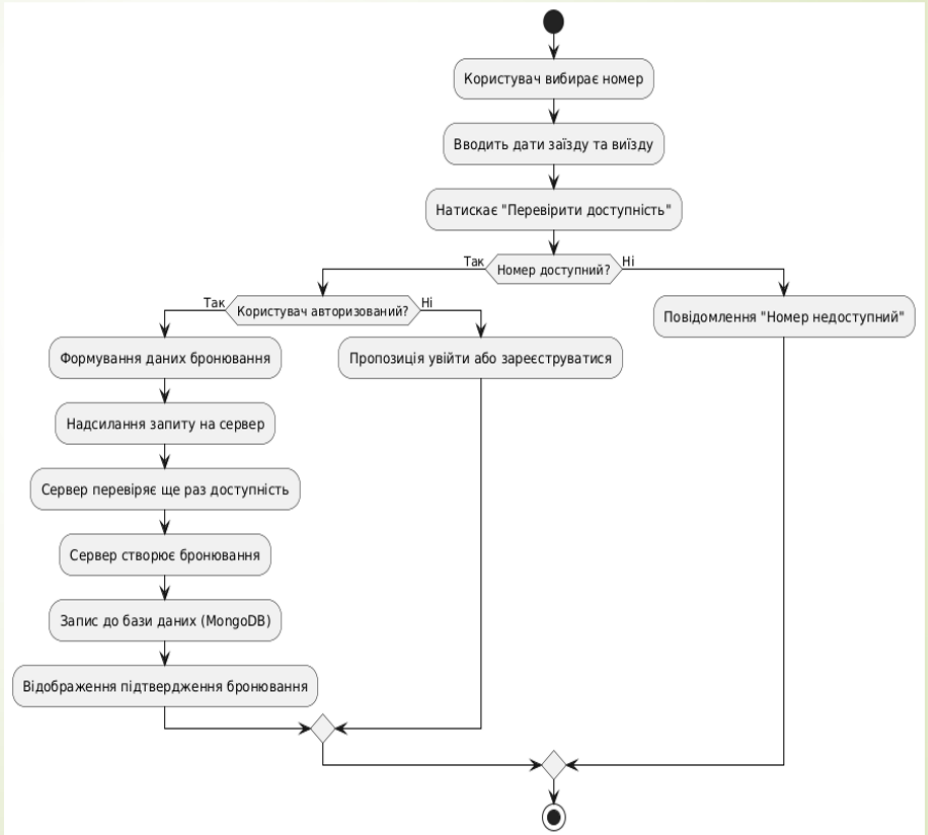
Візуалізація функцій та взаємодії між компонентами системи

Діаграма активності, відображає повну послідовність дій, які виконує звичайний користувач у веб-додатку для онлайн-бронювання готелів. Вона охоплює типовий життєвий цикл взаємодії з системою.



Візуалізація функцій та взаємодії між компонентами системи

Блок-схема процесу бронювання номера демонструє послідовність кроків, які виконує система при оформленні замовлення користувачем.



Обґрунтування вибору засобів розробки: Next.js

- SSR (Server-Side Rendering) та статична генерація
- Забезпечує швидке завантаження сторінок
- Дозволяє попередньо завантажувати дані (наприклад, список номерів) для зменшення часу очікування.
- Вбудований API-маршрутизатор
- Спрощує створення ендпоінтів
- Оптимізація зображень (Image Component)
- Автоматичне налаштування розмірів фото номерів для різних пристроїв, що підвищує юзабіліті.
- Легка інтеграція з бекендом

Обґрунтування вибору засобів розробки: Express.js

- Мінімалістичний фреймворк дозволяє швидко створювати REST API для обробки бронювань, користувачів та номерів.
- Масштабованість
- Дозволяє обробляти сотні запитів одночасно.
- Легка інтеграція з MongoDB
- Завдяки драйверам можна ефективно працювати з даними.
- Middleware для безпеки
- Валідація даних, JWT-аутентифікація та обмеження запитів (rate limiting) запобігають атакам.

