Міністерство освіти і науки України
Дрогобицький державний педагогічний університет імені Івана Франка
Кафедра фізики та інформаційних систем

«До захисту допускаю»
завідувач кафедри фізики
та інформаційних систем,
кандидат фіз-мат. наук, доцент
_____ В. Б. Гольський
« ___ » _____ 2026 р.

# Розроблення системи бронювання послуг

## Спеціальність 122 Комп’ютерні науки

Випускова робота
на здобуття кваліфікації – бакалавр з комп'ютерних наук

Автор роботи: **Кіс Іван Романович** _____
*підпис*

Науковий керівник: **старший викладач**
**Наум Олег Миколайович** _____
*підпис*

**Дрогобич 2026**

Дрогобицький державний педагогічний університет
імені Івана Франка

Зав. кафедрою

_____ _____
(підпис) (дата)

Завдання
**на підготовку кваліфікаційної бакалаврської роботи**

1. Тема:**Розроблення системи бронювання послуг**
2. Керівник ___*старший викладач Наум О. М.*_____
3. Студент ___*Кіс Іван Романович*_____
(прізвище, ім'я, по батькові)
4. Перелік питань, що підлягають висвітленню у кваліфікаційній роботі
   1. Аналіз предметної області
   2. Вимоги до системи.
   3. Засоби та технології побудови системи
   4. Проєктування системи у графічних файлах
   5. Програмна реалізація системи
5. Список рекомендованої літератури

1.Пономаренко В. С. Проектування інформаційних систем : навч. посібник. Київ : ВД «Академія», 2002.

2.Глушко О. В., Глушко М. В. Розробка серверних вебзастосунків на платформі Node.js: навчальний посібник. Тернопіль: ТНТУ ім. Івана Пулюя, 2024.

3.Борсук О. В., Прокопенко С. М. Проєктування та розробка REST API на платформі Node.js. Вісник Харківського національного університету. Серія: Математичне моделювання. Інформаційні технології. Автоматизовані системи управління. 2024.

6. Етапи підготовки роботи

| № | Назва етапу | Термін виконання | Термін звіту перед керівником, кафедрою |
|---|---|---|---|
| 1. | Аналіз предметної області | Жовтень 2025 | |
| 2. | Вимоги до системи | Листопад 2025 | |
| 3 | Засоби та технології побудова системи | Грудень 2025 | |
| 4. | Проєктування системи у графічних файлах | Грудень 2025 | |
| 5. | Програмна реалізація системи | Лютий-травень 2026 | |

7. Дата видачі завдання _____
8. Термін подачі роботи керівнику_____
9. З вимогами до виконання кваліфікаційної роботи і завданням ознайомлений _____
(підпис студента)
10. Керівник _____
(підпис)

**Захист кваліфікаційної (бакалаврської) роботи**

**Розроблення системи бронювання послуг**

Оцінка за стобальною шкалою: _____

Оцінка за національною чотирибальною шкалою: _____

Коротка мотивація захисту: _____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____ Голова ЕК _____ _____

*Дата* *підпис* *прізвище, ім'я*

Секретар ЕК _____ _____

*підпис* *прізвище, ім'я*

# АНОТАЦІЯ

Розробка платформи для онлайн-бронювання послуг. Кваліфікаційна робота, Дрогобицький державний педагогічний університет імені Івана Франка, Дрогобич 2026.

Робота присвячена проєктуванню та розробці платформи для онлайн-бронювання послуг різних бізнесів. Метою роботи є створення багатофункціонального вебдодатку, який дозволяє підприємцям реєструвати свій бізнес, налаштовувати графік роботи та приймати онлайн-записи від клієнтів, а клієнтам — зручно знаходити необхідні послуги та здійснювати бронювання через інтерактивний календар.

У розділі аналізу предметної області визначено призначення платформи та проаналізовано наявні аналоги. У розділі вимог та технологій розробки сформульовано функціональні вимоги та обрано засоби реалізації системи. У розділі проєктування описано архітектуру системи за допомогою UML-діаграм. У розділі програмної реалізації детально розкрито методи розробки основних модулів платформи із застосуванням сучасного стеку технологій.

У результаті виконаної роботи було розроблено вебплатформу, що має наступні можливості: реєстрація та автентифікація з підтримкою Google OAuth, рольова система доступу, інтерактивний календар бронювань з перевіркою доступності слотів, панель управління для власників бізнесу з аналітикою та графіками, система сповіщень у реальному часі, чат між клієнтом і бізнесом, автоматичні електронні нагадування та функції на основі штучного інтелекту.

Ключові слова: вебплатформа, бронювання послуг, React, Node.js, MongoDB, Socket.io, штучний інтелект.

## ABSTRACT

Development of a platform for online service booking. Qualification work, Drohobych Ivan Franko State Pedagogical University, Drohobych 2025.

The work is dedicated to designing and developing a platform for online booking of services provided by various businesses. The aim is to create a multifunctional web application that allows entrepreneurs to register their business, configure working hours, and accept online appointments from clients, while clients can conveniently find the services they need and make bookings via an interactive calendar.

The result of the work is a web platform with the following features: registration and authentication with Google OAuth support, role-based access control, an interactive booking calendar with slot availability checking, a business owner dashboard with analytics and charts, a real-time notification system, chat system for clients, automated email reminders, and artificial intelligence features.



Keywords: web platform, service booking, React, Node.js, MongoDB, Socket.io, artificial intelligence.

# ЗМІСТ

# ВСТУП

Актуальність теми. У сучасному світі цифровізація охоплює всі сфери людської діяльності, й ринок надання послуг не є винятком. За даними досліджень галузі, понад 60% споживачів надають перевагу онлайн-бронюванню послуг порівняно з телефонними дзвінками чи особистими візитами. Малий та середній бізнес, зокрема салони краси, медичні кабінети, фітнес-клуби, консалтингові компанії та десятки інших категорій підприємств, гостро потребує доступного інструменту для автоматизації процесу запису клієнтів.

Наявні на ринку рішення або є надто дорогими для малого бізнесу, або пропонують обмежений набір функцій, або орієнтовані виключно на англомовну аудиторію. Це зумовлює потребу у створенні сучасної платформи (Software as a Service), яка поєднує зручність використання, широкий функціонал та доступність для бізнесів різних масштабів.

Платформа такого типу повинна надавати власникам бізнесу інструменти для керування розкладом, послугами та бронюваннями, а клієнтам — зручний інтерфейс для пошуку та запису до фахівців. Додаткову цінність створюють сучасні технології: взаємодія в реальному часі через WebSocket-протокол, інтеграція з email, а також функції на основі штучного інтелекту для генерації описів, рекомендацій, аналізу даних, та відповіді користувачам.

Об'єкт дослідження: процеси проєктування та розробки платформи для онлайн-бронювання послуг.

Предмет дослідження: методи та засоби проєктування та розробки повнофункціональної вебплатформи для бронювання послуг з використанням стеку MERN та супутніх технологій.

Мета роботи: спроєктувати та розробити платформу для онлайн-бронювання послуг, яка охоплює повний цикл взаємодії між бізнесом та клієнтом, та інтеграцією ШІ а також Google API.

Для досягнення мети було поставлено наступні завдання:

1. провести аналіз предметної області та існуючих сервісів-аналогів;
2. сформулювати функціональні та нефункціональні вимоги до платформи;
3. обрати відповідний технологічний стек для реалізації;
4. спроєктувати архітектуру системи за допомогою UML-діаграм;
5. програмно реалізувати всі ключові модулі платформи;
6. протестувати і налагодити роботу розробленої системи.

Практичне значення роботи. Розроблена платформа є готовим до використання вебдодатком, який може бути розгорнутий як хмарний сервіс і застосований реальними бізнесами для автоматизації процесу запису клієнтів. Платформа реалізує повний цикл бронювання: від реєстрації бізнесу до отримання клієнтом підтвердження та нагадування.

Структура роботи: робота складається з вступу, чотирьох розділів, висновків та списку використаних джерел.

# 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

## 1.1. Призначення платформи для бронювання послуг

SaaS (Software as a Service) — це модель надання програмного забезпечення в хмарі. Ця модель найближче описує систему розроблену у цій роботі. Постачальник розробляє і підтримує послуги, надає для нього автоматичний доступ клієнтам через Інтернет. Оплата здійснюється тільки за те, чим користуються. Отже, замовники SaaS можуть суттєво скоротити витрати, розгортати, масштабувати й оновлювати бізнес-рішення швидше, ніж за підтримки локальних систем і програмного забезпечення, а також точніше прогнозувати сукупну вартість володіння. [8]

На початку 2000-х років перші рішення SaaS були розрізненими й негнучкими, кожне з них призначалося лише для однієї бізнес-проблеми. Відтоді SaaS значно розвинулися. Сучасні застосування можуть охоплювати й поєднувати все, від фінансів, керування персоналом, закупівель і процесів у ланцюгах постачання до рішень для комерції, маркетингу, продажів і послуг.

Призначення розроблюваної платформи полягає у створенні єдиного цифрового середовища, де підприємства різних галузей можуть керувати своїм розкладом та приймати онлайн-записи, а клієнти — зручно знаходити та бронювати необхідні послуги. Платформа орієнтована на наступні категорії бізнесу: заклади краси та wellness, медичні кабінети та стоматологія, фітнес-центри та спортивні секції, юридичні та консалтингові компанії, автомайстерні та побутові сервіси.

Основні цілі платформи можна окреслити наступним чином:

- автоматизація процесу запису клієнтів та зменшення навантаження на адміністраторів бізнесу;
- надання клієнтам зручного інтерфейсу для самостійного вибору послуги, дати та часу;

- попередження подвійного бронювання та оптимізація завантаженості розкладу;
- забезпечення своєчасних нагадувань для зменшення кількості неявок (no-show);
- надання власникам бізнесу аналітичних даних для прийняття обґрунтованих рішень;
- забезпечення безпечного та надійного середовища для всіх учасників платформи.

