ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Розробка та реалізація Web-застосунку для бронювання готельних номерів мовою JavaScript»

на здобуття освітнього ступеня бакалавра
зі спеціальності 121 Інженерія програмного забезпечення
освітньо-професійної програми «Інженерія програмного забезпечення»

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

_____ Денис ДЖИГА
(підпис)

Виконав: здобувач вищої освіти групи ПД-41

Денис ДЖИГА

Керівник: Ірина ЗАМРІЙ
д.т.н., доцент

Рецензент: _____

Київ 2024

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти Бакалавр

Спеціальність 121 Інженерія програмного забезпечення

Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного забезпечення
_____ Ірина ЗАМРІЙ
« ____ » _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Джига Денису Юрійовичу

1. Тема кваліфікаційної роботи: «Розробка та реалізація Web-застосунку для бронювання готельних номерів мовою JavaScript»
керівник кваліфікаційної роботи д.т.н., доцент, зав. кафедри ІПЗ Ірина ЗАМРІЙ,
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «27» лютого 2024 р. № 36.

2. Строк подання кваліфікаційної роботи «28» травня 2024 р.

3. Вихідні дані до кваліфікаційної роботи: теоретичні відомості про методи обробки природньої мови, опис методів вилучення інформації з тексту та формування тематичних словників, технічна документація з описом бібліотек для обробки природньої мови.

1. Офіційна документація бібліотек Python.
2. Офіційна документація Visual Studio Code
3. Опис методів та алгоритмів для обробки текстів природньою мовою

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Аналіз предметної галузі
2. Вибір засобів програмної реалізації
3. Проектування веб-додатку
4. Програмна реалізація веб-додатку

5. Перелік графічного матеріалу: *презентація*

1. Титульний лист
2. Мета, об’єкт та предмет дослідження
3. Задачі дипломної роботи
4. Аналіз аналогів
5. Вимоги до додатку
6. Програмні засоби реалізації
7. Структура бази даних
8. Карта сторінок веб-додатку
9. Демонстрація веб-додатку (Головна сторінка)
10. Демонстрація веб-додатку (Форма добавлення готелю)
11. Демонстрація веб-додатку (Форма добавлення номеру)
12. Демонстрація веб-додатку (Елемент готелю та доданого номеру)
13. Апробація результатів дослідження
14. Висновки

6. Дата видачі завдання «28» лютого 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Підбір та аналіз науково-технічної літератури	28.02.-06.03.2024	
2	Аналіз та дослідження існуючих аналогів	07.03-13.03.2024	
3	Огляд існуючих аналогів та побудова сайту бронювання готельних номерів	13.03-21.03.2024	

4	Проектування сайту бронювання готельних номерів	21.03-30.03.2024	
5	Програмна реалізація застосунку	30.03-17.04.2024	
6	Тестування застосунку	17.04-28.04.2024	
7	Оформлення роботи: вступ, висновки, реферат	29.04-05.05.2024	
8	Розробка демонстраційних матеріалів	06.05-12.05.2024	
9	Попередній захист роботи	13.05-31.05.2024	

Здобувач вищої освіти _____ Денис ДЖИГА
(підпис)

Керівник
кваліфікаційної роботи _____ Ірина ЗАМРІЙ
(підпис)

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 55 стор., 26 рис., 1 табл., 9 джерел.

Об'єкт дослідження: Процес бронювання готельних номерів у готельному бізнесі.

Предмет дослідження: Методи та засоби автоматизації бронювання готельних номерів за допомогою веб-додатку, включаючи інтерфейси користувача, алгоритми пошуку та фільтрації, а також інтеграцію з базами даних.

Мета роботи: Визначити ефективні способи підвищення зручності та автоматизації процесу бронювання готельних номерів за допомогою розробленого веб-додатку.

Для досягнення мети необхідно виконати наступні задачі:

1. Аналіз існуючих рішень: Огляд веб-додатків для бронювання, визначення переваг і недоліків, виділення ключових можливостей;
2. Проектування архітектури: Розробка архітектури додатку, визначення компонентів системи, та БД;
3. Реалізація функціональності: Пошук і фільтрація номерів, перегляд деталей і доступності, процес бронювання та адміністрування в додатку.

Галузь використання – готельний бізнес. Зокрема, проект може бути корисним для людей, які прагнуть автоматизувати процес бронювання та покращити зручність для своїх клієнтів.

КЛЮЧОВІ СЛОВА: FLASK, HTML, CSS, JAVASCRIPT, POSTGRESQL, АВТОМАТИЗАЦІЯ, БРОНЮВАННЯ, ГОТЕЛЬ, НОМЕР, ВЕБ-ДОДАТОК.

ЗМІСТ

ВСТУП

У сучасному світі цифрових технологій, готельний бізнес все більше залежить від ефективних онлайн-рішень для бронювання номерів. Зручність і швидкість процесу бронювання є ключовими факторами, що впливають на задоволення клієнтів та успіх готелів. Більшість існуючих платформ для бронювання часто не забезпечують належного рівня автоматизації та зручності, що призводить до втрати потенційних клієнтів.

Розробка спеціалізованого веб-додатку для бронювання готельних номерів, який би поєднував у собі передові технології та зручний інтерфейс, є надзвичайно актуальною. Такий додаток може значно спростити процес пошуку і бронювання номерів для користувачів, забезпечити інтеграцію з базами даних готелів, та надати потужні інструменти для фільтрації і сортування варіантів розміщення.

У зв’язку з високою конкуренцією в готельному бізнесі, готелі повинні постійно шукати нові способи залучення клієнтів та підвищення якості обслуговування. Автоматизація процесу бронювання через веб-додаток допоможе зменшити витрати на адміністрування та покращити взаємодію з клієнтами, забезпечуючи їм більш персоналізований та швидкий сервіс.

Об’єкт дослідження: Процес бронювання готельних номерів у готельному бізнесі.

Предмет дослідження: Методи та засоби автоматизації бронювання готельних номерів за допомогою веб-додатку, включаючи інтерфейси користувача, алгоритми пошуку та фільтрації, а також інтеграцію з базами даних.

Мета роботи: Визначити ефективні способи підвищення зручності та автоматизації процесу бронювання готельних номерів за допомогою розробленого веб-додатку.

Для досягнення мети необхідно виконати наступні задачі:

Аналіз існуючих рішень - огляд веб-додатків конкурентів для бронювання, визначення їх переваг і недоліків, виділення ключових можливостей;

Проектування архітектури - розробка архітектури додатку, визначення компонентів системи, та БД;

Реалізація функціональності - пошук і фільтрація номерів готелів, перегляд деталей і доступності для бронювання, процес бронювання та адміністрування в додатку.

Галузь використання – готельний бізнес. Зокрема, проект може бути корисним для людей, які прагнуть автоматизувати процес бронювання та покращити зручність для своїх клієнтів.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1. Принцип бронювання готельних номерів в інтернеті

У сучасному світі все більше людей використовують Інтернет для бронювання готельних номерів. Це забезпечує зручність, швидкість і можливість порівнювати різні варіанти в реальному часі. Завдяки онлайн-сервісам, користувачі можуть швидко знайти відповідний готель, ознайомитися з відгуками, переглянути фотографії номерів та зручно забронювати потрібний варіант. В умовах високої конкуренції серед готелів, наявність ефективного та зручного веб-додатку для бронювання стає ключовим фактором для залучення клієнтів.

Загальний процес бронювання готельних номерів в інтернеті можна окреслити наступними етапами [1]:

Пошук готелів, користувач починає з введення основних параметрів пошуку, таких як місто або регіон, дати заїзду та виїзду, кількість гостей і бажані зручності. На основі цих параметрів система відображає список доступних варіантів;

Фільтрація та сортування, щоб звузити результати пошуку, користувачі можуть використовувати різноманітні фільтри, наприклад, за ціною, рейтингом, зручностями (басейн, фітнес-центр тощо), відстанню від центру міста та іншими критеріями. Результати також можна сортувати за ціною, популярністю або рейтингом;

Огляд деталей, після застосування фільтрів користувач може переглянути деталі кожного готелю, включаючи фотографії номерів, опис зручностей, політику щодо бронювання та скасування, а також відгуки інших гостей. Це допомагає зробити більш обґрунтований вибір;

Вибір номера, користувач обирає конкретний номер на основі наданої інформації. Тут важливо враховувати тип номера, його вартість, доступні зручності та умови скасування;

Заповнення даних, після вибору номера користувач переходить до форми бронювання, де вказує особисті дані, такі як ім'я, контактна інформація та особливі побажання. На цьому етапі також зазвичай вказується інформація про платіж;

Підтвердження бронювання, після заповнення всіх необхідних даних користувач підтверджує бронювання. Система обробляє запит, перевіряє наявність номера та здійснює платіж, якщо це необхідно;

Отримання підтвердження, після успішного бронювання користувач отримує підтвердження на вказану електронну адресу. У підтвердженні містяться всі деталі бронювання, включаючи дати перебування, інформацію про номер та політику скасування;

Взаємодія з готелем, в деяких випадках користувач може спілкуватися з готелем для уточнення додаткових деталей або побажань. Це може бути здійснено через спеціальний чат або електронну пошту.

Онлайн-бронювання готельних номерів значно спрощує процес пошуку і бронювання, надаючи користувачам можливість швидко і зручно знайти найкращі варіанти розміщення. Завдяки цьому, готелі можуть залучати більше клієнтів, забезпечуючи їм високий рівень сервісу і зручності.

1.2. Аналіз аналогових веб-додатків

Аналіз існуючих веб-додатків для бронювання готельних номерів є важливим етапом розробки власного додатку. Це дозволяє визначити сильні та слабкі сторони конкурентів, виявити кращі практики та уникнути помилок, допущених іншими. Розуміння цих аспектів допоможе створити більш ефективний та зручний продукт, який відповідатиме потребам користувачів.

Booking.com – один з найбільших гравців на ринку онлайн-бронювання готелів. Він пропонує величезну базу даних готелів та зручний інтерфейс для користувачів [2]. Основною перевагою є широкий вибір варіантів розміщення,

можливість порівняння цін та доступність відгуків від інших користувачів (рисунок 1.1).

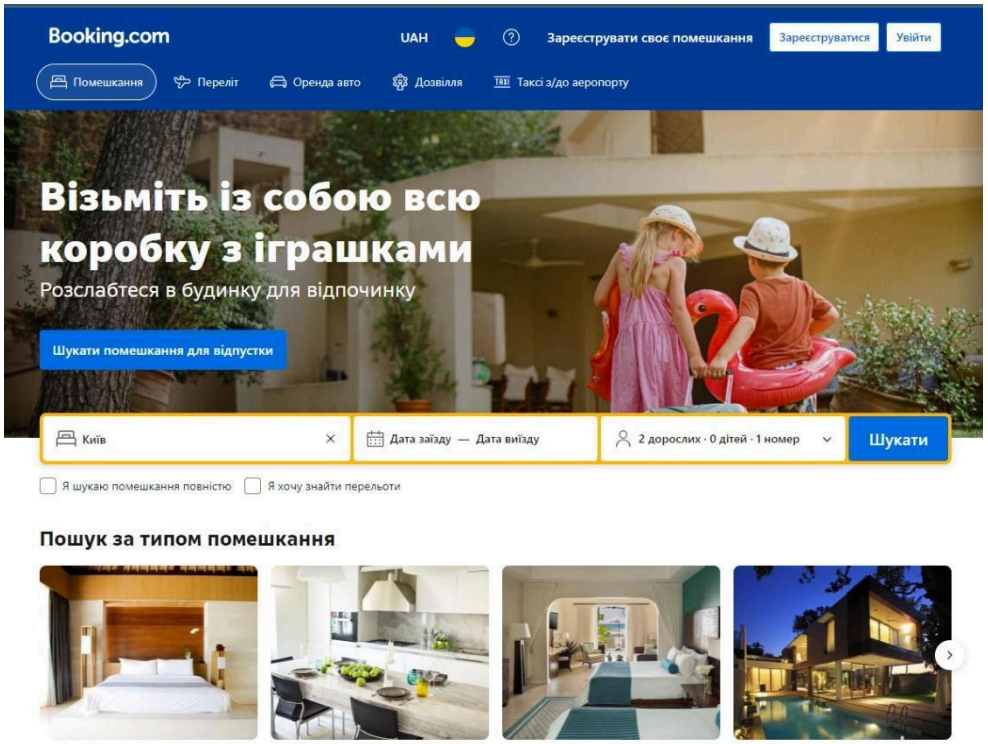


Рис. 1.1 Booking.com (Головна сторінка)

Сервіс також пропонує гнучкі умови бронювання та часто має вигідні пропозиції і знижки. Однак, високі комісії для готелів можуть бути значним недоліком, що може вплинути на вартість для кінцевого користувача. Іншим недоліком є можливість переповнення інформацією, що ускладнює вибір.

Airbnb спеціалізується на пропозиціях унікального житла, включаючи оренду квартир, будинків та навіть кімнат. Головними перевагами є інтеграція з соціальними мережами, можливість знаходити унікальні пропозиції та досвід «як вдома» [3]. Сервіс дозволяє користувачам знаходити житло в різних куточках світу та надає можливість безпосередньо спілкуватися з власниками (рисунок 1.2). Також Airbnb пропонує зручний мобільний додаток, який дозволяє користувачам здійснювати бронювання, спілкуватися з власниками та отримувати необхідну інформацію прямо зі своїх смартфонів.

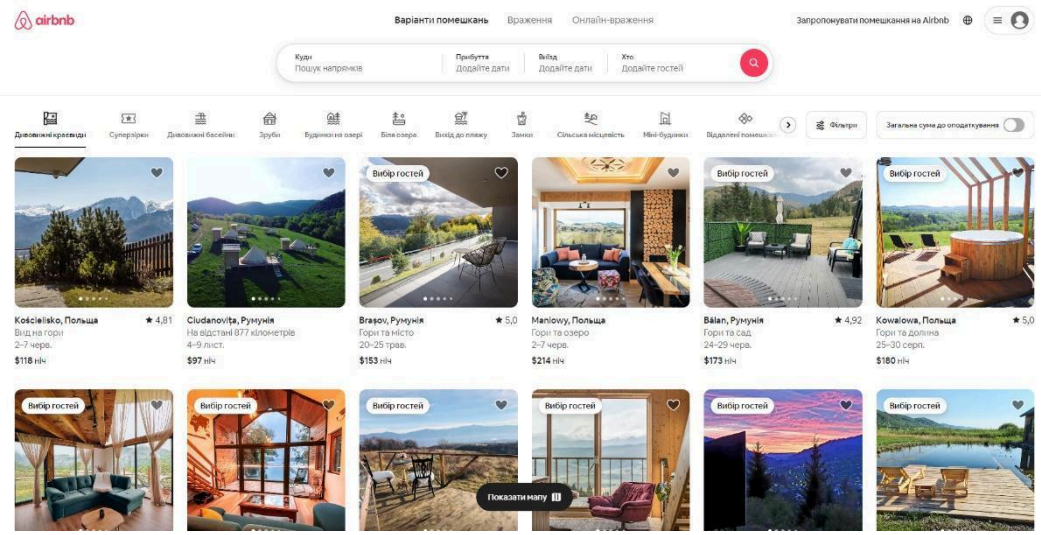


Рис. 1.2 Airbnb (Головна сторінка)

Недоліками є відсутність професійної підтримки у випадку проблем, обмежена кількість готелів, а також можливі проблеми з достовірністю оголошень.

Expedia надає великий вибір туристичних послуг, включаючи бронювання готелів, авіаквитків та оренду автомобілів. Перевагою є можливість планування комплексних поїздок, наявність програм лояльності та спеціальних пропозицій для зареєстрованих користувачів (рисунок 1.3) [4].

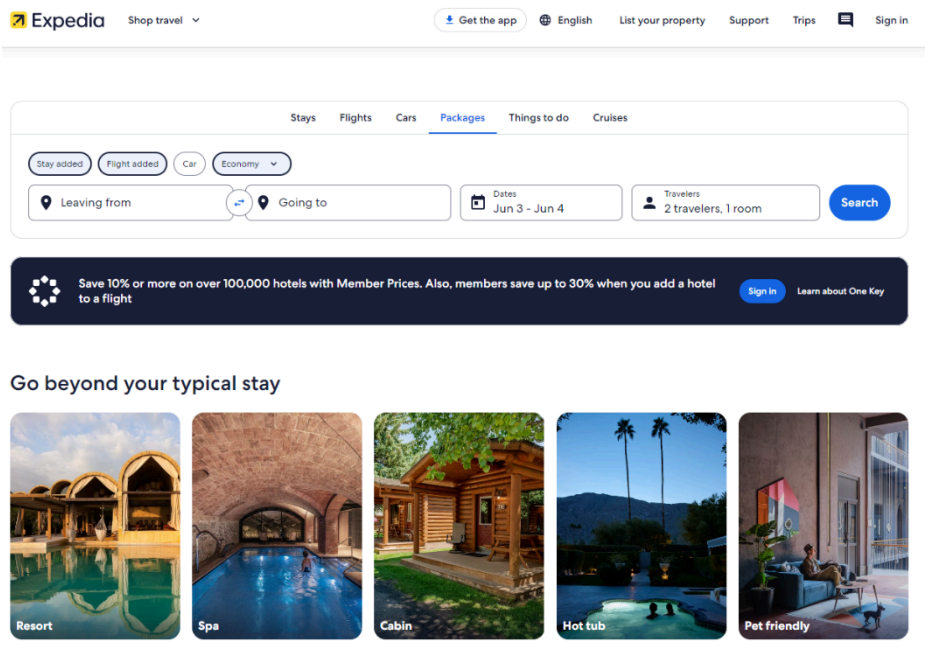


Рис. 1.3 Expedia (Головна сторінка)

Однак, недоліками є складний інтерфейс, велика кількість реклами, а також можливі проблеми з підтримкою клієнтів. Часто користувачі стикаються з труднощами у зміні або скасуванні бронювань, що може створювати незручності.

Критерії для порівняння аналогів були обрані на основі ключових функціональних можливостей, які впливають на якість та зручність користування веб-додатком для бронювання готелів:

Пошук і фільтрація номерів: Важливість зручного пошуку для користувачів полягає у швидкому знаходженні потрібного варіанту розміщення в готелі. Фільтрація за різними параметрами, такими як ціна, розташування, зручності в номері, допомагають звузити пошук і знайти оптимальний варіант для користувача;

Перегляд деталей і доступності: Надання повної інформації про номери та їх доступність є критично важливим для прийняття рішення користувачами для користування нашим веб-додатком . Це включає в себе фотографії, описи, політики бронювання , скасування та відгуки інших користувачів;

Процес бронювання: Зручність та швидкість процесу бронювання впливають на загальний досвід користувача. Інтуїтивно зрозумілий інтерфейс, мінімальна кількість кроків для завершення бронювання, можливість зберігати інформацію для майбутніх бронювань – все це сприяє позитивному враженню від сервісу;

Адміністрування бронювань: Можливість ефективного управління бронюваннями дозволяє користувачам легко змінювати або скасовувати свої бронювання, а також переглядати історію бронювань. Це важливо для підтримки довіри та задоволеності клієнтів;

Адаптивний дизайн: Забезпечення доступності на різних пристроях (компьютерах, планшетах, смартфонах) є необхідним для сучасних веб-додатків. Це дозволяє користувачам здійснювати бронювання з будь-якого місця і в будь-який час;

Інтеграція з платіжними системами: Зручність та безпека проведення платежів є важливими аспектами для завершення процесу бронювання. Підтримка різних платіжних методів, шифрування даних та захист від шахрайства підвищують довіру користувачів до сервісу;

Безпека даних: Забезпечення захисту персональної інформації користувачів є критично важливим для будь-якого веб-додатку. Це включає захист від несанкціонованого доступу, збереження конфіденційності даних та відповідність стандартам безпеки;

На основі цих критеріїв була створена порівняльна таблиця (див. табл. 1.1)

Таблиця 1.1

Порівняння аналогів з майбутнім веб-додатком «Reservia»

Параметр	**Booking.com**	**Airbnb**	**Expedia**	**Reservia**
Переваги	Велика база даних готелів, зручний інтерфейс, гнучкі умови бронювання,	Унікальні пропозиції житла, інтеграція з соцмережами," можливість безпосереднього спілкування з власниками	Великий вибір туристичних послуг, можливість планування комплексних поїздок, програми лояльності, спеціальні пропозиції	Простота використання, швидкість роботи, високий рівень безпеки даних
Недоліки	Високі комісії для готелів, можливість переповнення інформацією	Відсутність професійної підтримки, обмежена кількість готелів	Складний інтерфейс, багато реклами, проблеми з підтримкою клієнтів	Обмежена база даних
Пошук і фільтрація номерів	Так	Так	Так	Так
Перегляд деталей і доступності	Так	Так	Так	Так
Процес бронювання	Так	Так	Так	Так
Адміністрування бронювань	Так	Так	Так	Так
Адаптивний дизайн	Так	Так	Так	Так
Інтеграція з платіжними системами	Так	Так	Так	Ні
Безпека даних	Високий рівень	Високий рівень	Високий рівень	Високий рівень

Аналіз аналогових веб-додатків для бронювання готельних номерів показав, що кожен з розглянутих сервісів має свої унікальні переваги та недоліки. Booking.com виділяється великою базою даних готелів та зручним інтерфейсом, що робить його привабливим для широкого кола користувачів. Проте високі комісії для готелів і можливість переповнення інформацією можуть бути вагомими недоліками. Airbnb пропонує унікальні варіанти житла та інтеграцію з соціальними мережами, створюючи досвід «як вдома». Однак, відсутність професійної підтримки і проблеми з достовірністю оголошень можуть створювати незручності для користувачів. Expedia надає широкий спектр туристичних послуг і можливість планування комплексних поїздок, але складний інтерфейс і велика кількість реклами можуть знизити користувацький досвід.

Reservia демонструє простоту використання, швидкість роботи та високий рівень безпеки даних, що є вагомими перевагами. Обмежена база даних є значним недоліком, але проект володіє значними перспективами для розвитку. Загалом, кожен з розглянутих сервісів відповідає різним потребам користувачів. Reservia, зокрема, має потенціал для масштабування та розширення бази даних у майбутньому. Простий і зручний інтерфейс робить його привабливим для користувачів, а високий рівень безпеки забезпечує надійний захист даних. Незважаючи на обмеженість ресурсів для розробки, при подальшому розвитку проект може стати конкурентоспроможним рішенням на ринку онлайн-бронювання готелів.

