МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра комп'ютерних наук

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему: **«РОЗРОБКА ВЕБЗАСТОСУНКУ ДЛЯ ОНЛАЙН ОРЕНДИ ОБЛАДНАННЯ ТА ІНСТРУМЕНТІВ З БРОНЮВАННЯМ, ОПЛАТОЮ ТА ВІДСТЕЖЕННЯМ ДОСТУПНОСТІ»**

Виконав: студент 4 курсу, групи 6.1222
спеціальності 122 Комп'ютерні науки
(шифр і назва спеціальності)
освітньої програми Комп'ютерні науки
(назва освітньої програми)
А. О. Зорін
(ініціали та прізвище)
Керівник доцент кафедри комп'ютерних наук, к.т.н Добровольський Г.А.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)
Рецензент доцент кафедри програмної інженерії, доцент, к.ф.-м.н. Мильцев О. М.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя 2026

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра комп'ютерних наук

Рівень вищої освіти бакалавр

Спеціальність 122 Комп'ютерні науки
(шифр і назва)

Освітня програма Комп'ютерні науки

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри комп'ютерних наук, к.пед.н., доцент

_____ Пшенична О.С.
(підпис)

" ____ " _____ 2025 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Зоріну Артему Олександровичу
(прізвище, ім'я та по-батькові)

1. Тема роботи Розробка вебзастосунку для онлайноренди обладнання та інструментів з бронюванням, оплатою та відстеженням доступності

керівник роботи Добровольський Геннадій Анатолійович, к.т.н., доцент
(прізвище, ім'я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 02 » грудня 2025 року № 1741-с

2. Строк подання студентом роботи 05.06.2026

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Теоретичні основи розробки вебзастосунків для онлайн-оренди та бронювання ресурсів
2. Проєктування вебзастосунку для онлайн-оренди обладнання та інструментів.
3. Розробка та тестування вебзастосунку.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
презентація

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання 03.12.2025

К А Л Е Н Д А Р Н И Й П Л А Н

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	15.02.2026	
2.	Збір вихідних даних.	02.03.2026	
3.	Обробка методичних та теоретичних	14.03.2026	
	джерел.		
4.	Розробка першого та другого розділу.	23.04.2026	
5.	Розробка третього розділу.	14.05.2026	
6.	Оформлення та нормоконтроль	05.06.2026	
	кваліфікаційної роботи бакалавра.		
7.	Захист кваліфікаційної роботи.	18.06.2026	

Студент _____ А. О. Зорін
(підпис) (ініціали та прізвище)

Керівник роботи _____ Г. А. Добровольський
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ О.Г.Спиця
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота бакалавра «Розробка вебзастосунку для онлайноренди обладнання та інструментів з бронюванням, оплатою та відстеженням доступності»: 94 с., 21 рис., 7 табл., 23 джерел, 3 додатків.

БАЗА ДАНИХ, БРОНЮВАННЯ, ВЕБЗАСТОСУНОК, ЕЛЕКТРОННА КОМЕРЦІЯ, ОБЛАДНАННЯ, ОНЛАЙН-ОРЕНДА, ОПЛАТА, JAVASCRIPT, TYPESCRIPT, UNIT TESTING, WEB APPLICATION.

Об’єктом дослідження є процес організації онлайн-оренди обладнання та інструментів.

Предметом дослідження є методи, моделі та програмні засоби розробки вебзастосунків для автоматизації процесів оренди, бронювання, оплати та контролю доступності обладнання.

Метою роботи є розробка вебзастосунку для онлайн-оренди обладнання та інструментів, що автоматизує процеси бронювання, розрахунок доступності в реальному часі та облік фінансових операцій, підвищуючи зручність користування системою.

Методи дослідження включають аналіз предметної області, порівняння існуючих рішень, методи проєктування ПЗ, об’єктно-орієнтований підхід, моделювання баз даних і модульне тестування з використанням Jest.

У роботі проаналізовано сучасні вебсистеми оренди, визначено вимоги до платформи та особливості обробки даних у реальному часі. Спроєктовано архітектуру системи та базу даних, що охоплює користувачів, обладнання, категорії, бронювання та платежі. Реалізовано основні модулі: реєстрацію, бронювання, розрахунок вартості та контроль доступності обладнання. Проведено модульне тестування, яке підтвердило коректність роботи системи та її практичну придатність.