Система передбачає три ролі користувачів. Адміністратор платформи керує всіма користувачами та бізнесами, має доступ до загальної статистики та верифікує нові підприємства. Власник бізнесу реєструє своє підприємство, додає послуги, налаштовує графік роботи та керує бронюваннями. Клієнт переглядає каталог бізнесів, обирає послугу та здійснює бронювання через інтерактивний календар.

Платформа також включає інтеграцію з сервісами штучного інтелекту (OpenAI API) для автоматичної генерації описів бізнесів та послуг, персоналізованих рекомендацій, чатування та аналітичних висновків для власників бізнесу. Це виділяє розроблювану систему серед аналогів і забезпечує додаткову цінність для кінцевих користувачів.

## 1.2. Огляд та аналіз аналогів

У процесі дослідження предметної області було проаналізовано декілька існуючих платформ для онлайн-бронювання послуг:

Calendly — одна з найпопулярніших платформ для планування зустрічей та бронювання часу. Вона дозволяє фахівцям та командам автоматизувати процес планування, надаючи клієнтам посилання для самостійного вибору зручного часового слоту. Платформа інтегрується з основними календарними сервісами (Google Calendar, Outlook, iCloud), підтримує відеоконференції через Zoom та Microsoft Teams. [4]

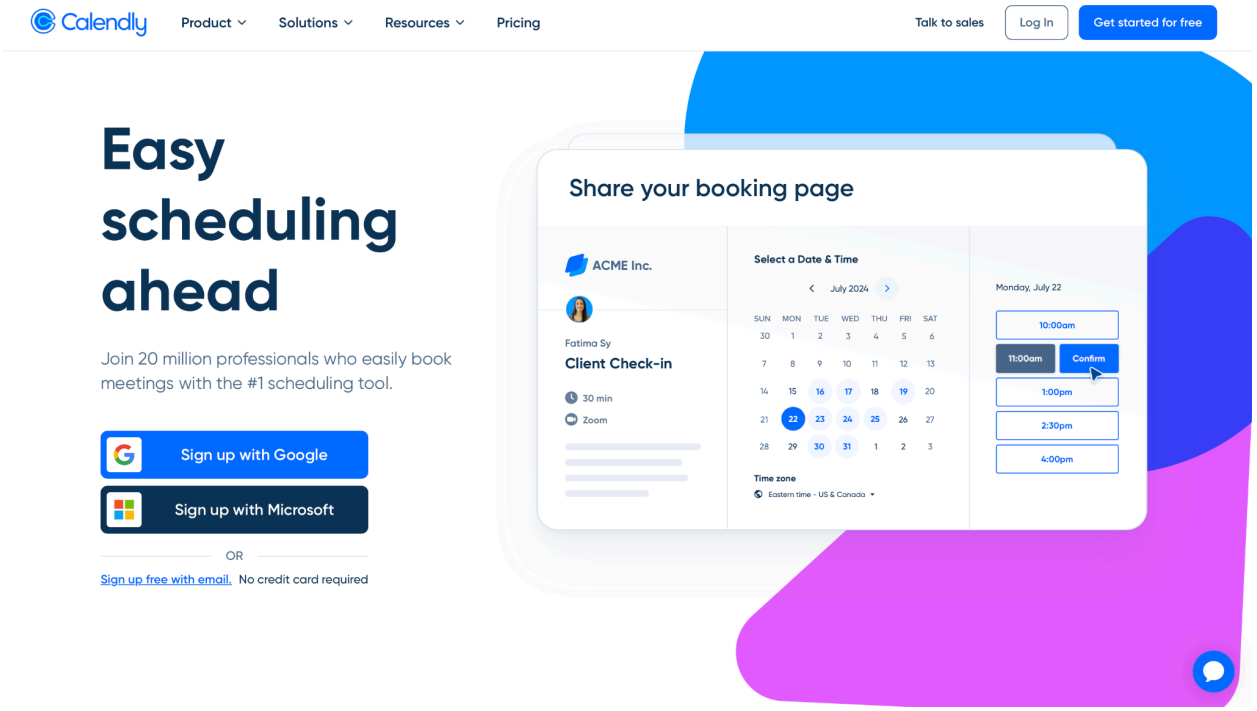


*Рисунок 1.1 — Головна сторінка Calendly*

Основні функції Calendly включають: Спрощення планування для понад 20 мільйонів

Інтуїтивно зрозуміла та потужна платформа автоматизації планування Calendly допомагає вам укладати угоди, наймати найкращих спеціалістів, будувати міцні стосунки та швидше розвивати свій бізнес.

До недоліків відноситься орієнтованість переважно на індивідуальних фахівців та команди, а не на бізнеси з широким спектром послуг, обмеженість безкоштовного плану та відсутність україномовного інтерфейсу.

SimplyBook.me — це комплексна система онлайн-бронювань, яка охоплює широкий спектр галузей: від салонів краси та спа до медичних установ та фітнес-центрів. Платформа надає власний вебсайт для бронювання, можливість інтеграції з існуючими сайтами через віджети та кнопки, а також мобільні застосунки. [5]

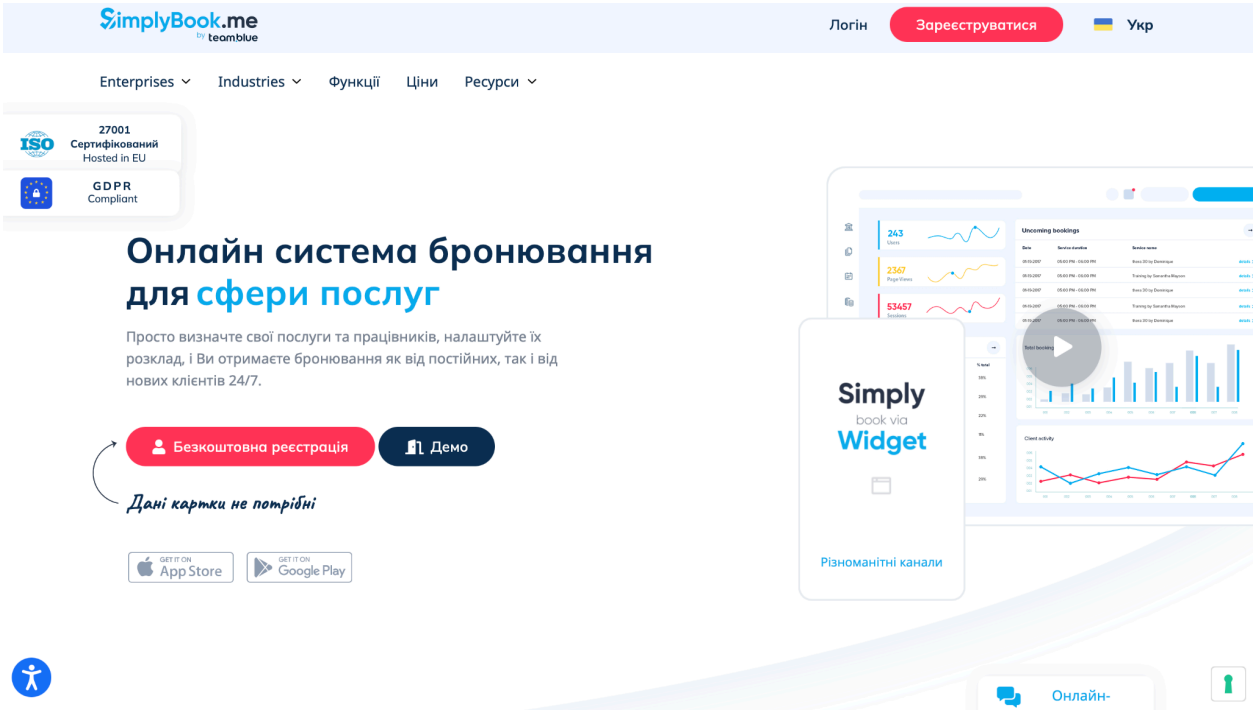


*Рисунок 1.2 — Інтерфейс SimplyBook.me*

SimplyBook.me пропонує широкий набір функцій: Онлайн-бронювання 24/7 клієнти бронюють зустрічі через різні канали такі як ваш веб-сайт для бронювання, бізнес веб-сайт, Facebook, Instagram чи Google Business Profile.

Надсилання відповідних нагадуваннь перед зустрічами. Зменшить ймовірність скасування в останній момент за допомогою авансів. Позбудьтесь накладок у резервуваннях за допомогою синхронізації з вашим персональним календарем.

Створюйте і налаштовуйте форму прийому, щоб зібрати інформацію про клієнта в процесі бронювання. Це можуть бути запити з текстовими або цифровими відповідями, прапорцями чи випадаючим меню.

Головне серед цього це є великий український онлайн сервіс такого роду.

