МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет водного господарства та природокористування
Навчально-науковий інститут кібернетики, інформаційних технологій та інженерії
Кафедра комп'ютерних технологій та економічної кібернетики

Допущено до захисту:

Завідувач кафедри
комп'ютерних технологій та
економічної кібернетики
д. е. н., проф. П. М. Грицюк

_____

«____» _____ 2026 р.

КВАЛІФІКАЦІЙНА РОБОТА НА ЗДОБУТТЯ ОСВІТНЬО-КВАЛІФІКАЦІЙНОГО РІВНЯ «БАКАЛАВР»

Автоматизація процесів коворкінгу з використанням інтерактивних інформаційних систем

Виконав:
здобувач вищої освіти за ОПП «Інформаційні системи та технології» спеціальності 126 «Інформаційні системи та технології», групи ІСТ-41
Куль Мухаммад Давид

Керівник:
к.е.н., доцент Волошин В. С.

Рецензент:
к.т.н., доцент Гладка О. М.

Рівне – 2026

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет водного господарства
та природокористування
Навчально-науковий інститут кібернетики,
інформаційних технологій та інженерії

Кафедра комп'ютерних технологій та економічної кібернетики

Освітньо-кваліфікаційний рівень - бакалавр
Освітньо-професійна програма «Інформаційні системи і технології»
Спеціальність 126 «Інформаційні системи та технології»

ЗАТВЕРДЖУЮ
Завідувач кафедри
комп'ютерних технологій та
економічної кібернетики
д.е.н., професор П.М. Грицюк
_____
"__"_____ 2026 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Куль Мухаммад Давид
(прізвище, ім'я, по батькові)

1.Тема роботи: Автоматизація процесів коворкінгу з використанням інтерактивних інформаційних систем

керівник роботи: **Волошин Володимир Степанорвич, к.е.н., доцент**
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджена наказом по університету С № 476 від 25.04.2026.

2. Термін здачі студентом закінченої роботи *08.06.2025 року*

3. Вихідні дані до роботи
посібники, наукові статті, інтернет- джерела за темою роботи.

4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити) *:*

•провести аналіз предметної області та існуючих рішень у сфері автоматизації коворкінгів;
•визначити функціональні вимоги до інформаційної системи;
•спроєктувати логічну модель даних та структуру бази даних;
•реалізувати серверну частину системи для обробки запитів і взаємодії з базою даних;
•розробити клієнтську частину веб-додатку з використанням сучасних підходів до побудови інтерфейсу;
•реалізувати функціонал реєстрації, авторизації, бронювання робочих місць, перегляду історії бронювань і платежів, а також роботи з відгуками;
•передбачити роль адміністратора з можливістю базового управління даними системи;
•інтегрувати платіжний функціонал із використанням Stripe API;

5. Перелік графічного матеріалу

Презентація за матеріалами бакалаврської роботи.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
1	Волошин В.С., доцент кафедри КТЕК		
2	Волошин В.С., доцент кафедри КТЕК		

Дата видачі завдання 19 січня 2026 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської роботи	Термін виконання етапів (роботи)	Примітка
1	Робота над Розділом 1	20.01.2026-01.03.2026	
2	Робота над Розділом 2	02.03.2026-25.04.2026	
3	Підготовка пояснювальної записки	26.04.2026-25.05.2026	
4	Написання вступу, висновків, реферату	26.05.2026-08.06.2026	
5	Попередній захист роботи	09.06.2026	
6	Підготовка презентації роботи	10.06.2026	
7	Відгук керівника, рецензування роботи, перевірка на плагіат	11.06.2026	
8	Допуск до захисту	12.06.2026	

Студент(ка) _____ *Куль Мухаммад Д.*
(підпис) (прізвище і ініціали)

Керівник кваліфікаційної роботи
_____ *Волошин В.С.*
(підпис) (прізвище і ініціали)

ЗАЯВА
щодо самостійності виконання випускної кваліфікаційної роботи

Я, _____(ПІБ), студент(ка) _____курсу ___ групи ___ ННІ _____

ЗАЯВЛЯЮ:

моя випускна кваліфікаційна робота на тему «_____ _____», яка надається в екзаменаційну комісію із захисту кваліфікаційних робіт зі спеціальності «_____» для захисту, виконана самостійно і не містить плагіату.

Всі запозичення з друкованих та електронних джерел, у тому числі із захищених раніше випускових кваліфікаційних робіт мають відповідні посилання.

Я не використовував(ла) шахрайські методи маніпуляції з текстом (заміна букв, інтервали, мікропробіли, білі знаки, парафрази та ін.).

Інструменти штучного інтелекту використовував(ла) без порушення академічної доброчесності.

Я ознайомлений(а) з чинним Порядком перевірки навчальних, кваліфікаційних, навчально-методичних та наукових робіт на наявність ознак академічного плагіату в НУВГП та Положенням про академічну доброчесність в НУВГП, за яким виявлення плагіату є підставою для відмови в допуску моєї роботи до захисту та застосування відповідних санкцій (академічної відповідальності).

АКТ
перевірки випускної кваліфікаційної роботи
автора _____
на рівень запозичень та можливих маніпуляцій з текстом

Відповідно до даних системи StrikePlagiarism файл «_____»

містить: ____ % коефіцієнта подібності; ____% коефіцієнта цитованості; ____ замінених букв; ____ інтервалів; ____ мікропробілів; ____ білих знаків; ____ % ймовірність використання ШІ.

За результатами перевірки засвідчую, що:
автор кваліфікаційної роботи _____:

- самостійно виконав(ла) кваліфікаційну роботу;
- коректно посилався(лась) на використані інформаційні джерела;
- в роботі відсутні ознаки академічної недоброчесності.

Керівник кваліфікаційної роботи *підпис* _____

(ПІП)

Метадані

ДОКУМЕНТ

Заголовок

2026_126_bakalavrska_KulMukhammadD_VoloshynVS.docx

Автор	Науковий керівник / Експерт	ІД документу
Куль Мухаммад Давид _	**Куль Мухаммад Давид _**	**334155369**

ОРГАНІЗАЦІЯ

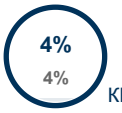
Назва організації	підрозділ
National University of Water and Environmental Engineering	**National University of Water and Environmental Engineering**

ЗВІТ

Дата звіту	Дата редагування
6/3/2026	**---**

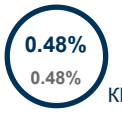
Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.



КП 1

7099
Кількість слів



КЦ

58102
Кількість символів

Тривога

У цьому розділі ви знайдете інформацію щодо текстових спотворень. Ці спотворення в тексті можуть говорити про МОЖЛИВІ маніпуляції в тексті. Спотворення в тексті можуть мати навмисний характер, але частіше характер технічних помилок при конвертації документа та його збереженні, тому ми рекомендуємо вам підходити до аналізу цього модуля відповідально. У разі виникнення запитань, просимо звертатися до нашої служби підтримки.

Заміна букв	ß	0
Інтервали	A→	0
Мікропробіли	·	12
Білі знаки	ß	0
Парафрази (SmartMarks)	a	18

Джерела

Нижче наведений список джерел. В цьому списку є джерела із різних баз даних. Колір тексту означає в якому джерелі він був знайдений. Ці джерела і значення Коефіцієнту Подібності не відображають прямого плагіату. Необхідно відкрити кожне джерело і проаналізувати зміст і правильність оформлення джерела.

10 найдовших фраз	Колір тексту

#	НАЗВА ТА АДРЕСА ДЖЕРЕЛА URL (НАЗВА БАЗИ)	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)

РЕФЕРАТ

Кваліфікаційна робота бакалавра: 60с., 26 рис., 5 табл., 15 літературних джерел.

Актуальність теми даної бакалаврської роботи полягає в тому, що в умовах розвитку цифрової економіки та зростання популярності гнучких форматів зайнятості (фріланс, віддалена робота, стартап-команди) коворкінги стають важливою частиною сучасної інфраструктури. Ефективне управління такими просторами потребує автоматизації ключових процесів, зокрема бронювання робочих місць, обліку користувачів, контролю оплат та збору зворотного зв'язку. Використання застарілих або розрізнених інструментів призводить до помилок, втрати даних і зниження якості обслуговування. У зв'язку з цим виникає необхідність у створенні сучасної інформаційної системи, яка забезпечує централізоване управління процесами коворкінгу та зручну взаємодію з користувачами.

Об'єктом дослідження бакалаврської роботи є процес організації та управління робочими місцями у коворкінгу, що включає взаємодію користувачів із сервісом бронювання, оплат та отримання послуг.

Предметною областю бакалаврської роботи є інформаційна система управління коворкінгом, зокрема її функціональні можливості, архітектура, структура бази даних, а також методи і засоби реалізації веб-додатку з інтуїтивним користувацьким інтерфейсом.

Метою бакалаврської роботи є проєктування та розробка інформаційної системи для автоматизації управління коворкінгом з використанням сучасних технологій веб-розробки, зокрема React, Node.js та PostgreSQL, а також забезпечення ефективної взаємодії користувачів із системою через зручний та адаптивний інтерфейс.

У бакалаврській роботі було: проведено аналіз предметної області та існуючих рішень; визначено основні функціональні вимоги до системи; розроблено логічну модель даних та структуру бази даних; реалізовано серверну та клієнтську частини веб-додатку; створено інтерфейс користувача

з можливістю реєстрації, авторизації, бронювання робочих місць, перегляду історії бронювань і платежів, а також роботи з відгуками; реалізовано роль адміністратора, яка передбачає базове управління системою, зокрема перегляд і редагування користувачів, керування робочими місцями та тарифами, а також контроль бронювань; забезпечено коректну взаємодію між компонентами системи та перевірено її працездатність на типових сценаріях використання.

КЛЮЧОВІ СЛОВА: ІНФОРМАЦІЙНА СИСТЕМА, КОВОРКІНГ, ВЕБ-ДОДАТОК, БРОНЮВАННЯ, БАЗА ДАНИХ, КОРИСТУВАЦЬКИЙ ІНТЕРФЕЙС, REACT, NODE.JS, POSTGRESQL.

ЗМІСТ

ВСТУП

У сучасних умовах розвитку цифрової економіки та поширення гнучких форматів зайнятості (фріланс, віддалена робота, стартап-команди) коворкінги стають важливою складовою інфраструктури бізнес-середовища. Вони забезпечують користувачам доступ до робочих місць, ресурсів і сервісів без необхідності довгострокової оренди офісів. Разом із цим зростає потреба в ефективному управлінні такими просторами, що включає бронювання робочих місць, облік користувачів, контроль платежів та організацію зворотного зв’язку.

У багатьох випадках управління коворкінгами здійснюється за допомогою розрізнених інструментів, таких як таблиці, месенджери або календарі, що призводить до помилок, дублювання інформації та ускладнення контролю за завантаженістю ресурсів. У зв’язку з цим актуальним є створення централізованої інформаційної системи, яка автоматизує основні бізнес-процеси коворкінгу та забезпечує зручну взаємодію користувачів із сервісом через веб-інтерфейс.

Сучасні веб-технології дозволяють реалізувати такі системи з високим рівнем зручності, продуктивності та масштабованості. Використання клієнт-серверної архітектури, інтерактивних інтерфейсів та надійних систем керування базами даних забезпечує стабільну роботу сервісу та можливість його подальшого розвитку.

Метою бакалаврської роботи є проєктування та розробка інформаційної системи для автоматизації управління коворкінгом з використанням сучасних технологій веб-розробки, а також створення зручного, інтуїтивно зрозумілого користувацького інтерфейсу.

Об’єктом дослідження є процес організації та управління робочими місцями у коворкінгу.

Предметом дослідження є інформаційна система управління коворкінгом, її функціональні можливості, архітектура, структура бази даних та принципи взаємодії користувачів із системою.

Завданнями бакалаврської роботи є:

- провести аналіз предметної області та існуючих рішень у сфері автоматизації коворкінгів;
- визначити функціональні та нефункціональні вимоги до інформаційної системи;
- спроєктувати архітектуру системи, логічну модель даних та структуру бази даних;
- розробити серверну та клієнтську частини вебзастосунку із реалізацією основних функцій управління коворкінгом;
- реалізувати механізми автентифікації користувачів, бронювання робочих місць, обробки платежів та адміністрування системи;
- провести тестування розробленої інформаційної системи та оцінити результати її функціонування.

Інструменти, використані під час виконання роботи: React, Node.js, PostgreSQL, JavaScript, HTML/CSS, SCSS, Axios, Visual Studio Code, Git, а також Stripe API для реалізації платіжного функціоналу.

РОЗДІЛ 1. ІНФОРМАЦІЙНА ПІДТРИМКА ОРГАНІЗАЦІЇ КОВОРКІНГУ

1.1. Поняття коворкінгу

Коворкінг (від англ. *coworking* — «спільна робота») — це форма організації робочого простору, за якої незалежні спеціалісти, підприємці, фрілансери або команди працюють у спільному середовищі, використовуючи загальну інфраструктуру та ресурси. На відміну від традиційних офісів, коворкінг передбачає гнучкі умови користування, що дозволяє орендувати робоче місце на короткий або довготривалий період.

Сучасні коворкінги поєднують у собі не лише фізичний простір для роботи, але й екосистему взаємодії між користувачами, що сприяє обміну досвідом, створенню нових проєктів і підвищенню продуктивності.

Основні характеристики коворкінгу

Коворкінг як форма організації праці має низку ключових характеристик:

- **Гнучкість використання** — можливість бронювання робочих місць на різні періоди (години, дні, місяці);
- **Спільна інфраструктура** — користування спільними ресурсами (інтернет, переговорні кімнати, офісна техніка);
- **Оптимізація витрат** — відсутність необхідності орендувати повноцінний офіс;
- **Соціальна взаємодія** — можливість комунікації між користувачами;
- **Доступність** — широкий вибір тарифів і форматів користування.

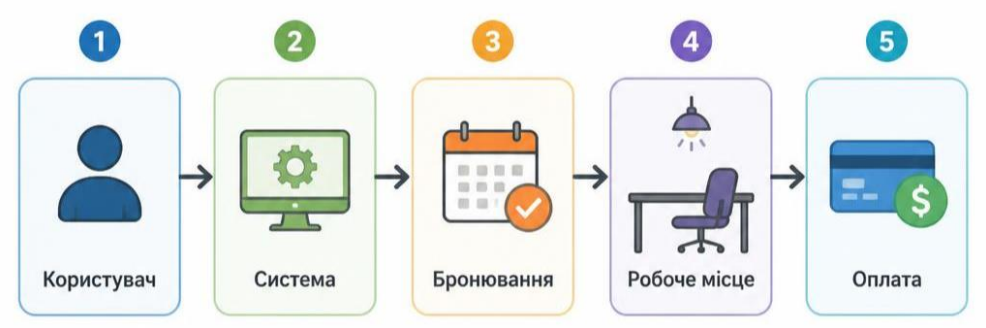


Рис. 1.1 – Загальна схема функціонування коворкінгу

Типи коворкінгів

Залежно від організації простору та цільової аудиторії виділяють кілька основних типів коворкінгів:

Таблиця 1.1 – Класифікація коворкінгів

Тип коворкінгу	Характеристика
Відкритий простір	Загальна зона з робочими місцями без фіксованих позицій
Приватні офіси	Окремі кімнати для команд або компаній
Спеціалізовані	Орієнтовані на певну сферу (IT, дизайн, стартапи)
Гібридні	Поєднання відкритих зон і приватних приміщень

Основні компоненти коворкінгу як системи

Коворкінг можна розглядати як складну систему, що включає кілька взаємопов’язаних компонентів:

- **Користувачі** — особи, які користуються послугами коворкінгу;
- **Робочі місця** — фізичні ресурси (столи, кімнати);
- **Тарифи** — моделі оплати за користування;
- **Бронювання** — процес резервування ресурсів;
- **Платежі** — фінансові операції;
- **Адміністрування** — управління всією системою.