1.3. Формування задач роботи

Для успішної реалізації проекту веб-додатку для бронювання готельних номерів необхідно виконати ряд задач, які забезпечують високу якість, зручність та функціональність додатку. Основні задачі можна розділити на кілька ключових етапів:

Визначення вимог до системи: визначення функціональних та нефункціональних вимог до системи, створення специфікацій системи;

Розробка загальної архітектури: вибір архітектурного стилю, визначення основних компонентів системи;

Дизайн бази даних: вибір типу бази даних, проектування схеми бази даних;

Пошук і фільтрація номерів: реалізація механізму пошуку за ключовими параметрами, впровадження фільтрів для звуження результатів пошуку, оптимізація алгоритмів пошуку;

Перегляд деталей і доступності: створення інтерфейсу для перегляду детальної інформації про номери, реалізація функції перевірки доступності номерів;

Процес бронювання: розробка форми бронювання, реалізація логіки обробки бронювання;

Адміністрування бронювань: створення панелі адміністратора для управління бронюваннями;

Забезпечення безпеки даних: реалізація системи аутентифікації та авторизації, проведення регулярних перевірок безпеки;

Виконання цих задач забезпечує створення надійного, зручного та функціонального веб-додатку для бронювання готельних номерів, який відповідатиме сучасним вимогам ринку та очікуванням користувачів.

2 ВИБІР ЗАСОБІВ ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

2.1. Принцип роботи фреймворку Flask

Flask це мікро фреймворк для розробки веб-додатків на мові програмування Python. Він був створений для забезпечення простоти використання та високої гнучкості, дозволяючи розробникам швидко створювати веб-додатки будь-якої складності. Основна концепція Flask полягає в мінімалізмі та модульності, що дозволяє розробникам додавати лише ті компоненти та функціональність, які їм необхідні. Flask є мікро фреймворком, що означає мінімальний набір базових функцій. Він не включає в себе жорстких обмежень або складних структур, що дозволяє розробникам самостійно вибирати компоненти, які вони хочуть використовувати [5]. Це робить Flask ідеальним для швидкої розробки прототипів або створення невеликих додатків, які можна легко масштабувати. Flask спроектований таким чином, щоб бути простим і інтуїтивно зрозумілим. Він використовує мінімальну кількість абстракцій і дозволяє розробникам створювати додатки, просто дотримуючись стандартних практик Python. Це знижує криву навчання і дозволяє швидко почати роботу з фреймворком.

Як функціонують проекти у Flask [6]:

Проекти у Flask організовані навколо кількох ключових компонентів:

Об’єкт додатку: У центрі кожного проекту Flask знаходиться об’єкт додатку. Він створюється з використанням класу Flask і служить головною точкою входу для додатку. Об’єкт додатку відповідає за маршрутизацію запитів, обробку відповідей і управління конфігурацією додатку;

Маршрутизація: Flask використовує систему маршрутизації для визначення того, які функції повинні обробляти запити до різних URL-адрес. Розробник може визначити маршрути, використовуючи декоратор @app.route(). Кожен маршрут прив’язаний до функції, яка обробляє запит і формує відповідь;

Обробка запитів і відповідей: Flask дозволяє легко обробляти HTTP-запити та відповідати на них. Запити містять інформацію про метод (GET, POST, PUT,

DELETE), заголовки, параметри та тіло запиту. Відповідь може включати HTML, JSON або інші типи даних, залежно від потреб додатку.

Також перевагами використання Flask є:

Шаблонний Jinja2: Flask інтегрується з Jinja2, потужним шаблонним двигуном для Python. Це дозволяє розробникам створювати динамічні HTML-сторінки, використовуючи шаблони. Jinja2 підтримує змінні, цикли, умовні оператори та інші конструкції, що дозволяють легко створювати складні інтерфейси;

Статичні файли: Flask автоматично обробляє статичні файли, такі як CSS, JavaScript та зображення. Вони зберігаються у папці static і можуть бути легко включені в шаблони або повернуті як частина відповіді на запит;

Конфігурація: Flask підтримує різні способи налаштування додатку. Конфігураційні параметри можуть бути задані через файли конфігурації, змінні середовища або безпосередньо у коді додатку. Це дозволяє легко адаптувати додаток до різних середовищ (розробка, тестування, продуктивність);

Розширення: Flask має багату екосистему розширень, які додають додаткові можливості, такі як підтримка форм (Flask-WTF), автентифікація (Flask-Login), взаємодія з базами даних (Flask-SQLAlchemy) та багато іншого. Розширення дозволяють легко розширювати функціональність додатку без написання великої кількості коду.

Flask дозволяє швидко створювати прототипи і розгортати додатки, що особливо корисно в умовах обмеженого часу, тому цей мікро-фреймворк був обраний для реалізації веб-додатку.

2.2. Середовище Visual Studio Code

Visual Studio Code (VS Code) є одним із найпопулярніших середовищ розробки, і для реалізації веб-додатку на Flask воно пропонує низку переваг, які роблять його найкращим вибором. Перш за все, VSCode є легким і швидким середовищем розробки, що забезпечує швидкий запуск та плавну роботу навіть на

менш потужних комп’ютерах. Це особливо важливо для розробників, які працюють на різноманітних пристроях і потребують стабільної продуктивності без затримок.

Однією з ключових переваг VS Code є його потужна підтримка розширень. Середовище має величезну кількість розширень, які полегшують роботу з Flask, Python, HTML, CSS, JavaScript та іншими технологіями. Завдяки цим розширенням, розробники отримують доступ до таких функцій, як налагодження, автозавершення коду, лінтинг та багато інших корисних можливостей, що спрощують і прискорюють процес розробки. Важливим аспектом є інтеграція з Git. VS Code має вбудовану підтримку системи контролю версій Git, що дозволяє легко керувати версіями коду, створювати коміти, гілки та зливання змін прямо з редактора. Це значно спрощує процес спільної роботи над проектами та забезпечує зручне керування історією змін [7].

Зручний інтерфейс є ще однією суттєвою перевагою VS Code. Інтерфейс інтуїтивно зрозумілий і легко налаштовується під індивідуальні потреби розробника. Можливість відкривати кілька файлів у вкладках, підтримка панелей і терміналів роблять роботу зручною і ефективною, що особливо цінується розробниками під час роботи над складними проектами. Завдяки цим перевагам, Visual Studio Code забезпечує ефективний, продуктивний і зручний процес розробки, роблячи його ідеальним вибором для реалізації веб-додатків на Flask. Його можливості та зручність використання допомагають розробникам зосередитися на створенні якісного програмного забезпечення, не відволікаючись на технічні дрібниці.

2.3. Клієнтська частина додатку: HTML, Javascript та CSS

Клієнтська частина веб-додатку складається з трьох основних технологій: HTML, JavaScript та CSS. Кожна з них виконує свою унікальну роль у створенні інтерактивних та візуально привабливих веб-сторінок.

HTML є основою будь-якої веб-сторінки. Це мова розмітки, яка використовується для структурування вмісту веб-додатків. HTML визначає, які

елементи будуть присутні на сторінці, такі як заголовки, параграфи, списки, посилання, зображення та форми. HTML працює як скелет веб-сторінки, надаючи базову структуру, яку можна стилізувати та оживити за допомогою CSS і JavaScript [8].

Основні особливості HTML:

Семантика: HTML5 представив нові семантичні елементи, такі як <header>, <footer>, <article>, <section>, які допомагають краще структурувати контент;

Форми: HTML дозволяє створювати інтерактивні форми, що використовуються для збору даних від користувачів;

Медіа: HTML підтримує інтеграцію мультимедійних елементів, таких як відео (<video>) та аудіо (<audio>).

CSS використовується для стилізації HTML-елементів, роблячи веб-сторінки візуально привабливими і естетичними. CSS визначає, як елементи HTML повинні відображатися на екрані, включаючи кольори, шрифти, розташування, розміри та анімації. CSS дозволяє відокремити візуальне оформлення від структури документа, що робить код більш організованим і легшим для підтримки.

Основні особливості CSS:

Селектори: CSS використовує селектори для вибору HTML-елементів, до яких застосовуються стилі;

Каскадність: Стилі застосовуються у визначеному порядку, що дозволяє контролювати пріоритетність різних стилів;

Медіа-запити: CSS дозволяє створювати адаптивні дизайни, які змінюються в залежності від розміру екрану або пристрою, на якому переглядається сторінка;

Анімації та переходи: CSS підтримує створення анімацій та плавних переходів між станами елементів.

JavaScript мова програмування, яка додає інтерактивності та динамічної поведінки веб-сторінкам. JavaScript дозволяє змінювати вміст HTML і CSS у реальному часі, реагуючи на дії користувачів, такі як кліки, введення даних або прокручування сторінки. JavaScript є основним інструментом для розробки клієнтської логіки і взаємодії з користувачем.

Основні особливості JavaScript:

Динамічне маніпулювання DOM: JavaScript дозволяє змінювати структуру та вміст HTML-документа після його завантаження;

Події: JavaScript обробляє події користувача, такі як кліки, натискання клавіш та зміни у формах;

Взаємодія з сервером: JavaScript може асинхронно взаємодіяти з сервером через технології, такі як AJAX, дозволяючи оновлювати дані без перезавантаження сторінки;

Бібліотеки та фреймворки: Існує багато бібліотек (наприклад, jQuery) та фреймворків (наприклад, React, Angular, Vue), які спрощують розробку складних веб-додатків.

Ці три технології тісно інтегровані між собою, створюючи повноцінний клієнтський інтерфейс веб-додатків. HTML забезпечує базову структуру сторінки, CSS відповідає за її стилізацію, а JavaScript додає динаміку і інтерактивність. Разом вони дозволяють створювати сучасні, адаптивні та інтерактивні веб-додатки, які забезпечують чудовий користувацький досвід. Розробка клієнтської частини веб-додатку за допомогою HTML, CSS та JavaScript є основою сучасного веб-розробки. Кожна з цих технологій виконує свою унікальну роль, і разом вони створюють можливості для створення функціональних, привабливих та ефективних веб-додатків.

2.4. Серверна частина: Python

Серверна частина веб-додатку є критично важливою для його функціональності та ефективності. Вона відповідає за обробку запитів від клієнтської частини, управління базами даних, виконання бізнес-логіки та забезпечення безпеки. Для розробки серверної частини веб-додатків часто використовується мова програмування Python, завдяки її потужним можливостям, простоті та читабельності коду. Python є одним із найпопулярніших мов

програмування для веб-розробки з кількох причин. По-перше, Python має синтаксис, який легко читається і пишеться, що прискорює процес розробки і знижує ймовірність помилок. Це робить Python особливо привабливим для новачків та досвідчених розробників. По-друге, Python має величезну кількість бібліотек і фреймворків, таких як Flask, Django, Pyramid, які полегшують розробку веб-додатків. Ці інструменти надають готові рішення для поширених завдань, таких як маршрутизація запитів, автентифікація користувачів, робота з базами даних та багато іншого. Крім того, Python підтримує безліч бібліотек для роботи з даними, машинного навчання, наукових обчислень та інших областей, що робить його універсальним інструментом для різних типів проектів [9].

Маршрутизація є основним аспектом серверної частини, що визначає, як обробляти запити до різних URL-адрес. Python-фреймворки надають інструменти для легкої конфігурації маршрутів, що дозволяє направляти запити до відповідних функцій або методів. Серверна частина обробляє HTTP-запити від клієнтів, виконує необхідну бізнес-логіку і повертає відповідні відповіді. Це може включати взаємодію з базами даних, обчислення, виклики зовнішніх API та інше.

Взаємодія з базами даних є ще одним важливим аспектом серверної частини. Серверна частина часто взаємодіє з базами даних для збереження та отримання даних. Python підтримує різні бібліотеки для роботи з базами даних, такі як SQLAlchemy, Peewee для реляційних баз даних і PyMongo для нереляційних баз даних.

Вибір Python для серверної частини також обумовлений великою та активною спільнотою, яка постійно розробляє нові інструменти та надає підтримку. Це дозволяє швидко знаходити рішення на виникаючі питання і проблеми, забезпечуючи стабільний і продуктивний процес розробки.

3 ПРОЕКТУВАННЯ ВЕБ-ДОДАТКУ

3.1. Формування вимог до додатку

Формування вимог до додатку є важливим етапом розробки програмного забезпечення. Вимоги визначають, що саме повинен робити додаток, а також яким чином він повинен функціонувати. Цей процес допомагає розробникам зрозуміти очікування користувачів та зацікавлених сторін, що дозволяє створити продукт, який відповідає їхнім потребам. Вимоги можна поділити на дві основні категорії: функціональні та нефункціональні.

Функціональні вимоги описують конкретні функції, які повинен виконувати додаток. Вони визначають взаємодію користувача з системою та описують, як саме додаток повинен реагувати на різні дії користувача.

У контексті веб-додатку для бронювання готелів, функціональні вимоги включають наступне:

Пошук і фільтрація готелів: Додаток повинен надавати користувачам можливість шукати готелі за різними критеріями, такими як місцезнаходження, дати перебування, ціна, та інші зручності. Користувачі повинні мати змогу застосовувати фільтри для звуження результатів пошуку відповідно до своїх потреб;

Перегляд деталей і доступності номерів: Додаток повинен надавати детальну інформацію про кожен готель та його номери, включаючи фотографії, опис, ціну та доступність у вибрані дати. Користувачі повинні мати можливість переглядати цю інформацію перед бронюванням.

Процес бронювання номерів: Додаток повинен забезпечувати користувачам зручний інтерфейс для бронювання номерів. Користувачі повинні отримувати підтвердження свого бронювання з усіма деталями.

Адміністрування бронювань: Додаток повинен надавати адміністраторам можливість керувати бронюваннями, включаючи перегляд, редагування та скасування бронювань.

Нефункціональні вимоги визначають якісні характеристики системи, такі як продуктивність, безпека та зручність використання. Вони не стосуються конкретних функцій додатку, але є критично важливими для забезпечення задоволення користувачів та стабільної роботи системи.

У контексті веб-додатку для бронювання готелів, ключові нефункціональні вимоги включають:

Продуктивність: Додаток повинен забезпечувати високу швидкість роботи і швидкий відгук на дії користувачів. Час завантаження сторінок, обробки запитів та виконання операцій повинен бути мінімальним;

Безпека даних: Додаток повинен забезпечувати захист персональних даних користувачів від несанкціонованого доступу та витоку. Це включає використання шифрування для передачі даних, безпечного зберігання даних та механізмів аутентифікації та авторизації;

Адаптивний дизайн: Додаток повинен мати адаптивний дизайн, який забезпечує коректне відображення та зручність використання на різних пристроях, включаючи комп'ютери, планшети та смартфони;

Надійність: Додаток повинен бути стабільним і надійним, забезпечуючи безперебійну роботу навіть при високому навантаженні. Це включає стійкість до збоїв, ефективне управління ресурсами та регулярне резервне копіювання даних;

Зручність використання: Інтерфейс додатку повинен бути інтуїтивно зрозумілим і простим у використанні, забезпечуючи позитивний досвід для користувачів. Це включає логічну організацію елементів, зрозумілу навігацію та доступність інформації.

Формування вимог до додатку є основою для розробки успішного програмного продукту. Вимоги дозволяють чітко визначити, які функції та характеристики повинен мати додаток, щоб задовольнити потреби користувачів і відповідати очікуванням зацікавлених сторін. Детальне опрацювання функціональних і нефункціональних вимог допомагає уникнути непорозумінь у процесі розробки і забезпечує створення якісного та надійного продукту.

3.2. Архітектура веб-додатку, основні сторінки

Архітектура веб-додатку визначає його структуру, взаємодію між різними компонентами та логіку роботи. Додаток, побудований на Flask, зазвичай складається з трьох основних частин: інтерфейсу, серверної частини та бази даних (рисунок 3.1). Ця архітектура забезпечує чітке розділення функцій, що полегшує розробку, тестування та підтримку додатку.

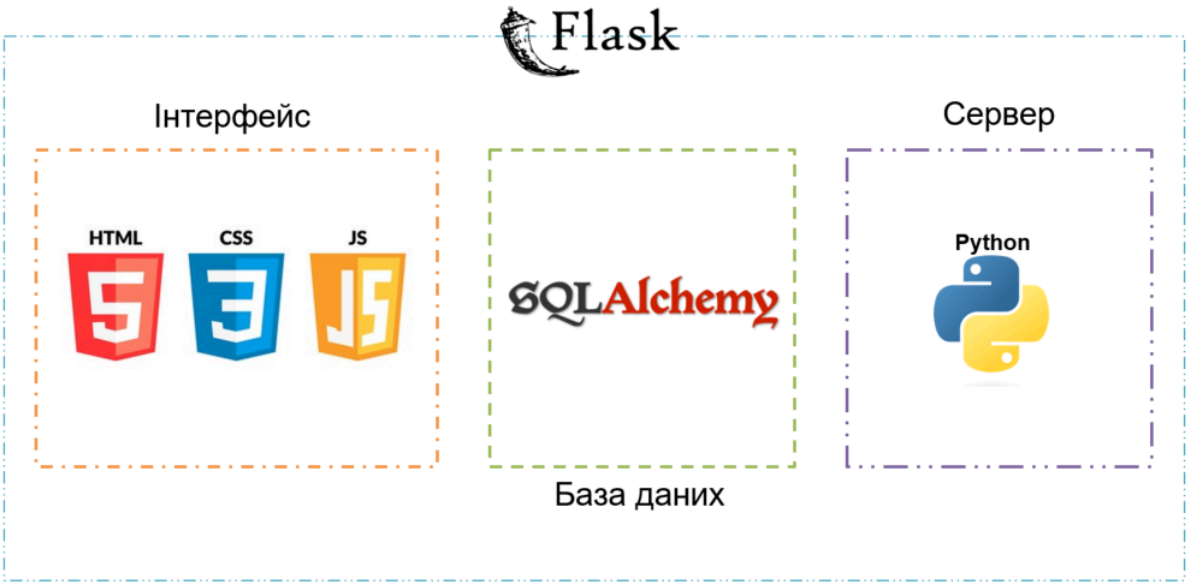


Рис. 3.1 Архітектура веб-додатку

Основні компоненти архітектури:

Інтерфейс користувача (Frontend) відповідає за відображення даних та взаємодію з користувачем. Він складається з HTML для структури сторінок, CSS для стилізації та JavaScript для додавання інтерактивності. HTML визначає структуру

веб-сторінок, CSS відповідає за їх оформлення, а JavaScript забезпечує динамічну взаємодію з користувачем;

Серверна частина (Backend) реалізована на Python з використанням фреймворку Flask. Сервер обробляє запити від клієнта, виконує бізнес-логіку, взаємодіє з базою даних та повертає відповіді клієнту. Сервер також забезпечує безпеку додатку, управління сесіями та автентифікацію користувачів;

База даних відповідає за збереження та управління даними додатку. Для взаємодії з базою даних використовується ORM (Object-Relational Mapping) бібліотека SQLAlchemy, яка дозволяє працювати з базами даних у зручний спосіб, використовуючи об'єктно-орієнтований підхід.

Карта сторінок веб-додатку допомагає візуалізувати структуру додатку та взаємозв’язки між різними сторінками. Вона визначає основні маршрути та навігацію, полегшуючи користувачам орієнтацію в додатку (рисунок 3.2).

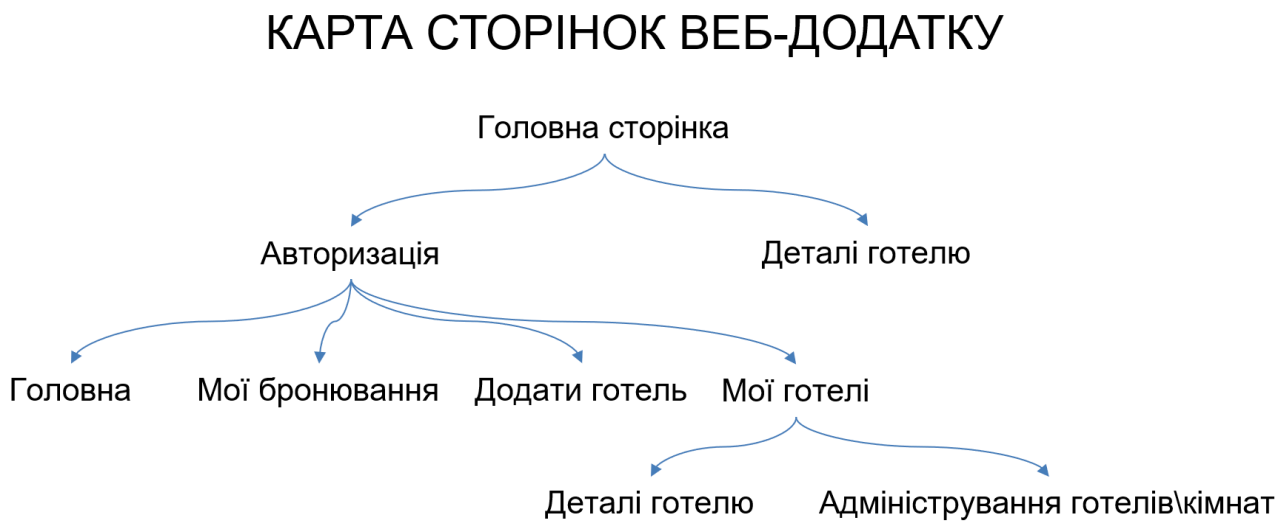


Рис. 3.2 Карта сторінок веб-додатку

Основні сторінки веб-додатку:

Головна сторінка: Це початкова сторінка додатку, яка надає загальну інформацію та доступ до основних функцій;

Авторизація: Сторінка авторизації дозволяє користувачам входити в систему, використовуючи свої облікові дані. Після успішного входу користувачі отримують доступ до персоналізованих функцій;

Мої бронювання: Ця сторінка надає користувачам можливість переглядати свої активні та минулі бронювання. Користувачі можуть отримувати детальну інформацію про кожне бронювання, а також скасовувати або редагувати їх;

Додати готель: Сторінка для додавання нових готелів до системи;

Мої готелі: Сторінка, на якій адміністратори або власники готелів можуть переглядати список своїх готелів, редагувати інформацію про них та керувати номерами;

Деталі готелю: Сторінка з детальною інформацією про конкретний готель, включаючи опис, фотографії, доступні номери та ціни. Користувачі можуть переглядати цю інформацію перед бронюванням;

Адміністрування готелів/кімнат: Ця сторінка дозволяє адміністраторам керувати готелями та номерами, включаючи додавання, редагування та видалення записів.

Архітектура веб-додатку та карта сторінок допомагають організувати розробку та забезпечити логічну структуру додатку. Це сприяє підвищенню продуктивності розробки, поліпшенню користувацького досвіду та забезпеченню гнучкості при майбутніх змінах або розширеннях функціональності. Крім того, чітке розмежування між різними компонентами системи полегшує тестування та виявлення помилок, а також сприяє командній роботі, оскільки різні розробники можуть працювати над окремими частинами додатку незалежно один від одного. Такий підхід гарантує масштабованість і підтримку високих стандартів якості коду.