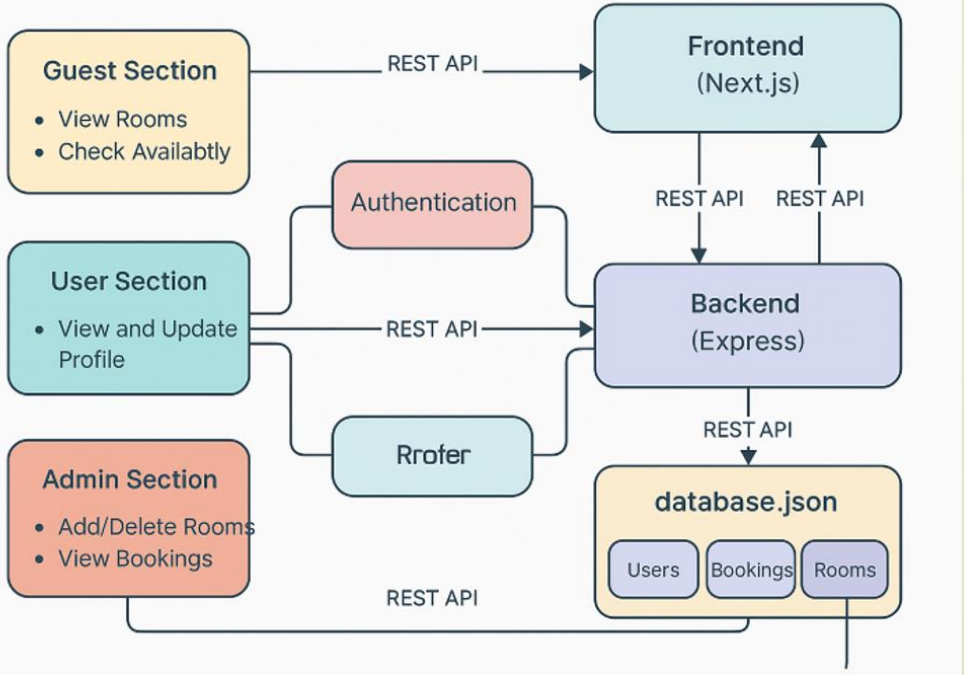
Обґрунтування вибору засобів розробки: MongoDB

- Немає жорсткої структури, що дозволяє легко додавати нові поля (наприклад, зручності номерів).
- Швидкість роботи з даними
- Оптимізована для операцій читання/запису, що важливо для часто оновлюваних бронювань.
- JSON-подібний формат
- Ідеально підходить для JavaScript-стеку (Next.js + Express.js).
- Підтримує горизонтальне масштабування для великої кількості бронювань.

Проєктування маршрутизації

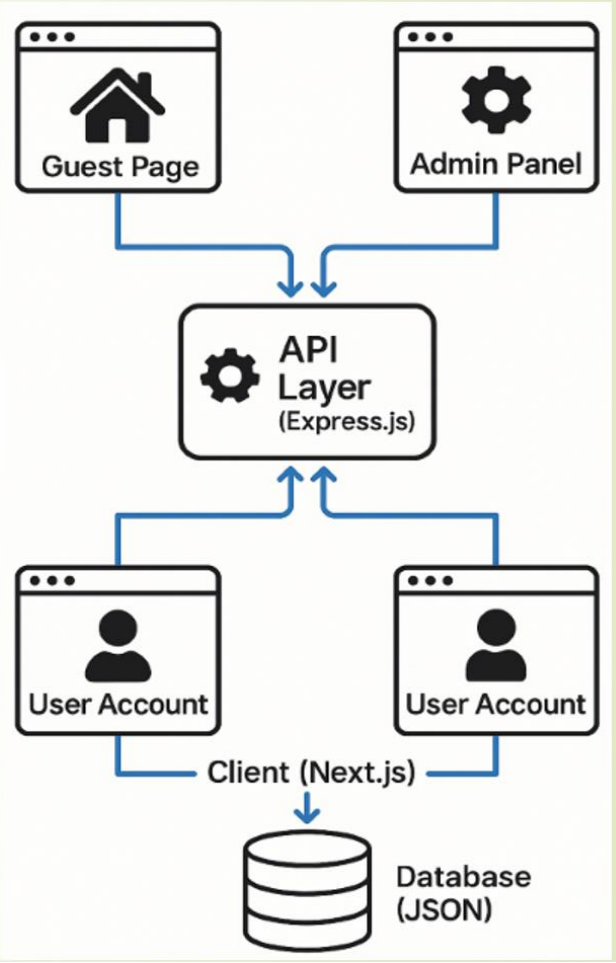
Маршрутизація у даній системі побудована за принципом розділення відповідальності між клієнтською (frontend) та серверною (backend) частинами. Фронтенд відповідає за відображення інтерфейсу, взаємодію з користувачем, відправку HTTP-запитів та отримання відповіді від серверу.