Рис. 1.2 – Структурна модель коворкінгу як інформаційної системи

Бізнес-процеси у коворкінгу

Функціонування коворкінгу базується на наборі типових бізнес-процесів:

Таблиця 1.2 – Основні бізнес-процеси коворкінгу

Процес	Опис
Реєстрація користувача	Створення облікового запису
Бронювання місця	Вибір робочого місця та часу
Оплата послуг	Проведення фінансових операцій
Управління тарифами	Налаштування цін і планів
Зворотний зв’язок	Отримання відгуків від користувачів

Переваги використання коворкінгу

Використання коворкінгів має ряд переваг як для користувачів, так і для власників:

- зменшення витрат на інфраструктуру;
- підвищення гнучкості робочого процесу;
- зручність доступу до ресурсів;

- можливість масштабування;
- формування професійного середовища.

Проблеми та обмеження

Незважаючи на переваги, існують і певні недоліки:

- складність управління великою кількістю бронювань;
- ризик дублювання записів;
- необхідність постійного контролю завантаженості;
- залежність від цифрових систем.

Роль інформаційних систем у коворкінгах

Для вирішення зазначених проблем використовуються інформаційні системи, які забезпечують автоматизацію ключових процесів. Такі системи дозволяють:

- централізовано зберігати дані;
- автоматично обробляти бронювання;
- контролювати оплату послуг;
- надавати користувачам зручний інтерфейс;
- забезпечувати адміністративний контроль.

Таким чином, коворкінг є складною організаційно-технічною системою, ефективне функціонування якої неможливе без використання сучасних інформаційних технологій. Автоматизація процесів дозволяє підвищити якість обслуговування, зменшити кількість помилок і забезпечити стабільність роботи сервісу.

1.2. Порівняльний аналіз серверних фреймворків на платформі Node.js та Laravel

У процесі розробки сучасних веб-додатків важливим етапом є вибір серверної технології, яка забезпечує обробку запитів, взаємодію з базою даних, бізнес-логіку та безпеку системи. Серед найбільш популярних підходів до серверної розробки сьогодні виділяються платформа **Node.js** та PHP-фреймворк **Laravel**.

Обидві технології широко використовуються для створення веб-систем різного рівня складності, однак вони базуються на різних концепціях, архітектурних підходах і технологічних стеків. Це робить доцільним проведення їх порівняльного аналізу для обґрунтування вибору технології у межах даної роботи.

Рис. 1.3 – Порівняння архітектури Node.js та Laravel

Загальна характеристика Node.js

Node.js — це серверна платформа, що дозволяє виконувати JavaScript поза браузером. Вона побудована на рушії V8 і використовує асинхронну неблокуючу модель введення-виведення.

Основні особливості:

- асинхронна обробка запитів;
- event-driven архітектура;
- висока продуктивність при роботі з великою кількістю запитів;
- єдина мова програмування (JavaScript) для frontend і backend

Node.js часто використовується разом із фреймворками, такими як:

- Express.js
- NestJS

Загальна характеристика Laravel

Laravel — це популярний PHP-фреймворк, який реалізує архітектуру MVC (Model-View-Controller) та надає готові інструменти для швидкої розробки веб-додатків.

Основні особливості:

- чітка структура проєкту;
- вбудована система маршрутизації;
- ORM (Eloquent);
- система автентифікації;
- зручна робота з базами даних;
- велика кількість готових рішень "з коробки".

Laravel орієнтований на швидку розробку класичних веб-додатків.

Архітектурні відмінності

Node.js використовує неблокуючу модель обробки запитів, що дозволяє обслуговувати тисячі одночасних підключень без створення окремого потоку для кожного користувача.

Laravel працює за класичною синхронною моделлю:

- кожен HTTP-запит обробляється окремо;
- використовується MVC;
- логіка чітко розділена.

Ключова різниця:

- Node.js — продуктивність і масштабованість
- Laravel — структурованість і швидкість розробки

Продуктивність та масштабованість

Node.js демонструє високу продуктивність у системах із великою кількістю одночасних користувачів:

- реальний час (чати, бронювання);
- API-сервіси;
- стрімінг.

Laravel поступається у високонавантажених системах, але добре підходить для:

- корпоративних систем;
- CRM;
- класичних веб-додатків.

Порівняння підходів до обробки запитів

Однією з ключових відмінностей між Node.js та Laravel є підхід до обробки HTTP-запитів, що безпосередньо впливає на продуктивність, масштабованість та поведінку системи під навантаженням.

У середовищі Node.js використовується асинхронна неблокуюча модель, яка базується на механізмі event loop. Це означає, що сервер не створює окремий потік для кожного запиту, а обробляє їх у межах одного потоку, делегуючи операції введення-виведення у фонові процеси. Такий підхід дозволяє ефективно працювати з великою кількістю одночасних підключень, що є критично важливим для систем із високим навантаженням.

Зокрема, при виконанні операцій, пов’язаних із базою даних або зовнішніми API, Node.js не блокує виконання основного потоку, а продовжує обробляти інші запити. Після завершення асинхронної операції викликається callback або Promise, що забезпечує повернення результату.

У Laravel використовується традиційна синхронна модель обробки запитів. Кожен HTTP-запит обробляється окремо, і під час виконання операцій, таких як звернення до бази даних, потік виконання блокується до отримання результату. Це може призводити до зниження продуктивності при великій кількості одночасних користувачів.

Проте така модель є більш зрозумілою та передбачуваною, що спрощує розробку та відлагодження системи. Вона добре підходить для додатків із помірним навантаженням, де важливіша стабільність і простота реалізації.

Таким чином, Node.js забезпечує більш ефективну обробку великої кількості запитів, тоді як Laravel пропонує більш традиційний і простий підхід до розробки серверної логіки.

Зручність розробки

Laravel має перевагу:

- багато готових рішень;
- менше коду "з нуля";
- швидкий старт.

Node.js:

- потребує більше налаштування;
- більше контролю;
- гнучкість.

Підтримка сучасних архітектурних підходів

Сучасні веб-додатки все частіше будуються на основі архітектурних підходів, таких як мікросервіси, REST API та SPA (Single Page Application). Вибір серверної технології значною мірою впливає на можливість реалізації таких підходів.

Node.js спочатку орієнтований на побудову API-сервісів. Він широко використовується для створення RESTful API, що взаємодіють із клієнтськими застосунками, розробленими на React, Angular або Vue. Завдяки використанню єдиної мови програмування (JavaScript) як на клієнті, так і на сервері, значно спрощується процес розробки та підтримки системи.

Крім того, Node.js добре підходить для побудови мікросервісної архітектури. Його легковаговість та модульність дозволяють розбивати систему на незалежні компоненти, які можуть масштабуватися окремо.

Laravel також підтримує побудову API, однак його основна орієнтація — це монолітні веб-додатки. Хоча з використанням додаткових інструментів можливо реалізувати мікросервісну архітектуру, це потребує більше зусиль і додаткової конфігурації.

У контексті даної роботи, де клієнтська частина реалізована на React, використання Node.js є більш логічним рішенням, оскільки забезпечує нативну інтеграцію з frontend та спрощує побудову API.

Робота з базами даних

Laravel:

- Eloquent ORM;
- прості запити;
- мінімум SQL.

Node.js:

- використання ORM (Sequelize, Prisma);
- або чистий SQL;
- більше контролю, але складніше.

Безпека

Laravel:

- захист від SQL injection;
- CSRF protection;
- готова система auth.

Node.js:

- безпека залежить від розробника;
- більше ризиків при неправильній реалізації.

Екосистема та популярність

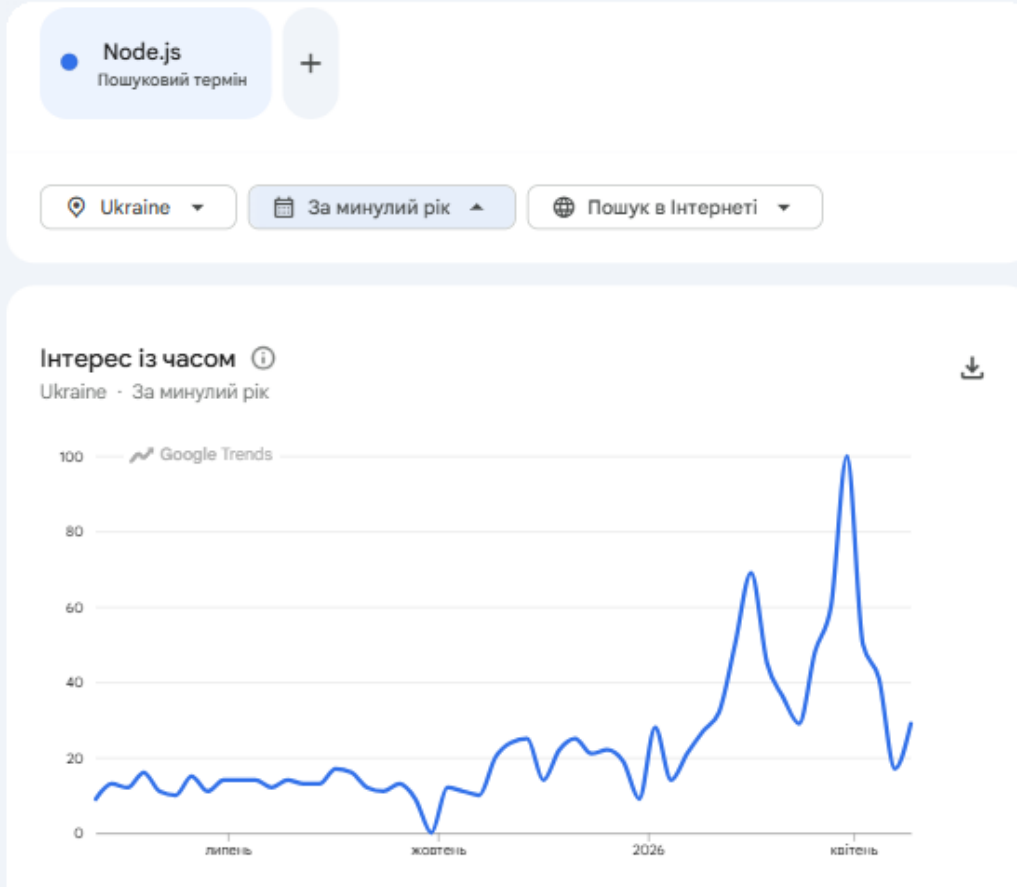


Рис. 1.4 – Популярність Node.js (usage statistics)

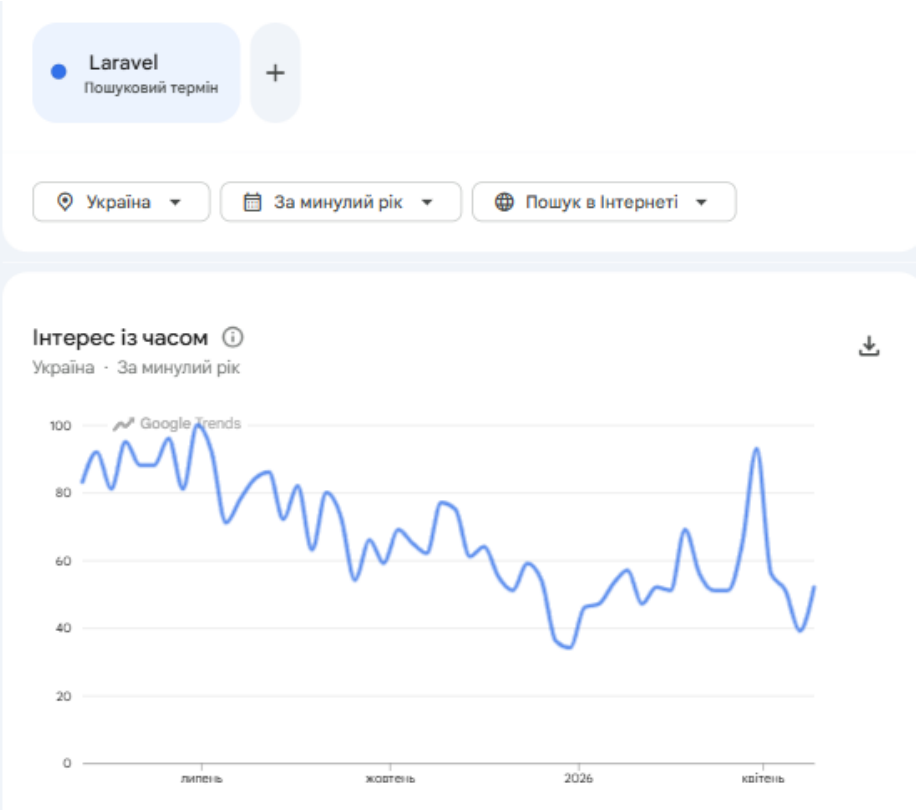


Рис. 1.5 – Популярність Laravel (usage statistics)

Порівняльна таблиця

Критерій	**Node.js**	**Laravel**
Мова	JavaScript	PHP
Архітектура	Event-driven	MVC
Продуктивність	Висока	Середня
Масштабованість	Висока	Обежена
Простота	Середня	Висока
Безпека	Залежить від dev	Вбудована
Гнучкість	Висока	Середня

Таблиця 1.3 – Порівняння Node.js та Laravel

У межах виконання курсового та дипломного проєкту було використано платформу Node.js для реалізації серверної частини системи управління коворкінгом.

Практичний досвід показав такі переваги:

- швидка обробка запитів;
- зручна робота з REST API;
- хороша інтеграція з React;
- можливість масштабування.

Також були виявлені недоліки:

- необхідність самостійної реалізації багатьох компонентів;
- більша складність у порівнянні з Laravel на початковому етапі.

Обґрунтування вибору технології

Для реалізації інформаційної системи управління коворкінгом було обрано **Node.js**, оскільки:

- система передбачає часту взаємодію з сервером;
- необхідна висока продуктивність;
- використовується React (єдина мова — JavaScript);
- потрібна масштабованість;
- планується інтеграція платіжних сервісів (Stripe API).

Інтеграція з зовнішніми сервісами

Сучасні інформаційні системи часто потребують інтеграції з зовнішніми сервісами, такими як платіжні системи, сервіси авторизації, email-розсилки та інші API.

Node.js має значну перевагу у цьому аспекті завдяки своїй орієнтації на роботу з асинхронними запитами та великій кількості бібліотек у npm. Інтеграція з такими сервісами, як Stripe, Firebase або сторонні REST API, реалізується швидко та з мінімальними витратами.

Laravel також надає можливості інтеграції, проте вони часто потребують використання додаткових пакетів та більш складної конфігурації. Водночас Laravel має добре документовані рішення для типових задач, що спрощує інтеграцію для розробників із досвідом роботи з PHP.

У рамках даної роботи передбачається інтеграція платіжної системи Stripe, що дозволить реалізувати функціонал онлайн-оплати. У цьому випадку Node.js забезпечує більш гнучкий та швидкий підхід до реалізації взаємодії з API платіжного сервісу.

Висновки до підрозділу:

У результаті проведеного аналізу встановлено, що Node.js та Laravel мають різні сфери застосування та переваги. Laravel є зручним для швидкої розробки структурованих веб-додатків, тоді як Node.js забезпечує високу продуктивність і гнучкість.

З урахуванням вимог до системи управління коворкінгом, зокрема необхідності масштабованості, швидкої обробки запитів та інтеграції з сучасними веб-технологіями, вибір Node.js є обґрунтованим.

1.3. Дослідження вебплатформ коворкінгів України

Розвиток коворкінг-індустрії в Україні супроводжується активною цифровізацією процесів управління робочими просторами. Сучасні коворкінги дедалі частіше використовують веб-платформи для автоматизації бронювання, обліку користувачів, управління тарифами та взаємодії з клієнтами. Це дозволяє зменшити навантаження на адміністративний персонал, мінімізувати кількість помилок і підвищити загальну ефективність сервісу.

У ході аналізу було розглянуто як міжнародні, так і локальні рішення, що застосовуються у сфері коворкінгів, з метою визначення їхніх переваг, недоліків та рівня функціональної зрілості.

Міжнародні платформи коворкінгів

Одним із найвідоміших прикладів є глобальна платформа **WeWork platform**, яка забезпечує комплексне управління робочими просторами. Вона надає користувачам можливість бронювання робочих місць, переговорних кімнат та офісів через веб-інтерфейс або мобільний застосунок. Система підтримує інтеграцію з календарями, автоматизовані платежі та персоналізовані пропозиції для користувачів.