Booksy — платформа для бронювання послуг у сфері краси та особистого догляду. Вона об'єднує клієнтів та майстрів, дозволяючи здійснювати пошук фахівців за розташуванням та спеціалізацією, переглядати портфоліо та відгуки, а також бронювати послуги онлайн. [6]

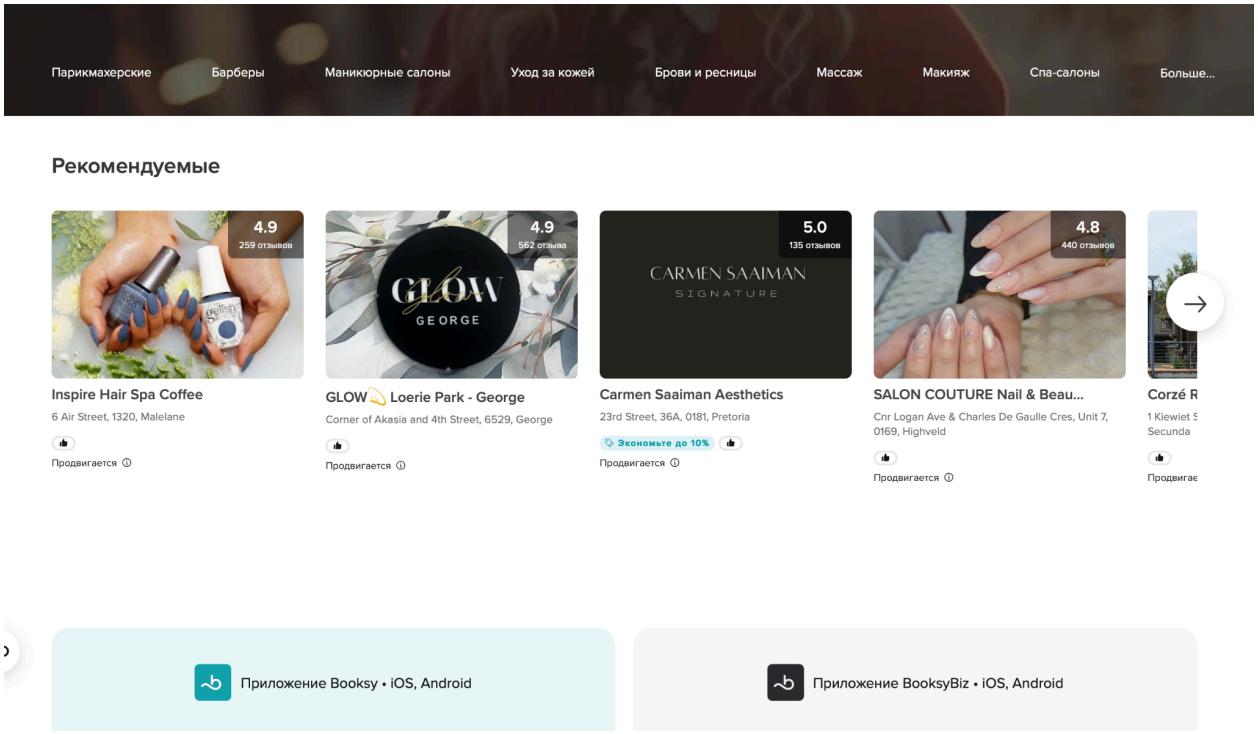


*Рисунок 1.3 — Сторінка бізнесу в Booksy*

Платформа орієнтована переважно на сферу краси, що є одночасно і перевагою (глибока спеціалізація), і обмеженням (вузька ніша). Booksy недоступна в Україні у повному обсязі та не підтримує украномовний інтерфейс.

Порівняльний аналіз аналогів дозволяє виявити ключові можливості для диференціації розроблюваної платформи. Головними перевагами створюваного рішення є: відкрита мультикатегорійна модель (будь-яка сфера бізнесу), інтеграція штучного інтелекту для генерації контенту та аналітики, сповіщення в реальному часі через WebSocket, сучасний та адаптивний інтерфейс, підтримка темної та світлої теми інтерфейсу.

**Таблиця 1.1 — Порівняльний аналіз платформ**

| Функція | Calendly | SimplyBook | Booksy | Проект (розробка) |
|---|---|---|---|---|
| Мультикатегорійність | – | ✓ | – | ✓ |
| AI-функції | – | – | – | ✓ |
| Real-time сповіщення | частково | – | частково | ✓ |
| Українська мова | – | ✓ | – | ✓ |
| Відкритий API | частково | ✓ | – | ✓ |
| Темна тема | – | ✓ | ✓ | ✓ |
| Безкоштовний план | ✓ | ✓ | ✓ | ✓ |

# 2. ВИМОГИ, ЗАСОБИ ТА ТЕХНОЛОГІЇ РОЗРОБКИ

## 2.1. Функціональні вимоги до платформи

Розробка будь-якої інформаційної системи починається з формулювання чітких вимог. Функціональні вимоги описують, що система повинна робити, тобто конкретні функції та поведінку системи у відповідь на дії користувача.

Для розроблюваної платформи сформульовано наступні функціональні вимоги:

**Модуль автентифікації та авторизації:**

- реєстрація нових користувачів з вибором ролі (клієнт або власник бізнесу);
- вхід в систему за допомогою email і пароля;
- автентифікація через Google OAuth 2.0;
- відновлення паролю через email;
- розмежування доступу на основі ролей (RBAC).

**Модуль управління бізнесом:**

- реєстрація бізнесу з вказанням категорії, опису, контактів та адреси;
- налаштування графіку роботи по кожному дню тижня;
- додавання, редагування та видалення послуг з вказанням тривалості та вартості;
- завантаження логотипу та зображень бізнесу;
- генерація AI-опису для бізнесу та послуг.

**Модуль бронювань:**

- перегляд доступних часових слотів в інтерактивному календарі;
- перевірка доступності слоту в режимі реального часу;
- здійснення бронювання з заповненням контактних даних;

– підтвердження або скасування бронювання власником бізнесу;
– скасування або перенесення бронювання клієнтом;
– відповідно до налаштованої політики скасування.

**Модуль сповіщень:**

– push-сповіщення в реальному часі через WebSocket;
– email-повідомлення про підтвердження, скасування бронювань;
– автоматичні нагадування за 24 та 1 годину до запису;
– центр сповіщень з можливістю позначити як прочитані.

Нефункціональні вимоги до системи включають: продуктивність (час відповіді API не більше 500 мс), безпеку (JWT-автентифікація, шифрування паролів bcrypt, захист від ін'єкцій), масштабованість (архітектура дозволяє горизонтальне масштабування), адаптивність (коректне відображення на мобільних пристроях) та доступність (підтримка темної і світлої теми інтерфейсу)

## 2.2. Середовище і засоби розробки

Для розробки платформи було обрано сучасний технологічний стек, який забезпечує як ефективність розробки, так і якість кінцевого продукту. Стек включає технології для розробки клієнтської частини (фронтенд), серверної частини (бекенд) та бази даних.

React — це бібліотека JavaScript для побудови користувацьких інтерфейсів, розроблена та підтримувана компанією Meta. React використовує концепцію компонентного підходу, де інтерфейс розбивається на незалежні, перевикористовувані компоненти. Ключовою особливістю React є Virtual DOM — абстракція реального DOM-дерева, яка дозволяє мінімізувати кількість дорогих операцій з реальним DOM та забезпечує високу продуктивність. [9]

Для даного проєкту React є оптимальним вибором завдяки: великій екосистемі та спільноті, хорошій продуктивності при роботі з динамічними даними, компонентній архітектурі, що сприяє повторному використанню коду. Разом з React використовується Vite — сучасний інструмент збірки, який забезпечує миттєвий старт dev-сервера та оптимізовану збірку для продакшн.

Tailwind CSS — CSS-фреймворк із відкритим вихідним кодом, створеним Адамом Уетеном і підтримуваним Tailwind Labs. Особливість цієї бібліотеки в тому, що вона не передбачає CSS-класи окремих елементів. Замість цього вона надає службові класи, які можна об’єднати для стилізації кожного елемента. [10]

Node.js — це середовище виконання JavaScript на стороні сервера, побудоване на рушії V8 від Google Chrome. платформа з відкритим кодом для виконання високопродуктивних мережевих застосунків, написаних мовою JavaScript. Засновником платформи є Раян Дал (Ryan Dahl). Якщо раніше JavaScript застосовувався для обробки даних в браузері користувача, то node.js надав можливість виконувати JavaScript-скрипти на сервері та відправляти користувачеві результат їхнього виконання. Платформа Node.js перетворила JavaScript на мову загального використання з великою спільнотою розробників. [11]

Express.js — це мінімалістичний і гнучкий фреймворк для Node.js, який надає набір функцій для розробки веб- та мобільних застосунків. Express спрощує маршрутизацію HTTP-запитів, роботу з middleware та обробку помилок. Архітектура middleware в Express дозволяє будувати чистий, модульний код: кожен middleware відповідає за конкретне завдання (автентифікація, валідація, обробка помилок). [14]

MongoDB — це документно-орієнтована NoSQL база даних, що зберігає дані у форматі BSON (Binary JSON). MongoDB Atlas дозволяє користувачам готувати організації Atlas та керувати ними з Azure. Вона

спільно розробляється та керується корпорацією Майкрософт та MongoDB як власна інтеграція Azure. Використовуйте власні інтеграції Azure для легкої підготовки, управління та тісної інтеграції програмного забезпечення та служб із компаній з розробки програмного забезпечення в Azure.

MongoDB Atlas використовується для зберігання, керування та обробки операційних даних для хмарних програм. Він підтримує такі сценарії, як аналітика в режимі реального часу, повнотекстовий та векторний пошук, а також потокова обробка для програм, керованих подіями. [13]

Socket.io — це бібліотека для двостороннього, подієво-орієнтованого спілкування між браузером та сервером. Socket.io будується на основі технології WebSocket, проте надає додаткові можливості: автоматичне відновлення з'єднання, кімнати та простори імен, підтримку fallback на HTTP long-polling. У платформі Socket.io використовується для реалізації сповіщень у реальному часі та оновлення статусів бронювань без перезавантаження сторінки. [14]

Nodemailer — це модуль для Node.js, призначений для надсилання електронних листів. Бібліотека підтримує різні транспортні протоколи (SMTP, sendmail, SES) та формати листів (HTML, текст, вкладення). У платформі Nodemailer використовується для відправки підтверджень бронювання, автоматичних нагадувань та сповіщень про зміни статусу.