SUMMARY

Bachelor's Qualifying Thesis «Development o f a Web Application for Online Rental of Equipment and Tools with Booking, Payment and Availability Tracking»: 94 pages, 21 figures, 7 tables, 23 sources, 3 supplements.

DATABASE, BOOKING, WEB APPLICATION, E-COMMERCE, EQUIPMENT, ONLINE RENTAL, PAYMENT, JAVASCRIPT, TYPESCRIPT, UNIT TESTING, WEB APPLICATION.

The object of research is the process of organizing online rental of equipment and tools.

The subject of research is methods, models, and software tools for developing web applications aimed at automating rental processes, booking, payment, and equipment availability control.

The aim of the work is to develop a web application for online rental of equipment and tools that automates booking processes, real-time availability calculation, and financial transaction management, improving overall user experience and system usability.

The research methods include domain analysis, comparison of existing solutions, software design methods, object-oriented approach, database modeling, and unit testing using the Jest framework.

The work analyzes modern web-based rental systems, defines platform requirements and real-time data processing features. The system architecture and database structure are designed, covering users, equipment, categories, bookings, and payments. The main modules are implemented, including registration, booking, cost calculation, and equipment availability control. Unit testing was conducted, confirming the correctness of the system and its practical applicability.

ЗМІСТ

Завдання на кваліфікаційну роботу……..2
Реферат 4
Summary 5
Вступ..... 8
1 Теоретичні основи розробки вебзастосунків для онлайн-оренди та бронювання ресурсів..... 10
1.1 Поняття та особливості вебзастосунків для систем оренди та бронювання..... 10
1.2 Аналіз сучасних систем онлайн-оренди обладнання та інструментів…..... 12
1.3 Постановка мета та завдання роботи 17
2 Проєктування вебзастосунку для онлайн-оренди обладнання та інструментів 18
2.1 Визначення вимог до вебзастосунку та побудова архітектури системи…..... 18
2.2 Проєктування бази даних та логіки управління бронюваннями й оплатами..... 23
2.3 Проєктування користувацького інтерфейсу та вибір технологій реалізації.. 32
3 Розробка та тестування вебзастосунку 37
3.1 Реалізація функціоналу системи (реєстрація, оренда, бронювання, оплата)…..... 37
3.2 Розробка модуля відстеження доступності обладнання в реальному часі…… 44
3.3 Тестування вебзастосунку та аналіз результатів роботи системи 50
3.4 Інструкція користувача..... 52
Висновки 54

Перелік посилань..... 58
Додаток А Коди для реалізації функціоналу системи..... 61
Додаток Б Лістинг для розробки модуля відстеження доступності обладнання в реальному часі 76
Додаток В Інтерфейс розробленого вебзастосунку..... 90

ВСТУП

Сучасний розвиток інформаційних технологій та цифровізація бізнес-процесів сприяють активному впровадженню веборієнтованих систем для автоматизації різноманітних видів діяльності. Одним із перспективних напрямів є створення сервісів онлайн-оренди обладнання та інструментів, які дозволяють користувачам швидко знаходити необхідні ресурси, здійснювати їх бронювання, оплачувати послуги та контролювати доступність обладнання через мережу Інтернет.

Сфера оренди обладнання охоплює широкий спектр інструментів і технічних засобів, що використовуються у будівництві, ремонті, виробництві та побутовій діяльності. Традиційні методи організації оренди часто передбачають телефонні дзвінки, особисте відвідування пунктів прокату та ручне ведення обліку замовлень. Такий підхід може призводити до помилок, дублювання бронювань, неактуальності інформації про наявність обладнання та додаткових витрат часу як для клієнтів, так і для працівників підприємства.

Використання сучасних вебтехнологій дає можливість автоматизувати основні процеси оренди, забезпечити цілодобовий доступ до сервісу, спростити взаємодію між користувачами та адміністраторами системи, а також підвищити ефективність управління ресурсами. Особливого значення набуває реалізація функцій онлайн-бронювання, електронної оплати та відстеження доступності обладнання в режимі реального часу, що дозволяє зробити процес оренди більш прозорим і зручним.