3.3. Формування бази даних

Формування бази даних є важливим етапом розробки будь-якого веб-додатку, оскільки саме база даних забезпечує збереження, організацію та доступ до інформації. У випадку додатку для бронювання готелів було вирішено створити чотири основні таблиці: User, Hotel, Room та Booking (рисунок 3.3). Ці таблиці були обрані для забезпечення ефективного зберігання та управління даними, пов’язаними з користувачами, готелями, номерами та бронюваннями.

Рис. 3.3 Структура бази даних

Таблиця User зберігає інформацію про користувачів додатку. Вона включає такі поля:

id (Integer, primary_key=True): Унікальний ідентифікатор користувача;

username (String(80), unique=True, nullable=False): Унікальне ім'я користувача;

email (String(120), unique=True, nullable=False): Унікальна електронна пошта користувача;

password (String(120), nullable=False): Хешований пароль користувача;

is_active (Boolean, default=False, nullable=False): Поле, що вказує, чи активний користувач;

hotels (relationship): Відношення «один до багатьох» з таблицею Hotel, що дозволяє зберігати готелі, створені користувачем;

bookings (relationship): Відношення «один до багатьох» з таблицею Booking, що дозволяє зберігати бронювання, зроблені користувачем;

Таблиця User дозволяє зберігати та керувати інформацією про користувачів, включаючи їхні облікові дані та активність.

Таблиця Hotel зберігає інформацію про готелі. Вона включає такі поля:

id (Integer, primary_key=True): Унікальний ідентифікатор готелю;

name (String(120), nullable=False): Назва готелю;

address (String(255), nullable=False): Адреса готелю;

contact (String(120), nullable=False): Контактна інформація готелю;

description (Text, nullable=False): Опис готелю;

photo_urls (JSON): URL фотографій готелю;

services (JSON, nullable=True): Список послуг, що надаються готелем;

owner_id (Integer, ForeignKey('user.id'), nullable=False): Ідентифікатор власника готелю;

rooms (relationship): Відношення «один до багатьох» з таблицею Room, що дозволяє зберігати номери, доступні в готелі.

Таблиця Hotel забезпечує збереження та управління інформацією про готелі, включаючи їхні назви, адреси, контактну інформацію, описи, фотографії та послуги.

Таблиця Room зберігає інформацію про номери в готелях. Вона включає такі поля:

id (Integer, primary_key=True): Унікальний ідентифікатор номера;

name (String(120), nullable=False): Назва номера;

is_available (Boolean, default=True, nullable=False): Поле, що вказує на доступність номера;

price (Float, nullable=False): Вартість номера за ніч;

description (Text, nullable=False): Опис номера;

capacity (Integer, nullable=False): Кількість гостей, яких може прийняти номер;

hotel_id (Integer, ForeignKey('hotel.id'), nullable=False): Ідентифікатор готелю, до якого належить номер;

available_from (Date, nullable=False): Дата, з якої номер доступний для бронювання;

available_to (Date, nullable=False): Дата, до якої номер доступний для бронювання;

Таблиця Room дозволяє зберігати та керувати інформацією про номери, включаючи їхні назви, доступність, ціну, опис та місткість.

Таблиця Booking зберігає інформацію про бронювання номерів. Вона включає такі поля:

id (Integer, primary_key=True): Унікальний ідентифікатор бронювання;

hotel_name (String(120), nullable=False): Назва готелю, в якому заброньовано номер;

room_name (String(120), nullable=False): Назва забронюваного номера;

start_date (Date, nullable=False): Дата початку бронювання;

end_date (Date, nullable=False): Дата завершення бронювання;

price (Float, nullable=False): Вартість бронювання;

user_id (Integer, ForeignKey('user.id'), nullable=False): Ідентифікатор користувача, який здійснив бронювання.

Таблиця Booking забезпечує збереження та управління інформацією про бронювання, включаючи дати, вартість та дані користувача, який здійснив бронювання.

Обрані таблиці та їх поля дозволяють ефективно зберігати та керувати інформацією, необхідною для функціонування додатку для бронювання готелів. Вони забезпечують чітке розділення даних за користувачами, готелями, номерами та бронюваннями, що сприяє підвищенню продуктивності та зручності роботи з базою даних.

3.4. Структура проекту

Організація структури проекту є важливим аспектом розробки веб-додатків. Правильна структура дозволяє легко орієнтуватися в коді, забезпечує зрозумілий розподіл обов'язків між компонентами та сприяє масштабованості додатку. Загальна структура проекту (рисунок 3.4).

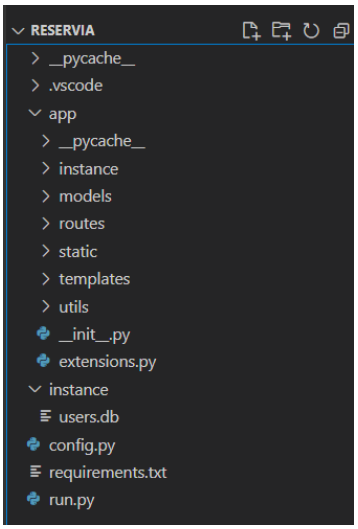


Рис. 3.4 «Reservia» (структура додатку)

Основні директорії та файли проекту розташовані наступним чином:

app/ – головна директорія додатку, що містить основний код додатку;

__init__.py – файл ініціалізації додатку;

extensions.py – файл для підключення розширень Flask;

instance/ – директорія для файлів, які змінюються під час виконання додатку (наприклад, база даних);

models/ – директорія для моделей бази даних;

routes/ – директорія для маршрутизації;

static/ – директорія для статичних файлів (CSS, JavaScript, зображення);

templates/ – директорія для HTML-шаблонів;

utils/ – директорія для допоміжних функцій і утиліт;

instance/ – директорія для конфігураційних файлів та бази даних;

users.db – файл бази даних SQLite;

config.py – файл конфігурації додатку;

config.py – файл глобальної конфігурації додатку;

requirements.txt – файл з переліком залежностей проекту;

run.py – файл запуску додатку.

models/ директорія містить моделі бази даних (рисунок 3.5), кожна з яких представляє певну таблицю:

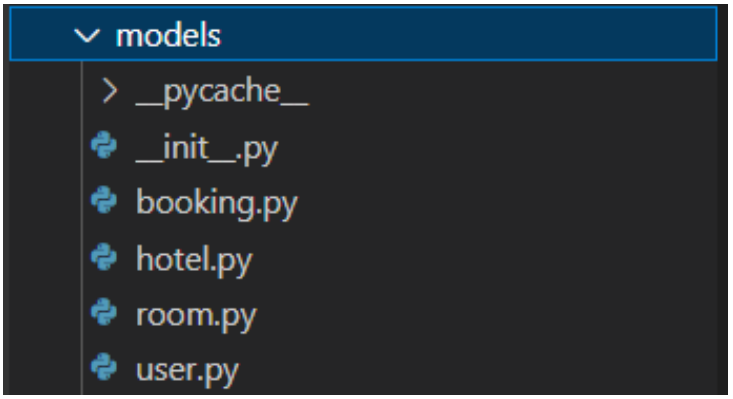


Рис. 3.4 Директорія (models/)

__init__.py – файл ініціалізації моделей;

booking.py – модель для таблиці Booking, яка зберігає інформацію про бронювання;

hotel.py – модель для таблиці Hotel, яка зберігає інформацію про готелі;

room.py – модель для таблиці Room, яка зберігає інформацію про номери в готелях;

user.py – модель для таблиці User, яка зберігає інформацію про користувачів.

routes/ директорія містить файли маршрутизації (рисунок 3.5), які визначають шляхи для обробки запитів:

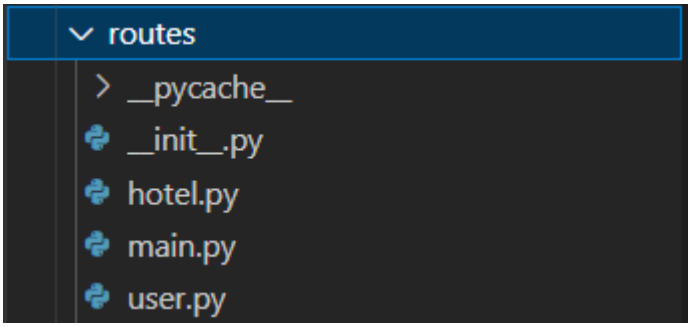


Рис. 3.5 Директорія (routes/)

__init__.py – файл ініціалізації маршрутів;

hotel.py – маршрути для обробки запитів, пов’язаних з готелями;

main.py – основні маршрути для додатку;

user.py – маршрути для обробки запитів, пов'язаних з користувачами.

static/директорія містить статичні файли (рисунок 3.6), які відправляються клієнту без змін:

js/ – директорія для JavaScript-файлів;

home.js – скрипти для головної сторінки;

hotel_details.js – скрипти для сторінки деталей готелю;

hotel.js – скрипти для управління готелями;

register.js – скрипти для реєстрації користувачів;

uploads/ – директорія для завантажених файлів, таких як фотографії готелів;

CSS файли – файли стилів для різних сторінок додатку, включаючи home.css, hotel_card_main.css, hotel_card.css, hotel_details.css та index.css.

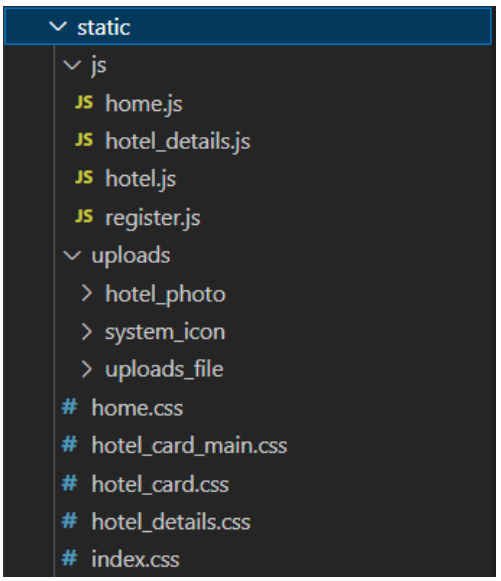


Рис. 3.6 Директорія (static/)

templates/ директорія містить HTML-шаблони (рисунок 3.7) для рендерингу сторінок на стороні сервера:

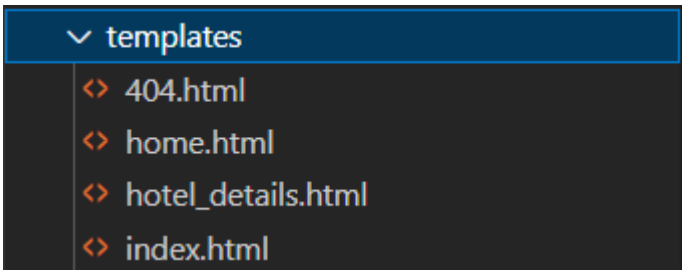


Рис. 3.7 — Директорія templates/

404.html – шаблон сторінки помилки 404;
home.html – шаблон головної сторінки;
hotel_details.html – шаблон сторінки деталей готелю;
index.html – основний шаблон сторінки.

Організація проекту у вигляді чіткої структури з розподілом на різні директорії та файли забезпечує зручність розробки, легкість підтримки та масштабування додатку. Така структура дозволяє ефективно управляти кодом, розподіляти обов'язки між різними компонентами та забезпечує зрозумілий і логічний розподіл функціональності.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ ВЕБ-ДОДАТКУ

4.1. Авторизація користувача

Авторизація користувача є важливим елементом будь-якого веб-додатку, що дозволяє користувачам безпечно входити в систему та отримувати доступ до персоналізованих функцій. Реєстрація нового користувача починається з отримання даних форми, таких як електронна пошта, логін та пароль. Важливо переконатися, що дані відповідають мінімальним вимогам щодо довжини та унікальності. Ось ключові етапи процесу:

Отримання даних форми:

```
if request.method == 'POST':
    email = request.form['email']
    username = request.form['reg_username']
    password = request.form['reg_password']
```

Перевірка довжини логіна, паролю та електронної пошти:

```
if len(username) < 8 or len(password) < 8 or len(email) < 5:
    flash('Логін, пароль та email мають бути достатньо довгими!', 'error')
    return render_template('index.html')
```

Перевірка унікальності логіна та електронної пошти:

```
existing_user = User.query.filter_by(username=username).first()
existing_email = User.query.filter_by(email=email).first()
if existing_user or existing_email:
    flash('Користувач з таким логіном або email вже існує.', 'error')
    return render_template('index.html')
```

Створення нового користувача та збереження його в базі даних:

```
new_user = User(username=username, email=email, password=password, is_active=True)
db.session.add(new_user)
db.session.commit()
user_login(new_user)
flash('Реєстрація пройшла успішно!', 'success')
return redirect(url_for('user.user_home'))
```

Процес входу користувача передбачає перевірку введеного логіна та паролю з тими, що зберігаються в базі даних. Якщо дані коректні, користувач успішно входить в систему.

Отримання даних форми:

```
    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']
```

Перевірка довжини логіна та паролю:

```
    if len(username) < 8 or len(password) < 8:
        flash('Логін та пароль повинні містити не менше 8 символів!', 'error')
        return render_template('index.html')
```

Перевірка наявності користувача в базі даних:

```
    user = User.query.filter_by(username=username, password=password).first()
    if user:
        user.is_active = True
        db.session.commit()
        user_login(user)
        flash('Вхід успішний!', 'success')
        return redirect(url_for('user.user_home'))
    else:
        flash('Помилка входу! Перевірте дані та спробуйте ще раз.', 'error')
```

Коли користувач намагається отримати доступ до персоналізованих сторінок, таких як user_home, необхідно перевірити, чи активна сесія користувача. Перевірка наявності користувача в сесії:

```
    if 'user_id' in session:
        user_id = session['user_id']
        user = User.query.get(user_id)
        if user:
             return render_template('home.html', user=user, username=user.username,
user_id=user.id)
        else:
            return render_template('404.html'), 404
    else:
        return render_template('404.html'), 404
```

Процес авторизації користувача включає кілька важливих етапів, таких як отримання даних форми, перевірка їх довжини та унікальності, створення нового користувача, збереження його в базі даних, перевірка даних при вході та управління сесіями.

4.2. Адміністрування готелю та номеру

Адміністрування готелів та номерів включає створення, перегляд та видалення нових готелів та номерів до існуючих готелів. Цей процес важливий для забезпечення актуальності інформації про доступні готелі та номери.

4.2.1 Створення

Процес створення готелю починається з отримання даних форми, перевірки наявності готелю з такою ж назвою, збереження завантажених фотографій та збереження нової інформації в базі даних.

HTML-форма для створення готелю забезпечує введення необхідних даних користувачем та завантаження фотографій:

```
<div id="create_hotel" class="content">
                    <form    id="hotelForm"    action="/create_hotel"    method="post"
enctype="multipart/form-data">
                <input type="hidden" id="user_id" name="user_id" value="{{ user_id
}}">
            <label for="name">Назва готелю:</label>
            <input type="text" id="name" name="name" required><br><br>
            <label for="description">Опис:</label>
                                <textarea    id="description"    name="description"
required></textarea><br><br>
            <label for="address">Адреса:</label>
            <input type="text" id="address" name="address" required><br><br>
            <label for="contact">Контакти:</label>
            <input type="text" id="contact" name="contact" required><br><br>
            <label for="photo">Фото готелю:</label>
                        <input   type="file"   id="photo"   name="photo"   multiple
accept="image/*"><br><br>
            <div id="preview"></div>
            <label>Сервіси:</label>
            <div class="checkboxes">
                <!-- Перелік чекбоксів для сервісів -->
            </div><br><br>
            <button type="submit" class="btn">ДОДАТИ ГОТЕЛЬ</button>
        </form>
</div>
```

Отримання даних форми:

```
@hotel.route('/create_hotel', methods=['POST'])
def create_hotel():
    name = request.form['name']
    description = request.form['description']
    address = request.form['address']
    contact = request.form['contact']
    services = request.form.getlist('services')
    user_id = request.form['user_id']
```

Перевірка наявності готелю з такою ж назвою:

```
existing_hotel = Hotel.query.filter(Hotel.name.ilike(name)).first()
if existing_hotel:
    return jsonify({'error': 'Готель з такою назвою вже існує!'}), 400
```

Збереження завантажених фотографій:

```
photo_urls = []
for file in request.files.getlist('photo'):
    filename = secure_filename(file.filename)
    file_path = os.path.join(Config.HOTEL_IMG, filename)
    file.save(file_path)
    photo_urls.append(filename)
```

Збереження нової інформації про готель в базі даних:

```
hotel = Hotel(
    name=name,
    description=description,
    address=address,
    contact=contact,
    photo_urls=photo_urls,
    services=services,
    owner_id=user_id
)
db.session.add(hotel)
db.session.commit()
return jsonify({'message': 'Готель успішно створено!'})
```

Процес додавання номера до готелю включає отримання даних форми, перевірку заповнення всіх обов'язкових полів, перетворення дат у відповідний формат та збереження інформації про новий номер в базі даних. Отримання даних форми та перевірка заповнення обов'язкових полів:

```
@hotel.route('/add_room', methods=['POST'])
def add_room():
     required_fields = ['name', 'price', 'description', 'capacity', 'hotel_id',
'available_from', 'available_to']
     if not all(field in request.json for field in required_fields):
         return jsonify({'error': 'Missing data!'}), 400
```

Перетворення дат у відповідний формат:

```
try:
            available_from = datetime.strptime(request.json['available_from'],
'%d.%m.%Y').date()
               available_to = datetime.strptime(request.json['available_to'],
'%d.%m.%Y').date()
except ValueError:
    return jsonify({'error': 'Invalid date format!'}), 400
```

Збереження нової інформації про номер в базі даних:

```
new_room = Room(
    name=request.json['name'],
    price=request.json['price'],
    description=request.json['description'],
    capacity=request.json['capacity'],
    hotel_id=request.json['hotel_id'],
    is_available=True,
    available_from=available_from,
    available_to=available_to
)
db.session.add(new_room)
db.session.commit()
return jsonify({'message': 'Номер успішно додано!', 'room_id': new_room.id}),
201
```

Процес додавання готелів та номерів включає створення нових готелів, перевірку унікальності їхніх назв, збереження фотографій та додавання номерів до існуючих готелів. Правильна організація цього процесу забезпечує актуальність та точність інформації про доступні готелі та номери, що є важливим для користувачів та адміністраторів системи.

4.2.2 Перегляд

Перегляд створених готелів та номерів є важливим елементом адміністрування в додатку. Він дозволяє власникам готелів переглядати та керувати інформацією про свої готелі та номери, а також користувачам переглядати деталі доступних варіантів. Процес перегляду готелів, що належать певному користувачу, починається з отримання ідентифікатора користувача, перевірки його існування та отримання списку готелів, що належать цьому користувачу.

Отримання готелів користувача:

```
@hotel.route('/user/<int:user_id>/hotels')
def get_user_hotels(user_id):
    user = User.query.get(user_id)
    if not user:
        return jsonify({'error': 'User not found'}), 404
```

Формування даних про готелі та номери:

```
hotels = []
for hotel in Hotel.query.filter_by(owner_id=user_id).all():
    rooms = Room.query.filter_by(hotel_id=hotel.id).all()
    rooms_data = [{
        'id': room.id,
```

```
                'name': room.name,
                'is_available': room.is_available,
                'price': room.price,
                'description': room.description,
                'capacity': room.capacity,
                       'available_from':  room.available_from.strftime('%d.%m.%Y')  if
room.available_from else None,
                          'available_to':  room.available_to.strftime('%d.%m.%Y')  if
room.available_to else None
            } for room in rooms]
            hotels.append({
                'id': hotel.id,
                'name': hotel.name,
                'address': hotel.address,
                'contact': hotel.contact,
                'description': hotel.description,
                'photo_urls': hotel.photo_urls,
                'services': hotel.services,
                'owner_id': hotel.owner_id,
                'rooms': rooms_data
            })
        return jsonify(hotels)
```

Процес перегляду деталей готелю включає отримання ідентифікаторів готелю та користувача, перевірку існування готелю, отримання списку номерів, що належать цьому готелю, та рендеринг відповідної HTML-сторінки з даними.

Отримання ідентифікаторів готелю та користувача:

```
@hotel.route('/hotel_details')
def hotel_details():
    serviceIcons = {
        "WiFi": "fa-wifi",
        "Парковка": "fa-car",
        "Басейн": "fa-swimmer",
        "Спортзал": "fa-dumbbell",
        "Спа": "fa-spa",
        "Ресторан": "fa-utensils",
        "Бар": "fa-glass-martini-alt",
        "Пральня": "fa-soap",
        "Трансфер": "fa-shuttle-van",
        "Конференц зала": "fa-users",
        "Можна з тваринами": "fa-paw",
        "Кондиціонер": "fa-wind",
        "Казино": "fa-dice",
        "Пляж": "fa-umbrella-beach",
        "Сніданок": "fa-bread-slice"
    }
    hotel_id = request.args.get('hotelId', type=int)
    user_id = request.args.get('userId', type=int)
    if not hotel_id:
        return "Hotel ID is required", 400
    if not user_id:
        return "User ID is required", 400
```

Перевірка існування готелю та формування даних про номери:

```
hotel = Hotel.query.get(hotel_id)
if not hotel:
    return "Hotel not found", 404

rooms = Room.query.filter_by(hotel_id=hotel.id).all()
rooms_data = [{
    'id': room.id,
    'name': room.name,
    'is_available': room.is_available,
    'price': room.price,
    'description': room.description,
    'capacity': room.capacity,
    'available_from': room.available_from,
    'available_to': room.available_to
} for room in rooms]

hotel_data = {
    'id': hotel.id,
    'name': hotel.name,
    'address': hotel.address,
    'contact': hotel.contact,
    'description': hotel.description,
    'photo_urls': hotel.photo_urls,
    'services': hotel.services,
    'owner_id': hotel.owner_id,
    'rooms': rooms_data
}
```

Рендеринг HTML-сторінки з даними про готель та номери:

```
    return render_template('hotel_details.html', hotel=hotel_data,
serviceIcons=serviceIcons, user_id=user_id)
```

Процес перегляду всіх готелів передбачає отримання списку всіх готелів з бази даних та формування JSON-відповіді з цими даними.