OpenAI API — це хмарний сервіс, що надає доступ до великих мовних моделей (LLM) компанії OpenAI, зокрема GPT-4o-mini. Інтеграція API дозволяє реалізувати функції штучного інтелекту без необхідності навчання власних моделей. Усі AI-функції реалізовані з graceful degradation: якщо API недоступний або ключ не налаштований, система продовжує коректно працювати без AI-функцій. [16]

JSON Web Token (JWT) — це відкритий стандарт (RFC 7519) для передачі інформації між сторонами у вигляді JSON-об'єкта. JWT є компактним, самодостатнім та безпечно підписаним токеном. У платформі

JWT використовується для реалізації stateless-автентифікації: після входу в систему сервер генерує токен з ідентифікатором та роллю користувача, який клієнт зберігає і передає з кожним запитом.

Паспортна система (Passport.js) — це middleware для Node.js, що надає гнучку та модульну автентифікацію. Passport підтримує понад 500 стратегій автентифікації, включаючи JWT, Google OAuth, Facebook та інші. Для платформи використовуються стратегії passport-jwt та passport-google-oauth20. [17]

# 3. ПРОЄКТУВАННЯ СИСТЕМИ

## 3.1. Загальна архітектура системи

Система побудована за архітектурним патерном клієнт-сервер з чітким розподілом відповідальності між рівнями. Фронтенд (React SPA) відповідає за відображення інтерфейсу та взаємодію з користувачем, бекенд (Node.js/Express REST API) — за бізнес-логіку та роботу з даними, база даних (MongoDB) — за зберігання даних.

Серверна частина організована відповідно до принципів чистої архітектури (Clean Architecture) та патерну MVC: моделі описують схеми даних MongoDB, контролери містять бізнес-логіку, маршрути визначають API-ендпоінти. Middleware-функції відповідають за наскрізні задачі: автентифікацію, авторизацію, валідацію, обмеження запитів та обробку помилок.

Клієнтська частина організована за принципом розділення відповідальності між шарами: сторінки (pages) — топрівневі компоненти React Router, компоненти (components) — перевикористовувані UI-елементи, контексти (contexts) — глобальний стан (Auth, Theme, Socket), сервіс (services) — обгортки над HTTP-запитами через Axios.

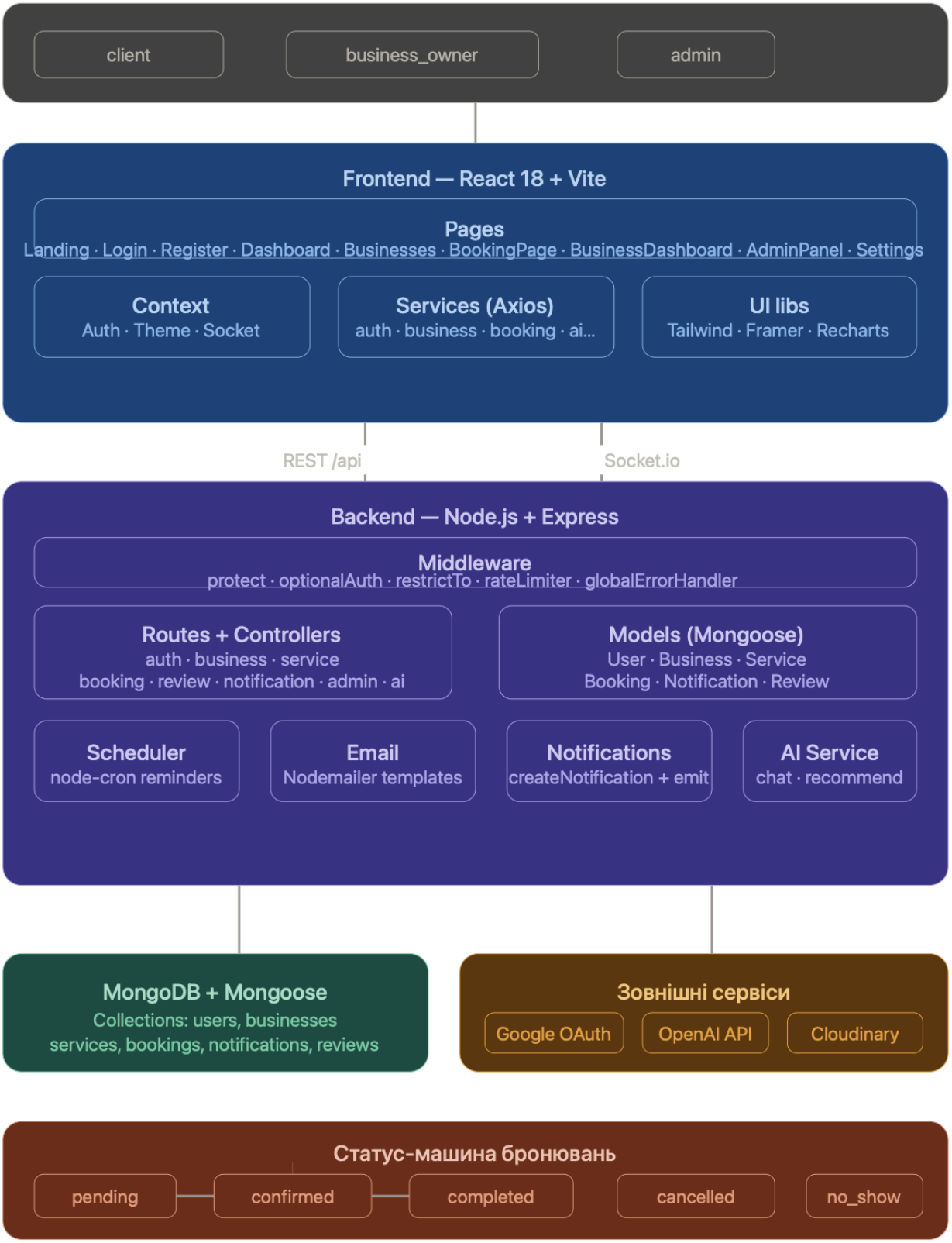


*Рисунок 3.1 — Загальна архітектура системи*

### 3.2. Діаграма використання (Use Case diagram)

Діаграма використання (Use Case diagram) є одним з типів UML-діаграм, що відображає функціональні вимоги до системи з точки зору зовнішніх акторів. Вона показує, що система може робити для кожного з її користувачів, не деталізуючи технічну реалізацію.

В системі передбачено три актори: Клієнт, Власник бізнесу та Адміністратор. Клієнт може виконувати наступні дії: реєструватися та авторизуватися в системі, переглядати каталог бізнесів з фільтрами та пошуком, переглядати профіль бізнесу та його послуги, здійснювати бронювання через інтерактивний календар, переглядати та керувати своїми бронюваннями, отримувати сповіщення та нагадування, залишати відгуки на завершені послуги.

Власник бізнесу, окрім функцій клієнта, може: реєструвати та налаштовувати власний бізнес, додавати та редагувати послуги, налаштовувати графік роботи, підтверджувати та скасовувати бронювання клієнтів, переглядати аналітику та статистику бізнесу, генерувати AI-описи для бізнесу та послуг. Адміністратор платформи має доступ до всіх функцій, а також може управляти користувачами, верифікувати бізнеси та переглядати

загальну статистику платформи.