Актуальність теми зумовлена зростанням попиту на цифрові сервіси у сфері оренди та необхідністю створення комплексних інформаційних систем, які поєднують функції керування обладнанням, бронюванням, оплатою та моніторингом доступності ресурсів. Розробка такого вебзастосунку сприятиме

підвищенню якості обслуговування клієнтів, оптимізації бізнес-процесів та ефективнішому використанню наявного обладнання.

Об’єктом дослідження є процес організації онлайн-оренди обладнання та інструментів.

Предметом дослідження є методи, моделі та програмні засоби розробки вебзастосунків для автоматизації процесів оренди, бронювання, оплати та контролю доступності обладнання.

Практичне значення роботи полягає у створенні вебзастосунку, який може бути використаний підприємствами та організаціями, що надають послуги оренди обладнання та інструментів. Запропоноване рішення дозволяє автоматизувати процеси обробки замовлень, забезпечити актуальність інформації про наявність ресурсів та підвищити ефективність взаємодії з клієнтами.

Результати дослідження можуть бути використані для подальшого розвитку інформаційних систем у сфері оренди обладнання та інших сервісів спільного використання ресурсів.

1 ТЕОРЕТИЧНІ ОСНОВИ РОЗРОБКИ ВЕБЗАСТОСУНКІВ ДЛЯ ОНЛАЙН-ОРЕНДИ ТА БРОНЮВАННЯ РЕСУРСІВ

1.1 Поняття та особливості вебзастосунків для систем оренди та бронювання

Вебзастосунки для систем оренди та бронювання є сучасним різновидом інформаційних систем, що забезпечують автоматизацію процесів збору, обробки, зберігання та надання інформації користувачам з метою підтримки управлінських рішень і організації сервісної діяльності. У загальному розумінні інформаційна система являє собою сукупність взаємопов’язаних компонентів, які інтегрують дані, технології та процеси для ефективного управління інформацією в організації та забезпечення її використання у бізнес-процесах [1]. У цьому контексті вебзастосунки для оренди та бронювання виступають як приклад прикладних інформаційних систем, орієнтованих на автоматизацію взаємодії між постачальником послуг та кінцевим користувачем.

З погляду функціонального призначення такі системи належать до інформаційно-пошукових і довідкових систем, основною метою яких є організація швидкого доступу до структурованих даних, їх обробка та надання релевантної інформації користувачу. Інформаційно-пошукові системи характеризуються здатністю здійснювати пошук, індексацію та вибірку необхідних даних із великих масивів інформації, виступаючи посередником між користувачем і інформаційними ресурсами [2]. У випадку систем оренди це проявляється у можливості швидко знаходити доступне обладнання, перевіряти його стан та здійснювати резервування в режимі реального часу.

Архітектурно вебзастосунки для оренди та бронювання базуються на клієнт-серверній моделі, де взаємодія між користувачем і системою здійснюється через веббраузер. Клієнтська частина відповідає за інтерфейс, відображення даних та формування запитів, тоді як серверна частина виконує

обробку запитів, реалізує бізнес-логіку та забезпечує роботу з базою даних. Такий підхід дозволяє централізовано керувати інформацією, забезпечує масштабованість системи та спрощує її супровід і розвиток [3].

Важливою особливістю вебзастосунків у сфері оренди та бронювання є робота з актуальними даними про доступність ресурсів у режимі реального часу. Це означає, що система повинна постійно оновлювати інформацію про стан обладнання або інструментів, щоб уникати ситуацій подвійного бронювання та забезпечувати коректність відображення доступних ресурсів. Саме механізми синхронізації та оновлення даних є критично важливими для таких систем, оскільки вони безпосередньо впливають на якість сервісу.

Наступною суттєвою характеристикою є автоматизація процесів оренди, яка охоплює повний життєвий цикл взаємодії користувача із системою: від пошуку обладнання до оформлення бронювання, розрахунку вартості та підтвердження замовлення. Автоматизація дозволяє значно зменшити час обробки заявок, мінімізувати людський фактор та підвищити ефективність управління ресурсами [4]. Додатково система може автоматично формувати сповіщення користувачам щодо статусу бронювання, термінів оренди або необхідності оплати.