Бекенд обробляє запити, працює з базою даних, здійснює валідацію та реалізує бізнес-логіку додатку.



Проєктування маршрутизації

З архітектурної точки зору, розподіл на Frontend (Next.js) та Backend (Express.js) дозволяє організувати чітке розмежування обов'язків між візуальним інтерфейсом та логікою обробки запитів, що спрощує підтримку та розвиток проєкту. Вся взаємодія між клієнтською частиною та сервером відбувається через REST API, що дозволяє централізовано контролювати авторизацію, перевірку прав доступу та обробку даних.



Розробка користувацького інтерфейсу

Гостьова частина (Public Pages) — доступна всім відвідувачам сайту без авторизації. Включає:

- Головну сторінку з переліком доступних номерів;
- Сторінку з деталями окремого номера;
- Перевірки доступності за датами;
- Сторінки реєстрації та авторизації.

Особистий кабінет користувача — доступний після входу в систему. Реалізовано функції:

- Перегляд особистих бронювань;
- Можливість скасування або зміни бронювання;
- Редагування профілю (ім’я, пароль).

Адміністративна панель — доступна лише користувачам з правами адміністратора. Дозволяє:

- Додавати, редагувати або видаляти готельні номери;
- Переглядати список усіх бронювань;
- Переглядати список зареєстрованих користувачів.

Розробка користувацького інтерфейсу

Next.js — фреймворк на основі React, який забезпечує серверний рендеринг, високу продуктивність і підтримку SEO. Це дозволяє створювати масштабовані й ефективні веб-застосунки зі швидкою реакцією на дії користувача

Усі компоненти інтерфейсу побудовані із дотриманням принципів адаптивного дизайну, що гарантує коректну роботу на різних типах пристроїв — від комп'ютерів до мобільних телефонів.


```
import Link from 'next/link';
import { useEffect, useState } from 'react';
import 'bootstrap/dist/css/bootstrap.min.css';

export default function Home() {
  const [rooms, setRooms] = useState([]);
  const [user, setUser] = useState(null);

  useEffect(() => {
    fetch('/api/rooms')
      .then(res => res.json())
      .then(data => setRooms(data));

    const stored = localStorage.getItem('currentUser');
    if (stored) {
      setUser(JSON.parse(stored));
    }
  }, []);

  const handleLogout = () => {
    localStorage.removeItem('currentUser');
    window.location.reload();
  };