Отримання списку всіх готелів:

```
@main.route('/hotels')
def get_hotels():
    hotels = Hotel.query.all()
    hotels_data = [{
        'id': hotel.id,
        'name': hotel.name,
        'address': hotel.address,
        'description': hotel.description,
        'photo_urls': hotel.photo_urls,
        'services': hotel.services
    } for hotel in hotels]
    return jsonify(hotels_data)
```

JavaScript-код для обробки подій на сторінці, що дозволяє переходити до деталей готелю при натисканні на відповідну кнопку:

```
document.addEventListener('DOMContentLoaded', function() {
    const hotelsContainer = document.getElementById('my_hotels');
    if (!hotelsContainer) {
        console.error('Unable to find the container for hotels');
        return;
    }
    hotelsContainer.addEventListener('click', function(event) {
        if (!event.target.classList.contains('more')) return;
        const hotelId = event.target.getAttribute('data-hotel-id');
        const userID = event.target.getAttribute('data-user-hotel-id');
        if (!hotelId || !userID) {
            console.error('No hotel ID or user ID found');
            return;
        }
                                                    window.location.href      =
`/hotel_details?hotelId=${hotelId}&userId=${userID}`;
    });});
```

Процес перегляду створених готелів та номерів включає отримання даних про готелі та номери з бази даних, формування відповідних JSON-відповідей та рендеринг HTML-сторінок з детальною інформацією. Це забезпечує зручність перегляду та управління готелями та номерами для власників, а також полегшує користувачам доступ до інформації про доступні варіанти розміщення.

4.2.3 Видалення

Видалення готелів та номерів є важливою частиною адміністрування веб-додатку для бронювання готелів. Цей процес дозволяє власникам готелів видаляти непотрібні або застарілі дані про готелі та номери. Процес видалення готелю включає отримання ідентифікатора готелю, перевірку його існування, видалення всіх номерів, що належать цьому готелю, та видалення самого готелю з бази даних.

Отримання ідентифікатора готелю та перевірка його існування:

```
@hotel.route('/delete_hotel/<int:hotel_id>', methods=['POST'])
def delete_hotel(hotel_id):
    hotel = Hotel.query.get(hotel_id)
    if not hotel:
        return jsonify({'error': 'Hotel not found'}), 404
```

Видалення номерів та готелю з бази даних:

```
try:
    Room.query.filter_by(hotel_id=hotel_id).delete()
    db.session.delete(hotel)
    db.session.commit()
      return jsonify({'message': 'Hotel and all its rooms have been deleted
successfully!'})
except Exception as e:
    db.session.rollback()
```

```
        return jsonify({'error': str(e)}), 500
```

JavaScript-код для обробки подій на сторінці, що дозволяє видаляти готель при натисканні на відповідну кнопку:

```
document.addEventListener('DOMContentLoaded', function () {
    const hotelsContainer = document.getElementById('my_hotels');
    hotelsContainer.addEventListener('click', function(event) {
        const isDeleteButton = event.target.classList.contains('delete');
        const hotelId = event.target.getAttribute('data-hotel-id');
        const hotelName = event.target.getAttribute('data-hotel-name');
        const hotelCard = event.target.closest('.hotel-card');
        if (isDeleteButton) {
            Swal.fire({
                title: 'Ви впевнені?',
                   text: `Ви справді бажаєте видалити готель "${hotelName}" та
всі його номери?`,
                icon: 'warning',
                showCancelButton: true,
                confirmButtonColor: '#db0a06',
                cancelButtonColor: '#00c3ff',
                confirmButtonText: 'Так, видалити!',
                cancelButtonText: 'Скасувати'
            }).then((result) => {
                if (result.isConfirmed) {
                    fetch(`/delete_hotel/${hotelId}`, {
                        method: 'POST'
                    })
                    .then(response => response.json())
                    .then(data => {
                        Swal.fire(
                            'Видалено!',
                              'Готель та всі його номери було успішно видалено.
Перезавантажте сторінку для відображення змін.',
                            'success'
                        );
                        if (hotelCard) {
                            hotelCard.remove();
                        }
                    })
                    .catch(error => {
                        console.error('Error:', error);
                        Swal.fire(
                            'Помилка!',
                            'Під час видалення виникла помилка.',
                            'error'
                        );
                    });
                }
            });
        }
    });
});
```

Процес видалення номера включає отримання ідентифікатора номера, перевірку його існування та видалення цього номера з бази даних.

Отримання ідентифікатора номера та перевірка його існування:

```
@hotel.route('/delete_room/<int:room_id>', methods=['POST'])
```

```
def delete_room(room_id):
    room = Room.query.get(room_id)
    if not room:
        return jsonify({'error': 'Room not found'}), 404
```

Видалення номера з бази даних:

```
try:
    db.session.delete(room)
    db.session.commit()
    return jsonify({'message': 'Room has been deleted successfully!'})
except Exception as e:
    db.session.rollback()
    return jsonify({'error': str(e)}), 500
```

JavaScript-код для обробки подій на сторінці, що дозволяє видаляти номер при натисканні на відповідну кнопку.

```
document.addEventListener('DOMContentLoaded', function () {
    const hotelsContainer = document.getElementById('my_hotels');
    hotelsContainer.addEventListener('click', function(event) {
        if (event.target.classList.contains('delete_room')) {
            const roomId = event.target.getAttribute('data-room-id');
            const roomName = event.target.getAttribute('data-room-name');
            const roomElement = event.target.closest('.hotel-room');
            Swal.fire({
                title: 'Ви впевнені?',
                text: `Ви справді бажаєте видалити кімнату "${roomName}"?`,
                icon: 'warning',
                showCancelButton: true,
                confirmButtonColor: '#db0a06',
                cancelButtonColor: '#00c3ff',
                confirmButtonText: 'Так, видалити!',
                cancelButtonText: 'Скасувати'
            }).then((result) => {
                if (result.isConfirmed) {
                    fetch(`/delete_room/${roomId}`, {
                        method: 'POST'
                    })
                    .then(response => response.json())
                    .then(data => {
                        Swal.fire(
                            'Видалено!',
                            'Кімната була успішно видалена.',
                            'success'
                        );
                        if (roomElement) {
                            roomElement.remove();
                        }
                    })
                    .catch(error => {
                        console.error('Error:', error);
                        Swal.fire(
                            'Помилка!',
                            'Під час видалення виникла помилка.',
                            'error'
                        );
                    });
                }
            });
        }
```

```
        });
    });
```

Процес видалення готелів та номерів включає отримання відповідних ідентифікаторів, перевірку існування записів у базі даних та видалення цих записів. Правильна організація цього процесу забезпечує ефективне управління даними та підтримання актуальності інформації у системі. JavaScript-код для обробки подій на стороні клієнта дозволяє забезпечити інтерактивний досвід для користувачів, надаючи можливість підтверджувати або скасовувати видалення.

4.3. Бронювання

Процес бронювання номера в готелі є ключовим функціоналом веб-додатку для бронювання готелів. Він включає отримання даних від користувача, перевірку коректності даних, збереження бронювання в базі даних, перегляд бронювань користувача та можливість скасування бронювання.

Процес створення бронювання починається з отримання даних форми, перевірки наявності необхідних полів, перетворення дат у відповідний формат та збереження інформації про нове бронювання в базі даних.

Отримання даних та перевірка їх наявності:

```
@main.route('/book_room', methods=['POST'])
def book_room():
    data = request.get_json()
    print("Received data:", data)
        if not all(key in data for key in ['room_name', 'hotel_name', 'start_date', 'end_date', 'price', 'user_id']):
            return jsonify({'error': 'Missing required data!'}), 400
```

Перетворення дат у відповідний формат та перевірка їх коректності:

```
try:
    start_date = datetime.strptime(data['start_date'], '%Y-%m-%d').date()
    end_date = datetime.strptime(data['end_date'], '%Y-%m-%d').date()
except ValueError:
    return jsonify({'error': 'Invalid date format!'}), 400
if start_date >= end_date:
    return jsonify({'error': 'End date must be after start date!'}), 400
```

Збереження інформації про нове бронювання в базі даних:

```
booking = Booking(
```

```
        hotel_name=data['hotel_name'],
        room_name=data['room_name'],
        start_date=start_date,
        end_date=end_date,
        price=data['price'],
        user_id=data['user_id']
    )

    db.session.add(booking)
    db.session.commit()

    return jsonify({'message': 'Booking successfully created!', 'booking_id': booking.id}), 201
```

Процес перегляду бронювань користувача включає отримання ідентифікатора користувача, пошук усіх бронювань, пов'язаних із цим користувачем, та формування відповіді з даними про ці бронювання.

Отримання бронювань користувача:

```
@main.route('/user/<int:user_id>/bookings')
def user_bookings(user_id):
    bookings = Booking.query.filter_by(user_id=user_id).all()
    bookings_data = [{
        'id': booking.id,
        'hotel_name': booking.hotel_name,
        'room_name': booking.room_name,
        'start_date': booking.start_date.strftime('%Y-%m-%d'),
        'end_date': booking.end_date.strftime('%Y-%m-%d'),
        'price': booking.price
    } for booking in bookings]

    return jsonify(bookings_data)
```

Процес скасування бронювання включає отримання ідентифікатора бронювання, перевірку його існування та видалення цього бронювання з бази даних. Отримання ідентифікатора бронювання та перевірка його існування:

```
@main.route('/cancel_booking/<int:booking_id>', methods=['POST'])
def cancel_booking(booking_id):
    booking = Booking.query.get(booking_id)
    if not booking:
        return jsonify({'error': 'Booking not found!'}), 404
```

Видалення бронювання з бази даних:

```
db.session.delete(booking)
db.session.commit()
return jsonify({'message': 'Booking cancelled successfully!'}), 200
```

JavaScript-код для обробки подій на сторінці, що дозволяє скасувати бронювання при натисканні на відповідну кнопку:

```
function cancelBooking(bookingId) {
```

```
        Swal.fire({
            title: 'Ви впевнені?',
            text: "Це дію не можна буде відмінити!",
            icon: 'warning',
            showCancelButton: true,
            confirmButtonColor: '#dc3545',
            cancelButtonColor: '#6c757d',
            confirmButtonText: 'Так, відмінити!',
            cancelButtonText: 'Скасувати'
        }).then((result) => {
            if (result.isConfirmed) {
                fetch(`/cancel_booking/${bookingId}`, { method: 'POST' })
                    .then(response => {
                              if (!response.ok) throw new Error('Failed to cancel
booking');
                        return response.json();
                    })
                    .then(data => {
                        Swal.fire(
                            'Відмінено!',
                             'Ваше бронювання було відмінено. Оновіть сторінку для
відображення змін.',
                            'success'
                        );
                    })
                    .catch(error => {
                        Swal.fire(
                            'Помилка!',
                            'Не вдалося відмінити бронювання.',
                            'error'
                        );
                        console.error('Error:', error);
                    });
            }
        });
    }
```

Процес бронювання номера включає отримання даних від користувача, перевірку їх коректності, збереження бронювання в базі даних, перегляд бронювань користувача та можливість скасування бронювання. Правильна організація цього процесу забезпечує зручність для користувачів та ефективне управління бронюваннями. JavaScript-код для обробки подій на стороні клієнта дозволяє забезпечити інтерактивний досвід для користувачів, надаючи можливість підтверджувати або скасовувати бронювання.

4.4. Тестування та демонстрація результатів

Тестування та демонстрація результатів є важливими етапами розробки веб-додатку, що дозволяють переконатися в коректності роботи системи та

показати кінцевий продукт користувачам. Демонстрація результатів включає показ кінцевого продукту користувачам, акцентуючи увагу на функціональності та зручності використання головної сторінки. Користувачі можуть побачити, як легко можна знайти потрібний готель, використовуючи панель пошуку та фільтри, переглянути деталі готелю та здійснити бронювання. Першим, що побачить користувач, буде головна сторінка додатку (рисунок 4.1). Головна сторінка веб-додатку RESERVIA є центральним місцем для користувачів, де вони можуть почати пошук готелів, переглядати деталі та фільтрувати результати за своїми уподобаннями.

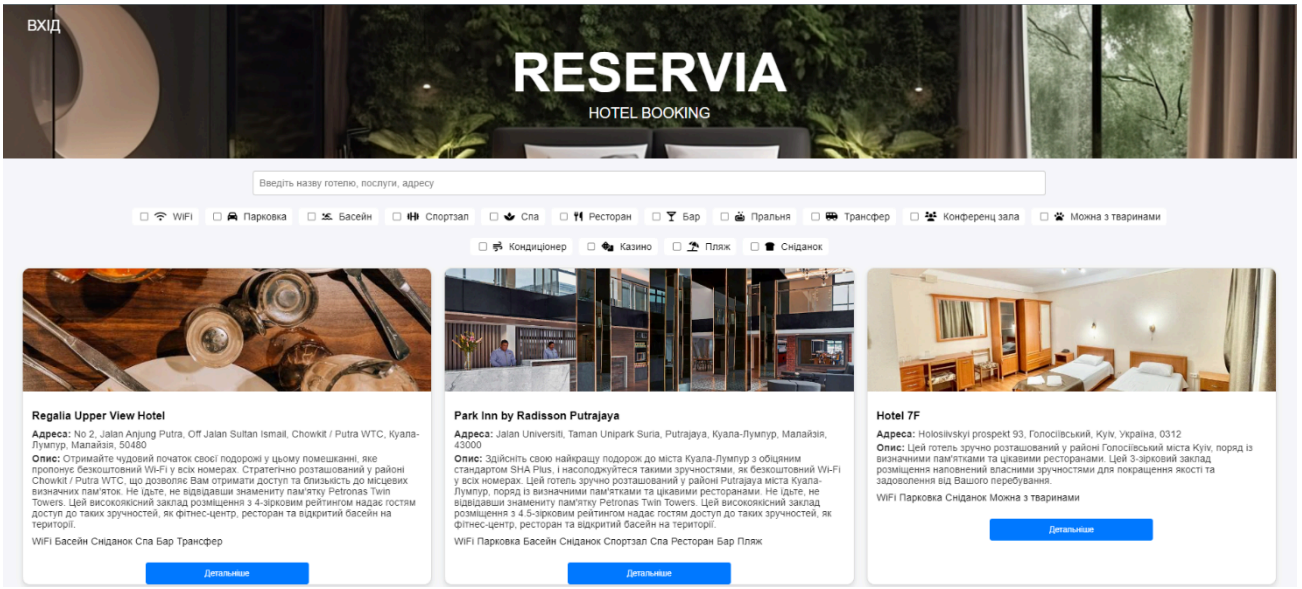


Рис. 4.1 Додаток (Головна Сторінка)

Головна сторінка складається з декількох ключових елементів, які забезпечують зручність і функціональність. У верхній частині сторінки розташований логотип RESERVIA з підзаголовком «HOTEL BOOKING». Цей елемент виконує роль ідентифікації бренду та забезпечує простий доступ до головної сторінки. Праворуч розміщена кнопка «ВХІД», яка дозволяє користувачам увійти до свого облікового запису або зареєструватися. Центральний елемент сторінки, панель пошуку, де користувачі можуть вводити назву готелю, послуги чи адресу для швидкого пошуку.

Поруч з панеллю пошуку розміщені фільтри (рисунок 4.2), які дозволяють користувачам звужувати результати за певними критеріями, такими як WiFi, парковка, басейн, спортзал, спа, ресторан, бар, пральня, трансфер, конференц-зала, можливість з тваринами, кондиціонер, казино, пляж та сніданок.

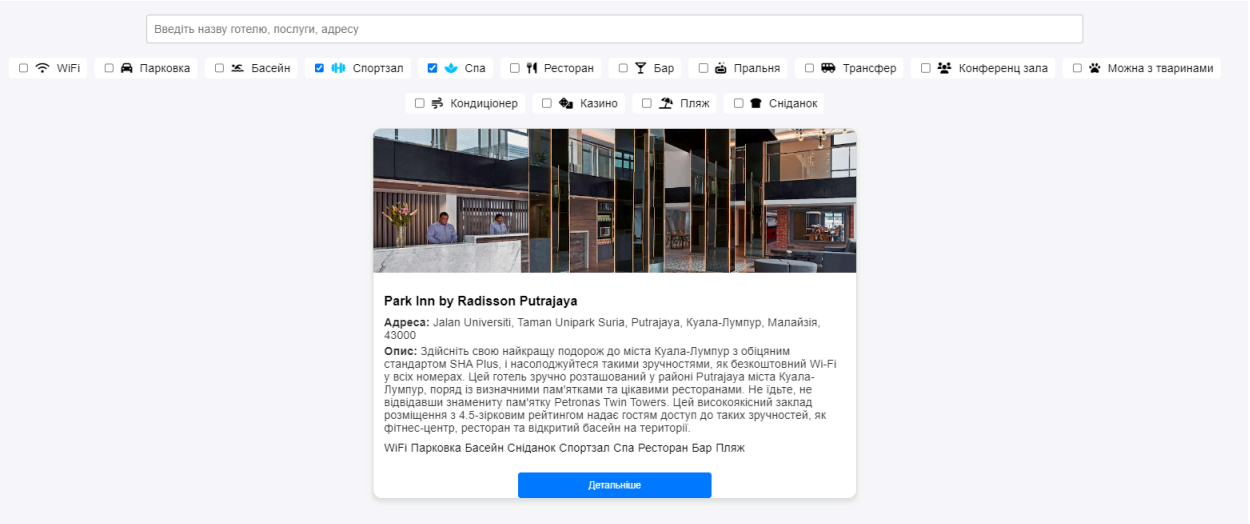


Рисунок 4.2 Демонстрація роботи (фільтра)

Нижче панелі пошуку розташований список готелів, які відповідають заданим критеріям. Кожен готель представлений у вигляді картки, яка містить фотографію готелю, назву, адресу, короткий опис та перелік послуг. Кнопка «Детальніше» на кожній картці дозволяє перейти до сторінки з детальною інформацією про обраний готель.

Коли користувач натискає на кнопку «Детальніше» на головній сторінці, він переходить на сторінку з детальною інформацією про обраний готель (рисунок 4.3). Ця сторінка дозволяє переглянути деталі готелю та його номерів, а також забронювати номер.

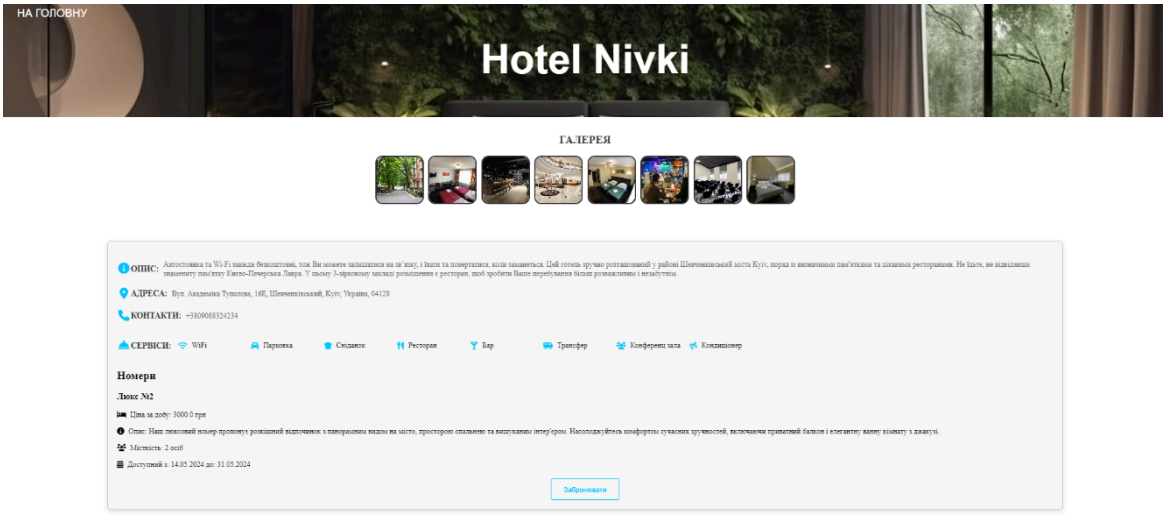


Рис. 4.3 Сторінка з детальною інформацією (про готелі)

Сторінка детальної інформації про готель містить повну інформацію про готель, його номери та доступні сервіси. Це забезпечує користувачів всім необхідним для прийняття рішення про бронювання. Загальна структура сторінки:
Заголовок: Назва готелю;
Галерея: Набір фотографій готелю;
Інформація про готель: Опис, адреса, контакти та перелік доступних сервісів;
Список номерів: Детальна інформація про доступні номери, включаючи ціну, опис, місткість та доступні дати;
Кнопка бронювання: Кнопка для здійснення бронювання номера.

Якщо користувач не увійшов у систему, при натисканні кнопки «Забронювати» з’являється повідомлення про необхідність входу (рис. 4.4). Це забезпечує безпеку та захист даних користувачів.

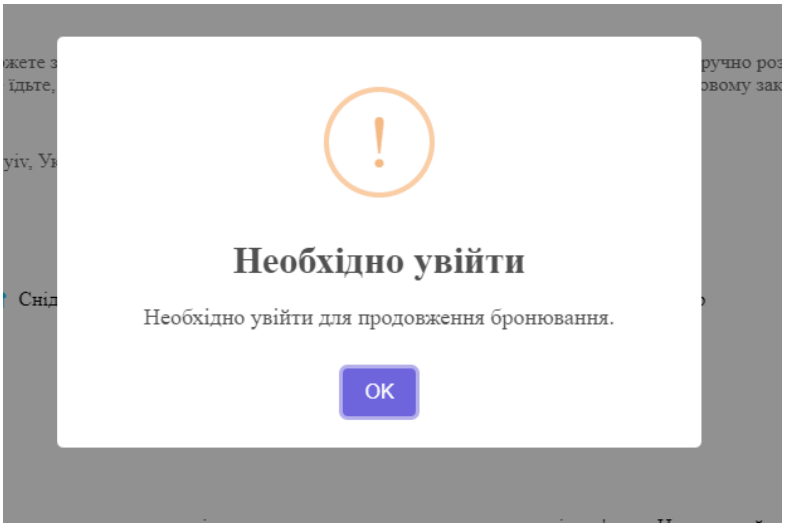


Рис. 4.4 Повідомлення про необхідність входу в систему

Коли користувач натискає на кнопку «ВХІД» на головній сторінці, з’являється модальне вікно з формами для входу та реєстрації. Це дозволяє користувачам легко отримати доступ до своїх акаунтів або створити новий акаунт, якщо вони ще не зареєстровані. Форма входу включає два основні поля: логін і пароль (рисунок 4.5).