Не менш важливим функціональним аспектом є реалізація інтерактивного пошуку та фільтрації, що дозволяє користувачам швидко знаходити необхідне обладнання за різними критеріями, такими як тип, ціна, доступність або часовий інтервал оренди. Це значно підвищує зручність використання системи та скорочує час прийняття рішення [1].

Сучасні вебзастосунки для оренди також активно інтегрують платіжні системи, що дозволяє користувачам здійснювати оплату онлайн безпосередньо в межах платформи. Така інтеграція забезпечує прозорість фінансових операцій, підвищує рівень довіри до сервісу та спрощує процес розрахунків між сторонами.

Окрім базових функцій, сучасні системи оренди часто включають розширені можливості, серед яких персоналізація користувацького досвіду, що

базується на аналізі попередніх дій користувача та дозволяє формувати індивідуальні рекомендації щодо доступного обладнання. Також важливою функцією є синхронізація календарів бронювання, яка дозволяє уникати конфліктів у розкладі та забезпечує точність планування ресурсів. Додатково системи забезпечують автоматизовану комунікацію з користувачами, включаючи повідомлення про підтвердження бронювання, нагадування про оплату та інші сервісні сповіщення [5]. Важливою складовою також є адаптивність інтерфейсу, що забезпечує коректну роботу системи на різних пристроях, включаючи мобільні телефони та планшети.

З технічного погляду вебзастосунки оренди та бронювання реалізуються як багаторівневі системи, що включають клієнтську частину, серверну логіку та базу даних. Серверна частина відповідає за обробку бізнес-логіки, управління бронюваннями, перевірку доступності ресурсів і взаємодію з платіжними сервісами, тоді як клієнтська частина забезпечує інтерфейс користувача та взаємодію із системою.

Отже, вебзастосунки для систем оренди та бронювання є складними багатофункціональними інформаційними системами, які поєднують у собі елементи автоматизації, обробки даних у реальному часі, інтерактивної взаємодії з користувачем та фінансових операцій. Їх ключові особливості полягають у забезпеченні зручного доступу до ресурсів, підвищенні ефективності управління ними та мінімізації ручної праці, що робить такі системи важливим інструментом цифрової трансформації сервісної економіки.

1.2 Аналіз сучасних систем онлайн-оренди обладнання та інструментів

Стрімкий розвиток цифрових технологій суттєво трансформував ринок оренди обладнання та інструментів, перемістивши значну частку транзакцій у онлайн-простір. За даними аналітичної компанії IBISWorld, галузь оренди інструментів та обладнання у США демонструє стале зростання: сукупний

середньорічний темп зростання (CAGR) за п’ять років до 2025 року склав 3,9%, а загальний обсяг доходів галузі досяг 5,7 млрд доларів США [6]. Водночас ринок програмного забезпечення для управління орендою обладнання у 2024 році оцінювався у 48,2 млрд доларів США, а до 2033 року очікується його зростання до 55,1 млрд доларів США [7]. Ці тенденції свідчать про масштабний перехід від традиційних форм оренди до цифрових платформ.

Особливу роль у розвитку галузі відіграє мобільний сегмент. Ринок мобільних застосунків для оренди інструментів, оцінений у 1,79 млрд доларів США у 2024 році, прогнозовано досягне 6,98 млрд доларів США до 2034 року з CAGR 14,6% [8]. Основними чинниками зростання є зміщення переваг споживачів у бік оренди замість купівлі, широке проникнення смартфонів та формування так званої «економіки спільного використання» (sharing economy). За оцінками аналітиків, глобальний обсяг ринку sharing economy до 2025 року перевищить 335 млрд доларів США [14].

Сучасні системи онлайн-оренди обладнання та інструментів можна класифікувати за декількома основними моделями: B2C (бізнес – споживач), B2B (бізнес – бізнес) та P2P (рівний з рівним, peer-to-peer). Кожна з моделей має свої функціональні особливості, цільову аудиторію та ступінь технологічної зрілості.