Основною перевагою WeWork є високий рівень автоматизації та продумана екосистема сервісів. Водночас система орієнтована переважно на великі корпоративні клієнти, що робить її менш доступною для малого бізнесу або локальних коворкінгів.

Іншим прикладом є платформа **Regus platform**, яка спеціалізується на оренді офісних просторів і коворкінгів у різних країнах світу. Regus пропонує гнучкі тарифні плани, можливість онлайн-бронювання та масштабовану інфраструктуру для корпоративних клієнтів.

Попри високу функціональність, дана система має обмеження з точки зору гнучкості UX/UI та локалізації під конкретні ринки. Вона більше орієнтована на стандартизовані бізнес-процеси, ніж на індивідуальні сценарії використання.

Українські рішення у сфері коворкінгів

На українському ринку переважають менш складні цифрові рішення. Більшість коворкінгів використовують або власні веб-сайти з базовим функціоналом, або сторонні сервіси бронювання, які не завжди адаптовані під специфіку локального ринку.

Типовими інструментами є прості веб-форми бронювання, календарні сервіси або CRM-системи загального призначення. Такі рішення забезпечують лише базову функціональність і часто не покривають повний цикл взаємодії з користувачем.

Основними недоліками існуючих українських рішень є:

- відсутність єдиної централізованої платформи;

- обмежені можливості управління бронюваннями;
- слабка або відсутня інтеграція платіжних систем;
- низький рівень автоматизації процесів;
- недостатньо розвинений користувацький інтерфейс;
- відсутність аналітики використання ресурсів.

Порівняльний аналіз платформ На основі проведеного дослідження сформовано узагальнююче порівняння найбільш характерних платформ за ключовими критеріями функціонування.

Критерій	**WeWork**	**Regus**	**Локальні українські рішення**
Бронювання робочих місць	Повністю автоматизоване	Автоматизоване	Частково або вручну
Платіжна система	Інтегрована	Інтегрована	Часто відсутня або зовнішні сервіси
Користувацький інтерфейс	Сучасний, адаптивний	Стандартний корпоративний	Спрощений, іноді застарілий
Масштабованість	Висока	Висока	Низька
Локалізація під ринок	Обмежена	Обмежена	Висока
Аналітика та звітність	Розширена	Базова/середня	Мінімальна або відсутня
Автоматизація процесів	Повна	Часткова	Обмежена
Орієнтація на користувача	Індивідуальні та корпоративні клієнти	Переважно корпоративні	Локальні користувачі

Таблиця 1.4 – Порівняльний аналіз веб-платформ коворкінгів

Висновки

Проведений аналіз показав, що сучасні веб-платформи коворкінгів мають суттєво різний рівень функціональної зрілості залежно від цільового ринку. Міжнародні рішення забезпечують високий рівень автоматизації, але є складними та часто недоступними для малого бізнесу. Локальні українські рішення, у свою чергу, залишаються простими, але функціонально обмеженими.

Таким чином, існує чіткий розрив між корпоративними системами та базовими локальними рішеннями, що обґрунтовує необхідність розробки проміжної інформаційної системи. Така система повинна поєднувати:

- зручний сучасний інтерфейс;
- централізоване управління даними;
- автоматизацію бронювань;
- підтримку фінансових операцій;
- можливість подальшого масштабування.

РОЗДІЛ 2. РОЗРОБКА ІНТЕРАКТИВНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ КОВОРКІНГУ

2.1. Проєктування логічної моделі даних для інформаційної системи на базі PostgreSQL

Логічне проєктування бази даних є одним із ключових етапів створення інформаційної системи коворкінгу, оскільки саме на цьому рівні визначається структура зберігання даних, взаємозв’язки між сутностями та правила цілісності інформації. Коректно спроєктована модель даних забезпечує стабільну роботу системи, зменшує надмірність даних та спрощує подальший розвиток функціоналу.

Розроблена інформаційна система призначена для автоматизації процесів управління коворкінг-простором і охоплює повний цикл взаємодії користувачів із сервісом. До основних функцій належать реєстрація та автентифікація користувачів, перегляд доступних робочих місць, створення та керування бронюваннями, здійснення оплат, а також формування відгуків про якість обслуговування. Окремо реалізовано адміністративний модуль, який забезпечує керування всіма сутностями системи та надання аналітичної інформації.

Архітектура системи побудована на основі клієнт-серверної моделі, де клієнтська частина реалізована за допомогою React, серверна — на базі Express.js, а система зберігання даних — PostgreSQL. Такий підхід дозволяє забезпечити гнучкість, масштабованість та високу продуктивність обробки запитів.

На логічному рівні модель даних включає набір взаємопов’язаних сутностей, які відображають предметну область:

- користувачі (users);

- робочі місця (workspaces);
- тарифні плани (tariffs);
- бронювання (bookings);
- платежі (payments);
- відгуки (reviews);
- коментарі адміністратора до відгуків (review_comments).

Кожна з наведених сутностей виконує окрему функціональну роль у системі та містить набір атрибутів, що характеризують відповідні об’єкти предметної області (табл. 2.1).

Логічна модель бази даних складається із семи взаємопов’язаних таблиць, які забезпечують реалізацію всіх функціональних можливостей інформаційної системи коворкінгу. На відміну від базової моделі, структура даних була розширена таблицею «КоментаріДоВідгуків», що дозволяє адміністраторам відповідати на відгуки користувачів безпосередньо через систему. Такий підхід забезпечує централізоване управління інформацією, підтримує цілісність даних та сприяє підвищенню якості комунікації між користувачами та адміністрацією коворкінгу.

Типи зв'язків між таблицями

Для забезпечення взаємодії між сутностями інформаційної системи у базі даних реалізовано систему зв'язків між таблицями. Використання зв'язків дозволяє підтримувати цілісність даних, виключати дублювання інформації та забезпечувати коректну роботу всіх функціональних модулів системи.

У розробленій базі даних переважно використовуються зв'язки типу **«один-до-багатьох» (1:N):**

- **Користувачі → Бронювання**

  Один користувач може створювати необмежену кількість бронювань протягом роботи із системою. Водночас кожне бронювання належить лише одному користувачеві.

Назва таблиці	Назви полів
Користувачі	**– КодКористувача** – Ім’я – Прізвище – ЕлектроннаПошта – ХешПароля – Роль (користувач, адміністратор)
Робочі місця	**– КодМісця** – НомерМісця – ТипМісця (стандартне, преміум, переговорна кімната) – Статус (доступне, заброньоване, технічне обслуговування)
Тарифи	**– КодТарифу** – ТипТарифу (standard, premium, meeting_room) – НазваТарифу – Ціна – Іконка – Опис
Бронювання	**– КодБронювання** – КодКористувача – КодМісця – КодТарифу – ДатаПочатку – ДатаЗакінчення – СтатусОплати (неоплачено, оплачено, очікується) – НомерМісця – Вартість – Скасовано (так, ні)
Оплати	**– КодОплати** – КодБронювання – ДатаОплати – Сума – МетодОплати (картка, готівка, банківський переказ)
Відгуки	**– КодВідгуку** – КодКористувача – ДатаВідгуку – ТекстВідгуку – Оцінка (1–5)

КоментаріДоВідгуків	**– КодКоментаря** – КодВідгуку – КодАдміністратора – ТекстКоментаря – ДатаСтворення

Таблиця 2.1 – Сутності та атрибути бази даних

- **РобочіМісця → Бронювання**

  Одне робоче місце може використовуватись у багатьох бронюваннях у різні часові проміжки. При цьому кожне бронювання пов'язане лише з одним робочим місцем.

- **Тарифи → Бронювання**

  Один тарифний план може застосовуватися для багатьох бронювань різних користувачів. Кожне бронювання використовує лише один тариф.

- **Бронювання → Оплати**

  Кожне бронювання може мати один або декілька записів про оплату залежно від особливостей проведення фінансових операцій. Кожен платіж належить конкретному бронюванню.

- **Користувачі → Відгуки**

  Один користувач може залишати декілька відгуків про роботу коворкінгу. Кожен відгук пов'язаний лише з одним користувачем.

- **Відгуки → КоментаріДоВідгуків**

  На один відгук може бути надано декілька коментарів адміністрації. Кожен коментар належить лише одному відгуку.

- **Адміністратори → КоментаріДоВідгуків**

  Один адміністратор може залишати відповіді на різні відгуки користувачів. Кожен коментар створюється одним адміністратором.

Для реалізації зв'язків між таблицями використовуються зовнішні ключі (**Foreign Keys**), які забезпечують підтримання посилальної цілісності даних.

Наприклад:

- у таблиці **Бронювання** поле **КодКористувача** є зовнішнім ключем, що посилається на таблицю **Користувачі**;
- поле **КодМісця** у таблиці **Бронювання** пов'язує бронювання з відповідним робочим місцем із таблиці **РобочіМісця**;
- поле **КодТарифу** у таблиці **Бронювання** посилається на таблицю **Тарифи**;
- поле **КодБронювання** у таблиці **Оплати** забезпечує зв'язок між платежем та відповідним бронюванням;
- поле **КодКористувача** у таблиці **Відгуки** визначає автора відгуку;
- поле **КодВідгуку** у таблиці **КоментаріДоВідгуків** забезпечує зв'язок коментаря з відповідним відгуком;
- поле **КодАдміністратора** у таблиці **КоментаріДоВідгуків** визначає адміністратора, який залишив відповідь на відгук.

Запропонована структура зв'язків забезпечує:

- цілісність та узгодженість даних;
- мінімізацію дублювання інформації;
- ефективне виконання запитів до бази даних;
- коректне функціонування механізмів бронювання та оплати;
- можливість розширення функціоналу системи без суттєвої зміни існуючої структури бази даних;
- підтримку адміністративного модуля, аналітичного дашборду та системи роботи з відгуками користувачів.

Інформаційна система коворкінгу складається із семи взаємопов'язаних сутностей, які забезпечують збереження, обробку та аналіз даних. На **рисунку 2.1** представлено логічну модель даних системи, що відображає взаємозв'язки між користувачами, робочими місцями, тарифними планами, бронюваннями, платежами, відгуками та коментарями адміністраторів.

Рисунок 2.1 – ER-діаграма логічної моделі даних

Як видно зі схеми, користувачі можуть створювати бронювання, здійснювати оплату послуг та залишати відгуки. Кожне бронювання пов'язане з конкретним робочим місцем і тарифним планом, що дозволяє здійснювати точний облік використання ресурсів коворкінгу. Адміністратори мають можливість переглядати відгуки користувачів та залишати до них коментарі, забезпечуючи ефективний зворотний зв'язок.

Розроблена логічна модель даних формує основу для стабільної роботи інформаційної системи, підтримує всі основні бізнес-процеси коворкінгу та забезпечує можливість подальшого розвитку програмного забезпечення без суттєвого ускладнення структури бази даних.

Нормалізація структури бази даних

Під час проєктування логічної моделі даних особлива увага приділялася нормалізації структури бази даних. Використання принципів нормалізації дозволило усунути надлишковість інформації, зменшити ризик виникнення аномалій при додаванні, оновленні або видаленні даних та підвищити загальну ефективність роботи системи.

Структура бази даних відповідає вимогам третьої нормальної форми (3NF), оскільки кожна таблиця описує окрему сутність предметної області, усі атрибути залежать від первинного ключа, а між неключовими атрибутами відсутні транзитивні залежності.

Наприклад, інформація про тарифні плани винесена в окрему таблицю «Тарифи», що дозволяє використовувати один тариф для багатьох бронювань без дублювання його характеристик. Аналогічно інформація про робочі місця зберігається окремо від таблиці бронювань, що забезпечує централізоване управління ресурсами коворкінгу.

Застосування нормалізації також спрощує підтримку системи та створює можливість подальшого розширення функціоналу без необхідності істотної зміни існуючої структури даних.

Забезпечення цілісності та безпеки даних

Для гарантування коректності інформації в базі даних використовується комплекс механізмів контролю цілісності.

Кожна таблиця містить первинний ключ, який забезпечує унікальну ідентифікацію записів. Для встановлення зв'язків між сутностями використовуються зовнішні ключі, які запобігають появі неузгоджених даних та забезпечують посилальну цілісність.

Додатково в базі даних реалізовано систему обмежень типу CHECK, які контролюють допустимі значення окремих атрибутів. Наприклад, для поля ролі користувача допускаються лише значення «user» та «admin», для оцінки відгуку — значення від 1 до 5, а для статусів бронювання, тарифів та робочих місць визначено фіксовані набори допустимих значень.

Для захисту персональних даних користувачів у системі не зберігаються паролі у відкритому вигляді. Замість цього використовується механізм хешування паролів перед записом до бази даних, що підвищує рівень інформаційної безпеки та відповідає сучасним вимогам до веб-застосунків.

Оптимізація доступу до даних

Для підвищення продуктивності роботи інформаційної системи в базі даних створено індекси для найбільш використовуваних полів. Зокрема, індексація застосовується для електронних адрес користувачів, ідентифікаторів робочих місць, бронювань та інших атрибутів, які часто використовуються під час пошуку та фільтрації інформації.

Необхідність використання індексів обумовлена наявністю функцій пошуку, сортування та фільтрації, реалізованих як у користувацькій частині системи, так і в адміністративній панелі. Завдяки цьому забезпечується швидке отримання інформації навіть при збільшенні обсягу даних.

Крім того, реалізована структура бази даних дозволяє ефективно формувати аналітичні запити для відображення статистичної інформації на адміністративному дашборді. Адміністратор може отримувати дані про кількість користувачів, бронювань, платежів, відгуків та інші показники функціонування коворкінгу в режимі реального часу.

Висновки до підрозділу

У результаті виконаного логічного проєктування було створено структуру бази даних, яка повністю відображає бізнес-процеси інформаційної системи управління коворкінгом. Модель включає сім взаємопов'язаних сутностей, що забезпечують підтримку процесів реєстрації користувачів, управління робочими місцями, бронювання ресурсів, здійснення оплат та збору відгуків.

Запропонована структура забезпечує високий рівень цілісності даних, підтримує розмежування прав доступу між користувачами та адміністраторами, а також створює основу для ефективної роботи як користувацького інтерфейсу, так і адміністративної панелі. Використання PostgreSQL дозволило реалізувати надійне зберігання інформації та створити масштабовану архітектуру, придатну для подальшого розвитку системи.

2.2. Програмна реалізація інтерфейсів засобами бібліотеки React

Загальна архітектурна модель клієнтської частини

Клієнтська частина вебзастосунку реалізована як **Single Page Application (SPA)** на базі **React + TypeScript**. Основною метою архітектурного проєктування було забезпечення стабільної масштабованої структури інтерфейсу з чітким розділенням:

- презентаційного рівня (UI)
- логіки стану (state management)
- рівня взаємодії з API
- системи маршрутизації та доступу

Архітектура побудована відповідно до **component-driven design**, де кожен інтерфейсний елемент є незалежною одиницею з власним станом і поведінкою.