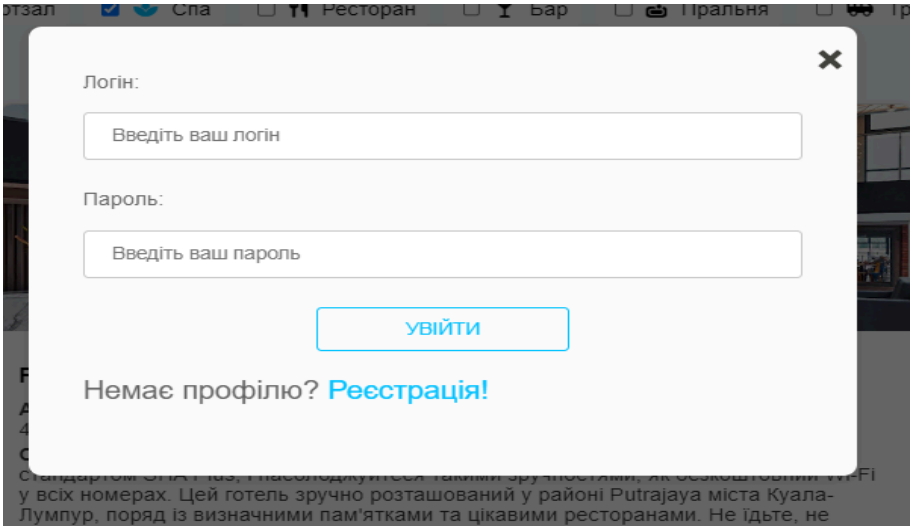


Рис. 4.5 Форма (входу)

Користувач вводить свої дані та натискає кнопку «УВІЙТИ». У разі помилки (наприклад, неправильний логін або пароль) користувач отримає відповідне повідомлення.Форма реєстрації включає три основні поля: електронна пошта, логін та пароль (рисунок 4.6).

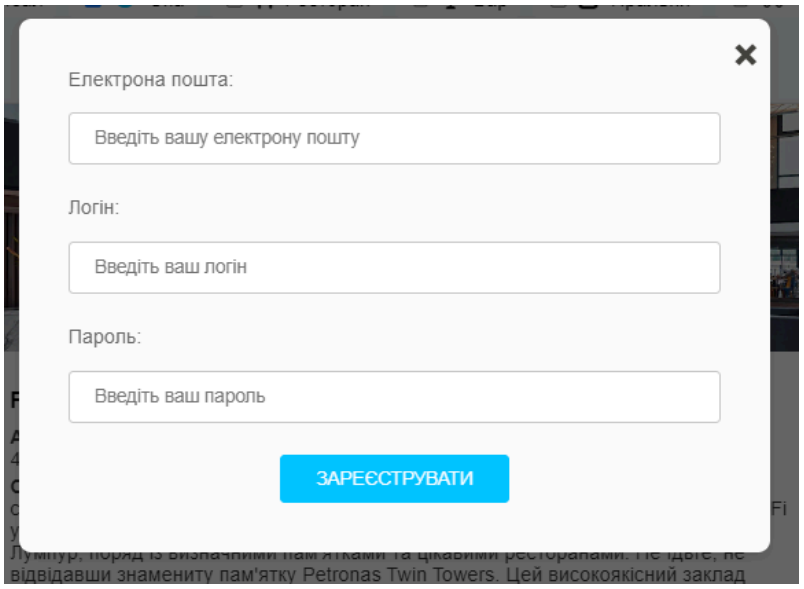


Рис. 4.6 Форма (реєстрації)

Користувач вводить необхідні дані та натискає кнопку «ЗАРЕЄСТРУВАТИ». У разі помилки (наприклад, логін або електронна пошта вже існують) користувач отримає відповідне повідомлення.

Враховуючи те що всі елементи головні сторінки працюють належним чином, та реагують на дії користувача тестування цієї сторінки системи можна вважати успішно пройденим.

Після успішного входу користувача в систему, він потрапляє на свою домашню сторінку (рисунок 4.7), яка надає доступ до основних функцій додатку, таких як перегляд та управління бронюваннями, додавання нових готелів, а також перегляд власних готелів.

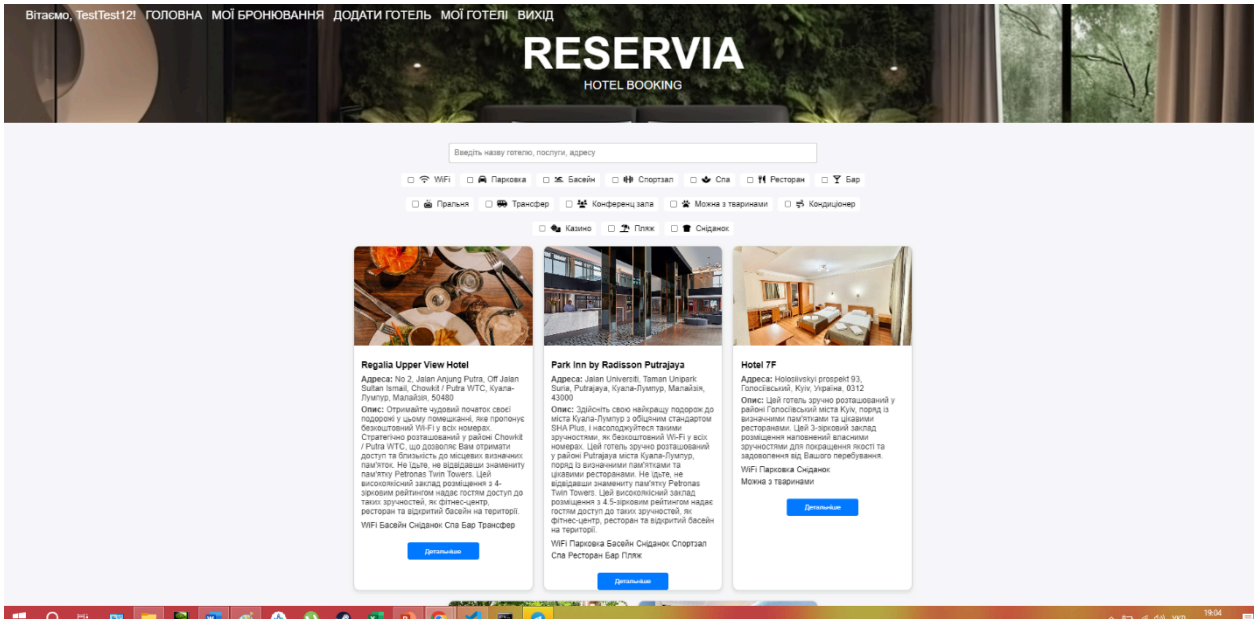


Рис. 4.7 Домашня сторінка (користувача)

Коли користувач переходить на сторінку «ДОДАТИ ГОТЕЛЬ», він бачить форму для заповнення інформації про новий готель (рисунок 4.8).

Ця форма включає поля для введення назви готелю, опису, адреси, контактної інформації, завантаження фотографій та вибору доступних сервісів. Після заповнення всіх полів користувач натискає кнопку «ДОДАТИ ГОТЕЛЬ».

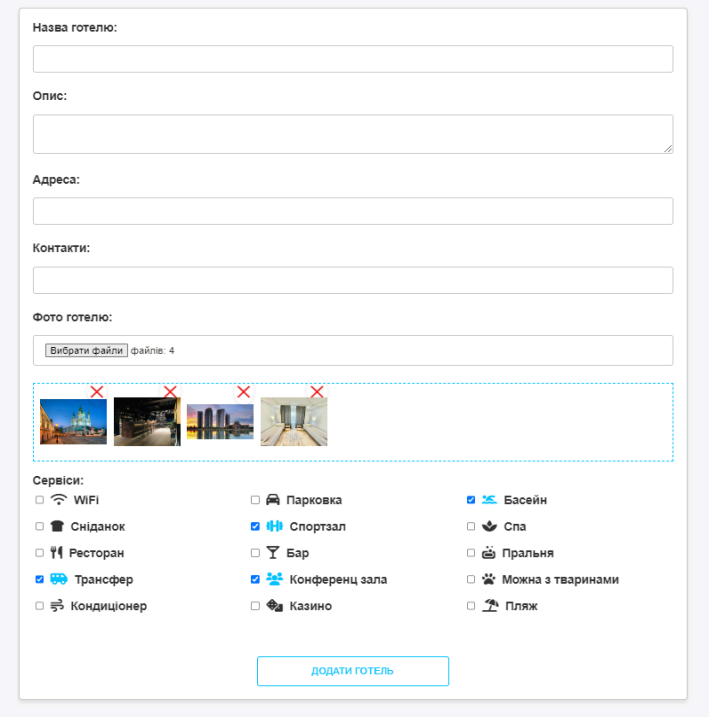


Рис. 4.8 Форма додавання (готелю)

Після успішного створення готелю, новий готель відображається на сторінці «МОЇ ГОТЕЛІ». Тут користувач бачить картку готелю з основною інформацією, фотографіями, переліком сервісів та можливістю додавання кімнат, перегляду деталей та видалення готелю (рисунок 4.9).

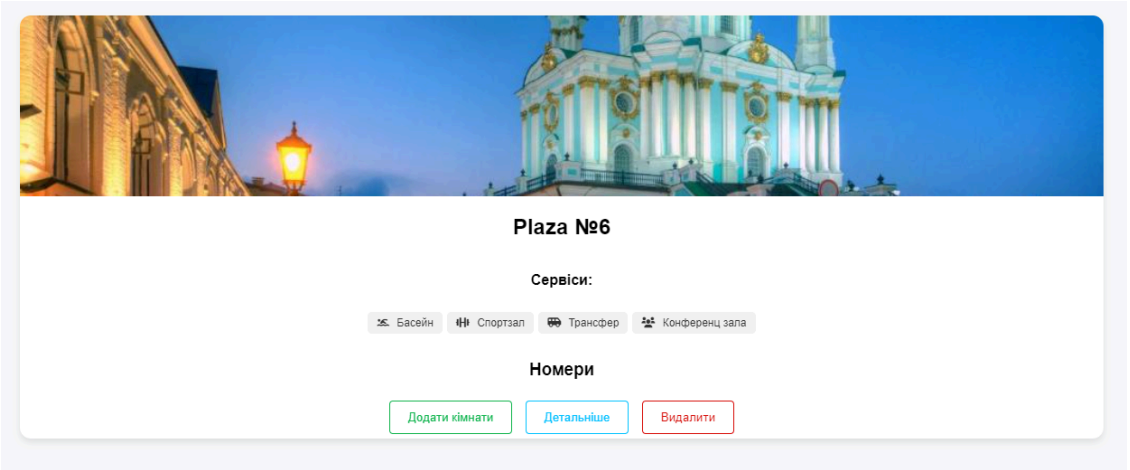


Рис. 4.9 Карта створеного (готелю)

Коли користувач натискає кнопку «Видалити» на картці готелю, він отримує сповіщення з підтвердженням дії (рисунок 4.10) Це сповіщення містить питання, чи впевнений користувач у видаленні готелю разом з усіма його кімнатами.

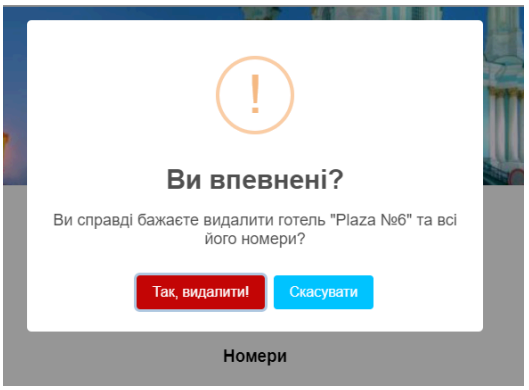


Рис. 4.10 Повідомлення про підтвердження видалення (готелю)

Сповіщення має дві кнопки: червону «Так, видалити!» і синю «Скасувати». Якщо користувач підтвердить дію, готель разом з усіма кімнатами буде видалено з системи. Якщо користувач натисне «Скасувати», дія буде відмінена. При

натисканні кнопки «Детальніше» на картці готелю, користувач перейде на сторінку з детальною інформацією про готель.

При натисканні кнопки «Додати кімнату» на картці готелю відкривається форма для додавання нової кімнати (рисунок 4.11).

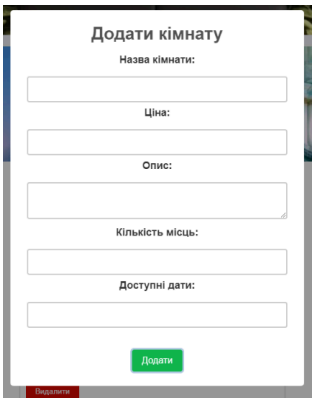


Рис. 4.11 Повідомлення про підтвердження видалення (готелю)

Користувач заповнює форму, вводячи назву кімнати, ціну, опис, кількість місць та доступні дати. Після заповнення форми, нова кімната з’являється на картці готелю (рисунок 4.12).

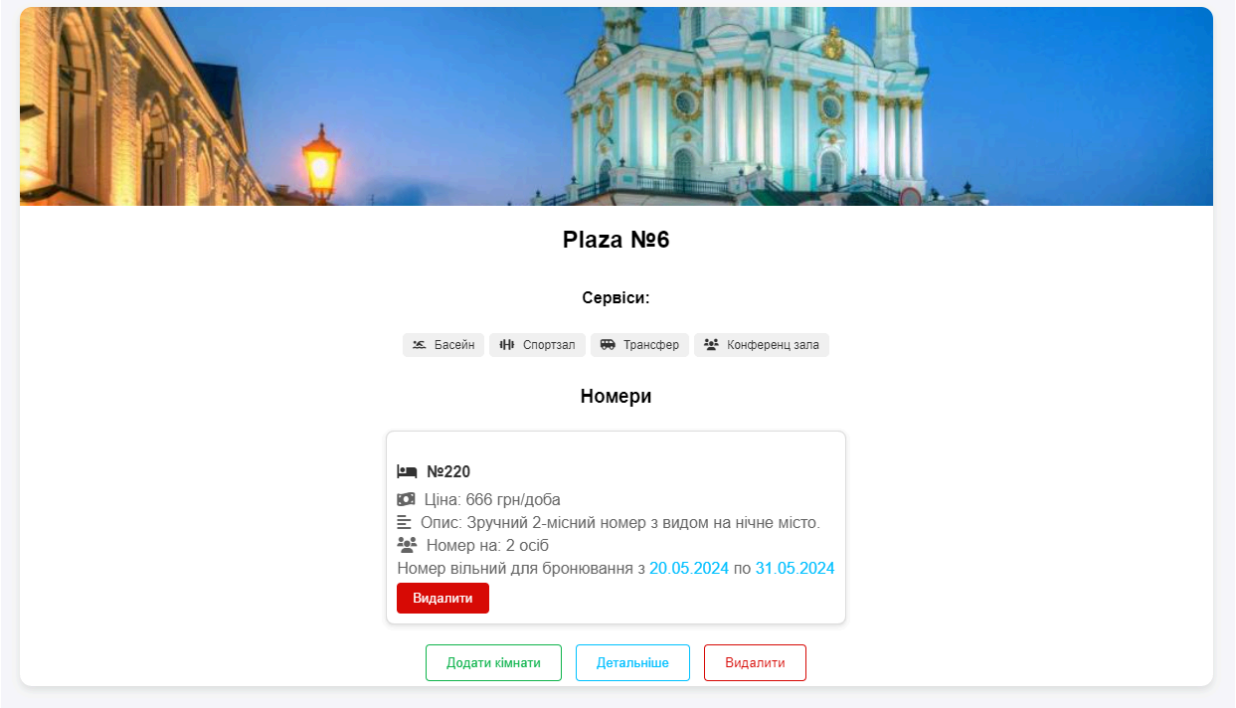


Рис. 4.12 Появлення номеру на картці (готелю)

Якщо користувач на сторінці детальної інформації натисне на кнопку «Забронювати» для конкретного номеру готелю, відкриється форма бронювання (рисунок 4.13).

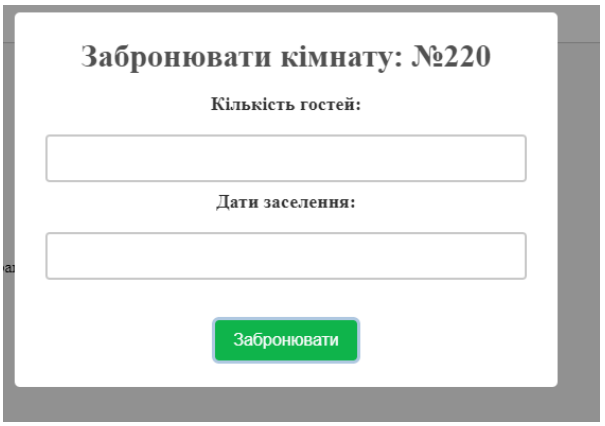


Рис. 4.13 Форма бронювання (номеру)

Ця форма вимагає ввести кількість гостей та дати заселення. Після заповнення форми та вибору дат перебування, додаток автоматично розрахує загальну ціну за всі дні перебування в готелі (рисунок 4.14). Якщо користувач підтвердить бронювання, він отримає повідомлення про успішне бронювання.

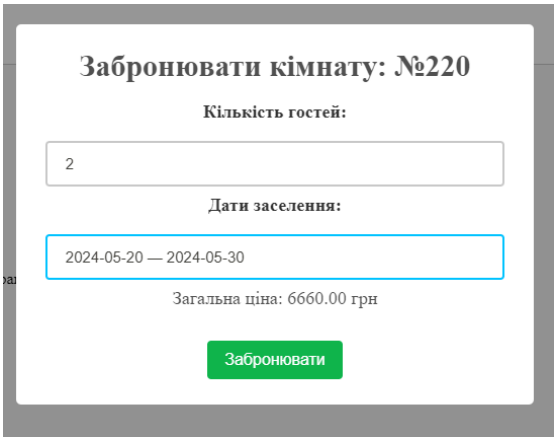


Рис. 4.14 Заповнена форма бронювання (номеру)

Успішне бронювання відобразиться у розділі «Мої бронювання» (рисунок 4.15). Користувач зможе побачити деталі свого бронювання, такі як дати перебування, номер готелю, ціну та кнопку для скасування бронювання, якщо це необхідно.

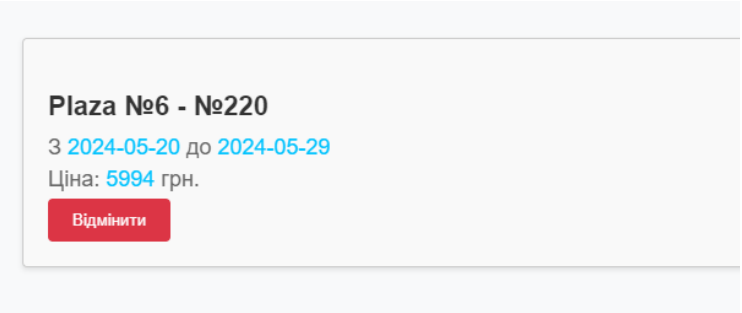


Рис. 4.14 Заповнена форма бронювання (номеру)

Це забезпечує зручність і простоту управління бронюваннями для користувачів додатку. Так як всі елементи додатку для авторизованого користувача працюють правильно, то тестування можна вважати успішно пройденим.

ВИСНОВКИ

У даній кваліфікаційній роботі проведено аналіз предметної галузі та реалізовано веб-додаток для бронювання готельних номерів.

На початку роботи розглянуто основні аспекти та актуальність онлайн-бронювання готельних номерів, що дозволяє користувачам швидко і зручно знаходити та бронювати номери. Проведено порівняльний аналіз існуючих веб-додатків для бронювання, таких як Booking.com, Airbnb, Expedia та Reservia. Визначено їх переваги та недоліки, що допомогло окреслити основні вимоги до власного додатку. Визначено ключові задачі, необхідні для успішної реалізації проекту, включаючи проектування архітектури додатку, розробку бази даних та реалізацію основної функціональності.

Досліджено та описано принципи роботи фреймворку Flask, що є основою для розробки серверної частини додатку. Описано переваги використання Visual Studio Code як основного середовища розробки, включаючи його легкість, розширюваність та інтеграцію з Git. Розглянуто використання HTML, JavaScript та CSS для створення інтерактивного та зручного інтерфейсу користувача. Описано реалізацію серверної логіки за допомогою мови програмування Python, що забезпечує обробку запитів, управління базою даних та автентифікацію користувачів.

Розроблено архітектуру веб-додатку та основні сторінки, що забезпечують логічну структуру та зручність використання додатку. Описано структуру бази даних, яка включає таблиці для зберігання інформації про користувачів, готелі, номери та бронювання. Представлено структуру проекту, що включає основні директорії та файли, необхідні для функціонування додатку.

Реалізовано функціонал реєстрації та авторизації користувачів, що забезпечує безпечний доступ до додатку. Створено інтерфейс та функціонал для додавання, редагування та видалення готелів і номерів. Реалізовано процес бронювання номерів користувачами, включаючи обробку даних та підтвердження

бронювання. Проведено тестування додатку та демонстрацію основних функціональних можливостей.

В результаті виконання кваліфікаційної роботи отримано веб-додаток, який дозволяє користувачам зручно бронювати готельні номери, а власникам готелів – ефективно управляти своїми об’єктами та номерами. Додаток забезпечує високу продуктивність, безпеку даних та зручність використання, що робить його конкурентоспроможним на ринку.

Апробація результатів дослідження:

Джига Д.Ю., Замрій І.В. Інтеграція property management systems (PMS) у веб-додаток для бронювання готельних номерів. *Всеукраїнська науково-технічна конференції «Застосування програмного забезпечення вінформаційно-комунікаційних технологіях».* 24 квітня 2024 р., Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. К.: ДУІКТ, 2024, С. 421 - 423.

ПЕРЕЛІК ПОСИЛАНЬ

1. Секрети і тонкощі: як працювати онлайн готеями [Електронний ресурс] // Ribas.u. – 2024. – Режим доступу до ресурсу: https://ribas.ua/blog/sekreti_i_tonkosti:_kak_rabotaty_s_bookingcom.

2. Booking [Електронний ресурс] // Booking. – 2024. – Режим доступу до ресурсу: booking.com.

3. Airbnb [Електронний ресурс] // Airbnb. – 2024. – Режим доступу до ресурсу: https://www.airbnb.com.ua/.

4. Expedia [Електронний ресурс] // Expedia. – 2024. – Режим доступу до ресурсу: https://www.expedia.com/.

5. Вступ до Python фреймворків: які вони бувають і для чого використовуються [Електронний ресурс] // Foxminded. – 2023. – Режим доступу до ресурсу: https://foxminded.ua/freimvorky-python/.

6. Flask [Електронний ресурс] // Flask. – 2024. – Режим доступу до ресурсу: https://flask.palletsprojects.com/en/3.0.x/.

7. 10 кращих редакторів коду для програмістів [Електронний ресурс] // Mate.AcademY. – 2023. – Режим доступу до ресурсу: https://mate.academy/blog/front-end-and-js/top-10-text-editors/.

8. HTML, CSS I JAVASCRIPT: ОСНОВИ ВЕБ-РОЗРОБКИ [Електронний ресурс] // Pravda. – 2022. – Режим доступу до ресурсу: https://pravda.if.ua/html-css-i-javascript-osnovy-veb-rozrobky/.