Найбільші гравці ринку, зокрема United Rentals (США) та Sunbelt Rentals (США/Велика Британія), активно інвестують у розбудову власних цифрових екосистем. За підсумками 2024–2025 років United Rentals зафіксувала, що 76% доходів компанії надходить від клієнтів, які використовують цифрові інструменти, а онлайн-доходи зросли на 22% у річному вимірі [10]. Платформа Total Control та мобільний застосунок компанії дозволяють орендарям відслідковувати доставку обладнання в режимі реального часу, подавати заявки на технічне обслуговування, вносити платежі та контролювати використання активів дистанційно [11].

Sunbelt Rentals у рамках п’ятирічної стратегії Sunbelt 4.0 розгортає мультихмарну платформу на базі Salesforce та партнерство з Trackunit IrisX, що

забезпечує єдиний інтерфейс управління автопарком, аналітику використання ресурсів, контроль витрат палива та показники сталого розвитку [12]. Показово, що компанія досягла потрійного прискорення завантаження сторінок і подвоїла конверсію з незавершених замовлень (abandoned cart), що принесло до 2,4 млн доларів США додаткових річних доходів від e-commerce [13].

Паралельно з корпоративними розвиваються платформи peer-to-peer оренди, серед яких виділяються Fat Llama (Велика Британія / США), Dozr та EquipmentShare (США). Fat Llama, заснована у 2016 році та придбана шведською компанією Hygglo у 2022 році, стала однією з найбільш показових прикладів успішної P2P-моделі: до 2024 року кумулятивні виплати власникам переданого в оренду майна на платформі перевищили 200 млн доларів США [9]. Ключовою конкурентною перевагою платформи є власна система машинного навчання для оцінки ризиків із точністю 99,8% при виявленні підозрілих транзакцій, що суттєво знижує рівень шахрайства та крадіжок [9].

Платформи P2P-оренди надають можливість здавати в оренду різноманітне обладнання: електроінструменти, камери, садову техніку, будівельні пристрої. Система онлайн-бронювання з календарем доступності, інтегровані механізми страхування, цифровий підпис документів та вбудовані платіжні системи формують повний цикл транзакції без потреби у фізичній взаємодії між сторонами [14].

Окрему категорію формують SaaS-рішення, призначені для автоматизації орендного бізнесу: EZRentOut, Rentrax, Rides Rental Software, Anolla та інші. Станом на 2024 рік більше 61% підприємств орендної галузі використовували спеціалізоване програмне забезпечення для управління орендними операціями, відстеження інвентарю та виставлення рахунків [7]. Типовий функціональний склад таких систем включає: онлайн-бронювання з перевіркою наявності в реальному часі; управління клієнтами з історією оренди; автоматичне виставлення рахунків та інтеграцію з платіжними шлюзами (Stripe, PayPal); управління документами з електронним підписом; планування технічного обслуговування; звітні дашборди [13].

Платформа Anolla позиціонується як рішення корпоративного рівня з підтримкою кількох складів і філій, гнучкими моделями ціноутворення (погодинна, добова, сезонна оренда) та централізованим управлінням, яке, за даними замовників, на 33,1% ефективніше порівняно зі звичайними рішеннями [12].

Технологічний прогрес суттєво розширює можливості платформ оренди. Частка хмарних рішень серед користувачів ПЗ для оренди обладнання зросла з 49% у 2021 році до 62% у 2024 році, що зумовлено потребами у віддаленому доступі, оновленнях у реальному часі та сумісності з різними пристроями [7]. Більше 58% користувачів орендних платформ здійснюють доступ до них із смартфонів або планшетів [7].

Інтернет речей (IoT) та штучний інтелект (AI) стають ключовими технологіями галузі. Датчики IoT, встановлені на обладнанні, дозволяють відстежувати години роботи, показники ефективності та стан вузлів, а алгоритми прогнозного обслуговування автоматично генерують заявки на ремонт при досягненні порогових значень [15].

За даними 2024 року, понад 3800 орендних компаній використовують функції прогнозного технічного обслуговування на базі AI, що дозволяє скоротити простої на 23% та витрати на ремонт – на 31% [7]. Флоти, оснащені IoT-пристроями, у 2025 році зафіксували на 25% менше поломок та зниження витрат на обслуговування на 15–20% [15].