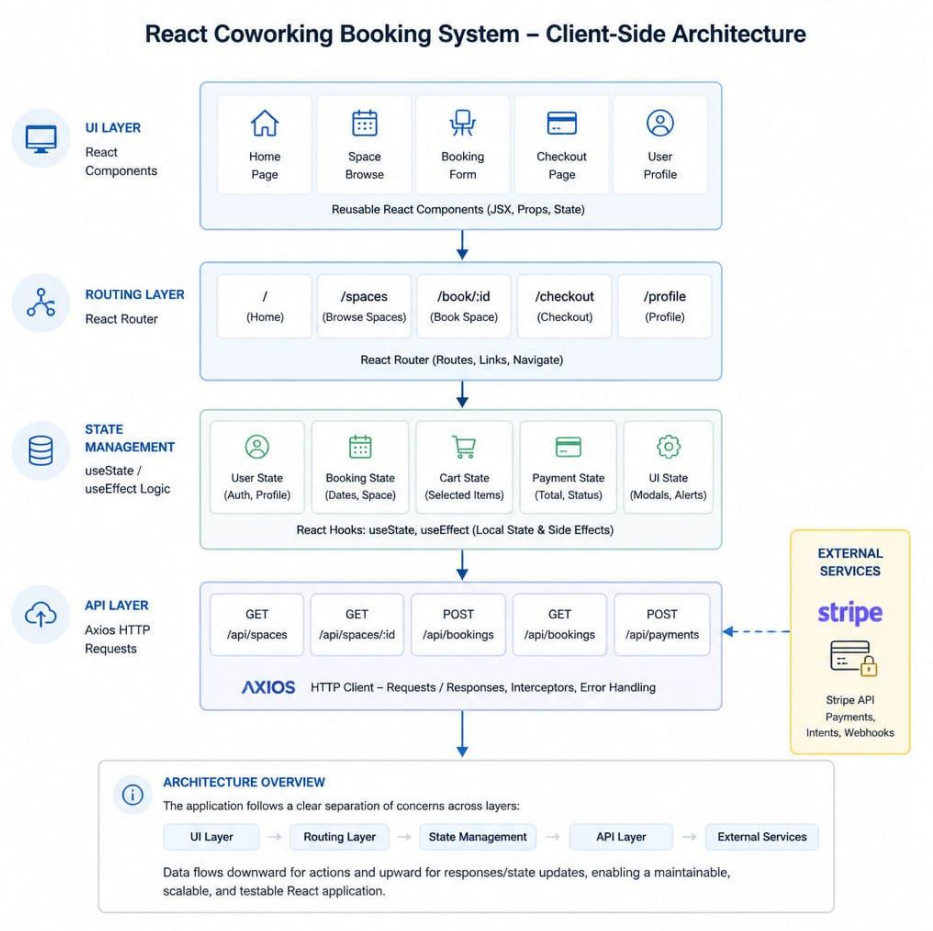


Рисунок 2.2 – Узагальнена схема клієнтської архітектури React застосунку

Рівнева структура інтерфейсу

Інтерфейс системи розділено на три логічні рівні:

1. Рівень сторінок (Pages Layer)

Сторінки є контейнерним рівнем і визначають основний маршрут користувача в системі. Вони не містять складної бізнес-логіки, а агрегують секції.

Приклади:

- Home
- Login / SignUp
- Dashboard
- Workspaces
- Admin панель

Функціонально сторінки відповідають за композицію UI.

---

2. Рівень функціональних секцій (Sections Layer)

Секції є основною одиницею побудови інтерфейсу. Кожна сторінка складається з набору секцій.

Наприклад:

- Home → Hero, Services, Reviews
- About → Mission, Team, Values
- Dashboard → Statistics, Recent activity

Цей підхід забезпечує:

- повторне використання UI блоків
- ізоляцію логіки
- незалежність секцій

---

3. Рівень базових компонентів (UI Layer)

Базові компоненти є універсальними елементами інтерфейсу:

- Header

- Footer
- Loader
- ScrollToTop

Вони використовуються глобально і не прив’язані до конкретної сторінки.

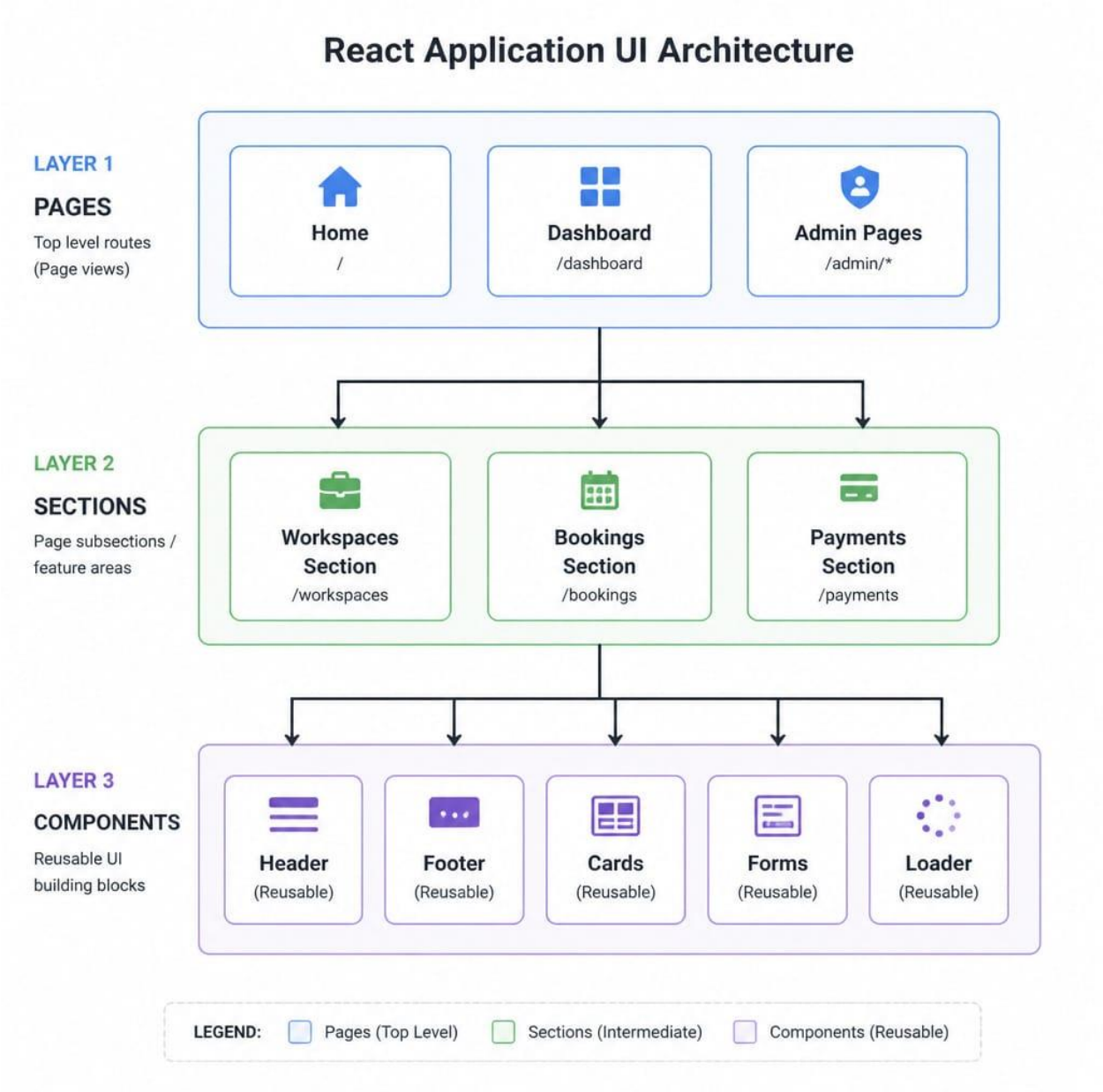


Рисунок 2.3 – Рівнева модель UI (Pages → Sections → Components)

Організація файлової структури

Структура проєкту побудована відповідно до доменно-компонентного поділу:

- pages/ — маршрутизовані екрани
- components/ — повторно використовувані UI елементи
- components/sections/ — функціональні блоки сторінок
- styles/ — SCSS стилі з модульною структурою
- router.tsx — конфігурація маршрутизації
- assets/ — статичні ресурси

Такий поділ мінімізує зв’язність між модулями та спрощує масштабування системи.

Система маршрутизації та контроль доступу

Маршрутизація реалізована за допомогою **React Router v6 (createBrowserRouter)** і базується на принципі **role-based routing**.

Основні типи маршрутів:

- Public routes — без авторизації
- Protected routes — для авторизованих користувачів
- Admin routes — для адміністратора системи

Логічна модель доступу

Доступ до сторінок визначається на основі:

- наявності JWT токена
- ролі користувача (user/admin)

```
const getUser = () => {
  try {
    const raw = localStorage.getItem("user");
    return raw ? JSON.parse(raw) : null;
  } catch {
    return null;
  }
};
```

Принцип роботи layout-ів

- **PublicLayout** — виконує редірект авторизованих користувачів
- **ProtectedLayout** — блокує неавторизованих
- **AdminLayout** — додатково перевіряє роль

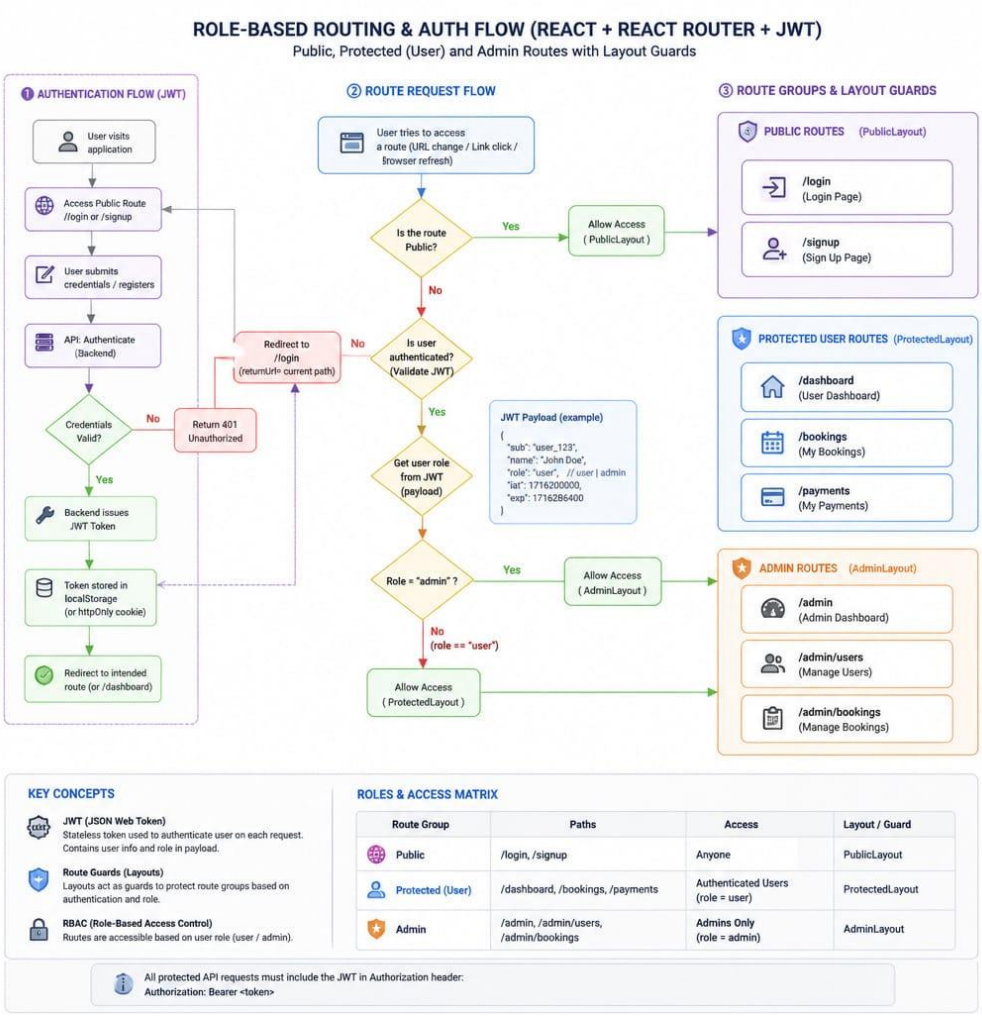


Рисунок 2.4 – Схема role-based routing

State-driven архітектура інтерфейсу

Інтерфейс побудований за принципом **state-driven UI**, де стан визначає:

- доступність елементів
- візуальне представлення
- поведінку компонентів

Основні локальні стани:

- selectedPlan
- selectedDate
- selectedSpace

- tariffs
- spaces

Зміна будь-якого стану автоматично тригерить перебудову UI без ручного оновлення DOM.

UI/UX модель взаємодії користувача

Інтерфейс проєктувався за принципом мінімізації когнітивного навантаження та зменшення кількості дій користувача.

---

1. Багатокрокова модель взаємодії

Процес бронювання розділено на послідовні кроки:

1. вибір тарифу
2. вибір дати
3. вибір робочого місця
4. підтвердження бронювання

Це реалізує принцип **step-by-step flow**, який знижує складність прийняття рішення.

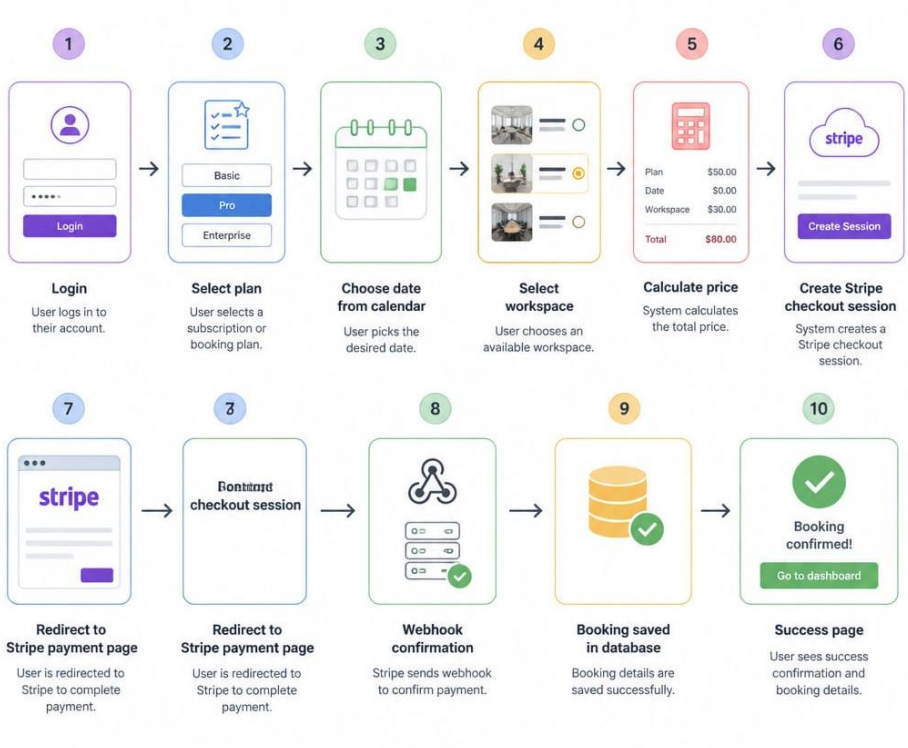


Рисунок 2.5 – Step-by-step booking flow

---

2. Принцип керованого стану (Controlled UI)

Всі ключові елементи інтерфейсу є керованими:

- input форми
- календар вибору дати
- список workspace
- кнопка підтвердження

UI не має автономної логіки — лише відображає state.

---

3. Візуальна ієрархія

Інтерфейс структурований за рівнями важливості:

- заголовки (H1–H3)
- картки (cards)
- CTA-кнопки
- допоміжний текст

Це забезпечує швидке сканування інформації користувачем.

---

4. Card-based модель інтерфейсу

Основні сутності представлені через картковий дизайн:

- тарифи
- workspace
- адміністративні записи

Переваги:

- уніфікована структура
- легке порівняння елементів
- адаптивність

Інтеграція з API та асинхронна модель даних

Взаємодія з backend реалізована через REST API з використанням Axios.

Основні типи запитів:

- GET — отримання даних
- POST — створення сутностей
- PATCH — оновлення стану
- DELETE — видалення

Асинхронна модель роботи

Дані не зберігаються локально, а отримуються динамічно:

- тарифи — /api/tariffs
- workspace — /api/workspaces
- бронювання — /api/bookings

UI синхронізується з backend у реальному часі.