9. How To Make a Web Application Using Flask in Python 3 [Електронний ресурс] // Digitalocean. – 2022. – Режим доступу до ресурсу: https://www.digitalocean.com/community/tutorials/how-to-make-a-web-application-using-flask-in-python-3.

Додаток А ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ (Презентація)

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Розробка та реалізація веб-додатку для бронювання готельних номерів

Виконав (ла) студент(ка) 4 (5) курсу
групи …
Прізвище Ім'я По-батькові
Керівник роботи
К.т.н, доц, доцент кафедри ІПЗ Прізвище Ім'я По-батькові

Київ – 2024

МЕТА, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

- **Мета роботи**: Визначити ефективні способи підвищення зручності та автоматизації процесу бронювання готельних номерів за допомогою розробленого веб-додатку.
- **Об'єкт дослідження**: Процес бронювання готельних номерів у готельному бізнесі.
- **Предмет дослідження**: Методи та засоби автоматизації бронювання готельних номерів за допомогою веб-додатку, включаючи інтерфейси користувача, алгоритми пошуку та фільтрації, а також інтеграцію з базами даних.

2

ЗАДАЧІ ДИПЛОМНОЇ РОБОТИ

1. **Аналіз існуючих рішень**: Огляд веб-додатків для бронювання, визначення переваг і недоліків, виділення ключових можливостей.
2. **Проектування архітектури**: Розробка архітектури додатку, визначення компонентів системи, та БД.
3. **Реалізація функціональності**: Пошук і фільтрація номерів, перегляд деталей і доступності, процес бронювання та адміністрування в додатку.

3

АНАЛІЗ АНАЛОГІВ

Параметр	Booking.com	Airbnb	Expedia	Reservia
Переваги	Велика база даних готелів, зручний інтерфейс	Унікальні пропозиції житла, інтеграція з соцмережами	Великий вибір туристичних послуг (готелі, авіаквитки, прокат авто)	Простота використання, швидкість роботи
Недоліки	Високі комісії для готелів	Відсутність професійної підтримки, обмежена кількість готелів	Складний інтерфейс, багато реклами	Обмежена база даних
Пошук і фільтрація номерів	Так	Так	Так	**Так**
Перегляд деталей і доступності	Так	Так	Так	**Так**
Процес бронювання	Так	Так	Так	**Так**
Адміністрування бронювань	Так	Ні	Так	**Так**
Адаптивний дизайн	Так	Так	Так	**Так**
Інтеграція з платіжними системами	Так	Так	Так	**Ні**
Безпека даних	Високий рівень	Високий рівень	Високий рівень	**Високий рівень**

4

ВИМОГИ ДО ДОДАТКУ

1. **Функціональні вимоги:**
   - **Пошук і фільтрація готелів;**
   - **Перегляд деталей і доступності номерів;**
   - **Процес бронювання номерів;**
   - **Адміністрування бронювань.**

2. **Ключові нефункціональні вимоги:**
   - **Продуктивність;**
   - **Безпека даних;**
   - **Адаптивний дизайн;**
   - **Надійність;**
   - **Зручність використання.**

5

ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ

6

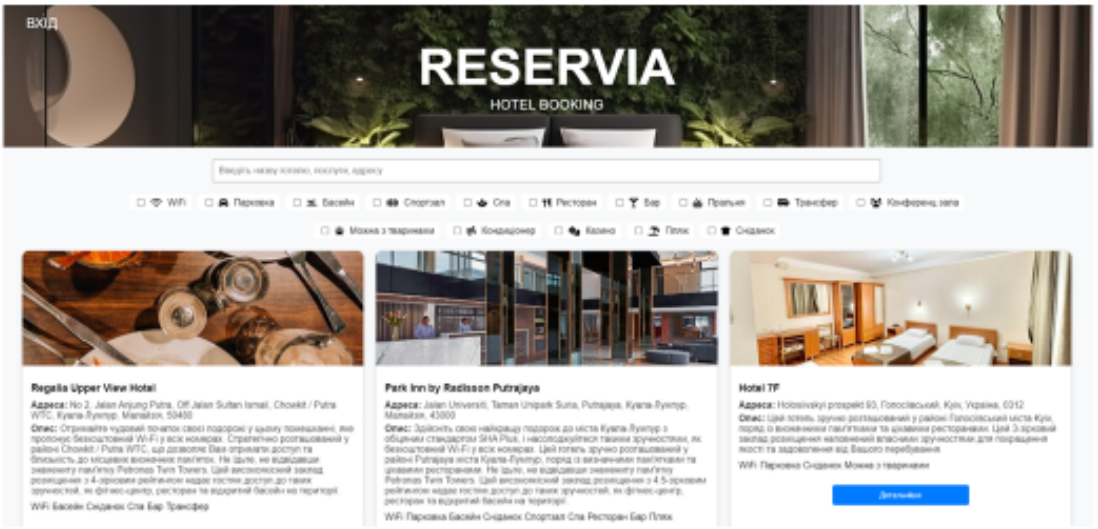
СТРУКТУРА БАЗИ ДАНИХ

7

КАРТА СТОРІНОК ВЕБ-ДОДАТКУ

8

ДЕМОНСТРАЦІЯ ВЕБ-ДОДАТКУ



«Головна сторінка»

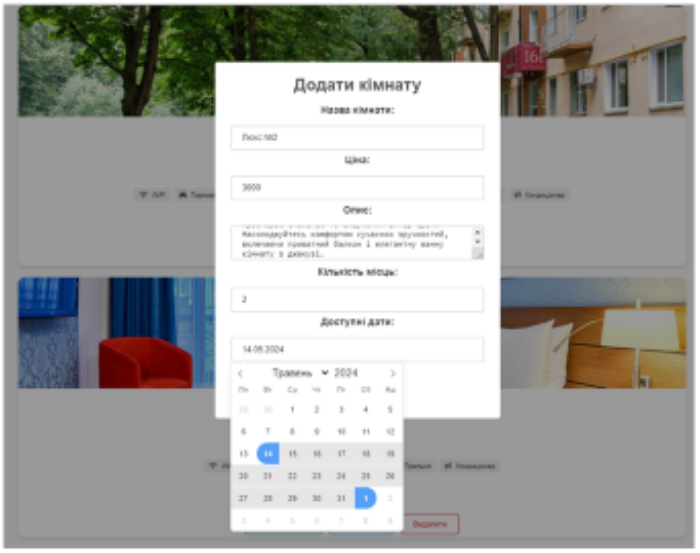
9

ДЕМОНСТРАЦІЯ ВЕБ-ДОДАТКУ

«Форма добавлення готелю»

10

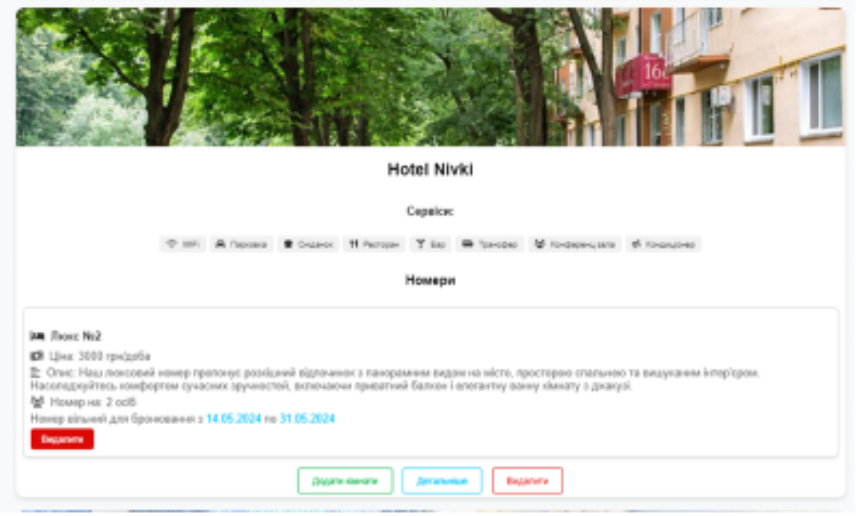
ДЕМОНСТРАЦІЯ ВЕБ-ДОДАТКУ



«Форма добавлення номеру»

11

ДЕМОНСТРАЦІЯ ВЕБ-ДОДАТКУ



«Елемент готелю та доданого номеру»

12

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

1. Джига Д.Ю., Замрій І.В. інтеграція property management systems (PMS) у веб додаток для бронювання готельних номерів Матеріали Всеукраїнської науково-технічної конференції «Автоматизація та управління бізнес процесами». 24 квітня 2024 р., Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. К.: ДУІКТ, 2024, С. 421 - 423.

13

ВИСНОВКИ

У результаті виконання роботи було розроблено веб-додаток для бронювання готельних номерів, який враховує переваги та недоліки існуючих рішень. Додаток має інтуїтивно зрозумілий інтерфейс, що забезпечує зручне використання для користувачів. Архітектура системи дозволяє ефективно обробляти запити та керувати бронюваннями, а також пропонує всі необхідні функціональні можливості для пошуку, фільтрації та перегляду доступності номерів.

14

Додаток Б. ЛІСТИНГ ДОДАТКУ

Клієнтська частина (сторінки):

404.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
        <meta name="viewport"
content="width=device-width,
initial-scale=1.0">
        <title>404 - Page Not
Found</title>
    <link rel="stylesheet" href="{{
url_for('static',
filename='css/style.css') }}">
    <style>
        .not-found-container {
            display: flex;
            justify-content: center;
            align-items: center;
            height: 100vh;
            flex-direction: column;
            text-align: center;
        }
        .not-found-container h1 {
            font-size: 72px;
            margin-bottom: 20px;
        }
        .not-found-container p {
            font-size: 24px;
            margin-bottom: 40px;
        }
        .not-found-container button
{
            padding: 10px 20px;
            font-size: 18px;
            color: #fff;
                background-color:
#007bff;
            border: none;
            border-radius: 4px;
            cursor: pointer;
                    transition:
background-color 0.2s;
        }
            .not-found-container
button:hover {
                background-color:
#0056b3;
        }
    </style>
</head>
<body>
                                <div
class="not-found-container">
        <h1>404</h1>
        <p>Користувача не знайдено,
будь-ласка увійдіть ще раз!</p>
                                <button
onclick="goHome()">На головну</button>
    </div>
    <script>
        function goHome() {
            window.location.href =
"/";
        }
    </script>
</body>
</html>
```

home.html:

```
<!DOCTYPE html>
<html>
<head>
    <title>RESERVIA</title>
            <link rel="icon"
href="static/uploads/system_icon/favicon.p
ng" type="image/x-icon">
    <link rel="stylesheet" href="{{
url_for('static', filename='home.css')
}}">
    <link rel="stylesheet" href="{{
url_for('static',
filename='hotel_card.css') }}">
    <link rel="stylesheet" href="{{
url_for('static',
filename='hotel_card_main.css') }}">
        <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/li
bs/font-awesome/6.0.0-beta3/css/all.min.cs
s">
        <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/flatpic
kr/dist/flatpickr.min.css">
</head>
<body>


        <div class="brand-logo"ë
style="background:
url('static/uploads/system_icon/background
.png') no-repeat center center / cover;
height: 250px; width: auto;
backdrop-filter: blur(5px);">
                                <span
class="brand-reservia">RESERVIA</span>
                                <span
class="brand-tagline">HOTEL BOOKING</span>
    </div>

    <div class="header-links">
                            <a href="#"
class="header-link">Вітаємо, {{
username}}!</a>
                            <a href="#"
class="header-link"
onclick="showContent('my_page')">ГОЛОВНА</
a>
                            <a href="#"
class="header-link"
onclick="showContent('my_booking')">МОЇ
БРОНЮВАННЯ</a>
                            <a href="#"
class="header-link"
onclick="showContent('create_hotel')">ДОДА
ТИ ГОТЕЛЬ</a>
                            <a href="#"
class="header-link"
onclick="showContent('my_hotels')">МОЇ
ГОТЕЛІ</a>
```

```
                            <a   href="{{
url_for('main.logout')                }}"
class="header-link">ВИХІД</a>
          </div>
                <div  id="dynamic-content"
class="hidden">

                        <div  id="my_page"
class="content">
                                    <div
class="search-section">
                         <input type="text"
id="search_input"     placeholder="Введіть
назву готелю, послуги, адресу">
                    </div>
                                    <div
class="filters-section-main">
                                    <div
class="filters">
                            <label><input
type="checkbox" value="WiFi"> <i class="fa
fa-wifi"            aria-hidden="true"></i>
WiFi</label>
                            <label><input
type="checkbox"    value="Парковка">    <i
class="fa  fa-car"  aria-hidden="true"></i>
Парковка</label>
                            <label><input
type="checkbox"     value="Басейн">     <i
class="fa                    fa-swimmer"
aria-hidden="true"></i> Басейн</label>
                            <label><input
type="checkbox"    value="Спортзал">    <i
class="fa                     fa-dumbbell"
aria-hidden="true"></i> Спортзал</label>
                            <label><input
type="checkbox"  value="Спа"> <i class="fa
fa-spa"             aria-hidden="true"></i>
Спа</label>
                            <label><input
type="checkbox"    value="Ресторан">    <i
class="fa                     fa-utensils"
aria-hidden="true"></i> Ресторан</label>
                            <label><input
type="checkbox" value="Бар"> <i class="fa
fa-glass-martini-alt"
aria-hidden="true"></i> Бар</label>
                            <label><input
type="checkbox"     value="Пральня">     <i
class="fa fa-soap" aria-hidden="true"></i>
Пральня</label>
                            <label><input
type="checkbox"    value="Трансфер">    <i
class="fa                   fa-shuttle-van"
aria-hidden="true"></i> Трансфер</label>
                            <label><input
type="checkbox" value="Конференц зала"> <i
class="fa                          fa-users"
aria-hidden="true"></i>          Конференц
зала</label>
                            <label><input
type="checkbox" value="Можна з тваринами">
<i            class="fa            fa-paw"
aria-hidden="true"></i>      Можна      з
тваринами</label>
                            <label><input
type="checkbox"   value="Кондиціонер">   <i
class="fa fa-wind" aria-hidden="true"></i>
Кондиціонер</label>
                            <label><input
type="checkbox"     value="Казино">     <i
class="fa fa-dice" aria-hidden="true"></i>
Казино</label>
                            <label><input
type="checkbox" value="Пляж"> <i class="fa
fa-umbrella-beach" aria-hidden="true"></i>
Пляж</label>
                            <label><input
type="checkbox"    value="Сніданок">    <i
class="fa                fa-bread-slice"
aria-hidden="true"></i> Сніданок</label>
                        </div>
                    </div>
                                    <div
id="hotel-list-main"
class="hotel-list-main"></div>
                </div>
                        <div  id="my_booking"
class="content">
                            <div  id="userID"
data-user-id="{{ user.id }}"></div>
                </div>
                    <div  id="create_hotel"
class="content">
                      <form id="hotelForm"
action="/create_hotel"       method="post"
enctype="multipart/form-data">
                        <input type="hidden"
id="user_id"    name="user_id"    value="{{
user_id }}">
                                    <label
for="name">Назва готелю:</label>
                          <input type="text"
id="name" name="name" required><br><br>

                                    <label
for="description">Опис:</label>
                                  <textarea
id="description"        name="description"
required></textarea><br><br>

                                    <label
for="address">Адреса:</label>
                          <input type="text"
id="address"                name="address"
required><br><br>

                                    <label
for="contact">Контакти:</label>
                          <input type="text"
id="contact"                name="contact"
required><br><br>

                                    <label
for="photo">Фото готелю:</label>
                          <input type="file"
id="photo"      name="photo"      multiple
accept="image/*"><br><br>
                        <div id="preview">
                                    <div
class="image-container">
                            </div>
                        </div>


<label>Сервіси:</label>
                                    <div
class="checkboxes">
```

```
                            <label><input
type="checkbox"             name="services"
value="WiFi"><i     class="fa     fa-wifi"
aria-hidden="true"></i> WiFi</label>
                            <label><input
type="checkbox"             name="services"
value="Парковка"><i    class="fa    fa-car"
aria-hidden="true"></i> Парковка</label>
                            <label><input
type="checkbox"             name="services"
value="Басейн"><i   class="fa   fa-swimmer"
aria-hidden="true"></i> Басейн</label>
                            <label><input
type="checkbox"             name="services"
value="Сніданок"><i               class="fa
fa-bread-slice"     aria-hidden="true"></i>
Сніданок</label>
                            <label><input
type="checkbox"             name="services"
value="Спортзал"><i class="fa fa-dumbbell"
aria-hidden="true"></i> Спортзал</label>
                            <label><input
type="checkbox"             name="services"
value="Спа"><i      class="fa      fa-spa"
aria-hidden="true"></i> Спа</label>
                            <label><input
type="checkbox"             name="services"
value="Ресторан"><i class="fa fa-utensils"
aria-hidden="true"></i> Ресторан</label>
                            <label><input
type="checkbox"             name="services"
value="Бар"><i                    class="fa
fa-glass-martini-alt"
aria-hidden="true"></i> Бар</label>
                            <label><input
type="checkbox"             name="services"
value="Пральня"><i    class="fa    fa-soap"
aria-hidden="true"></i> Пральня</label>
                            <label><input
type="checkbox"             name="services"
value="Трансфер"><i               class="fa
fa-shuttle-van"     aria-hidden="true"></i>
Трансфер</label>
                            <label><input
type="checkbox"             name="services"
value="Конференц     зала"><i     class="fa
fa-users"           aria-hidden="true"></i>
Конференц зала</label>
                            <label><input
type="checkbox"             name="services"
value="Можна   з   тваринами"><i   class="fa
fa-paw"  aria-hidden="true"></i>  Можна  з
тваринами</label>
                            <label><input
type="checkbox"             name="services"
value="Кондиціонер"><i  class="fa  fa-wind"
aria-hidden="true"></i>
Кондиціонер</label>
                            <label><input
type="checkbox"             name="services"
value="Казино"><i    class="fa    fa-dice"
aria-hidden="true"></i> Казино</label>
                            <label><input
type="checkbox"             name="services"
value="Пляж"><i                   class="fa
fa-umbrella-beach"  aria-hidden="true"></i>
Пляж</label>
                        </div><br><br>
                       <button type="submit"
class="btn">ДОДАТИ ГОТЕЛЬ</button>
                    </form>

                </div>
                        <div  id="my_hotels"
class="content">
                     <div class="hotel-list"
id="my_hotels">
                    </div>
                      <div class="room-list"
id="room_list">
                    </div>
                </div>
            </div>


        </body>
        <script   src="{{   url_for('static',
filename='js/home.js') }}"></script>
        <script   src="{{   url_for('static',
filename='js/hotel.js') }}"></script>
        <script
src="https://cdn.jsdelivr.net/npm/sweetale
rt2@11"></script>
        <script
src="https://cdn.jsdelivr.net/npm/flatpick
r"></script>
        <script
src="https://npmcdn.com/flatpickr/dist/l10
n/uk.js"></script>
        <script
src="https://cdn.jsdelivr.net/npm/flatpick
r/dist/l10n/uk.js"></script>
        <script>
        document.addEventListener('DOMConten
tLoaded', function() {
            var userId = "{{ user_id }}";
            loadUserHotels(userId);
        });
            </script>
        </html>
```

hotel_details.html:

```
        <!DOCTYPE html>
        <html>
        <head>
            <title>RESERVIA</title>
                        <link    rel="icon"
href="static/uploads/system_icon/favicon.p
ng" type="image/x-icon">
             <link rel="stylesheet" href="{{
url_for('static',      filename='home.css')
}}">
             <link rel="stylesheet" href="{{
url_for('static',
filename='hotel_details.css') }}">
             <link rel="stylesheet" href="{{
url_for('static',
filename='hotel_card.css') }}">
                    <link   rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/li
bs/font-awesome/6.0.0-beta3/css/all.min.cs
s">
                    <link   rel="stylesheet"
href="https://unpkg.com/swiper/swiper-bund
le.min.css"/>
                    <link   rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/li
bs/font-awesome/6.0.0-beta3/css/all.min.cs
s">
```

```
                <link   rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/flatpic
kr/dist/flatpickr.min.css">


      </head>
      <body>
                <div   class="brand-logo"
style="background:
url('static/uploads/system_icon/background
.png')  no-repeat  center  center  /  cover;
height:      250px;      width:      auto;
backdrop-filter: blur(5px);">
                                    <span
class="brand-reservia">{{       hotel.name
}}</span>
          </div>

          <div class="header-links">
                            <a   href="{{
url_for('user.user_home')              }}"
class="header-link">НА ГОЛОВНУ</a>
          </div>
          <div>
              <div class="gallery">
                                     <h1
class="label_hotel">ГАЛЕРЕЯ</h1>
                                    <div
class="photo-wrapper">
                      {% for photo_url in
hotel.photo_urls %}
                             <img  src="{{
url_for('static',
filename='uploads/hotel_photo/'           +
photo_url)     }}"     alt="Фото     готелю"
class="gallery-img">
                      {% endfor %}
                  </div>
              </div>
                        <div  id="modal"
class="modal">
                                    <span
class="close">&times;</span>
                                     <img
class="modal-content" id="modal-img">
                  <div id="caption"></div>
              </div>
              <div class="hotel-details">
                  <div class="detail">
                                    <span
class="icon"><i                  class="fas
fa-info-circle"></i></span>
                      <h3>ОПИС:</h3>
                                    <p>{{
hotel.description }}</p>
                  </div>
                  <div class="detail">
                                    <span
class="icon"><i                  class="fas
fa-map-marker-alt"></i></span>
                      <h3>АДРЕСА:</h3>
                       <p>{{ hotel.address
}}</p>
                  </div>
                  <div class="detail">
                                    <span
class="icon"><i                  class="fas
fa-phone"></i></span>
                      <h3>КОНТАКТИ:</h3>
                       <p>{{ hotel.contact
}}</p>
                  </div>
                  <div class="detail">
                                        <span
class="icon"><i                  class="fas
fa-concierge-bell"></i></span>
                      <h3>СЕРВІСИ:</h3>
                                         <ul
class="services-list">
                        {% for service
in hotel.services %}
                            <li>
                                <i class="fa
{{   serviceIcons[service]   }}"></i>   {{
service }}
                            </li>
                            {% endfor %}
                      </ul>
                  </div>
                                    <div
class="hotel-rooms">
                      <h2>Номери</h2>
                           {% for room in
hotel.rooms %}
                                    <div
class="room-detail"        data-room-id="{{
room.id      }}"      data-max-capacity="{{
room.capacity       }}"      data-price="{{
room.price    }}"   data-available-from="{{
room.available_from.strftime('%Y-%m-%d')
}}"                  data-available-to="{{
room.available_to.strftime('%Y-%m-%d')  }}"
data-hotel-name="{{     hotel.name     }}"
data-room-name="{{ room.name }}">
                            <h3>{{ room.name
}}</h3>
                                    <input
type="hidden"    id="user_id"    value="{{
user_id }}">
                             <p><i class="fa
fa-bed"></i> Ціна за добу: {{ room.price
}} грн</p>
                             <p><i class="fa
fa-info-circle"></i>        Опис:        {{
room.description }}</p>
                             <p><i class="fa
fa-users"></i> Місткість: {{ room.capacity
}} осіб</p>
                             <p><i class="fa
fa-calendar-alt"></i>   Доступний   з:   {{
room.available_from.strftime('%d.%m.%Y')
}}                  до:                 {{
room.available_to.strftime('%d.%m.%Y')
}}</p>
                                    <button
class="book-room" data-room-id="{{ room.id
}}">Забронювати</button>
                        </div>
                        {% endfor %}
                  </div>
              </div>
          </div>