United Rentals у 2025 році запровадила функцію Smart Suggestions на базі машинного навчання, яка рекомендує обладнання з урахуванням історії замовлень, даних про будівельний майданчик, сезонності та галузевих тенденцій, скорочуючи час підбору та замовлення обладнання на 27% [11]. Функція Equipment Fit AR дозволяє візуалізувати тривимірні моделі техніки безпосередньо на місці виконання робіт за допомогою мобільного пристрою [10].

За даними аналітиків, онлайн-платформи оренди обладнання вже забезпечують 30% усіх транзакцій у галузі [7]. Лідируючі позиції займає Північна Америка з часткою понад 50% глобального ринку та 36,1% ринку

мобільних застосунків для оренди інструментів [8]. Ринок оренди обладнання в Азійсько-Тихоокеанському регіоні, що займає близько 31% глобального ринку, демонструє стрімке зростання завдяки урбанізації та розгортанню масштабних інфраструктурних проектів [7]. Ринок P2P-застосунків для оренди у 2024 році оцінювався у 17,7 млрд доларів США і прогнозовано зросте з CAGR 10,9% до 2034 року [14].

Попри технологічний прогрес, існуючі системи онлайн-оренди мають низку суттєвих обмежень. По-перше, більшість великих корпоративних платформ орієнтована на промислові підприємства та будівельні компанії, тоді як потреби малого бізнесу та приватних осіб у доступних інструментах задовольняються недостатньо.

По-друге, P2P-платформи стикаються з проблемами довіри між учасниками, ризиками пошкодження або крадіжки обладнання, складнощами зі страхуванням та верифікацією особи.

По-третє, інвестиції у розбудову IoT-інфраструктури (GPS-трекери, датчики) залишаються недоступними для малих орендних підприємств: венчурне фінансування ринку ПЗ для оренди зосереджено насамперед у стартапах, що використовують AI та IoT, – його обсяг зріс на 25% у 2023–2024 роках [6]. Це формує значний технологічний розрив між великими гравцями та малими операторами ринку.

Таким чином, аналіз сучасних систем онлайн-оренди обладнання та інструментів засвідчує стрімку цифрову трансформацію галузі, виникнення різноманітних бізнес-моделей та активне впровадження технологій AI, IoT і хмарних платформ. Водночас залишаються незаповнені ніші, зокрема для локальних ринків (малих міст, регіонів з недостатнім рівнем проникнення цифрових послуг) та сегменту «останньої милі», що обумовлює актуальність розробки нових рішень з урахуванням потреб конкретних цільових аудиторій.

1.3 Постановка мета та завдання роботи

Метою роботи є розробка вебзастосунку для онлайн-оренди обладнання та інструментів з можливістю бронювання, оплати та відстеження доступності ресурсів.

Для досягнення поставленої мети необхідно вирішити такі завдання:

- дослідити теоретичні основи побудови вебзастосунків для систем оренди та бронювання;
- провести аналіз сучасних систем онлайн-оренди обладнання та інструментів;
- визначити вимоги до майбутнього програмного продукту;
- спроєктувати архітектуру системи та структуру бази даних;
- розробити користувацький інтерфейс вебзастосунку;
- реалізувати функціонал реєстрації користувачів, оренди, бронювання та оплати;
- реалізувати модуль відстеження доступності обладнання в режимі реального часу;
- провести тестування розробленого програмного продукту та проаналізувати результати його роботи/

2 ПРОЄКТУВАННЯ ВЕБЗАСТОСУНКУ ДЛЯ ОНЛАЙН-ОРЕНДИ ОБЛАДНАННЯ ТА ІНСТРУМЕНТІВ

2.1 Визначення вимог до вебзастосунку та побудова архітектури системи

На етапі проєктування вебзастосунку для оренди та бронювання обладнання здійснюється формалізація функціональних і нефункціональних вимог, а також визначення загальної архітектури системи. Це дозволяє чітко окреслити межі системи, ролі користувачів, сценарії взаємодії та технічні обмеження, які забезпечують стабільну та безпечну роботу вебзастосунку. У результаті формується цілісна модель системи, що поєднує бізнес-логіку, інтерфейс користувача та рівень зберігання даних.

Функціональні вимоги (табл. 2.1) визначають перелік можливостей системи відповідно до ролей користувачів: клієнта, менеджера та адміністратора. Вони описують, які саме дії може виконувати кожен користувач у межах системи, та формують основу бізнес-логіки вебзастосунку.