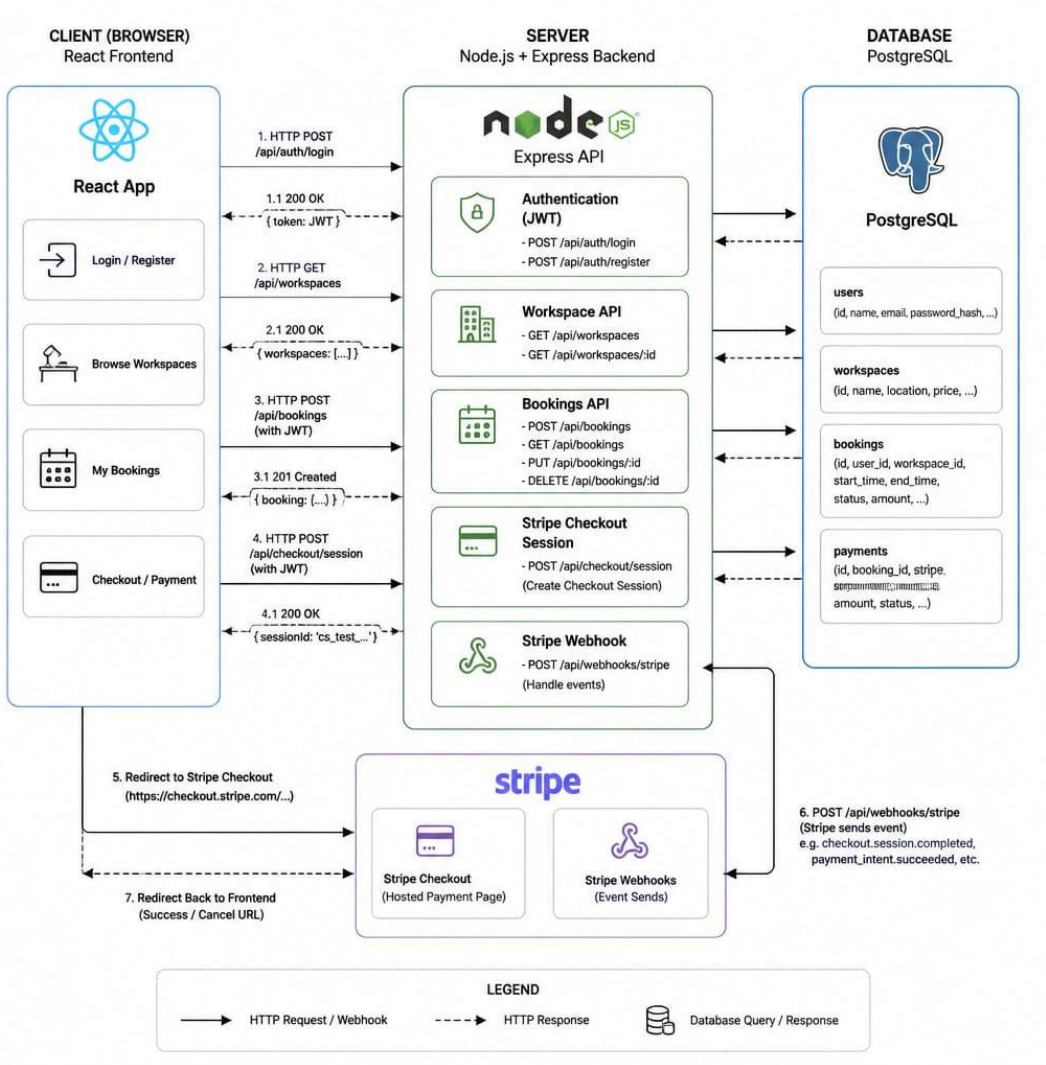


Рисунок 2.6 – Модель взаємодії Frontend ↔ Backend

UX-стани інтерфейсу

Кожен екран має три базові стани:

- loading — очікування даних
- success — відображення контенту
- error — обробка помилок

Додатково використовується fallback UI при відсутності даних.

Рольова модель інтерфейсу

Інтерфейс підтримує дві ролі:

- user
- admin

Від ролі залежить:

- доступ до маршрутів
- відображення UI елементів
- структура меню

Обробка станів авторизації та збереження сесії користувача

Окремим важливим аспектом клієнтської реалізації є механізм автентифікації та підтримки сесії користувача. Після успішного входу в систему сервер повертає JWT-токен, який використовується для ідентифікації користувача при наступних запитах до API.

Токен зберігається на стороні клієнта та застосовується для:

- доступу до захищених маршрутів;
- виконання авторизованих запитів (bookings, payments, reviews);
- визначення ролі користувача (user/admin).

Перед кожним запитом до backend виконується автоматичне додавання заголовка:

Authorization: Bearer <token>

Таким чином реалізується stateless модель авторизації, яка не потребує збереження сесій на сервері. Для контролю доступу до інтерфейсу використовується перевірка стану авторизації на рівні маршрутизатора. Якщо токен відсутній або є недійсним, користувач автоматично перенаправляється на сторінку входу.

Обробка помилок та стабільність клієнтської частини

Для підвищення стабільності інтерфейсу реалізовано централізовану модель обробки помилок API. Усі запити до backend виконуються в межах try/catch блоків, що дозволяє перехоплювати:

- помилки мережевого рівня;
- помилки авторизації (401/403);
- серверні помилки (500+);
- конфлікти бізнес-логіки (наприклад, дублювання бронювання).

У випадку помилки UI переходить у відповідний стан error, який відображає користувачу зрозуміле повідомлення без розкриття технічних деталей.

Додатково застосовується перевірка валідації на клієнті перед відправкою запитів, що зменшує кількість некоректних звернень до API та покращує загальну продуктивність системи.

Оптимізація взаємодії з API та продуктивність

Для зменшення навантаження на сервер і покращення UX використано кілька підходів оптимізації:

- мінімізація кількості запитів через агрегацію даних у backend;
- використання умовного завантаження даних (lazy fetching);
- оновлення UI тільки при зміні критичних станів;
- відсутність дублювання запитів при навігації між сторінками.

Окремо варто відзначити, що дані, які рідко змінюються (наприклад тарифи), завантажуються один раз і повторно використовуються в межах сесії користувача.

Особливості реалізації адміністративної частини інтерфейсу

Адміністративна панель реалізована як ізольований підсистема інтерфейсу з розширеним рівнем доступу. Вона використовує ті ж принципи component-driven архітектури, однак має додаткові функціональні можливості:

- управління користувачами (CRUD операції);
- модерація бронювань;
- керування тарифами та workspace;
- перегляд аналітики (dashboard метрики).

Всі адміністративні маршрути захищені подвійною перевіркою:

- наявність JWT токена;
- відповідність ролі admin.

Це гарантує відокремлення бізнес-логіки користувача і адміністратора на рівні UI та маршрутизації.

Реалізація платіжних сценаріїв та інтеграція Stripe

Окремим підсистемним компонентом клієнтської частини є механізм обробки платежів через зовнішній сервіс Stripe. Інтеграція реалізована через створення checkout-сесій на серверній стороні та перенаправлення користувача на сторінку оплати.

Після ініціації бронювання клієнт відправляє POST-запит до backend, де формується Stripe Checkout Session із передачею метаданих:

- ідентифікатор користувача;
- параметри бронювання (workspace, дата, час);
- сума платежу.

Далі користувач перенаправляється на сторінку Stripe для завершення транзакції. Успішна оплата завершується редиректом на сторінку success, а скасована — на cancel.

Ключовим елементом є використання webhook-механізму, який дозволяє серверу отримати підтвердження про оплату незалежно від клієнтської сторони. Після події checkout.session.completed backend автоматично:

- створює запис бронювання;
- формує запис платежу;
- оновлює статус оплати.

Це забезпечує консистентність даних навіть у випадку втрати зв’язку на стороні клієнта.

Узгодженість даних та транзакційна модель

У системі застосовано транзакційний підхід до критичних операцій, зокрема бронювання робочих місць. При створенні бронювання використовується механізм PostgreSQL транзакцій (BEGIN / COMMIT / ROLLBACK), що гарантує атомарність операцій:

- створення бронювання;
- створення платежу;
- перевірка доступності workspace.

У випадку помилки будь-яка з операцій відкочується, що виключає появу неконсистентних записів у базі даних (наприклад, існуючого платежу без бронювання).

Також реалізовано перевірку конфліктів часу бронювання через SQL-умови перетину інтервалів, що забезпечує коректну бізнес-логіку без накладення записів.

Безпека клієнт-серверної взаємодії

Безпека системи реалізована на кількох рівнях:

1. JWT-автентифікація

Кожен захищений запит перевіряється через middleware, який декодує токен і додає інформацію про користувача в request context.

2. Role-based access control (RBAC)

Розмежування доступу здійснюється на основі ролей:

- user — доступ до власних даних;
- admin — доступ до адміністративних ресурсів.

3. Захист маршрутів

Frontend не лише приховує UI-елементи, а й блокує доступ до маршрутів через layout-логіку, що зменшує ризик несанкціонованого доступу навіть при ручному введенні URL.

4. Валідація даних

Вхідні дані перевіряються як на клієнті, так і на сервері, що мінімізує ризик некоректних або шкідливих запитів.

Модель взаємодії Frontend–Backend

Взаємодія між клієнтською та серверною частинами реалізована за REST-архітектурним підходом із чітким поділом відповідальності.

Frontend виконує роль presentation layer, тоді як backend реалізує business logic та data layer. Основні принципи взаємодії:

- клієнт не містить бізнес-логіки;
- сервер є єдиним джерелом істини (single source of truth);
- всі зміни стану проходять через API;
- UI синхронізується з сервером після кожної критичної операції.

Цей підхід дозволяє масштабувати систему без змін у клієнтській логіці.

Підсумкова характеристика клієнтської реалізації

Реалізована клієнтська частина системи характеризується модульністю, високим рівнем ізоляції компонентів та чіткою структурою відповідальності між рівнями архітектури.

Основними технічними властивостями є:

- компонентна декомпозиція інтерфейсу;
- реактивна модель оновлення стану;
- централізована робота з API;
- рольова система доступу;
- інтеграція зовнішніх сервісів (Stripe);
- транзакційна модель критичних операцій.

Завдяки цьому забезпечується стабільна робота застосунку при одночасній підтримці розширюваності та супроводу коду.

Висновок

Клієнтська частина системи побудована як модульна SPA архітектура з чітким розділенням рівнів:

- pages (контейнери)
- sections (функціональні блоки)
- components (UI база)

Основними архітектурними принципами є:

- component-driven design
- state-driven UI
- role-based routing
- API-centric data flow
- step-by-step UX модель

Реалізована архітектура забезпечує масштабованість системи, простоту підтримки та передбачувану поведінку інтерфейсу.

2.3. Функціональні можливості розробленої вебплатформи

Розроблена вебплатформа для управління коворкінг-простором реалізує комплекс функціональних можливостей, що охоплюють повний цикл взаємодії користувача із системою — від ознайомлення з сервісом до здійснення бронювання, оплати та подальшого управління активностями. Архітектура функціональності побудована на основі рольової моделі доступу та принципу поділу системи на публічний, користувацький та адміністративний рівні.

Функціональні можливості системи безпосередньо відображають бізнес-логіку предметної області, забезпечуючи автоматизацію процесів управління робочими просторами, обліку бронювань та фінансових операцій.

Функціональні можливості публічної частини системи

Публічний рівень вебплатформи доступний без необхідності автентифікації та виконує інформаційно-презентаційну функцію. Його основною задачею є формування первинного користувацького досвіду та залучення потенційних клієнтів.

До функціоналу публічної частини належать:

- перегляд головної сторінки з описом переваг коворкінгу;
- ознайомлення з сервісами та інфраструктурою простору;
- перегляд інформаційної сторінки “About”, що містить опис місії, цінностей та команди;
- доступ до контактної інформації та комунікаційних каналів;
- перенаправлення на сторінки авторизації та реєстрації.

Даний рівень системи не взаємодіє безпосередньо з бізнес-даними, однак формує навігаційний та інформаційний шар, який є важливою частиною користувацького сценарію входу в систему.

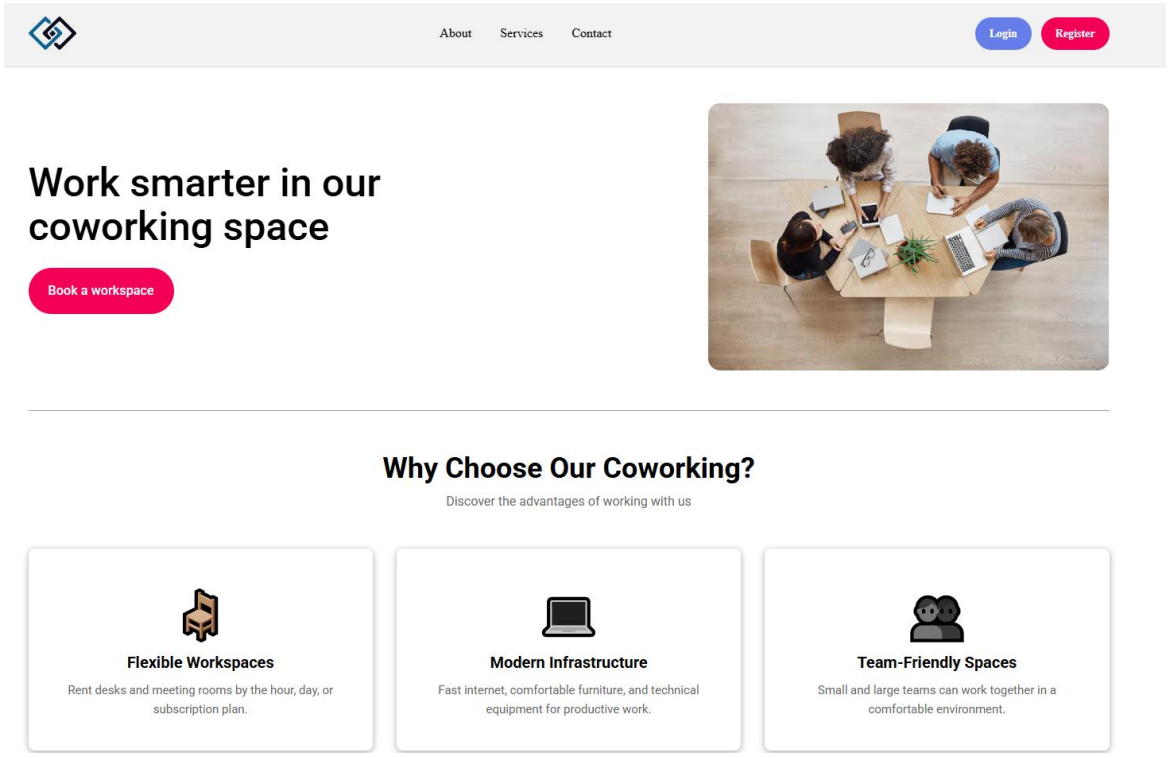


Рисунок 2.7. Публічна сторінка “Home”

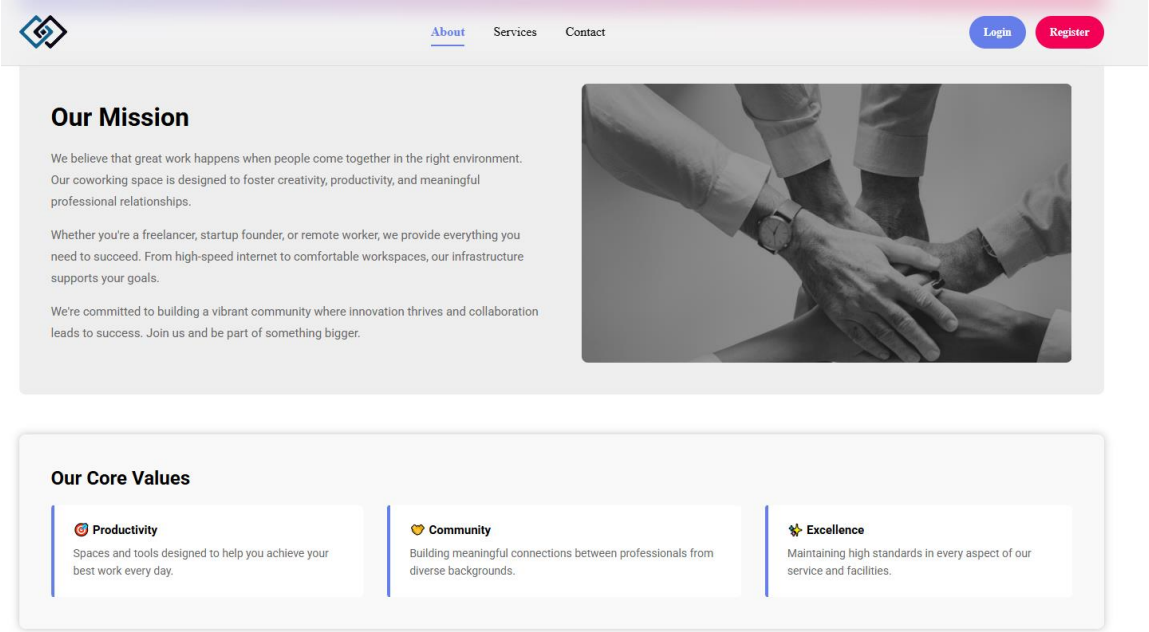


Рисунок 2.8. Публічна сторінка “About”