      </body>
      <script   src="{{   url_for('static',
filename='js/home.js') }}"></script>
```

```
      <script src="{{ url_for('static', filename='js/hotel_details.js') }}"></script>
      <script src="https://cdn.jsdelivr.net/npm/sweetalert2@11"></script>
      <script src="https://unpkg.com/swiper/swiper-bundle.min.js"></script>
      <script src="https://cdn.jsdelivr.net/npm/flatpickr@4.6.9/dist/l10n/uk.js"></script>
      <script src="https://cdn.jsdelivr.net/npm/flatpickr"></script>
      <script src="https://npmcdn.com/flatpickr/dist/l10n/uk.js"></script>
      <script src="https://cdn.jsdelivr.net/npm/flatpickr/dist/l10n/uk.js"></script>
      </html>
```

index.html:

```
      <!DOCTYPE html>
      <html>
      <head>
          <title>RESERVIA</title>
                      <link rel="icon" href="static/uploads/system_icon/favicon.png" type="image/x-icon">
           <link rel="stylesheet" href="{{ url_for('static', filename='index.css') }}">
           <link rel="stylesheet" href="{{ url_for('static', filename='home.css') }}">
           <link rel="stylesheet" href="{{ url_for('static', filename='hotel_card.css') }}">
           <link rel="stylesheet" href="{{ url_for('static', filename='hotel_card_main.css') }}">
                    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0-beta3/css/all.min.css">
                    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/flatpickr/dist/flatpickr.min.css">
      </head>
      <body>
                  <div class="brand-logo" style="background: url('static/uploads/system_icon/background.png') no-repeat center center / cover; height: 250px; width: auto; backdrop-filter: blur(5px);">
                                          <span class="brand-reservia">RESERVIA</span>
                                          <span class="brand-tagline">HOTEL BOOKING</span>
           </div>
           <div class="header-links">
                                  <a href="#" class="header-link">ВХІД</a>
           </div>
                     <div id="loginModal" class="modal">
                <div class="modal-content">
                                          <span class="close">&times;</span>
                      <form action="/login" method="post">
                                          <label for="username" class="label">Логін:</label>
                          <input type="text" id="username" name="username" placeholder="Введіть ваш логін" class="input-field" required><br><br>
                                          <label for="password" class="label">Пароль:</label>
                                          <input type="password" id="password" name="password" placeholder="Введіть ваш пароль" class="input-field" required><br><br>
                                           <div class="button-container">
                                      <button type="submit" class="login-btn">УВІЙТИ</button>
                          </div>
                                                  <p class="label_desc">Немає профілю? <a href="#" class="register-link"><span style="color: #00c3ff;">Реєстрація!</span></a></p>
                                          <div class="flashes">
                              {% with messages = get_flashed_messages() %}
                                          {% if messages %}
                                            <ul class="messages">
                                              {% for message in messages %}
                                              <li>{{ message }}</li>
                                              {% endfor %}
                                          </ul>
                                        {% endif %}
                                    {% endwith %}
                          </div>
                      </form>
                  </div>
              </div>
                     <div id="registerModal" class="modal">
                  <div class="modal-content">
                                          <span class="close-register">&times;</span>
                      <form action="/register" method="post">
                                  <label for="email" class="label">Електрона пошта:</label>
                                <input type="email" id="email" name="email" placeholder="Введіть вашу електрону пошту" class="input-field" required><br><br>
                                          <label for="reg_username" class="label">Логін:</label>
                                  <input type="text" id="reg_username" name="reg_username"
```

```
placeholder="Введіть ваш логін" class="input-field" required><br><br>
                                    <label for="reg_password" class="label">Пароль:</label>
                                    <input type="password" id="reg_password" name="reg_password" placeholder="Введіть ваш пароль" class="input-field" required><br><br>
                                    <div class="button-container">
                                    <button type="submit" class="login-btn">ЗАРЕЄСТРУВАТИ</button>
                        </div>
                                    <div class="flashes">
                            {% with messages = get_flashed_messages() %}
                                    {% if messages %}
                                    <ul class="messages">
                                    {% for message in messages %}

<li>{{ message }}</li>
                                    {% endfor %}
                                    </ul>
                                {% endif %}
                            {% endwith %}
                        </div>
                    </form>
                </div>
            </div>
                    <div id="my_page" class="my_page">
                <div class="search-section">
                        <input type="text" id="search_input" placeholder="Введіть назву готелю, послуги, адресу">
                </div>
                                    <div class="filters-section-main">
                    <div class="filters">
                                    <label><input type="checkbox" value="WiFi"> <i class="fa fa-wifi" aria-hidden="true"></i> WiFi</label>
                                    <label><input type="checkbox" value="Парковка"> <i class="fa fa-car" aria-hidden="true"></i> Парковка</label>
                                    <label><input type="checkbox" value="Басейн"> <i class="fa fa-swimmer" aria-hidden="true"></i> Басейн</label>
                                    <label><input type="checkbox" value="Спортзал"> <i class="fa fa-dumbbell" aria-hidden="true"></i> Спортзал</label>
                                    <label><input type="checkbox" value="Спа"> <i class="fa fa-spa" aria-hidden="true"></i> Спа</label>
                                    <label><input type="checkbox" value="Ресторан"> <i class="fa fa-utensils" aria-hidden="true"></i> Ресторан</label>
                                    <label><input type="checkbox" value="Бар"> <i class="fa fa-glass-martini-alt" aria-hidden="true"></i> Бар</label>
                                    <label><input type="checkbox" value="Пральня"> <i class="fa fa-soap" aria-hidden="true"></i> Пральня</label>
                                    <label><input type="checkbox" value="Трансфер"> <i class="fa fa-shuttle-van" aria-hidden="true"></i> Трансфер</label>
                                    <label><input type="checkbox" value="Конференц зала"> <i class="fa fa-users" aria-hidden="true"></i> Конференц зала</label>
                                    <label><input type="checkbox" value="Можна з тваринами"> <i class="fa fa-paw" aria-hidden="true"></i> Можна з тваринами</label>
                                    <label><input type="checkbox" value="Кондиціонер"> <i class="fa fa-wind" aria-hidden="true"></i> Кондиціонер</label>
                                    <label><input type="checkbox" value="Казино"> <i class="fa fa-dice" aria-hidden="true"></i> Казино</label>
                                    <label><input type="checkbox" value="Пляж"> <i class="fa fa-umbrella-beach" aria-hidden="true"></i> Пляж</label>
                                    <label><input type="checkbox" value="Сніданок"> <i class="fa fa-bread-slice" aria-hidden="true"></i> Сніданок</label>
                    </div>
                </div>
                    <div id="hotel-list-main" class="hotel-list-main"></div>
            </div>
            </div>
        </body>
        <script src="{{ url_for('static', filename='js/register.js') }}"></script>
        <script src="{{ url_for('static', filename='js/home.js') }}"></script>
        <script src="https://cdn.jsdelivr.net/npm/flatpickr@4.6.9/dist/l10n/uk.js"></script>
        </html>
```

Клієнтська частина (скрипти):

home.js:

```
        function showContent(contentId) {
                    const dynamicContent = document.getElementById('dynamic-content');
            if (!dynamicContent) {
                console.error("Couldn't find the element with ID 'dynamic-content'.");
                return;
            }

              dynamicContent.style.display = 'block';
```

```
                const contentSections = dynamicContent.querySelectorAll('.content');
            contentSections.forEach(section => {
                    section.style.display = 'none';
            });

                const selectedContent = document.getElementById(contentId);
            if (!selectedContent) {
                console.error(`Couldn't find the element with ID '${contentId}'.`);
                return;
            }

            selectedContent.style.display = 'block';

        }

        function loadUserHotels(userID) {
            fetch(`/user/${userID}/hotels`)
                    .then(response => response.json())
            .then(hotels => {
                const serviceIcons = {
                    "WiFi": "fa-wifi",
                    "Парковка": "fa-car",
                    "Басейн": "fa-swimmer",
                                "Спортзал": "fa-dumbbell",
                    "Спа": "fa-spa",
                                "Ресторан": "fa-utensils",
                                    "Бар": "fa-glass-martini-alt",
                    "Пральня": "fa-soap",
                                "Трансфер": "fa-shuttle-van",
                            "Конференц зала": "fa-users",
                        "Можна з тваринами": "fa-paw",
                                "Кондиціонер": "fa-wind",
                    "Казино": "fa-dice",
                                    "Пляж": "fa-umbrella-beach",
                                "Сніданок" : "fa-bread-slice"
                };
                        const container = document.getElementById('my_hotels');
                hotels.forEach(hotel => {
                                const card = document.createElement('div');
                        const firstPhotoUrl = "/static/uploads/hotel_photo/"+hotel.photo_urls[0]

                            card.className = 'hotel-card';
                    card.innerHTML = `
                                            <img src="${firstPhotoUrl}" alt="Hotel Photo">

<h2>${hotel.name}</h2>

<p><strong>Сервіси:</strong></p>
                                            <div class="hotel-services">

${hotel.services.map(service => `
                                            <span class="hotel-service">
                                                <i class="fa ${serviceIcons[service]}" aria-hidden="true"></i> ${service}
                                            </span>
                                `).join('')}
                                </div>

<div><h3>Номери</h3></div>
                                            <div class="hotel-rooms">

${hotel.rooms.map(room => `
                                            <div class="hotel-room">
                                        <h4><i class="fa fa-bed" aria-hidden="true"></i> ${room.name}</h4>
                                            <p><i class="fa fa-money-bill-wave" aria-hidden="true"></i> Ціна: ${room.price} грн/доба </p>
                                            <p><i class="fa fa-align-left" aria-hidden="true"></i> Опис: ${room.description}</p>
                                            <p><i class="fa fa-users" aria-hidden="true"></i> Номер на: ${room.capacity} осіб</p>
                                        <p>Номер вільний для бронювання з <span style="color:#00c3ff;">${room.available_from}</span> по <span style="color:#00c3ff;">${room.available_to}</span> </p>
                                        <button class="delete_room" data-room-id="${room.id}" data-room-name="${room.name}">Видалити</button>
                                </div>
                                    `).join('')}
                                </div>
                                            <div class="hotel-card-right">
                                        <button id="add_room" class="add_room" data-hotel-id="${hotel.id}">Додати кімнати</button>
                            <button class="more" data-hotel-id="${hotel.id}" data-user-hotel-id="${userID}">Детальніше</button>
                                        <button class="delete" data-hotel-id="${hotel.id}" data-hotel-name="${hotel.name}">Видалити</button>
                                </div>
                            `;

container.appendChild(card);
                });
```

```
        })
                    .catch(error => console.error('Error:', error));
    }

    document.addEventListener('DOMContentLoaded', function() {
        document.getElementById('my_hotels').addEventListener('click', function(event) {
            if (event.target.classList.contains('add_room')) {
                const hotelId = event.target.getAttribute('data-hotel-id');
                Swal.fire({
                    title: 'Додати кімнату',
                    html: `
                        <label for="name">Назва кімнати:</label>
                        <input type="text" id="name" name="name" required>
                        <label for="price">Ціна:</label>
                        <input type="number" id="price" name="price" required>
                        <label for="description">Опис:</label>
                        <textarea id="description" name="description" required></textarea>
                        <label for="capacity">Кількість місць:</label>
                        <input type="number" id="capacity" name="capacity" required>
                        <label for="date_range">Доступні дати:</label>
                        <input type="text" id="date_range" name="date_range" required>
                    `,
                    confirmButtonText: 'Додати',
                    confirmButtonColor: '#0fb64f',
                    didOpen: () => {
                        flatpickr.localize(flatpickr.l10ns.uk);
                        flatpickr("#date_range", {
                            mode: "range",
                            locale: "uk",
                            locale: {
                                rangeSeparator: ' — '
                            },
                            altInput: true,
                            altFormat: "d.m.Y",
                            dateFormat: "d.m.Y",
                        });
                    },
                    preConfirm: () => {
                        const name = Swal.getPopup().querySelector('#name').value;
                        const price = Swal.getPopup().querySelector('#price').value;
                        const description = Swal.getPopup().querySelector('#description').value;
                        const capacity = Swal.getPopup().querySelector('#capacity').value;
                        const dateRange = Swal.getPopup().querySelector('#date_range').value;
                        if (!name || !price || !description || !capacity || !dateRange) {
                            Swal.showValidationMessage(`Всі поля обов'язкові`);
                            return;
                        }
                        const dates = dateRange.split(" — ");
                        return { name: name, price: price, description: description, capacity: capacity, available_from: dates[0], available_to: dates[1] };
                    }
                }).then((result) => {
                    if (result.isConfirmed) {
                        console.log('Room data:', result.value);
                        fetch('/add_room', {
                            method: 'POST',
                            headers: {
                                'Content-Type': 'application/json'
                            },
                            body: JSON.stringify({
                                name: result.value.name,
                                price: parseFloat(result.value.price),
                                description: result.value.description,
                                capacity: parseInt(result.value.capacity),
                                available_from: result.value.available_from,
                                available_to: result.value.available_to,
                                hotel_id: parseInt(hotelId)
                            })
                        }).then(response => response.json())
                        .then(data => {
                            if (data.error) {
```

```
Swal.fire('Error!', data.error, 'error');
                                } else {

Swal.fire('Кімната додана. Перезавантажте сторінку для відображення змін.', data.message, 'success');

loadUserHotels(userID);
                                }
                            })
                             .catch(error => console.error('Error:', error));
                        }
                    });
                }
            });
        });
        document.addEventListener('DOMContentLoaded', function () {
                const hotelsContainer = document.getElementById('my_hotels');

hotelsContainer.addEventListener('click', function(event) {
                    const isDeleteButton = event.target.classList.contains('delete');
                        const hotelId = event.target.getAttribute('data-hotel-id');
                        const hotelName = event.target.getAttribute('data-hotel-name');
                        const hotelCard = event.target.closest('.hotel-card');
                if (isDeleteButton) {
                    Swal.fire({
                                title: 'Ви впевнені?',
                            text: `Ви справді бажаєте видалити готель "${hotelName}" та всі його номери?`,
                        icon: 'warning',
                           showCancelButton: true,
                         confirmButtonColor: '#db0a06',
                          cancelButtonColor: '#00c3ff',
                          confirmButtonText: 'Так, видалити!',
                           cancelButtonText: 'Скасувати'
                    }).then((result) => {
                                            if (result.isConfirmed) {

fetch(`/delete_hotel/${hotelId}`, {
                                        method: 'POST'
                                })
                                  .then(response => response.json())
                                .then(data => {
                                    Swal.fire(

'Видалено!',
                                        'Готель та всі його номери було успішно видалено. Перезавантажте сторінку для відображення змін.',

'success'
                                );
                                            if (hotelCard) {

hotelCard.remove();
                                }
                            })
                             .catch(error => {

console.error('Error:', error);
                                Swal.fire(

'Помилка!',
                                    'Під час видалення виникла помилка.',
                                    'error'
                                );
                            });
                        }
                    });
                }
            });
        });
        document.addEventListener('DOMContentLoaded', function () {
                const hotelsContainer = document.getElementById('my_hotels');

hotelsContainer.addEventListener('click', function(event) {
                                            if (event.target.classList.contains('delete_room')) {
                                const roomId = event.target.getAttribute('data-room-id');
                            const roomName = event.target.getAttribute('data-room-name');
                        const roomElement = event.target.closest('.hotel-room');

                    Swal.fire({
                                    title: 'Ви впевнені?',
                                text: `Ви справді бажаєте видалити кімнату "${roomName}"?`,
                            icon: 'warning',
                               showCancelButton: true,
                             confirmButtonColor: '#db0a06',
                              cancelButtonColor: '#00c3ff',
                              confirmButtonText: 'Так, видалити!',
                               cancelButtonText: 'Скасувати'
                    }).then((result) => {
                                            if (result.isConfirmed) {

fetch(`/delete_room/${roomId}`, {
                                        method: 'POST'
                                })
```

```
                            .then(response
=> response.json())
                        .then(data => {
                            Swal.fire(
'Видалено!',
                                'Кімната
була успішно видалена.',
'success'
                            );
                                    if
(roomElement) {
roomElement.remove();
                            }
                        })
                         .catch(error =>
{
console.error('Error:', error);
                            Swal.fire(
'Помилка!',
                                'Під час
видалення виникла помилка.',
                                'error'
                            );
                        });
                    }
                });
            }
        });
    });


    document.addEventListener('DOMConten
tLoaded', function() {
                const hotelsContainer =
document.getElementById('my_hotels');
        if (!hotelsContainer) {
                console.error('Unable to
find the container for hotels');
            return;
        }

hotelsContainer.addEventListener('click',
function(event) {
                                    if
(!event.target.classList.contains('more'))
return;
                        const hotelId =
event.target.getAttribute('data-hotel-id')
;
                         const userID =
event.target.getAttribute('data-user-hotel
-id');
            if (!hotelId || !userID) {
                 console.error('No hotel
ID or user ID found');
                return;
            }
                 window.location.href =
`/hotel_details?hotelId=${hotelId}&userId=
${userID}`;
        });
    });

    document.addEventListener('DOMConten
tLoaded', function() {
                        const userID =
document.getElementById('userID').getAttri
bute('data-user-id');

fetch(`/user/${userID}/bookings`)
                         .then(response =>
response.json())
            .then(bookings => {
                const bookingContainer =
document.getElementById('my_booking');
                bookings.forEach(booking
=> {
                     const bookingDiv =
document.createElement('div');
                    bookingDiv.className
= 'booking-detail';
                    bookingDiv.innerHTML
= `

<h3>${booking.hotel_name}                -
${booking.room_name}</h3>
                                    <p>З
<span>${booking.start_date}</span>      до
<span>${booking.end_date}</span></p>
                                    <p>Ціна:
<span>${booking.price} </span> грн.</p>
                                     <button
onclick="cancelBooking(${booking.id})">Від
мінити</button>
                    `;

bookingContainer.appendChild(bookingDiv);
                });
            })
                        .catch(error =>
console.error('Error loading bookings:',
error));
    });


    function cancelBooking(bookingId) {
        Swal.fire({
            title: 'Ви впевнені?',
             text: "Це дію не можна буде
відмінити!",
            icon: 'warning',
            showCancelButton: true,
                     confirmButtonColor:
'#dc3545',
                    cancelButtonColor:
'#6c757d',
            confirmButtonText: 'Так,
відмінити!',
                     cancelButtonText:
'Скасувати'
        }).then((result) => {
            if (result.isConfirmed) {

fetch(`/cancel_booking/${bookingId}`,    {
method: 'POST' })
                    .then(response => {
                                    if
(!response.ok) throw new Error('Failed to
cancel booking');
                                return
response.json();
                    })
                    .then(data => {
                        Swal.fire(
```

```
'Відмінено!',
                                    'Ваше
бронювання    було    відмінено.    Оновіть
сторінку для відображення змін.',
                                'success'
                            );
                        })
                        .catch(error => {
                            Swal.fire(
                                'Помилка!',
                                 'Не вдалося
відмінити бронювання.',
                                'error'
                            );

console.error('Error:', error);
                        });
                }
            });
        }

        document.addEventListener('DOMConten
tLoaded', function() {

document.getElementById('search_input').ad
dEventListener('input', searchHotels);

document.querySelectorAll('.filters
input[type="checkbox"]').forEach(checkbox
=> {

checkbox.addEventListener('change',
searchHotels);
            });

            loadHotels();
        });

        function searchHotels() {
                    const   searchInput   =
document.getElementById('search_input').va
lue.toLowerCase();
                const   selectedFilters   =
Array.from(document.querySelectorAll('.fil
ters
input[type="checkbox"]:checked')).map(cb
=> cb.value);
            fetch('/hotels')
                        .then(response   =>
response.json())
                .then(hotels => {

updateHotelList(searchInput,
selectedFilters, hotels);
                })
                        .catch(error   =>
console.error('Error:', error));
        }

        function loadHotels() {
            fetch('/hotels')
                        .then(response   =>
response.json())
                .then(hotels => {
                    updateHotelList('', [],
hotels);
                })
                        .catch(error   =>
console.error('Error:', error));
        }

        function
updateHotelList(searchInput,
selectedFilters, hotels) {
                        const   container   =
document.getElementById('hotel-list-main')
;
            container.innerHTML = '';

            hotels.forEach(hotel => {
                    const  firstPhotoUrl  =
"/static/uploads/hotel_photo/"            +
hotel.photo_urls[0];

                            const   card   =
document.createElement('div');
                            card.className   =
'hotel-card-main';
                card.innerHTML = `
                                            <img
src="${firstPhotoUrl}" alt="Hotel Photo">
                                            <div
class="hotel-info-main">

<h2>${hotel.name}</h2>

<p><strong>Адреса:</strong>
${hotel.address}</p>

<p><strong>Опис:</strong>
${hotel.description}</p>
                                            <div
class="hotel-services">

${hotel.services.map(service => `
                                            <span
class="hotel-service-main">${service}</spa
n>
                            `).join('')}
                        </div>
                    </div>
                                            <div
class="button-wrapper">
                                        <button
class="more-main"
data-hotel-id="${hotel.id}">Детальніше</bu
tton>
                    </div>
                `;

                    const matchesSearch =
hotel.name.toLowerCase().includes(searchIn
put) ||

hotel.address.toLowerCase().includes(searc
hInput) ||

hotel.description.toLowerCase().includes(s
earchInput);

                    const matchesFilters =
selectedFilters.length     ===     0     ||
selectedFilters.every(filter            =>
hotel.services.includes(filter));

                    if  (matchesSearch  &&
matchesFilters) {

container.appendChild(card);
```

```
            }
        });

        if (container.innerHTML === '') {
            container.innerHTML = `
                <div class="no-results">
                    <p>Немає готелів, що відповідають вашим критеріям пошуку.</p>
                </div>
            `;
        }
    }
    document.addEventListener('DOMContentLoaded', function() {
        const hotelsContainer = document.getElementById('my_page');
        if (!hotelsContainer) {
            console.error('Unable to find the container for hotels');
            return;
        }

hotelsContainer.addEventListener('click', function(event) {
                                    if (!event.target.classList.contains('more-main')) return;
                    const hotelId = event.target.getAttribute('data-hotel-id');
                window.location.href = `/hotel_details?hotelId=${hotelId}&userId=${99999}`;
        });
    });
```

hotel_details.js:

```
    document.addEventListener('DOMContentLoaded', function () {
                const images = document.querySelectorAll('.gallery-img');
                const modal = document.getElementById('modal');
                const modalImg = document.getElementById('modal-img');
                const captionText = document.getElementById('caption');
                const close = document.getElementsByClassName("close")[0];

        images.forEach(img => {
            img.onclick = function(){
                modal.style.display = "block";
                modalImg.src = this.src;
                captionText.innerHTML = this.alt;
            }
        });

        close.onclick = function() {
                modal.style.display = "none";
        }
    });
    document.addEventListener('DOMContentLoaded', function() {
        const hotelRoomsContainer = document.querySelector('.hotel-rooms');

hotelRoomsContainer.addEventListener('click', function(event) {
                                    if (event.target.classList.contains('book-room')) {
                const roomElement = event.target.closest('.room-detail');
                const roomId = roomElement.getAttribute('data-room-id');
                const maxCapacity = roomElement.getAttribute('data-max-capacity');
                const pricePerNight = roomElement.getAttribute('data-price');
                const minDate = roomElement.getAttribute('data-available-from');
                const maxDate = roomElement.getAttribute('data-available-to');
                const hotelName = roomElement.getAttribute('data-hotel-name');
                const roomName = roomElement.getAttribute('data-room-name');
                const userID = document.getElementById('user_id').value;
            if (userID === '99999') {
                Swal.fire({
                    title: 'Необхідно увійти',
                    text: 'Необхідно увійти для продовження бронювання.',
                    icon: 'warning',
confirmButtonText: 'OK'
                });
                return;
            }
            Swal.fire({
                title: `Забронювати кімнату: ${roomName}`,
                html: `
                    <label for="guests">Кількість гостей:</label>
                    <input type="number" id="guests" name="guests" min="1" max="${maxCapacity}" required>
                    <label for="booking_dates">Дати заселення:</label>
                    <input type="text" id="booking_dates" name="booking_dates" required>
                    <div id="calculated_price"></div>
                `,
                confirmButtonText: 'Забронювати',
                confirmButtonColor: '#0fb64f',
                didOpen: () => {
flatpickr.localize(flatpickr.l10ns.uk);
```

```
flatpickr("#booking_dates", {
                                    mode:
"range",
                                minDate:
minDate,
                                maxDate:
maxDate,
                             dateFormat:
"Y-m-d",
                                 locale:
"uk",
                            locale: {

rangeSeparator: ' — '
                            },
                               onChange:
function(selectedDates) {
                                    const
[start, end] = selectedDates;
                                       if
(start && end) {

const diff = (end - start) / (1000 * 3600
* 24);

const totalPrice = diff * pricePerNight;

document.getElementById('calculated_price'
).textContent     =     `Загальна     ціна:
${totalPrice.toFixed(2)} грн`;
                                    }
                                }
                            });
                        },
                        preConfirm: () => {
                              const guests =
Swal.getPopup().querySelector('#guests').v
alue;
                                    const
bookingDates                            =
Swal.getPopup().querySelector('#booking_da
tes').value;
                            if (!guests ||
!bookingDates) {

Swal.showValidationMessage(`Всі       поля
обов'язкові`);
                                return;
                            }
                                    const
[available_from,      available_to]      =
bookingDates.split(" — ");
                                    const
bookingData = {
                                  room_name:
roomName,
                                 hotel_name:
hotelName,

available_from,

available_to,
                                      price:
document.getElementById('calculated_price'
).textContent.split(":     ")[1].replace('
грн', ''),
                                userID
                            };

console.log('Booking data:', bookingData);
// Виводимо дані перед відправкою
                                    return
bookingData;
                        }
                    }).then((result) => {
                                        if
(result.isConfirmed) {

console.log('Booking               data:',
result.value);

fetch('/book_room', {
                                    method:
'POST',
                                headers: {

'Content-Type': 'application/json'
                                },
                                    body:
JSON.stringify({

room_name: result.value.room_name,

hotel_name: result.value.hotel_name,

start_date: result.value.available_from,

end_date: result.value.available_to,
                                    price:
parseFloat(result.value.price),
                                user_id:
userID
                                })
                            }).then(response
=> response.json())
                            .then(data => {
                                        if
(data.error) {

Swal.fire('Error!', data.error, 'error');
                                } else {

Swal.fire('Заброньовано!',           'Ваше
бронювання    успішно    зареєстровано.',
'success');
                                }
                            })
                             .catch(error =>
console.error('Error:', error));
                    }
                });
            }
        });
    });
```

hotel.js:

```
    document.addEventListener('DOMConten
tLoaded', function() {



document.getElementById('photo').addEventL
istener('change', function() {
                        const  preview  =
document.getElementById('preview');
            preview.innerHTML = '';

                  for  (let  i  =  0;  i  <
this.files.length; i++) {
```

```
                            const file =
this.files[i];

                                        if
(file.type.startsWith('image/')) {
                                const img =
document.createElement('img');
                                    img.src =
URL.createObjectURL(file);
                                img.onload =
function() {

URL.revokeObjectURL(this.src);
                        }

                            const container =
document.createElement('div');

container.style.position = 'relative';

container.appendChild(img);

                        const removeButton =
document.createElement('span');

removeButton.textContent = '✕';

removeButton.className = 'remove-img';
                        removeButton.onclick
= function() {

this.parentNode.remove();
                        };


container.appendChild(removeButton);

preview.appendChild(container);
                    }
                }
            });

      document.getElementById('hotelForm')
.addEventListener('submit',
function(event) {


document.getElementById('photo').addEventL
istener('change', function() {
                            const preview =
document.getElementById('preview');
                preview.innerHTML = '';

                    for (let i = 0; i <
this.files.length; i++) {
                            const file =
this.files[i];

                                        if
(file.type.startsWith('image/')) {
                                const img =
document.createElement('img');
                                    img.src =
URL.createObjectURL(file);
                                img.onload =
function() {

URL.revokeObjectURL(this.src);  // Очистка
пам'яті
                        }

                            const container =
document.createElement('div');

container.style.position = 'relative';

container.appendChild(img);

                        const removeButton =
document.createElement('span');

removeButton.textContent = '✕';

removeButton.className = 'remove-img';
                        removeButton.onclick
= function() {

this.parentNode.remove();     //  Видалення
блоку з зображенням
                        };


container.appendChild(removeButton);

preview.appendChild(container);
                    }
                }
            });

      });


document.getElementById('hotelForm').addEv
entListener('submit', function(event) {
                event.preventDefault();
                        const preview =
document.getElementById('preview');
                    var formData = new
FormData(this);

                Swal.fire({
                    title: "Ви впевнені?",
                        text: "Створити цей
готель?",
                    icon: "info",
                    showCancelButton: true,
                        confirmButtonText:
"Додати",
                        cancelButtonText:
"Скасувати",
                    confirmButtonColor:
'#0fb64f',
                        cancelButtonColor:
'#00c3ff',
                })
                .then((willAdd) => {
                    if (willAdd.isConfirmed)
{

fetch('/create_hotel', {
                            method: 'POST',
                            body: formData
                        })
                        .then(response =>
response.json())
                        .then(data => {

Swal.fire("Success!",         data.message,
"success");
```

```
document.getElementById('hotelForm').reset
();

preview.innerHTML = '';
                        })
                        .catch(error => {

console.error('Error:', error);

Swal.fire("Error!",   "Помилка   під   час
додавання   готелю!   Перевірте   дані   та
спробуйте ще раз.", "error");
                        });
                    }
                });

            });
        });
```

register.js:

```
        document.addEventListener("DOMConten
tLoaded", function() {
                    const    userImage    =
document.getElementById("user-image");
                const  profilePhotoInput  =
document.getElementById("profile_photo");


document.getElementById("image-overlay").a
ddEventListener("click", function() {
                profilePhotoInput.click();
            });


profilePhotoInput.addEventListener("change
", function() {
                    const  selectedFile  =
profilePhotoInput.files[0];
                if (selectedFile) {
                                const  url  =
URL.createObjectURL(selectedFile);
                    userImage.src = url;
                }
            });
        });
        var            loginModal           =
document.getElementById('loginModal');
        var          registerModal          =
document.getElementById('registerModal');

        var            loginLink            =
document.querySelector('.header-link[href=
"#"]');
        var          registerLink           =
document.querySelector('.register-link');

        var            closeLogin           =
document.getElementsByClassName("close")[0
];
        var          closeRegister          =
document.getElementsByClassName("close-reg
ister")[0];

        loginLink.onclick = function() {
            if (registerModal.style.display
=== "block") {
                registerModal.style.display
= "none";
            }
                loginModal.style.display  =
"block";
        }

        registerLink.onclick                =
function(event) {
            event.preventDefault();
            if (loginModal.style.display ===
"block") {
                loginModal.style.display =
"none";
            }
              registerModal.style.display  =
"block";
        }

        closeLogin.onclick = function() {
                loginModal.style.display  =
"none";
        }

        closeRegister.onclick = function() {
              registerModal.style.display  =
"none";
        }

        window.onclick = function(event) {
            if (event.target == loginModal)
{
                loginModal.style.display =
"none";
            }
                    if   (event.target   ==
registerModal) {
                registerModal.style.display
= "none";
            }
        }
```

Серверна частина (маршрути):

hotel.py:

```
        from  flask  import  render_template,
request,    session,    flash,    redirect,
url_for,Blueprint
        from flask import current_app as app
        from    flask    import    Blueprint,
request, jsonify, abort
        from flask_login import current_user
        from app.extensions import db
        from app.models import User, Hotel,
Room
        import os
        from      werkzeug.utils      import
secure_filename
        from datetime import datetime
        from sqlalchemy.sql import extract
        from collections import defaultdict
        from app import Config

        hotel = Blueprint('hotel', __name__)

        @hotel.route('/create_hotel',
methods=['POST'])
        def create_hotel():
            name = request.form['name']
                        description     =
request.form['description']
```

```
                                address      =
request.form['address']
                                contact      =
request.form['contact']
                               services      =
request.form.getlist('services')
                                user_id      =
request.form['user_id']

                           existing_hotel    =
Hotel.query.filter(Hotel.name.ilike(name))
.first()
          if existing_hotel:
                    return jsonify({'error':
'Готель з такою назвою вже існує!'}), 400

          photo_urls = []
                         for    file    in
request.files.getlist('photo'):
                                 filename   =
secure_filename(file.filename)
                               file_path   =
os.path.join(Config.HOTEL_IMG, filename)
               file.save(file_path)
               photo_urls.append(filename)

          hotel = Hotel(
              name=name,
              description=description,
              address=address,
              contact=contact,
              photo_urls=photo_urls,
              services=services,
              owner_id=user_id
          )
          db.session.add(hotel)
          db.session.commit()

               return  jsonify({'message':
'Готель успішно створено!'})

      @hotel.route('/add_room',
methods=['POST'])
      def add_room():
               required_fields  =  ['name',
'price',     'description',     'capacity',
'hotel_id',               'available_from',
'available_to']
          if not all(field in request.json
for field in required_fields):
                    return  jsonify({'error':
'Missing data!'}), 400
          try:
                           available_from  =
datetime.strptime(request.json['available_
from'], '%d.%m.%Y').date()
                            available_to   =
datetime.strptime(request.json['available_
to'], '%d.%m.%Y').date()
          except ValueError:
                    return  jsonify({'error':
'Invalid date format!'}), 400
          new_room = Room(
              name=request.json['name'],
              price=request.json['price'],

description=request.json['description'],

capacity=request.json['capacity'],

hotel_id=request.json['hotel_id'],
                is_available=True,

available_from=available_from,
                available_to=available_to
          )
          db.session.add(new_room)
          db.session.commit()
               return  jsonify({'message':
'Номер    успішно    додано!',    'room_id':
new_room.id}), 201


      @hotel.route('/user/<int:user_id>/ho
tels')
      def get_user_hotels(user_id):
          user = User.query.get(user_id)
          if not user:
                   return jsonify({'error':
'User not found'}), 404

          hotels = []
                           for    hotel    in
Hotel.query.filter_by(owner_id=user_id).al
l():
                                    rooms    =
Room.query.filter_by(hotel_id=hotel.id).al
l()
               rooms_data = [{
                   'id': room.id,
                   'name': room.name,
                            'is_available':
room.is_available,
                   'price': room.price,
                             'description':
room.description,
                                'capacity':
room.capacity,
                           'available_from':
room.available_from.strftime('%d.%m.%Y')
if room.available_from else None,
                             'available_to':
room.available_to.strftime('%d.%m.%Y')  if
room.available_to else None
               } for room in rooms]
               hotels.append({
                   'id': hotel.id,
                   'name': hotel.name,
                                'address':
hotel.address,
                                 'contact':
hotel.contact,
                              'description':
hotel.description,
                               'photo_urls':
hotel.photo_urls,
                                 'services':
hotel.services,
                                 'owner_id':
hotel.owner_id,
                   'rooms': rooms_data
               })
          return jsonify(hotels)


      @hotel.route('/hotel_details')
      def hotel_details():
          serviceIcons = {
```

```
            "WiFi": "fa-wifi",
            "Парковка": "fa-car",
            "Басейн": "fa-swimmer",
            "Спортзал": "fa-dumbbell",
            "Спа": "fa-spa",
            "Ресторан": "fa-utensils",
                                    "Бар":
"fa-glass-martini-alt",
            "Пральня": "fa-soap",
                                "Трансфер":
"fa-shuttle-van",
                        "Конференц зала":
"fa-users",
                    "Можна з тваринами":
"fa-paw",
            "Кондиціонер": "fa-wind",
            "Казино": "fa-dice",
            "Пляж": "fa-umbrella-beach",
            "Сніданок": "fa-bread-slice"
        }
                            hotel_id     =
request.args.get('hotelId', type=int)
                             user_id     =
request.args.get('userId', type=int)
        if not hotel_id:
                    return "Hotel ID is
required", 400
        if not user_id:
                     return "User ID is
required", 400

                                hotel      =
Hotel.query.get(hotel_id)
        if not hotel:
              return "Hotel not found",
404

                                rooms      =
Room.query.filter_by(hotel_id=hotel.id).al
l()
        rooms_data = [{
            'id': room.id,
            'name': room.name,
                           'is_available':
room.is_available,
            'price': room.price,
                            'description':
room.description,
            'capacity': room.capacity,
                       'available_from'  :
room.available_from,
                        'available_to'   :
room.available_to
        } for room in rooms]

        hotel_data = {
            'id': hotel.id,
            'name': hotel.name,
            'address': hotel.address,
            'contact': hotel.contact,
                            'description':
hotel.description,
                             'photo_urls':
hotel.photo_urls,
            'services': hotel.services,
            'owner_id': hotel.owner_id,
            'rooms': rooms_data
        }
                                        return
render_template('hotel_details.html',
hotel=hotel_data,
serviceIcons=serviceIcons,
user_id=user_id)



    @hotel.route('/delete_hotel/<int:hot
el_id>', methods=['POST'])
    def delete_hotel(hotel_id):
                                hotel      =
Hotel.query.get(hotel_id)
        if not hotel:
                return jsonify({'error':
'Hotel not found'}), 404

        try:

Room.query.filter_by(hotel_id=hotel_id).de
lete()
            db.session.delete(hotel)
            db.session.commit()
              return jsonify({'message':
'Hotel and all its rooms have been deleted
successfully!'})
        except Exception as e:
            db.session.rollback()
                return jsonify({'error':
str(e)}), 500

    @hotel.route('/delete_room/<int:room
_id>', methods=['POST'])
    def delete_room(room_id):
        room = Room.query.get(room_id)
        if not room:
                return jsonify({'error':
'Room not found'}), 404

        try:
            db.session.delete(room)
            db.session.commit()
              return jsonify({'message':
'Room has been deleted successfully!'})
        except Exception as e:
            db.session.rollback()
                return jsonify({'error':
str(e)}), 500
```

main.py:

```
    from flask import render_template,
request,    session,    flash,    redirect,
url_for,Blueprint
    from flask import current_app as app
    from    flask    import    Blueprint,
request, jsonify, abort
    from app.extensions import db
    from    app.models    import    User,
Booking, Hotel
    import os
    from      werkzeug.utils      import
secure_filename
    from datetime import datetime
    from sqlalchemy.sql import extract
    from collections import defaultdict
    main = Blueprint('main', __name__)

    @main.route('/')
    def index():
```

```
                                    return
render_template('index.html')



      @main.route('/login',
methods=['GET', 'POST'])
      def login():
          if request.method == 'POST':
                                username   =
request.form['username']
                                password   =
request.form['password']
                if len(username) < 8 or
len(password) < 8:
                    flash('Логін та пароль
повинні містити не менше 8 символів!',
'error')
                                    return
render_template('index.html')

                                  user   =
User.query.filter_by(username=username,
password=password).first()
              if user:
                  user.is_active = True
                  db.session.commit()
                  user_login(user)
                   flash('Вхід успішний!',
'success')
                                    return
redirect(url_for('user.user_home'))
              else:
                     flash('Помилка входу!
Перевірте дані та спробуйте ще раз.',
'error')

                                    return
render_template('index.html')

      @main.route('/register',
methods=['GET', 'POST'])
      def register():
          if request.method == 'POST':
                                   email   =
request.form['email']
                                username   =
request.form['reg_username']
                                password   =
request.form['reg_password']

               # Перевірка довжини логіна
та паролю
                 if len(username) < 8 or
len(password) < 8 or len(email) < 5:   #
Додаємо базову перевірку для email
                    flash('Логін, пароль та
email мають бути достатньо довгими!',
'error')
                                    return
render_template('index.html')

                  # Перевірка унікальності
email та username
                         existing_user   =
User.query.filter_by(username=username).fi
rst()
                        existing_email   =
User.query.filter_by(email=email).first()
              if existing_user:
                        flash('Користувач з
таким логіном вже існує.', 'error')
                                    return
render_template('index.html')
              if existing_email:
                        flash('Користувач з
таким email вже існує.', 'error')
                                    return
render_template('index.html')

                                 new_user   =
User(username=username,        email=email,
password=password, is_active=True)
              db.session.add(new_user)
              db.session.commit()
              user_login(new_user)
                 flash('Реєстрація пройшла
успішно!', 'success')
                                    return
redirect(url_for('user.user_home'))

                                    return
render_template('index.html')

      def user_login(user):
          user.is_active = True
          db.session.commit()
          session['user_id'] = user.id

      @main.route('/logout')
      def logout():
          if 'user_id' in session:
              user_id = session['user_id']
                                  user   =
User.query.get(user_id)
              if user:
                  user.is_active = False
                  db.session.commit()
          session.clear()
                                    return
redirect(url_for('main.index'))



      main.route('/create_hotel',
methods=['POST'])
      def create_hotel():
          data = request.form
          name = data['name']

                          existing_hotel    =
Hotel.query.filter(func.lower(Hotel.name)
== func.lower(name)).first()
          if existing_hotel:
                  return jsonify({'error':
'Hotel with this name already exists!'}),
400

                             description    =
data['description']
          address = data['address']
          contact = data['contact']
                              photo_urls    =
request.files.getlist('photo')
                               services     =
data.getlist('services')

                hotel  =  Hotel(name=name,
description=description,  address=address,
contact=contact,    photo_urls=photo_urls,
```

```
services=services,
owner_id=current_user.id)
          db.session.add(hotel)
          db.session.commit()

               return  jsonify({'message':
'Hotel created successfully!'})


      @main.route('/book_room',
methods=['POST'])
      def book_room():
          data = request.get_json()
          print("Received data:", data)
           if not all(key in data for key
in       ['room_name',       'hotel_name',
'start_date',     'end_date',     'price',
'user_id']):
                  return jsonify({'error':
'Missing required data!'}), 400

          try:
                              start_date   =
datetime.strptime(data['start_date'],
'%Y-%m-%d').date()
                               end_date   =
datetime.strptime(data['end_date'],
'%Y-%m-%d').date()
          except ValueError:
                  return jsonify({'error':
'Invalid date format!'}), 400

          if start_date >= end_date:
                  return jsonify({'error':
'End  date  must  be  after  start  date!'}),
400

          booking = Booking(

hotel_name=data['hotel_name'],
              room_name=data['room_name'],
              start_date=start_date,
              end_date=end_date,
              price=data['price'],
              user_id=data['user_id']
          )

          db.session.add(booking)
          db.session.commit()

               return  jsonify({'message':
'Booking      successfully      created!',
'booking_id': booking.id}), 201

      @main.route('/user/<int:user_id>/boo
kings')
      def user_bookings(user_id):
                            bookings     =
Booking.query.filter_by(user_id=user_id).a
ll()
          bookings_data = [{
              'id': booking.id,
                            'hotel_name':
booking.hotel_name,
                             'room_name':
booking.room_name,
                            'start_date':
booking.start_date.strftime('%Y-%m-%d'),
                              'end_date':
booking.end_date.strftime('%Y-%m-%d'),
              'price': booking.price
          } for booking in bookings]

          return jsonify(bookings_data)

      @main.route('/cancel_booking/<int:bo
oking_id>', methods=['POST'])
      def cancel_booking(booking_id):
                                booking      =
Booking.query.get(booking_id)
          if not booking:
                  return jsonify({'error':
'Booking not found!'}), 404

          db.session.delete(booking)
          db.session.commit()

               return  jsonify({'message':
'Booking cancelled successfully!'}), 200


      @main.route('/hotels')
      def get_hotels():
          hotels = Hotel.query.all()
          hotels_data = [{
              'id': hotel.id,
              'name': hotel.name,
              'address': hotel.address,
                              'description':
hotel.description,
                               'photo_urls':
hotel.photo_urls,
              'services': hotel.services
          } for hotel in hotels]

          return jsonify(hotels_data)
```

user.py:

```
      from  flask  import  render_template,
request,    session,    flash,    redirect,
url_for,Blueprint
      from flask import current_app as app
      from    flask    import    Blueprint,
request, jsonify, abort
      from app.extensions import db
      from app.models import User
      import os
      from      werkzeug.utils      import
secure_filename
      from datetime import datetime
      from sqlalchemy.sql import extract
      from collections import defaultdict
      user = Blueprint('user', __name__)
      @user.route('/user_home')
      def user_home():
          if 'user_id' in session:
              user_id = session['user_id']
                                    user    =
User.query.get(user_id)
              if user:
                                    return
render_template('home.html',     user=user,
username=user.username, user_id=user.id)
              else:
                                    return
render_template('404.html'), 404
          else:
                                    return
render_template('404.html'), 404
```