Як видно з таблиці 2.1, система підтримує багаторівневу модель доступу, де кожна роль має власний набір функцій. Клієнт орієнтований на взаємодію з каталогом і бронюванням, менеджер – на операційне управління бронюваннями, а адміністратор – на повне керування системою, включаючи фінансові та організаційні аспекти.

Нефункціональні вимоги (табл. 2.2) визначають якісні характеристики системи, такі як продуктивність, безпека, масштабованість та надійність. Вони не описують конкретні функції, але задають обмеження та стандарти, яких повинна дотримуватися система.

Таблиця 2.1 – Функціональні вимоги до вебзастосунку оренди та бронювання обладнання

Роль	Код	Функціональна вимога	Пріоритет
Клієнт	FR-C01	Перегляд каталогу обладнання з фільтрацією та пошуком	Високий
Клієнт	FR-C02	Перегляд детальної інформації про обладнання	Високий
Клієнт	FR-C03	Перевірка доступності обладнання на обрані дати	Високий
Клієнт	FR-C04	Створення бронювання на вибраний період	Високий
Клієнт	FR-C05	Онлайн-оплата бронювання	Високий
Клієнт	FR-C06	Перегляд історії бронювань	Середній
Клієнт	FR-C07	Скасування бронювання до початку оренди	Середній
Клієнт	FR-C08	Реєстрація та авторизація	Високий
Менеджер	FR-M01	Перегляд усіх бронювань із фільтрацією	Високий
Менеджер	FR-M02	Зміна статусу бронювання	Високий
Менеджер	FR-M03	Перегляд статистики бронювань	Середній
Менеджер	FR-M04	Управління доступністю обладнання	Високий
Адміністратор	FR-A01	CRUD-операції над обладнанням	Високий
Адміністратор	FR-A02	Управління категоріями	Високий
Адміністратор	FR-A03	Управління користувачами та ролями	Високий
Адміністратор	FR-A04	Перегляд фінансової звітності	Середній
Адміністратор	FR-A05	Налаштування цінової політики	Середній

Таблиця 2.2 – Нефункціональні вимоги до вебзастосунку

Категорія	Код	Вимога	Критерій
Продуктивність	NFR-01	Час відповіді сервера	< 500 мс для 95% запитів
Продуктивність	NFR-02	Завантаження сторінки	< 3 сек
Масштабованість	NFR-03	Кількість одночасних користувачів	до 1000
Безпека	NFR-04	Шифрування паролів	bcrypt (salt ≥ 10)
Безпека	NFR-05	Захист від SQL-ін’єкцій	параметризовані запити
Безпека	NFR-06	Захищене з’єднання	HTTPS (TLS 1.2+)
Доступність	NFR-07	Адаптивність інтерфейсу	320px–2560px
Доступність	NFR-08	Відповідність стандартам	WCAG 2.1 AA
Надійність	NFR-09	Доступність системи	99.5% uptime
Надійність	NFR-10	Резервне копіювання	щоденне автоматичне

З таблиці 2.2 видно, що основний акцент зроблено на безпеці користувацьких даних, стабільності роботи системи та її продуктивності при високому навантаженні. Це критично важливо для систем бронювання, де одночасно працює велика кількість користувачів і відбуваються транзакційні операції.

Нижче наведена діаграма варіантів використання (рис. 2.1), яка відображає взаємодію різних ролей користувачів із системою.

На діаграмі показано чотири основні актори системи: гість, клієнт, менеджер та адміністратор. Гість має обмежений доступ і може лише переглядати каталог та здійснювати реєстрацію. Після авторизації користувач переходить у роль клієнта, який отримує доступ до розширеного функціоналу: створення бронювання, перевірка доступності та оплата.

Менеджер відповідає за операційне управління бронюваннями, включаючи зміну статусів та оновлення доступності обладнання. Адміністратор має найширші повноваження, що включають керування користувачами, обладнанням, категоріями та фінансовою звітністю. Зв’язки між варіантами використання відображають логіку системи, зокрема залежності між діями (наприклад, бронювання неможливе без авторизації).

Архітектура вебзастосунку (рис. 2.2) побудована за трирівневою моделлю з використанням принципів MVC та Server-Side Rendering. Такий підхід забезпечує чіткий розподіл відповідальності між рівнями системи та спрощує масштабування і супровід програмного продукту.