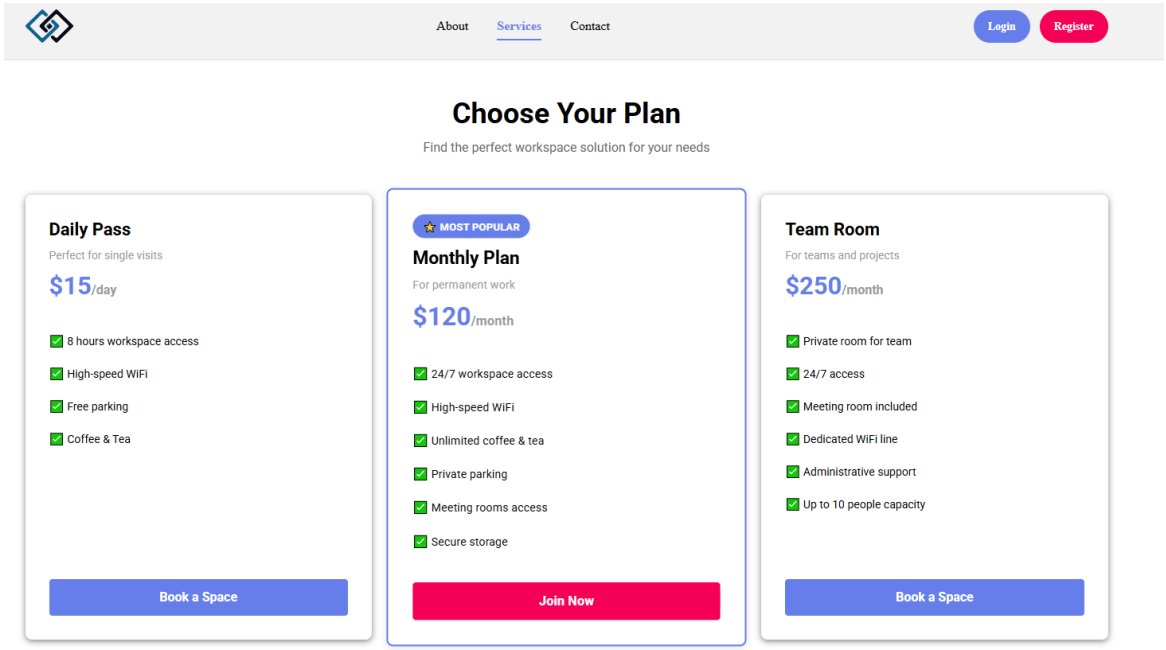


Рисунок 2.9. Публічна сторінка “Services”

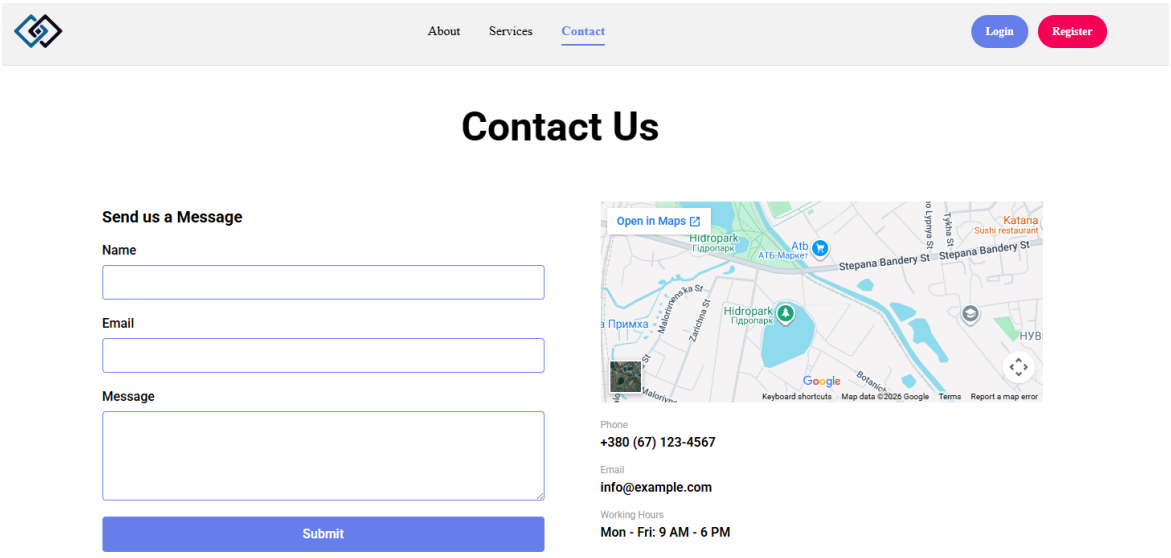


Рисунок 2.10. Публічна сторінка “Contact”

Функціональні можливості користувацької частини

Після проходження процедури автентифікації користувач отримує доступ до персоналізованого середовища (Dashboard), яке реалізує основні функції системи управління коворкінг-ресурсами.

1. Система бронювання робочих місць

Основною функціональністю користувацької частини є механізм бронювання робочих просторів. Система підтримує багатокроковий сценарій взаємодії, який включає:

- вибір тарифного плану (Daily Pass, Monthly Plan, Meeting Room);
- вибір дати та часового інтервалу;
- вибір конкретного робочого місця (workspace);
- підтвердження бронювання та ініціацію оплати.

Процес побудовано за принципом step-by-step workflow, що дозволяє зменшити когнітивне навантаження та уникнути помилок при заповненні параметрів бронювання. одатково реалізовано перевірку доступності робочих місць у реальному часі, що виключає можливість подвійного бронювання одного ресурсу в однаковий часовий інтервал.

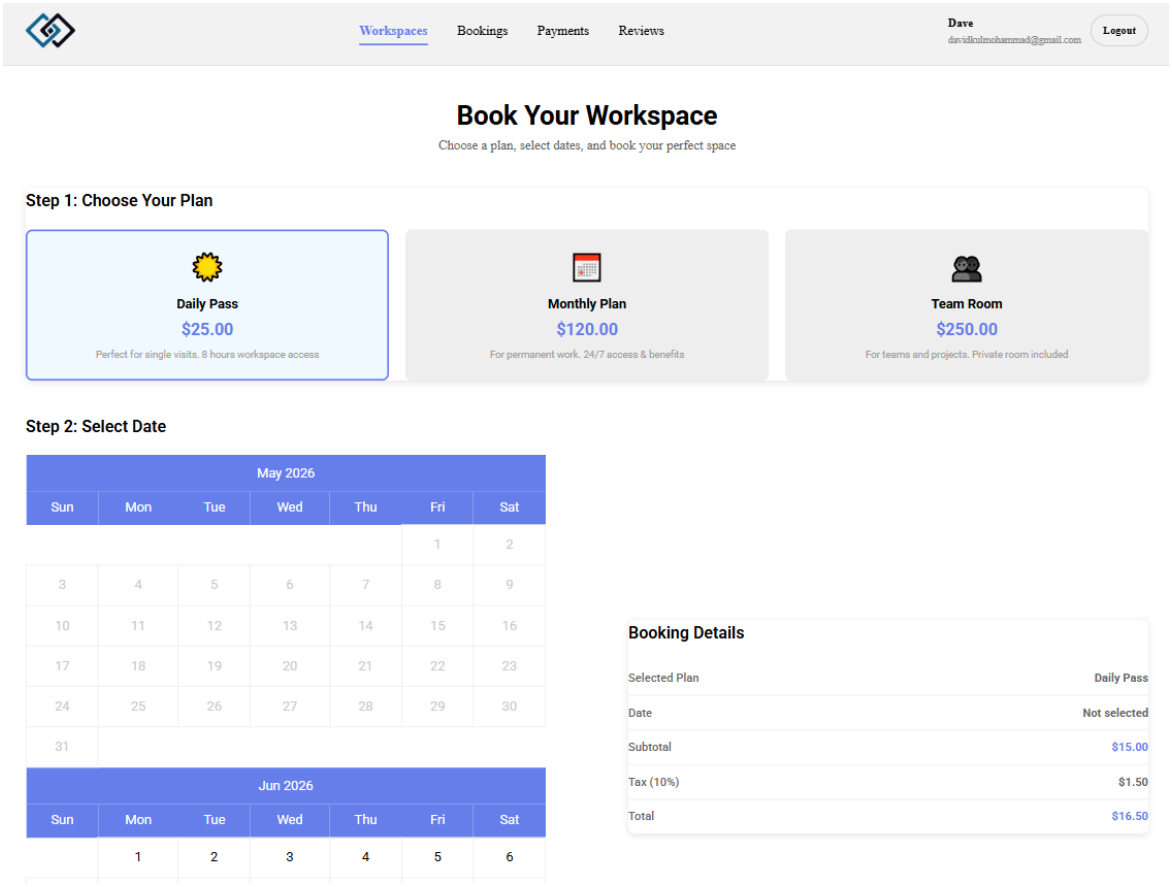


Рисунок 2.11. User-page “Workspaces”

---

2. Управління та перегляд бронювань

Користувач має доступ до персонального списку бронювань, який включає:

- активні (upcoming) бронювання;
- завершені (completed) бронювання;
- скасовані (cancelled) бронювання.

Для кожного запису відображається детальна інформація:

- назва робочого місця;
- дата та часовий інтервал;
- вартість;
- статус бронювання.

Також реалізована функція скасування активних бронювань, яка ініціює зміну стану запису в базі даних та звільнення відповідного workspace.

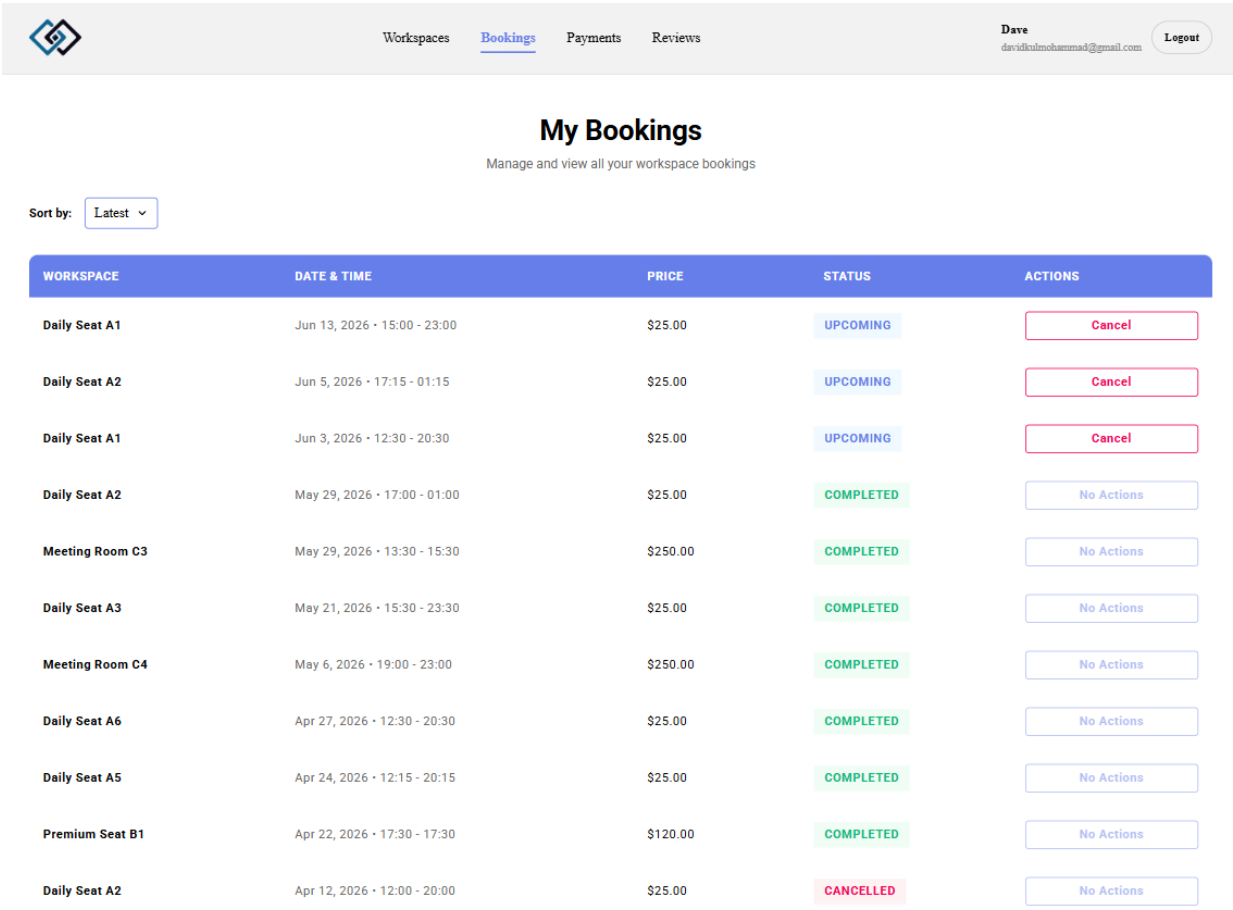


Рисунок 2.12. User-page “Bookings”

3. Платіжна підсистема

Фінансовий модуль забезпечує повний цикл обробки платежів, який включає:

- автоматичне створення платежу при ініціації бронювання;
- інтеграцію з платіжною системою Stripe;
- відображення історії транзакцій;
- підрахунок загальної суми витрат користувача.

Користувацький інтерфейс розділу Payments дозволяє відстежувати всі фінансові операції у хронологічному порядку з відображенням дати, суми та способу оплати.

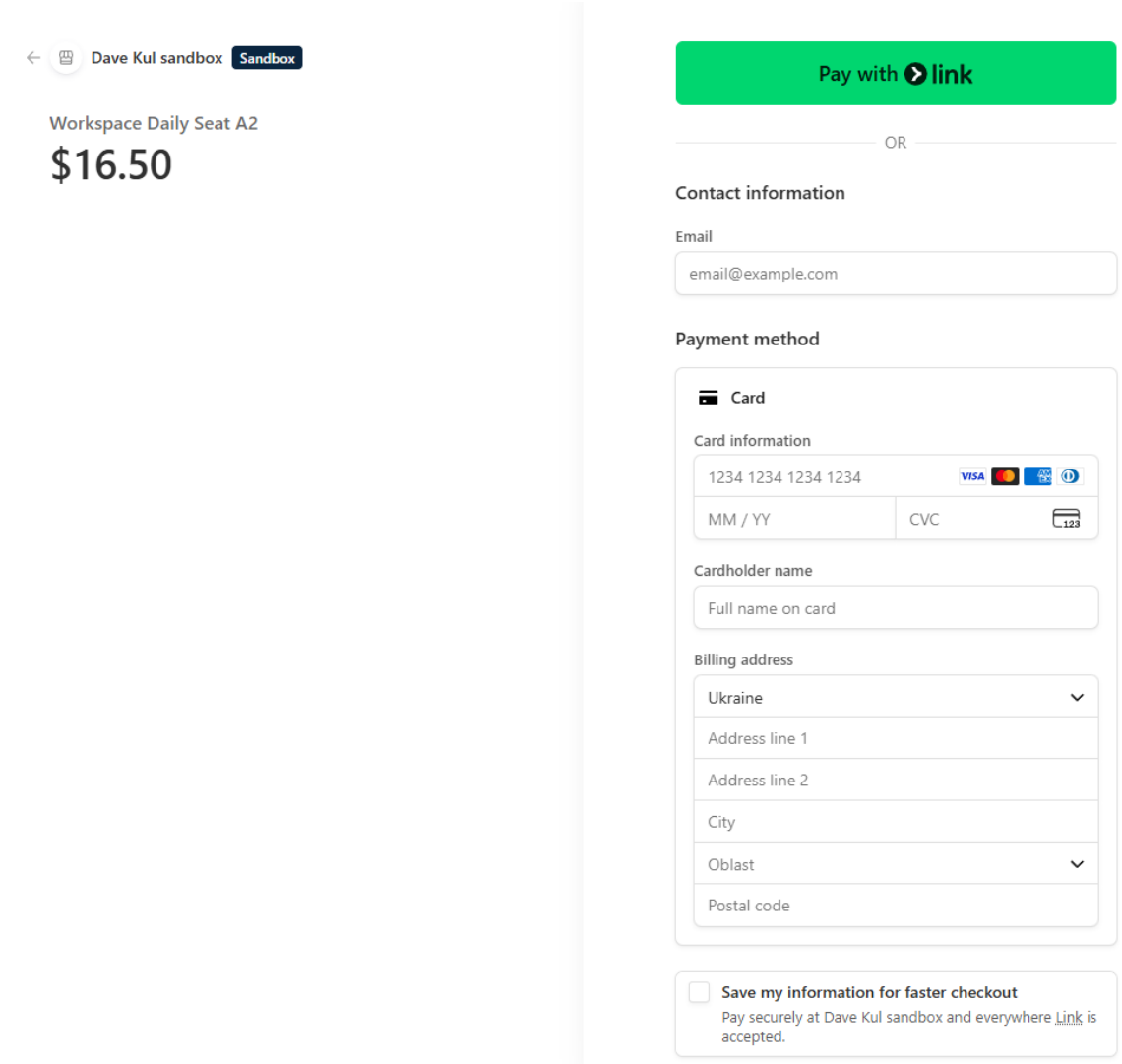


Рисунок 2.12. Платіжна система Stripe

4. Система відгуків та зворотного зв’язку

Платформа містить модуль відгуків, який дозволяє користувачам:

- залишати відгуки після використання сервісу;
- оцінювати якість обслуговування за рейтинговою шкалою;
- переглядати власні відгуки;
- взаємодіяти з відповідями адміністрації.