```

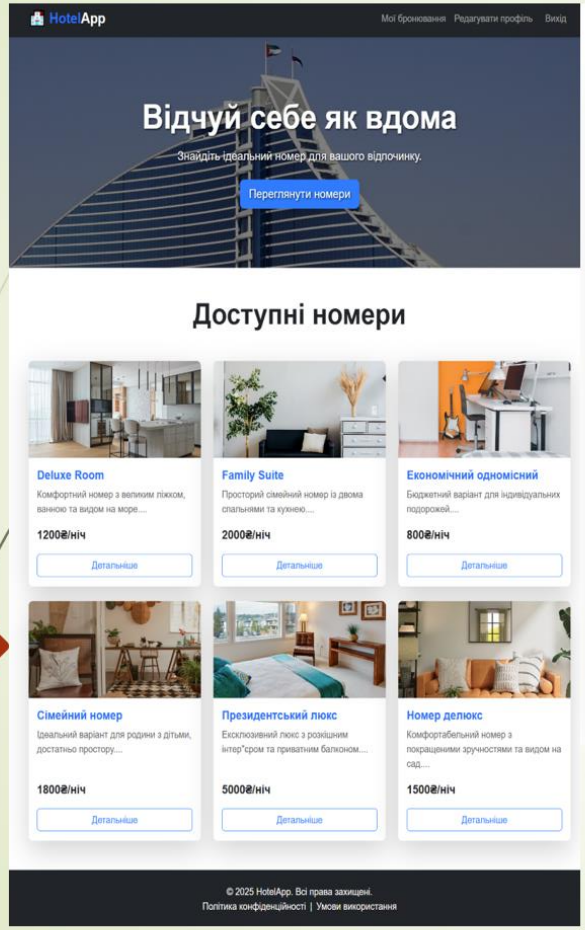
Точка входа index.js

Розробка користувацького інтерфейсу

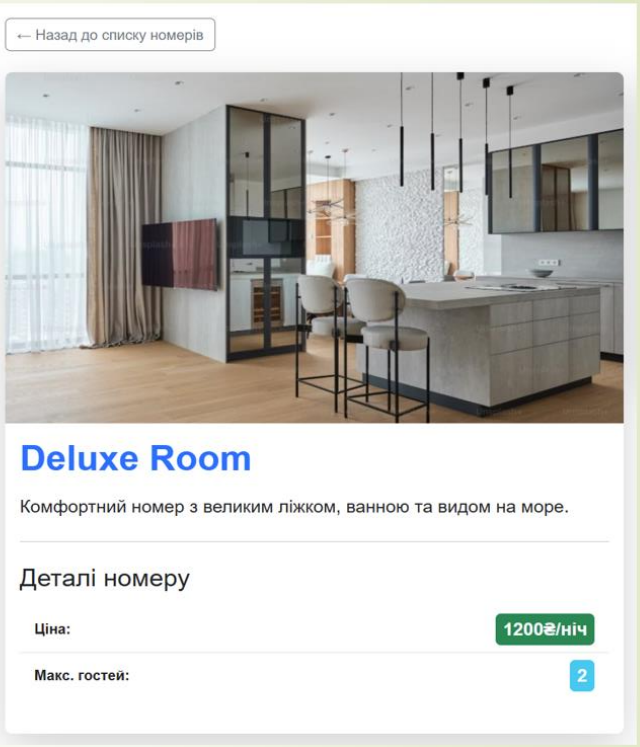
```
  return (
<div className="d-flex flex-column min-vh-100">
      {/* Хедер */}
      <header>
        <nav className="navbar navbar-expand-lg navbar-dark bg-dark shadow-sm">
          <div className="container">
            <Link href="/" className="navbar-brand fs-4 fw-bold">
              <span className="text-primary">Hotel</span>App
            </Link>
            <button
              className="navbar-toggler"
              type="button"
              data-bs-toggle="collapse"
              data-bs-target="#navbarNav"
              aria-controls="navbarNav"
              aria-expanded="false"
              aria-label="Toggle navigation"
            >
              <span className="navbar-toggler-icon"></span>
            </button>
            <div className="navbar-collapse" id="navbarNav">
              <ul className="navbar-nav ms-auto">
                {!user ? (
                  <>
                    <li className="nav-item">
                      <Link href="/register" className="nav-link">
                        Реєстрація
                      </Link>
                    </li>
                    <li className="nav-item">
                      <Link href="/login" className="nav-link btn btn-primary text-white ms-lg-2">
                        Вхід
                      </Link>
                    </li>
                  </>
                ) : (
                  <>
                    <li className="nav-item">
                      <Link href="/my-bookings" className="nav-link">
                        Мої бронювання
                      </Link>
                    </li>
                    <li className="nav-item">
                      <Link href="/edit-profile" className="nav-link">
                        Редагувати профіль
                      </Link>
                    </li>
                    <li className="nav-item">
                      <button
                        className="btn btn-outline-secondary nav-link ms-lg-2"
```