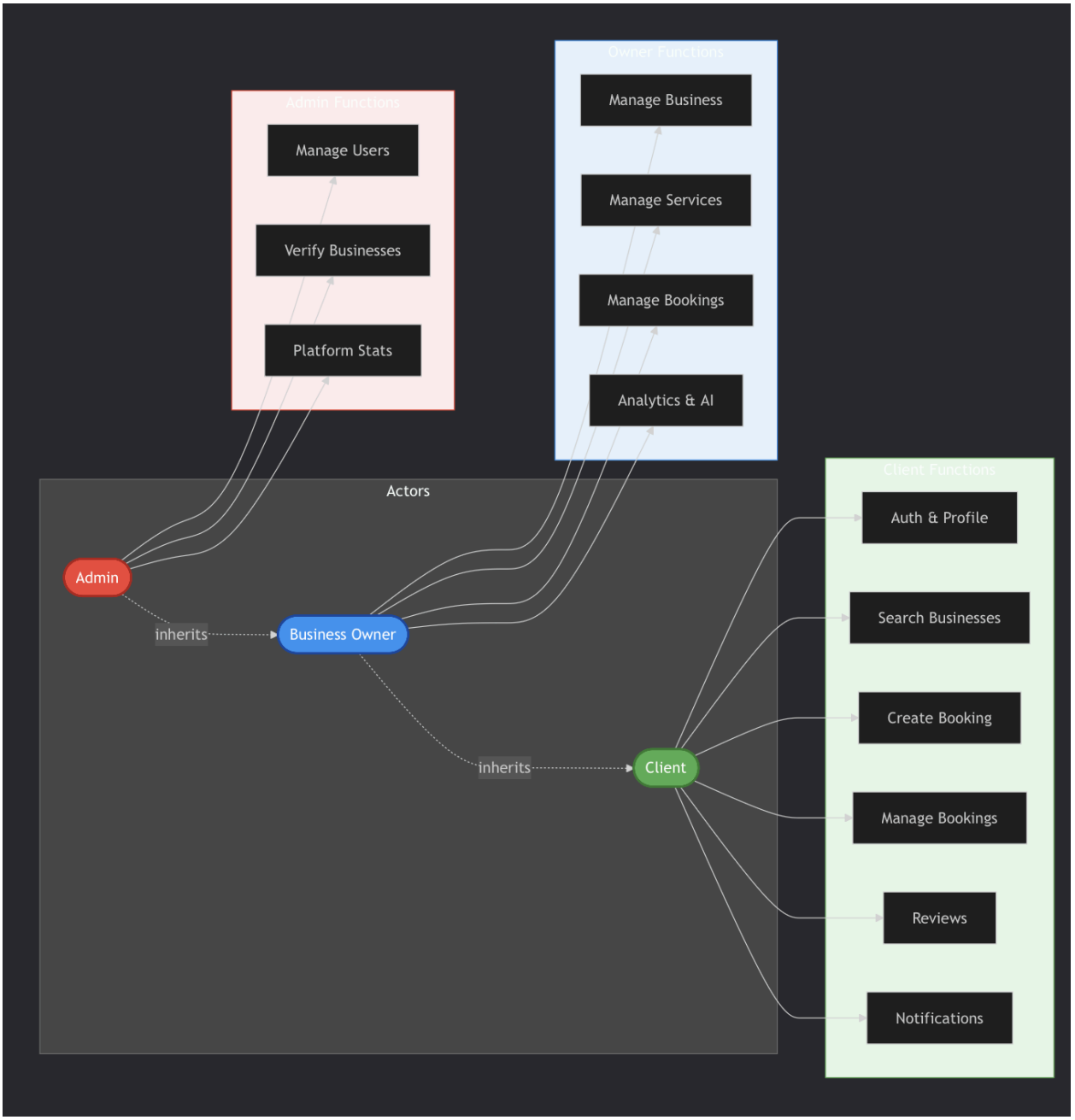


*Рисунок 3.2 — Діаграма використання (Use Case diagram)*

### 3.3. Діаграма послідовності

Діаграма послідовності відображає взаємодію між об'єктами системи у часі, показуючи порядок обміну повідомленнями між компонентами при виконанні певного сценарію.

На діаграмі послідовності зображено сценарій здійснення бронювання клієнтом. Клієнт відкриває сторінку бізнесу — фронтенд надсилає GET-запит до API для отримання даних бізнесу та послуг. Після вибору послуги та дати фронтенд запитує доступні часові слоти. Сервер перевіряє існуючі бронювання в MongoDB та повертає список вільних слотів. Клієнт обирає слот та підтверджує бронювання — POST-запит до /api/bookings. Сервер перевіряє доступність слоту атомарно (щоб уникнути race condition), створює запис в MongoDB, надсилає email-підтвердження та через Socket.io сповіщає власника бізнесу про нове бронювання.

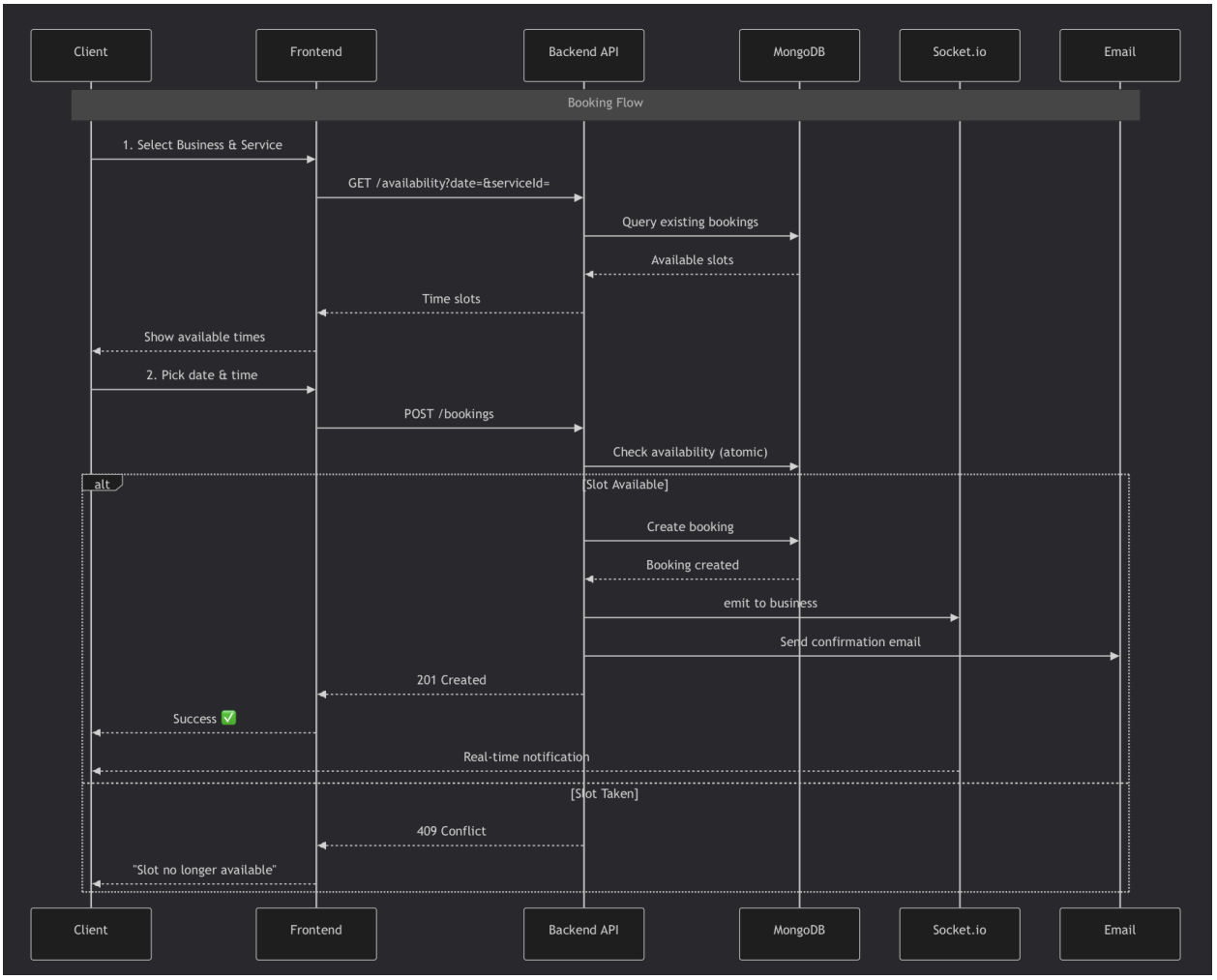


*Рисунок 3.3 — Діаграма послідовності (сценарій бронювання)*

### 3.4. ER-діаграма бази даних

ER-діаграма (Entity-Relationship diagram) відображає структуру бази даних: сутності, їхні атрибути та зв'язки між ними. В MongoDB як документно-орієнтованій БД зв'язки реалізуються через посилання (ObjectId references) між документами.

Основні сутності системи та їхні відносини: User (користувач) є власником (one-to-many) Business; Business (бізнес) містить (one-to-many) Service (послугу); User-клієнт може мати (one-to-many) Booking (бронювання); Booking посилається на Business та Service; Notification (сповіщення) належить User; Review (відгук) прив'язаний до конкретного Booking (one-to-one).

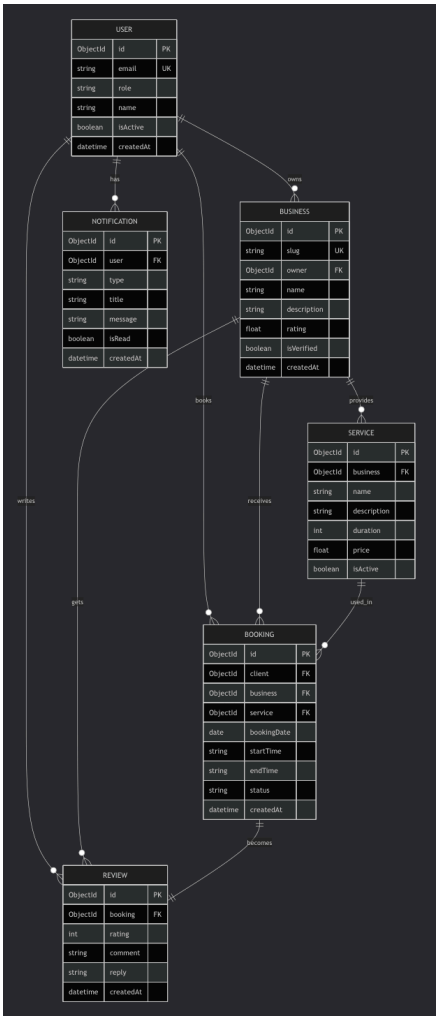


*Рисунок 3.4 — ER-діаграма бази даних*

# 4. ПРОГРАМНА РЕАЛІЗАЦІЯ ПЛАТФОРМИ

## 4.1. Автентифікація та система ролей

Автентифікація є фундаментальним модулем будь-якої багатокористувацької системи. Для платформи реалізовано два механізми автентифікації: класичний (email + пароль) та OAuth 2.0 через Google. Обидва механізми управляються через Passport.js та генерують JWT-токени для підтримки stateless-сесій.