Даний модуль виконує функцію зворотного зв’язку між користувачами та адміністрацією системи, забезпечуючи підвищення якості сервісу.

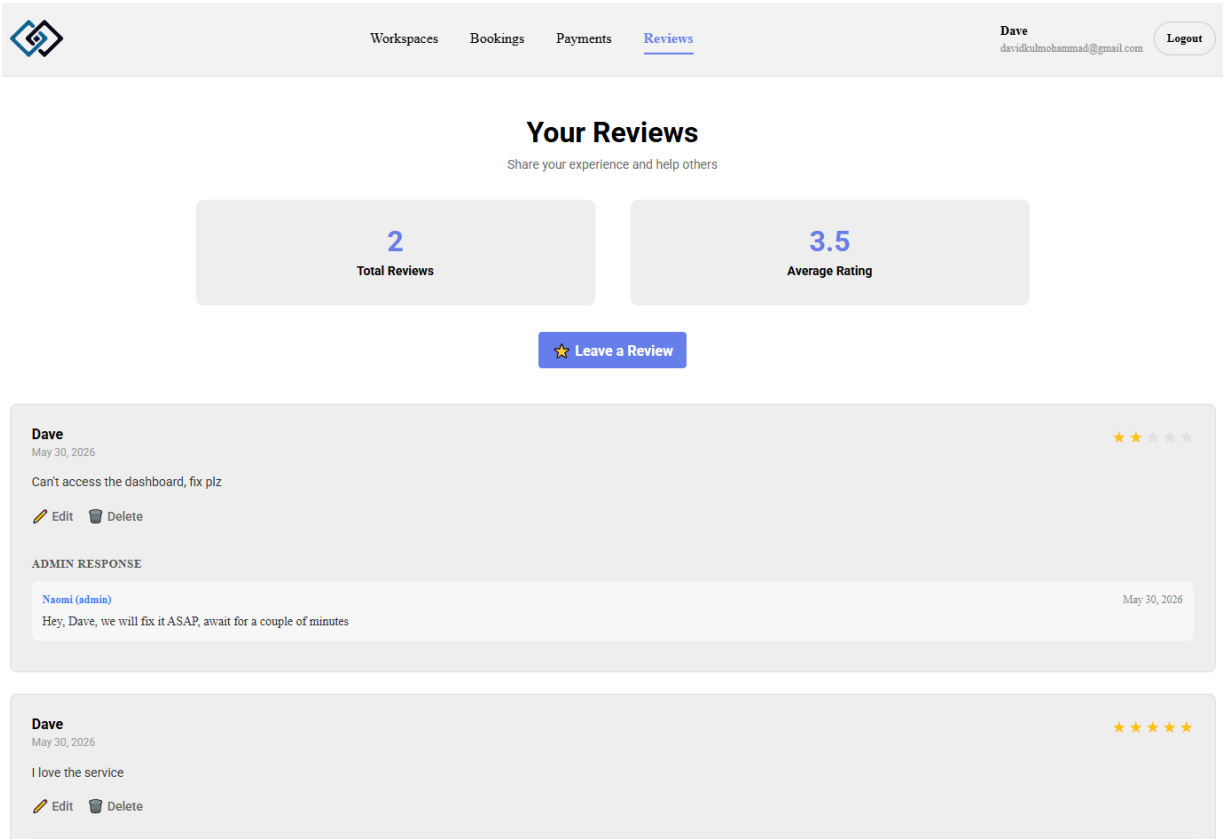


Рисунок 2.13. Платіжна система Stripe

---

5. Персоналізований Dashboard

Головна сторінка авторизованого користувача реалізована у вигляді інформаційної панелі, яка агрегує ключові показники:

- кількість майбутніх бронювань;
- загальна сума витрат;
- останні активності;

- швидкий доступ до основних функцій системи.

Dashboard реалізує принцип інформаційної ієрархії, надаючи користувачу найбільш релевантні дані без необхідності переходу між сторінками.

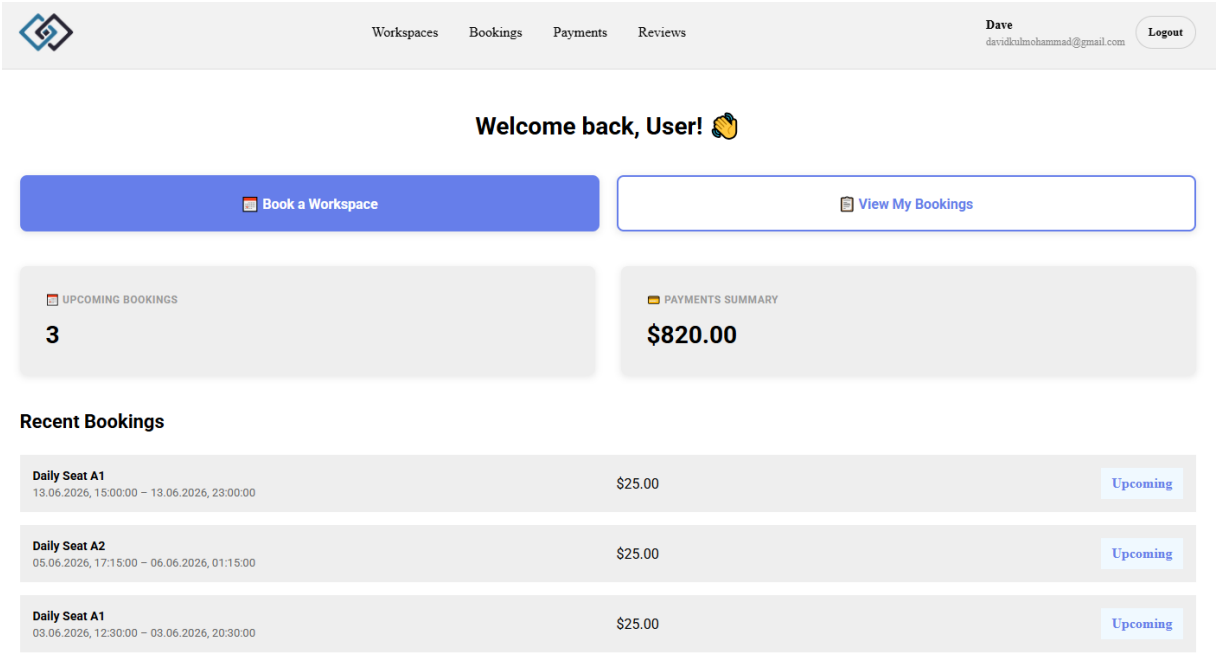


Рисунок 2.14. Dashboard користувача

Функціональні можливості адміністративної частини

Адміністративна панель є окремим функціональним модулем системи, призначеним для управління всіма бізнес-процесами платформи. Доступ до неї мають лише користувачі з роллю admin.

1. Управління користувачами

Адміністратор має можливість:

- переглядати список усіх користувачів системи;
- змінювати ролі користувачів (user ↔ admin);
- видаляти облікові записи;
- здійснювати пошук користувачів за email або ім’ям.

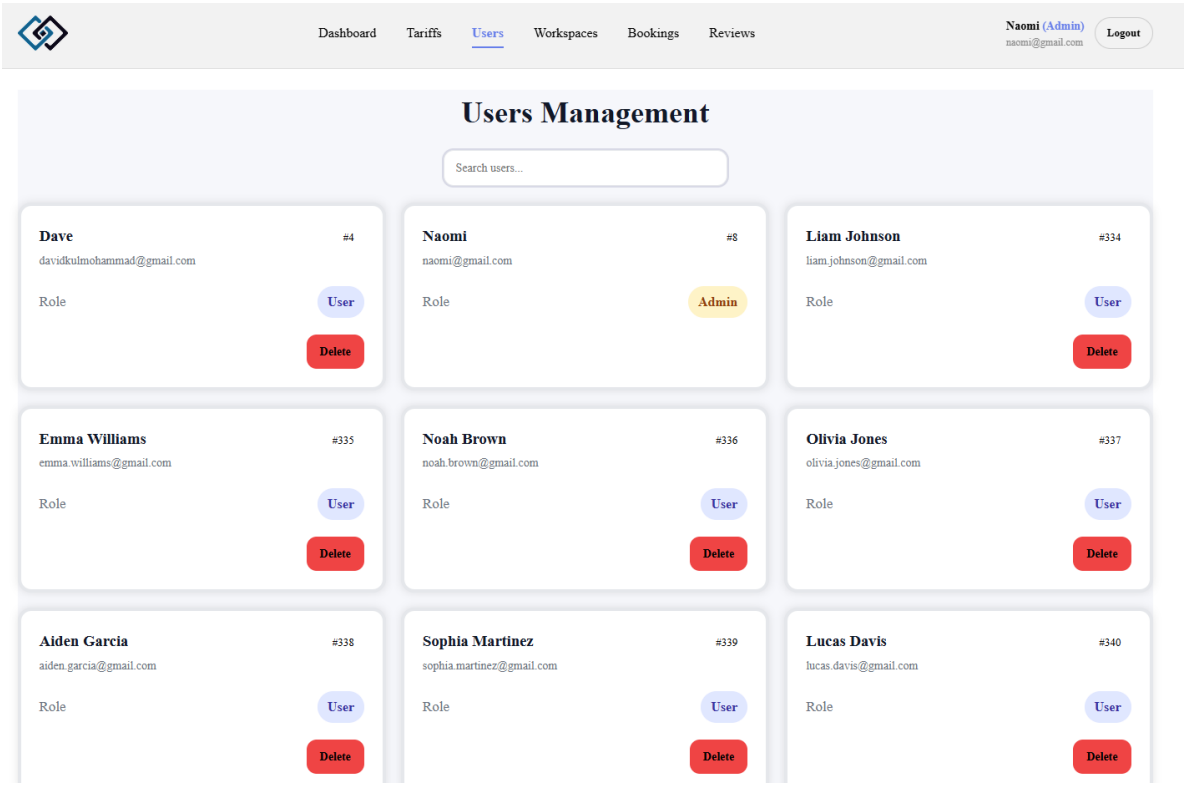


Рисунок 2.14. Admin-Page “Users”

---

2. Управління бронюваннями

Адміністративний модуль бронювань дозволяє:

- переглядати всі бронювання системи незалежно від користувача;
- фільтрувати записи за статусами (upcoming, completed, cancelled);
- здійснювати скасування бронювань;
- видаляти записи з системи.

Додатково відображається інформація про навантаження на простори, що дозволяє аналізувати ефективність використання ресурсів.

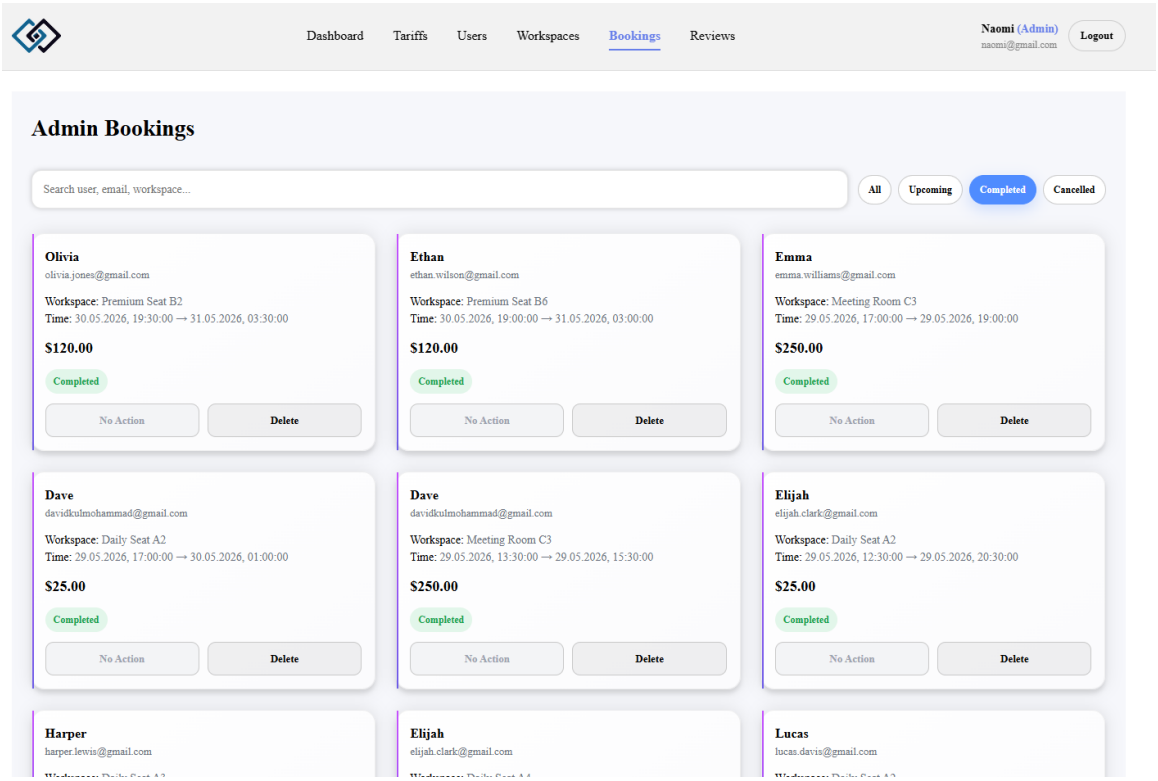


Рисунок 2.15. Admin-Page “Bookings”

---

3. Управління робочими просторами

Адміністратор може виконувати повний CRUD-цикл для workspace:

- створення нових робочих місць;
- редагування параметрів (номер, тип, статус);
- зміна стану доступності (available, booked, maintenance);
- видалення існуючих просторів.

Це дозволяє динамічно адаптувати інфраструктуру коворкінгу до потреб користувачів.