Верстка головної сторінки

Реалізація гостьової частини

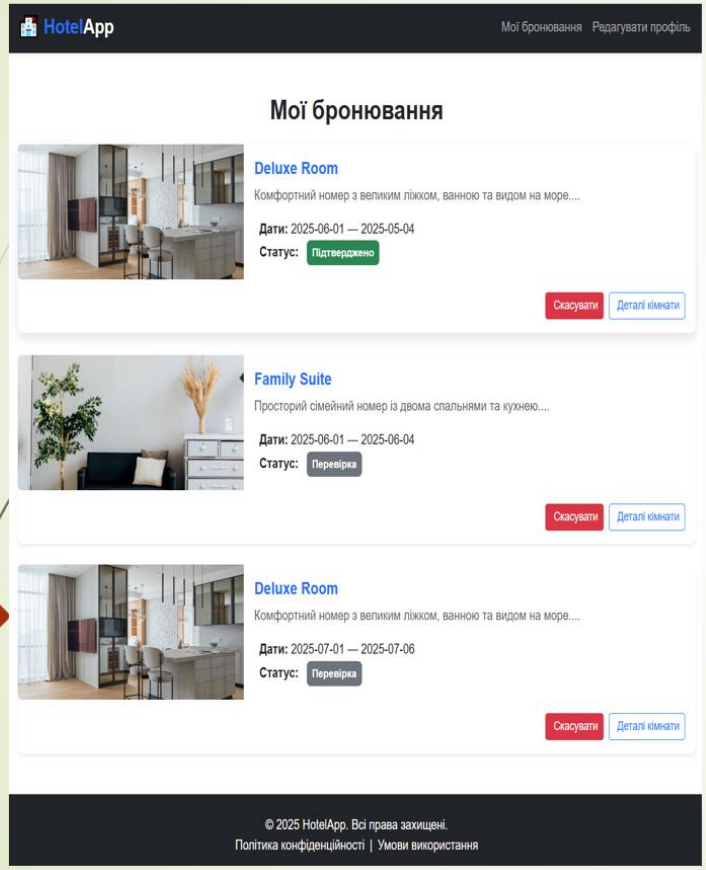


Головна сторінка

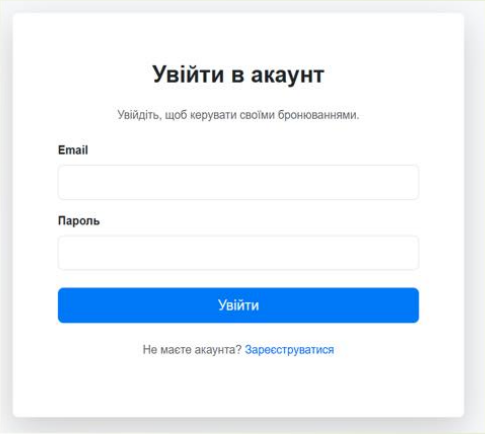


Сторінка з інформацією про номер

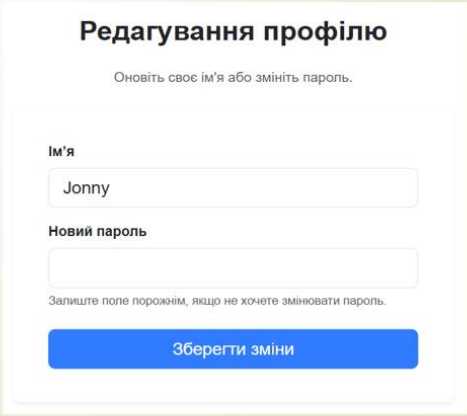
Реалізація функціоналу користувача



Сторінка всіх бронювань

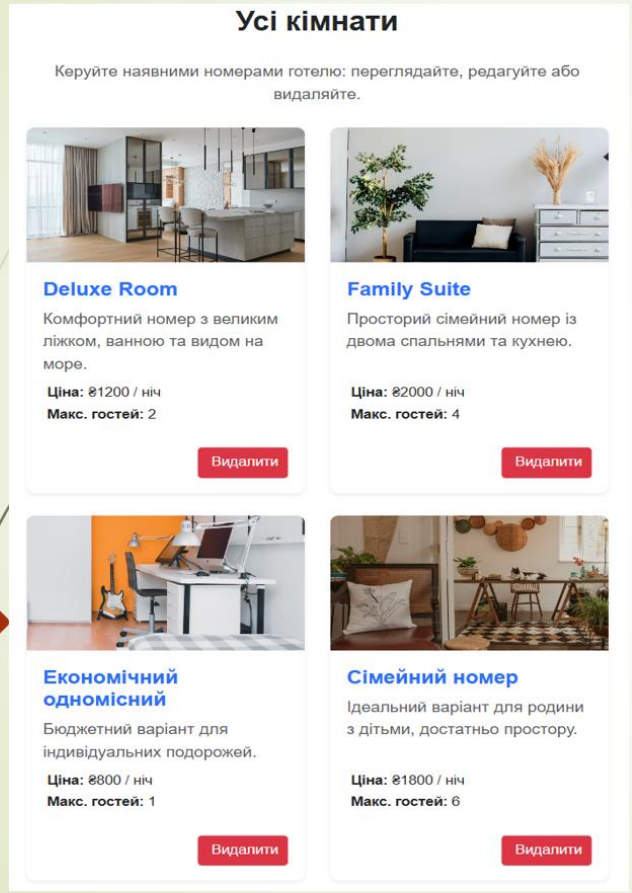


Форма авторизації

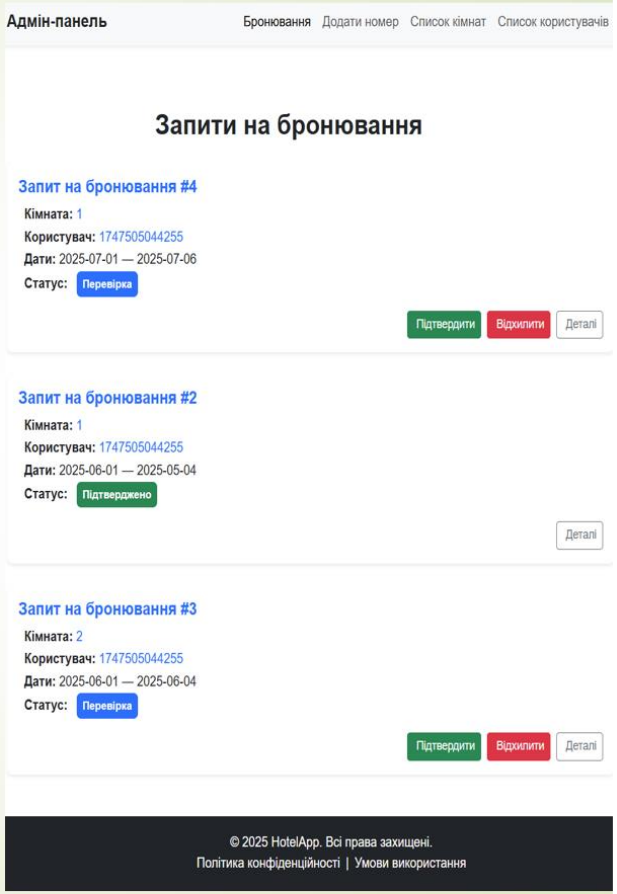


Форма редагування профіля

Реалізація функціоналу адміністратора



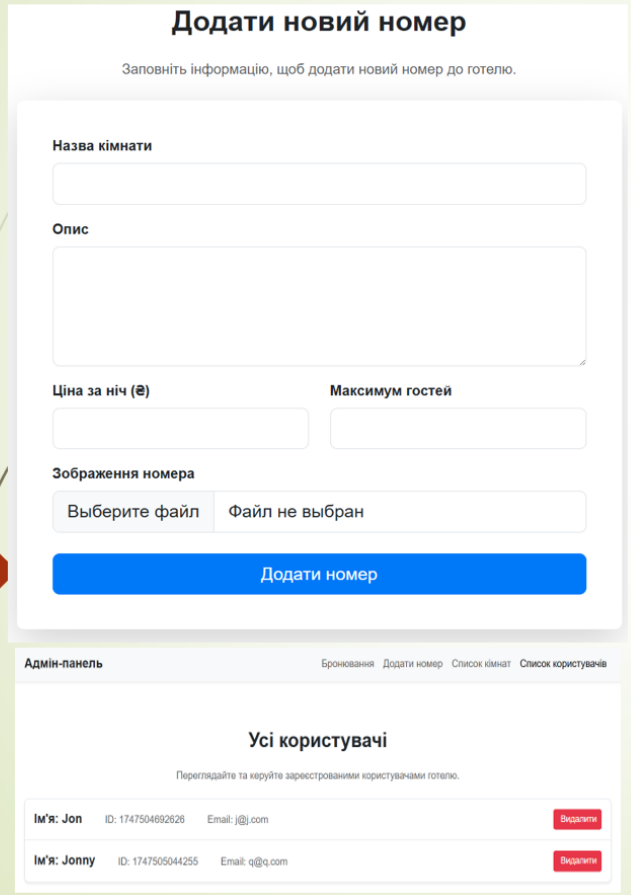
Список всіх кімнат



Список запитів на бронювання

Реалізація функціоналу адміністратора

Форма додавання даних



20

Додавання нового номеру в базу даних

```jsx
import { useState } from 'react';
import Link from 'next/link';

export default function AddRoom() {
  const [title, setTitle] = useState('');
  const [description, setDescription] = useState('');
  const [pricePerNight, setPricePerNight] = useState('');
  const [maxGuests, setMaxGuests] = useState('');