Модель користувача (User) успадковує базові поля від Mongoose Schema та містить наступні ключові поля: унікальний ідентифікатор, ім'я, email (унікальний), хешований пароль (bcrypt, 12 раундів), googleId для OAuth-автентифікації, роль (admin, business_owner, client), масив улюблених бізнесів та налаштування сповіщень. Модель включає методи для порівняння паролів та генерації JWT-токенів:

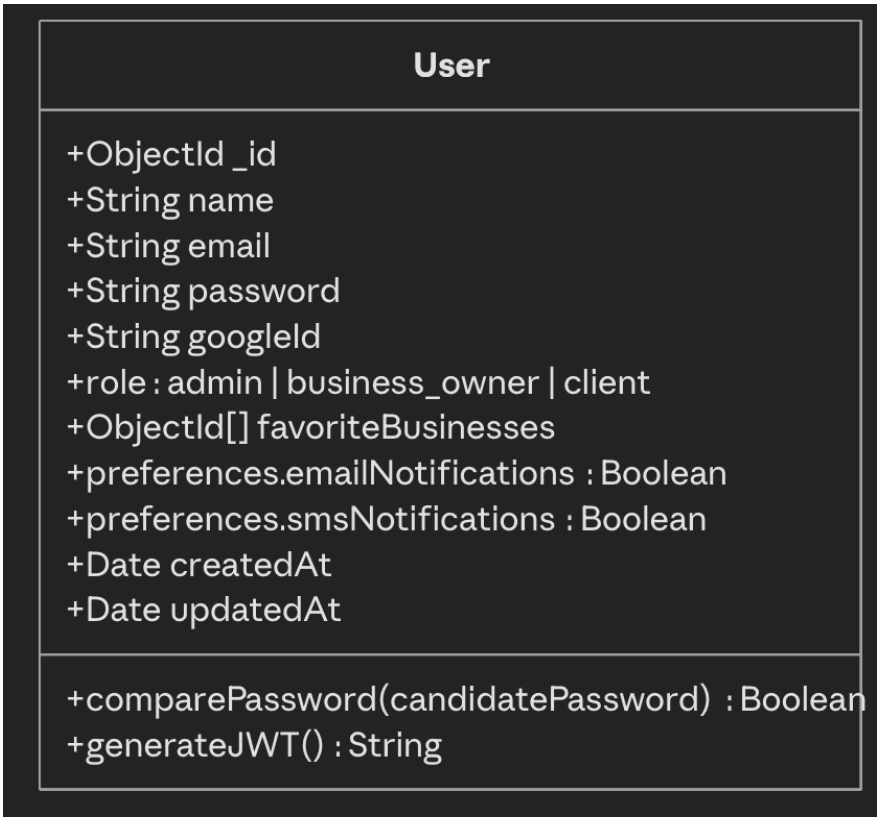


*Рисунок 4.1 — Модель User*

Нижче наведено ключовий фрагмент моделі User з методом генерації JWT-токена:

```
// Метод генерації JWT-токена
userSchema.methods.generateJWT = function() {
  return jwt.sign(
    { id: this._id, role: this.role, email: this.email },
    process.env.JWT_SECRET,
    { expiresIn: process.env.JWT_EXPIRES_IN || "7d" }
  );
};
```

Middleware автентифікації (protect) перехоплює кожен запит до захищених маршрутів та верифікує JWT-токен через Passport.js. У разі успішної верифікації об'єкт користувача додається до req.user для використання в контролерах:

```
// Захист маршрутів через JWT
const protect = (req, res, next) => {
  passport.authenticate("jwt", { session: false }, (err,
user) => {
    if (!user) return next(new ApiError("Authentication
required", 401));
    req.user = user;
    next();
  })(req, res, next);
};
```

Рольова система (RBAC — Role-Based Access Control) реалізована через middleware restrictTo, який перевіряє роль поточного користувача перед виконанням контролера. Це забезпечує захист адміністративних та бізнес-функцій від несанкціонованого доступу:

```
const restrictTo = (...roles) => (req, res, next) => {
  if (!roles.includes(req.user.role)) {
    return next(new ApiError("Access denied", 403));
  }
  next();
};
```

Контролер реєстрації обробляє POST-запит, перевіряє унікальність email, хешує пароль, створює запис у базі даних та повертає JWT-токен. Після реєстрації система автоматично надсилає вітальний лист через Nodemailer та створює внутрішнє сповіщення.

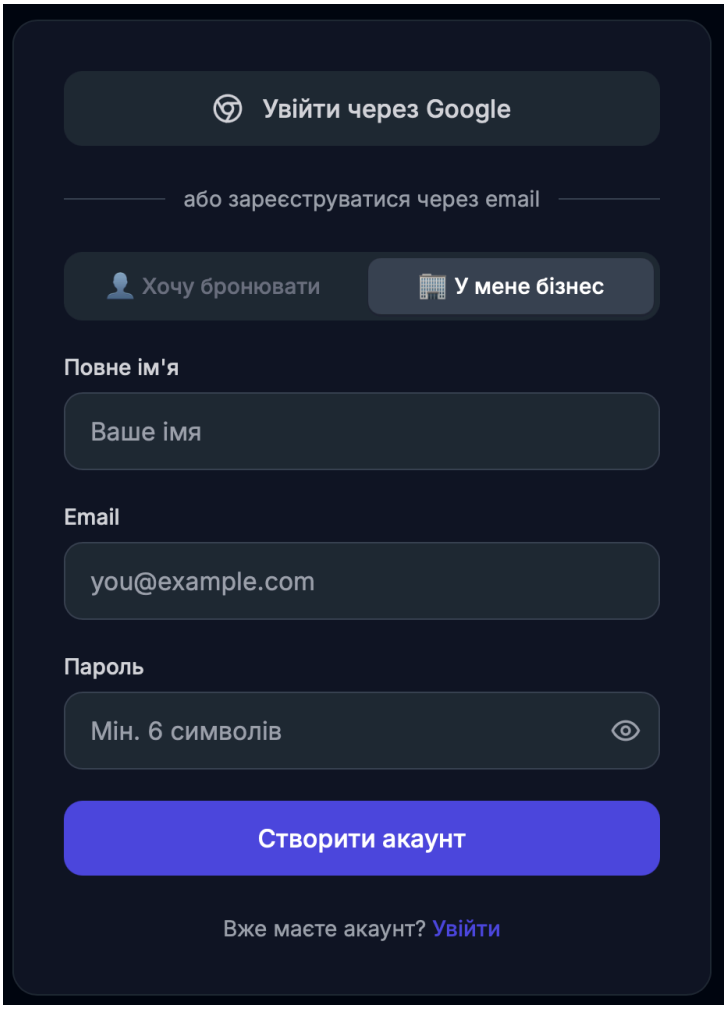


*Рисунок 4.2 — Сторінка реєстрації*

### 4.2. Управління бізнесами та послугами

Модуль управління бізнесами дозволяє власникам реєструвати своє підприємство, налаштовувати послуги та графік роботи. Це ключовий модуль платформи з точки зору business-логіки.

Модель Business містить повну інформацію про підприємство: назву, категорію, опис, адресу, контакти, логотип, зображення, графік роботи (workingHours) та налаштування бронювання (settings). Особливо важливою є

структура workingHours — масив об'єктів для кожного дня тижня з часом відкриття/закриття та переліком перерв:

```
// Схема робочих годин
const workingHoursSchema = new mongoose.Schema({
  day: { type: String, enum: ["monday",...,"sunday"] },
  isOpen: { type: Boolean, default: true },
  openTime: { type: String, default: "09:00" },
  closeTime: { type: String, default: "18:00" },
  breaks: [{ start: String, end: String }]
});
```

Налаштування бізнесу (settings) включають: буферний час між бронюваннями (bookingBuffer), максимальна кількість днів наперед для бронювання (advanceBooking), кількість годин до запису для скасування (cancellationPolicy) та опція автоматичного підтвердження бронювань (autoConfirm).

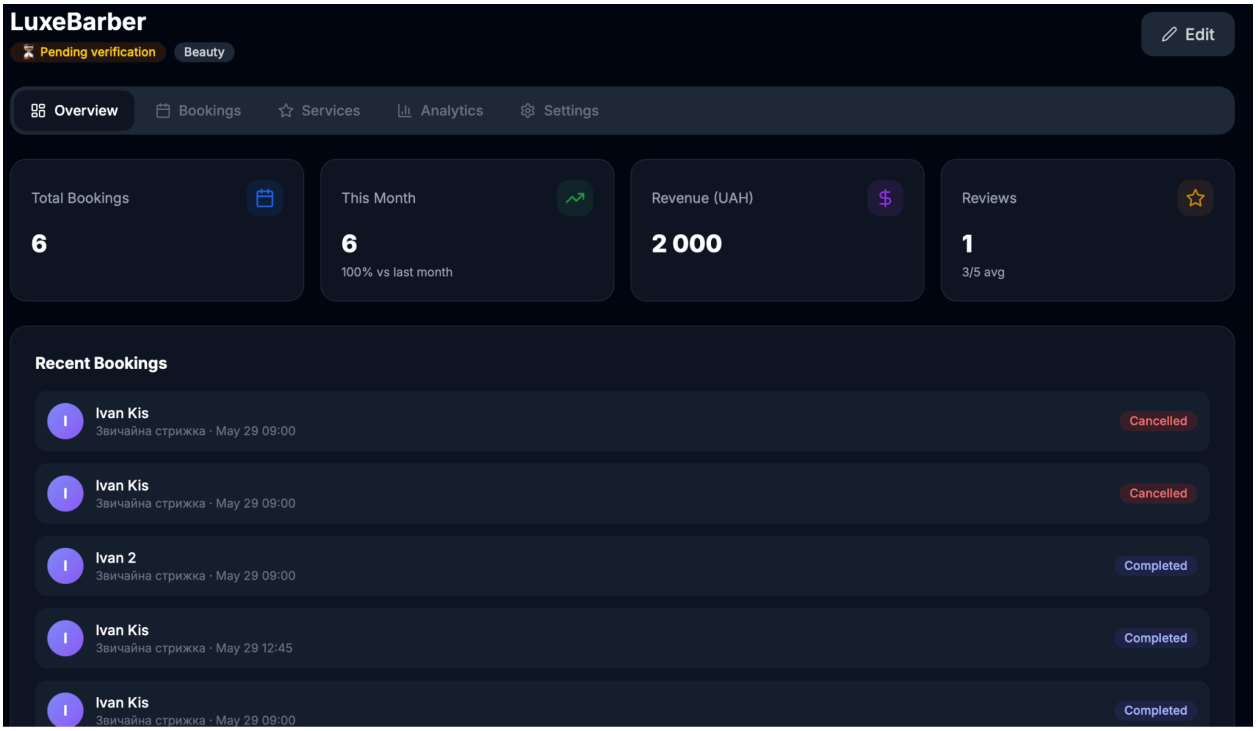


*Рисунок 4.4 — Панель управління власника бізнесу*

Важливою функцією є генерація AI-опису для бізнесу. При натисканні кнопки "Згенерувати з AI" фронтенд надсилає запит до