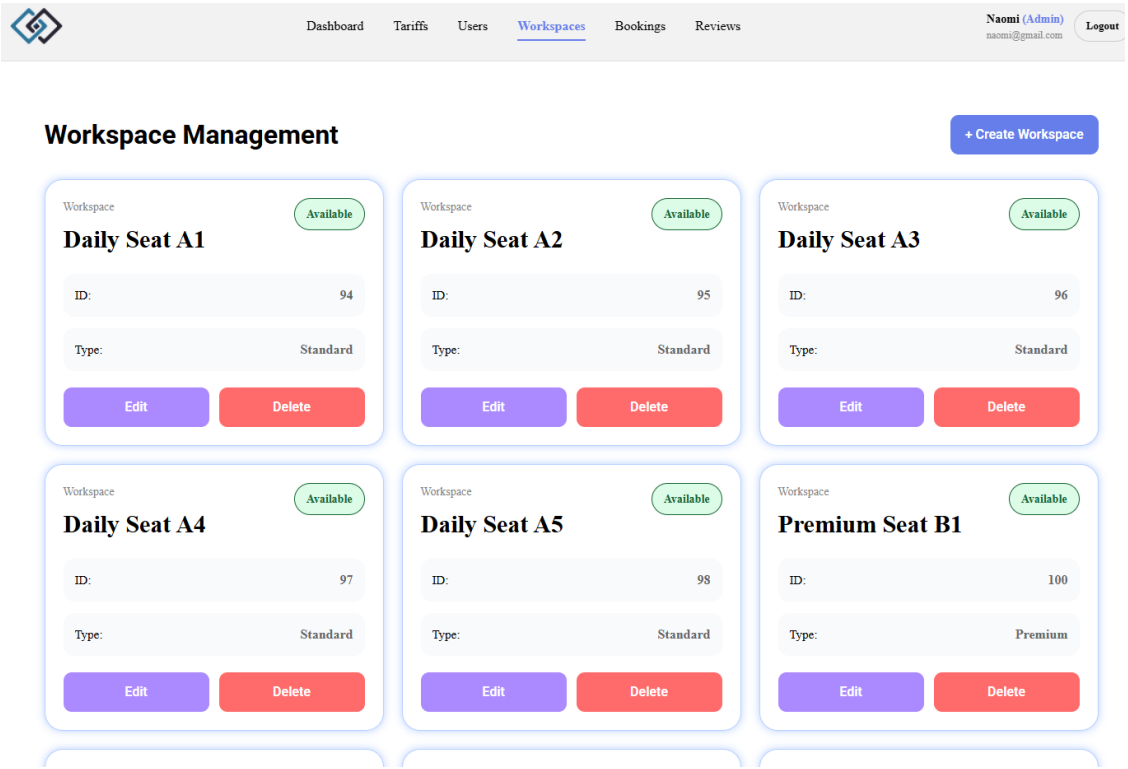


Рисунок 2.16. Admin-Page “Workspaces”

---

4. Управління тарифними планами

Система підтримує редагування тарифної політики без необхідності змін у коді застосунку:

- зміна вартості тарифів;
- оновлення описів;
- модифікація типів планів;
- керування доступними опціями для користувачів.

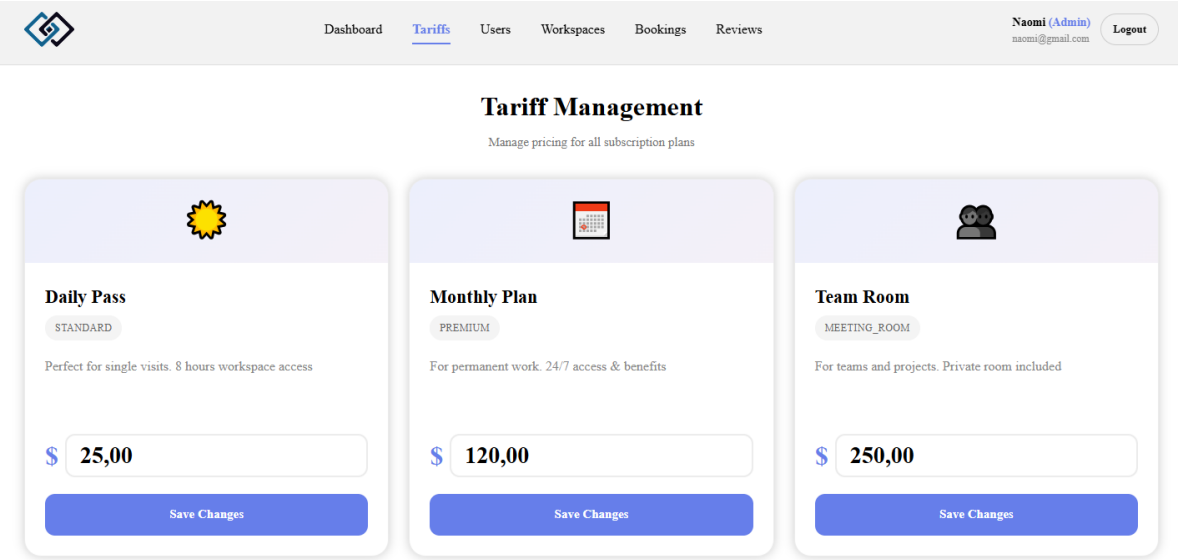


Рисунок 2.17. Admin-Page “Tariffs”

5. Модерація відгуків

Адміністративна частина включає функціонал модерації відгуків:

- перегляд усіх відгуків користувачів;
- додавання адміністративних відповідей;
- видалення некоректних або нерелевантних відгуків;
- контроль якості комунікації.

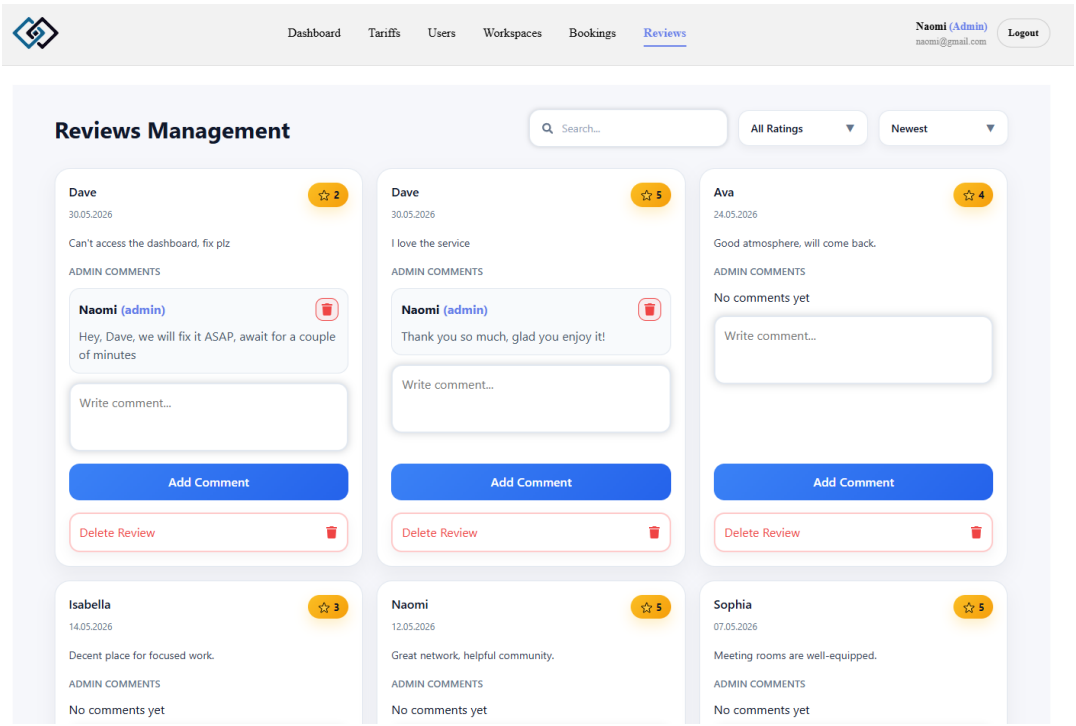


Рисунок 2.18. Admin-Page “Reviews”

6. Аналітична підсистема

Dashboard адміністратора реалізує базові аналітичні функції:

- загальна кількість користувачів;
- кількість бронювань;
- кількість відгуків;
- загальний дохід системи;
- графічне відображення динаміки бронювань і фінансових показників.

Дані представлені у вигляді графіків, що дозволяє оцінювати ефективність роботи платформи у часовій перспективі.

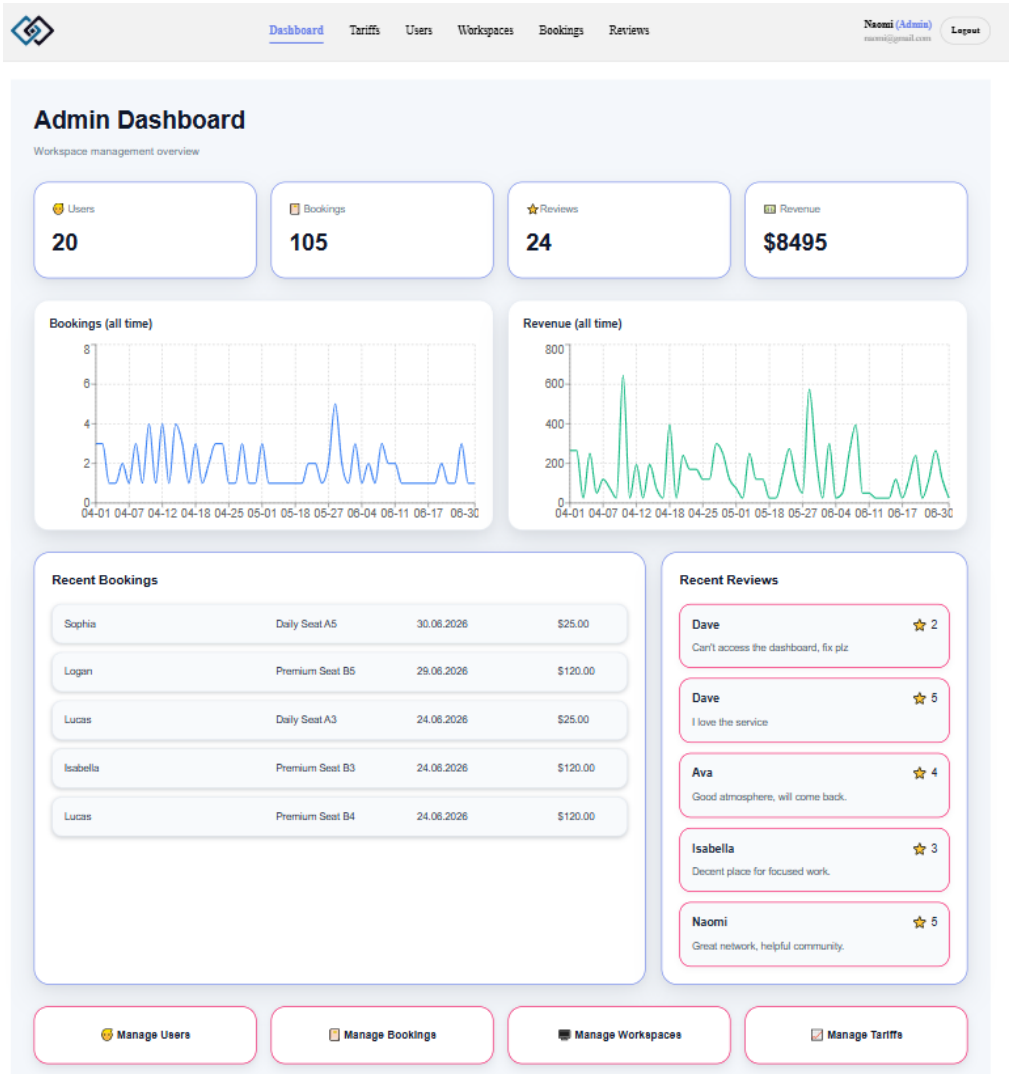


Рисунок 2.19. Dashboard адміністратора

Інтеграційні можливості системи

Функціонування вебплатформи базується на інтеграції кількох незалежних підсистем, що взаємодіють через стандартизований REST API:

- клієнтська частина (React SPA);
- серверна частина (Node.js + Express);
- база даних PostgreSQL;
- платіжний сервіс Stripe;
- система автентифікації JWT.

Такий підхід забезпечує слабку зв’язаність компонентів та дозволяє масштабувати систему без зміни її основної архітектури.

Висновок до функціональних можливостей

Розроблена вебплатформа забезпечує повний цикл управління коворкінг-простором, включаючи інформаційний доступ, бронювання ресурсів, фінансові операції та адміністративний контроль.

Функціональна модель системи побудована відповідно до принципів модульності, рольового доступу та централізованої обробки даних. Це забезпечує стабільність роботи, простоту розширення функціоналу та зручність використання для кінцевих користувачів і адміністраторів.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи було розроблено інформаційну систему для автоматизації процесів управління коворкінгом, яка забезпечує централізоване управління користувачами, робочими місцями, бронюваннями, платежами та відгуками. Актуальність виконаної роботи обумовлена зростанням популярності коворкінг-просторів та необхідністю використання сучасних цифрових інструментів для підвищення ефективності їх функціонування.

У ході виконання роботи було проведено аналіз предметної області, досліджено особливості організації діяльності коворкінгів та визначено основні бізнес-процеси, які потребують автоматизації. Також було здійснено порівняльний аналіз сучасних серверних технологій Node.js та Laravel, а також досліджено існуючі вебплатформи коворкінгів, що дозволило обґрунтувати вибір технологічного стеку та сформувати функціональні вимоги до майбутньої системи.

На етапі проєктування було розроблено логічну модель бази даних на основі СКБД PostgreSQL, визначено основні сутності предметної області та взаємозв’язки між ними. Структура бази даних відповідає принципам нормалізації, забезпечує цілісність інформації та створює умови для подальшого розширення функціональних можливостей системи.

У межах практичної реалізації було створено клієнтську частину вебзастосунку з використанням бібліотеки React та серверну частину на базі Node.js і Express. Реалізовано механізми реєстрації та авторизації користувачів із використанням JWT-токенів, рольову модель доступу, систему бронювання робочих місць, модуль керування тарифами, підсистему відгуків, а також адміністративну панель для управління основними сутностями системи.

Особливу увагу приділено реалізації фінансового модуля шляхом інтеграції платіжної системи Stripe, що дозволило автоматизувати процес онлайн-оплати послуг коворкінгу. Для забезпечення надійності роботи реалізовано механізми перевірки доступності робочих місць, контролю конфліктів бронювання, транзакційної обробки критичних операцій та захисту даних користувачів.

У результаті тестування було підтверджено коректність функціонування всіх основних модулів системи та їх узгоджену взаємодію. Розроблена вебплатформа забезпечує автоматизацію ключових процесів діяльності коворкінгу, спрощує адміністрування ресурсів, підвищує якість обслуговування користувачів та зменшує кількість операцій, що виконуються вручну.

Поставлена мета роботи досягнута в повному обсязі, а всі визначені завдання виконані. Створена інформаційна система може бути використана як основа для впровадження в реальних коворкінг-просторах та має перспективи подальшого розвитку. Серед можливих напрямів удосконалення можна виділити впровадження системи сповіщень, інтеграцію календарних сервісів, реалізацію мобільного застосунку, розширення аналітичного модуля та використання інструментів прогнозування завантаженості робочих місць на основі накопичених даних.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. React Documentation. URL: https://react.dev/ (дата звернення: 02.05.2026).
2. React Learn. URL: https://react.dev/learn (дата звернення: 03.05.2026).
3. Node.js Documentation. URL: https://nodejs.org/en/docs (дата звернення: 05.05.2026).
4. Express.js Documentation. URL: https://expressjs.com/ (дата звернення: 06.05.2026).
5. PostgreSQL Documentation. URL: https://www.postgresql.org/docs/ (дата звернення: 08.05.2026).
6. Stripe Documentation. URL: https://docs.stripe.com/ (дата звернення: 10.05.2026).
7. Axios Documentation. URL: https://axios-http.com/docs/intro (дата звернення: 12.05.2026).
8. React Router Documentation. URL: https://reactrouter.com/en/main (дата звернення: 14.05.2026).
9. JSON Web Tokens (JWT). URL: https://jwt.io/introduction (дата звернення: 16.05.2026).
10. What is Coworking? URL: https://www.coworkingresources.org/blog/what-is-coworking (дата звернення: 18.05.2026).
11. WeWork Official Website. URL: https://www.wework.com/ (дата звернення: 20.05.2026).
12. Regus Official Website. URL: https://www.regus.com/ (дата звернення: 22.05.2026).
13. Usage Statistics and Market Share of React. URL: https://w3techs.com/technologies/details/js-react (дата звернення: 25.05.2026).
14. Usage Statistics and Market Share of Node.js. URL: https://w3techs.com/technologies/details/ws-nodejs (дата звернення: 28.05.2026).
15. REST API Tutorial. URL: https://restfulapi.net/ (дата звернення: 31.05.2026).