  const [fakeImage, setFakeImage] = useState(null);
  const [message, setMessage] = useState('');
  const [loading, setLoading] = useState(false);

  const handleAdd = async () => {
    if (!title || !description || !pricePerNight || !maxGuests) {
      setMessage('Будь ласка, заповніть усі поля');
      return;
    }

    const res = await fetch('/api/admin/add-room', {
      method: 'POST',
      headers: { 'Content-Type': 'application/json' },
      body: JSON.stringify({
        title,
        description,
        pricePerNight: Number(pricePerNight),
        maxGuests: Number(maxGuests),
      }),
    });

    const data = await res.json();
    if (data.success) {
      setMessage('Номер успішно додано!');
      setTitle('');
      setDescription('');
      setPricePerNight('');
      setMaxGuests('');
      setFakeImage(null);
    } else {
      setMessage('Помилка при додаванні номера');
    }
  };
```

Список користувачів

Висновки

Практична реалізація системи демонструє ефективність обраного підходу. NextJS забезпечив швидке завантаження сторінок та адаптивний інтерфейс, ExpressJS показав стабільну роботу при обробці одночасних запитів на бронювання, а MongoDB довела свою ефективність у роботі з динамічно змінюваними даними. Розроблений механізм бронювання включає всі необхідні етапи - від вибору дат до підтвердження резервації, з можливістю подальшого скасування.

Розроблений веб-додаток представляє собою сучасне рішення для автоматизації процесів бронювання, яке поєднує високий рівень користувацького досвіду з ефективними інструментами управління. Використання сучасного JavaScript-стеку дозволяє легко інтегрувати додаткові функції, такі як система лояльності або інтеграція з зовнішніми платіжними сервісами. Система може бути успішно впроваджена в готельному бізнесі, сприяючи підвищенню якості обслуговування клієнтів та оптимізації роботи персоналу.