/api/businesses/:id/ai-description, сервер формує промпт з назвою бізнесу, категорією та переліком послуг, і відправляє його до OpenAI API:

```
const generateBusinessDescription = async ({ name,
category, services }) => {
  const prompt = `Generate a professional description for:
    Business: ${name}, Category: ${category},
    Services: ${services.join(", ")}`;
  const response = await client.chat.completions.create({
    model: "gpt-4o-mini",
    messages: [{ role: "user", content: prompt }],
    max_tokens: 200
  });
  return response.choices[0].message.content;
};
```

Усі AI-функції реалізовані з graceful degradation: якщо OPENAI_API_KEY не налаштований або API недоступний, функція повертає дефолтний текст без генерації виключення, що забезпечує стабільну роботу платформи у будь-яких умовах.

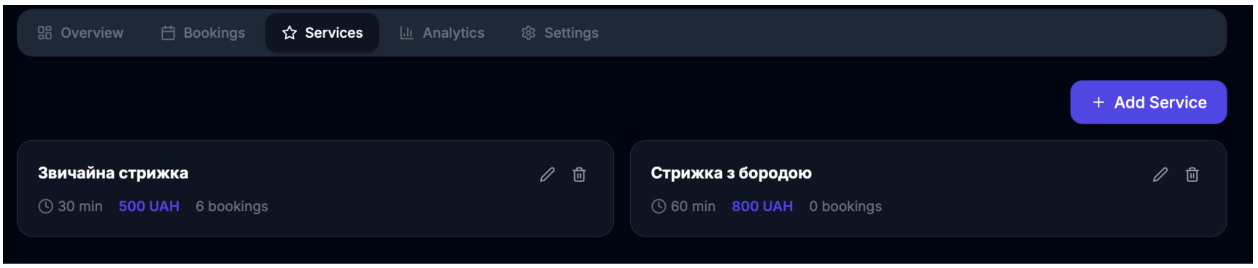


*Рисунок 4.6 — Управління послугами*

### 4.3. Система бронювання та календар

Система бронювання є центральним модулем платформи. Вона реалізує тристепове бронювання: вибір послуги, вибір дати та часу в інтерактивному календарі, підтвердження деталей.

Генерація доступних часових слотів є критично важливою функцією, що враховує багато факторів: час відкриття та закриття бізнесу, тривалість послуги з буферним часом та часом прибирання, наявні бронювання на обраний день, перерви в роботі, максимальний горизонт бронювання.

Алгоритм генерації слотів реалізований у функції generateSlots контролера бізнесів:

```
function generateSlots(openTime, closeTime, duration,
bookings, breaks) {
  const slots = [];
  const [openH, openM] = openTime.split(":").map(Number);
  let current = openH * 60 + openM;
  const end = closeH * 60 + closeM;
  while (current + duration <= end) {
    const startStr = formatTime(current);
    const endStr = formatTime(current + duration);
    const isBooked = bookings.some(
      b => b.startTime < endStr && b.endTime > startStr
    );
    if (!isBooked && !isInBreak(startStr, breaks)) {
      slots.push({ startTime: startStr, endTime: endStr });
    }
    current += duration;
  }
  return slots;
}
```

Для перевірки доступності слоту при збереженні бронювання використовується статичний метод Booking.checkAvailability, який виконує атомарний запит до бази даних. Це запобігає ситуації подвійного бронювання (race condition), коли два клієнти одночасно намагаються зайняти один слот:

```
bookingSchema.statics.checkAvailability = async function(
  businessId, serviceId, date, startTime, endTime,
excludeId
) {
  const conflict = await this.findOne({
    business: businessId,
    date: { $gte: startOfDay, $lte: endOfDay },
    status: { $in: ["pending", "confirmed"] },
    $or: [{ startTime: { $lt: endTime }, endTime: { $gt:
startTime } }]
  });
  return !conflict;
};
```

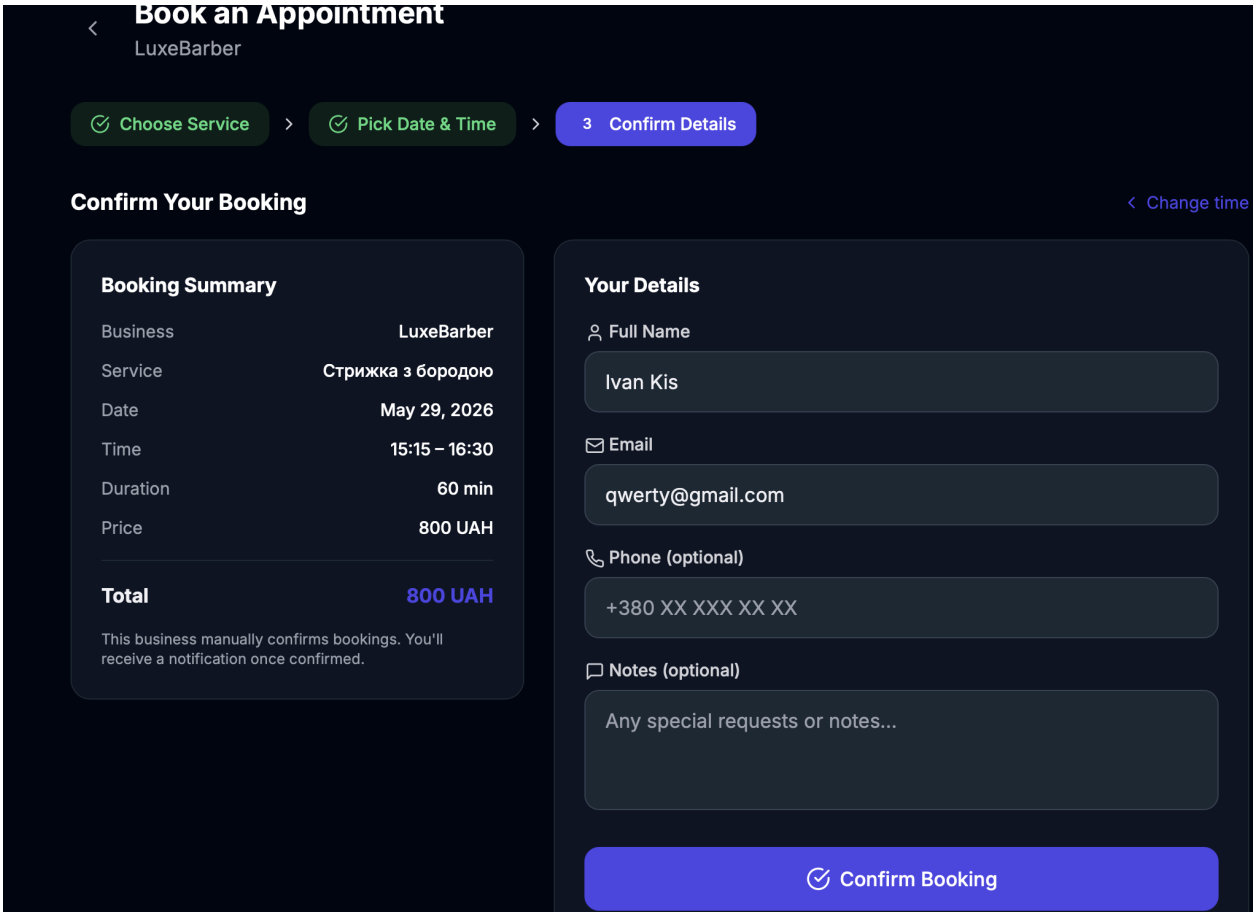


*Рисунок 4.7 — Сторінка бронювання*

Управління бронюваннями для власника бізнесу включає функції підтвердження, відмови та позначення виконаних записів. При підтвердженні або скасуванні бронювання система автоматично надсилає email-повідомлення клієнту та push-сповіщення через Socket.io.

### 4.4. Сповіщення в реальному часі

Для реалізації сповіщень у реальному часі використовується Socket.io — бібліотека, що забезпечує двостороннє спілкування між сервером та клієнтом через протокол WebSocket з fallback на HTTP polling.

Архітектура Socket.io в проєкті будується на концепції кімнат (rooms): кожен користувач при підключенні автоматично приєднується до особистої кімнати user:{userId}, а власник бізнесу може підписатися на кімнату

business:{businessId} для отримання сповіщень про нові бронювання. Для безпеки Socket.io-підключення вимагає валідний JWT-токен у auth-параметрах:

```
io.use((socket, next) => {
  const token = socket.handshake.auth?.token;
  try {
    const decoded = jwt.verify(token,
process.env.JWT_SECRET);
    socket.userId = decoded.id;
    next();
  } catch (err) {
    next(new Error("Invalid token"));
  }
});
```

На клієнті Socket.io інтегрований через SocketContext — React Context, який ініціалізує з'єднання при наявності JWT-токена та передає об'єкт socket компонентам через хук useSocket. Контекст відповідає за відображення toast-сповіщень та оновлення лічильника непрочитаних повідомлень у навігаційній панелі.

Система автоматичних email-нагадувань реалізована через node-cron — бібліотеку планування задач для Node.js. Cron-задача запускається щогодини та перевіряє бронювання, для яких потрібно надіслати нагадування (за 24 або 1 годину). Задача також позначає нагадування як відправлені, щоб уникнути дублювання:

```
cron.schedule("0 * * * *", async () => {
  const bookings = await Booking.find({
    date: { $gte: tomorrowStart, $lte: tomorrowEnd },
    status: "confirmed",
    "reminders.dayBefore.sent": false
  }).populate("client service business");
  for (const booking of bookings) {
    await sendEmail({ template: "bookingReminder", ... });
    await Booking.updateOne({ _id: booking._id },
      { "reminders.dayBefore.sent": true });
  }
});
```

### 4.5. Функції на основі штучного інтелекту

Інтеграція штучного інтелекту є однією з ключових відмінностей розроблюваної платформи від аналогів. Для реалізації AI-функцій використовується OpenAI API з моделлю GPT-4o-mini, яка забезпечує оптимальний баланс між якістю генерації та вартістю запитів.

AI-асистент для клієнтів реалізований у вигляді чат-інтерфейсу, де користувач може отримати допомогу у виборі послуги, дізнатися про можливості платформи або отримати відповідь на будь-яке питання. Асистент отримує контекст поточного бізнесу та доступних послуг для надання релевантних відповідей. Повна історія діалогу передається з кожним запитом для збереження контексту розмови.

AI-рекомендації будуються на основі рейтингу та популярності бізнесів у базі даних, поточної категорії та місця розташування, без необхідності складних ML-моделей. Функція аналітики бізнесу збирає агреговані дані зі статистики бронювань (пікові години, розподіл за статусами, популярні послуги) та формує текстові інсайти через OpenAI API:

```
const analyzeBusiness = async (businessId, userId) => {
  const [bookingStats, peakHours] = await Promise.all([
    Booking.aggregate([...]),
    Booking.aggregate([...])
  ]);
  const prompt = `Analyze booking data: ${statsText}.
    Provide 3 actionable insights.`;
  const insight = await openai.chat.completions.create({
    model: "gpt-4o-mini",
    messages: [{ role: "user", content: prompt }]
  });
  return { bookingStats, peakHours, aiInsight: insight };
};
```

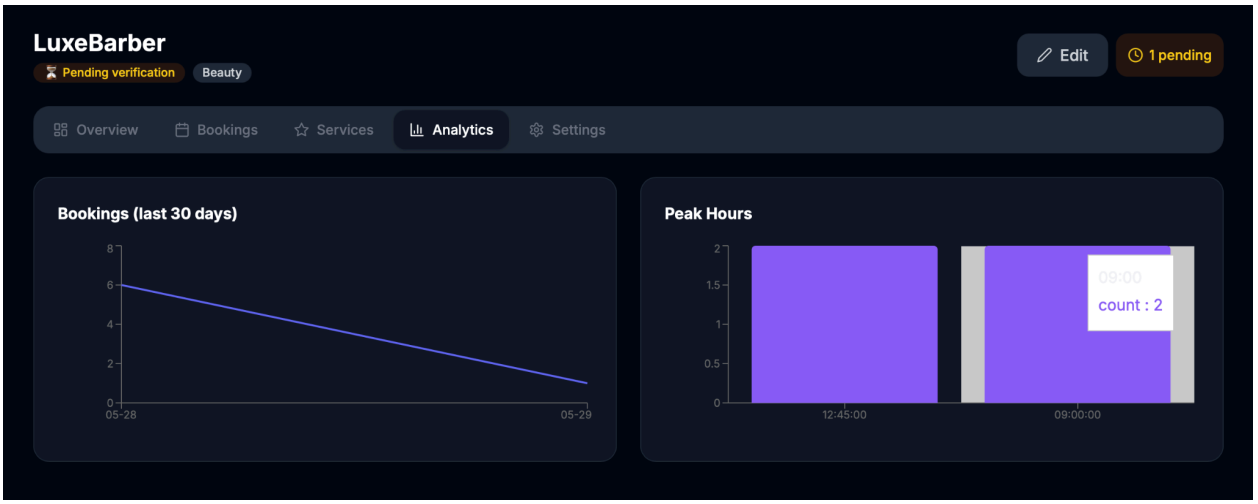


*Рисунок 4.13 — Секція аналітики з AI-інсайтами*

### 4.6. Адміністративна панель

Адміністративна панель надає суперкористувачам (role: admin) інструменти для управління платформою. Доступ до адмін-панелі захищено подвійним middleware: protect (перевірка JWT) та restrictTo("admin") (перевірка ролі).

Панель відображає загальну статистику платформи: кількість зареєстрованих користувачів, активних бізнесів, загальна кількість бронювань та сукупний дохід платформи. Усі дані агрегуються в реальному часі з MongoDB через MongoDB Aggregation Pipeline.

Управління користувачами дозволяє переглядати список всіх користувачів з фільтрацією за роллю та пошуком за іменем/email, а також активувати та деактивувати облікові записи. Управління бізнесами включає перегляд усіх зареєстрованих підприємств з детальною інформацією та верифікацію бізнесів — присвоєння відмітки "Верифіковано", яка відображається на публічному профілі.

Безпека платформи забезпечується комплексом заходів: паролі хешуються через bcrypt з 12 раундами солі, що унеможливлює їх відновлення навіть при витоку бази даних. JWT-токени мають обмежений термін дії (7 днів). Всі вхідні дані санітизуються через express-mongo-sanitize для захисту від NoSQL-ін'єкцій. Rate limiting обмежує кількість запитів: 100 запитів за 15

хвилин для загального API та 10 запитів за 15 хвилин для ендпоінтів автентифікації. Helmet.js встановлює безпечні HTTP-заголовки, що захищає від поширених атак (XSS, clickjacking).

# ВИСНОВКИ

В ході виконання кваліфікаційної роботи було проведено детальний аналіз предметної області та існуючих платформ для онлайн-бронювання послуг. Аналіз показав, що наявні рішення мають ряд обмежень: відсутність україномовного інтерфейсу, вузька галузева спеціалізація, висока вартість для малого бізнесу та відсутність сучасних AI-функцій.

На основі проведеного аналізу сформульовано функціональні та нефункціональні вимоги до розроблюваної системи, обрано відповідний технологічний стек. Архітектура платформи базується на принципах розподіленого клієнт-серверного додатку: React SPA на фронтенді, Node.js/Express REST API на бекенді та MongoDB як основна база даних.

Систему спроєктовано з використанням UML-діаграм (Use Case, послідовності, ER-діаграма, компонентів), що забезпечило чітке розуміння архітектури перед початком програмної реалізації. Проєктування дозволило виявити потенційні проблеми (зокрема, race condition при одночасному бронюванні) та закласти відповідні механізми їх вирішення на рівні архітектури.

У результаті виконаної роботи розроблено повнофункціональну платформу для онлайн-бронювання послуг, яка включає наступні модулі: систему автентифікації та авторизації з підтримкою Google OAuth 2.0 та JWT, рольову систему доступу (admin, business_owner, client), модуль управління бізнесом з налаштуванням графіку роботи та послуг, інтерактивний календар бронювань з алгоритмом генерації вільних слотів та захистом від подвійного бронювання, панель управління для власника бізнесу з аналітикою та графіками завантаженості, систему сповіщень у реальному часі на базі Socket.io, автоматичні email-нагадування через cron-задачі, функції на основі штучного інтелекту (AI-асистент, генерація описів, аналіз бізнес-даних).

## СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ


1. Mell, P., & Grance, T. (2011). The NIST Definition of Cloud Computing. National Institute of Standards and Technology. – Режим доступу: https://csrc.nist.gov/publications/detail/sp/800-145/final

2. Sommerville, I. (2016). Software Engineering (10th ed.). Pearson Education. – 809 с.

3. Calendly [Електронний ресурс] – Режим доступу: https://calendly.com/

4. Calendly Review: Pros, Cons and Features [Електронний ресурс] – Режим доступу: https://www.techradar.com/best/best-calendly-alternatives

5. SimplyBook.me [Електронний ресурс] – Режим доступу: https://simplybook.me/

6. Booksy [Електронний ресурс] – Режим доступу: https://booksy.com/

7. Wiegers, K., & Beatty, J. (2013). Software Requirements (3rd ed.). Microsoft Press. – 672 с.

8. SaaS як приклад [Електронний ресурс] – https://www.oracle.com/ua/applications/what-is-saas/

9. React Documentation [Електронний ресурс] – Режим доступу: https://react.dev/

10. Tailwind CSS [Електронний ресурс] – https://ru.wikipedia.org/wiki/Tailwind_CSS

11. Використання Node js [Електронний ресурс] – https://uk.wikipedia.org/wiki/Node.js

12. Express.js Documentation [Електронний ресурс] – Режим доступу: https://expressjs.com/

13. MongoDB Atlas [Електронний ресурс] – https://learn.microsoft.com/ru-ru/azure/partner-solutions/mongo-db/overview

14. Socket.io Documentation [Електронний ресурс] – Режим доступу: https://socket.io/docs/v4/

15. Nodemailer Documentation [Електронний ресурс] – Режим доступу: https://nodemailer.com/about/
16. OpenAI API Documentation [Електронний ресурс] – Режим доступу: https://platform.openai.com/docs
17. Passport.js Documentation [Електронний ресурс] – Режим доступу: https://www.passportjs.org/docs/
19. Subramanian, S. (2019). Practical MEAN Stack: Architecting and Building Modern JavaScript Applications. Apress.
20. Holmes, S. (2019). Getting MEAN with Mongo, Express, Angular, and Node (2nd ed.). Manning Publications.