Як видно з архітектурної моделі, система складається з трьох основних рівнів. Клієнтський рівень відповідає за взаємодію з користувачем і реалізується через браузер та React-компоненти. Прикладний рівень містить бізнес-логіку системи, включаючи сервіси автентифікації, бронювання, управління обладнанням та оплатами. Рівень даних відповідає за зберігання інформації в базі даних SQLite, де зберігаються користувачі, бронювання, категорії та фінансові транзакції.

Окрему роль відіграють API Routes і Server Actions, які забезпечують зв’язок між клієнтською частиною та серверною логікою. Middleware використовується для перевірки автентифікації та контролю доступу до захищених ресурсів.

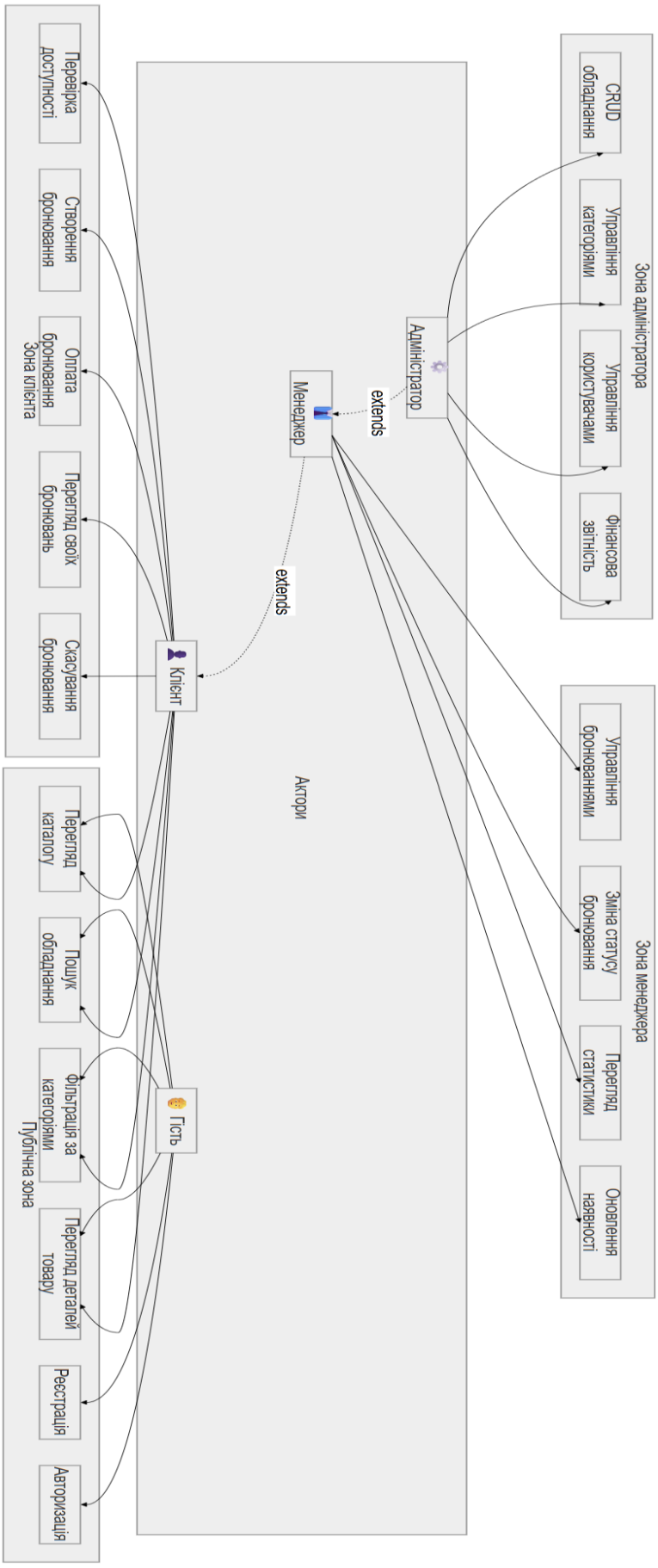


Рисунок 2.1 – Діаграма варіантів використання

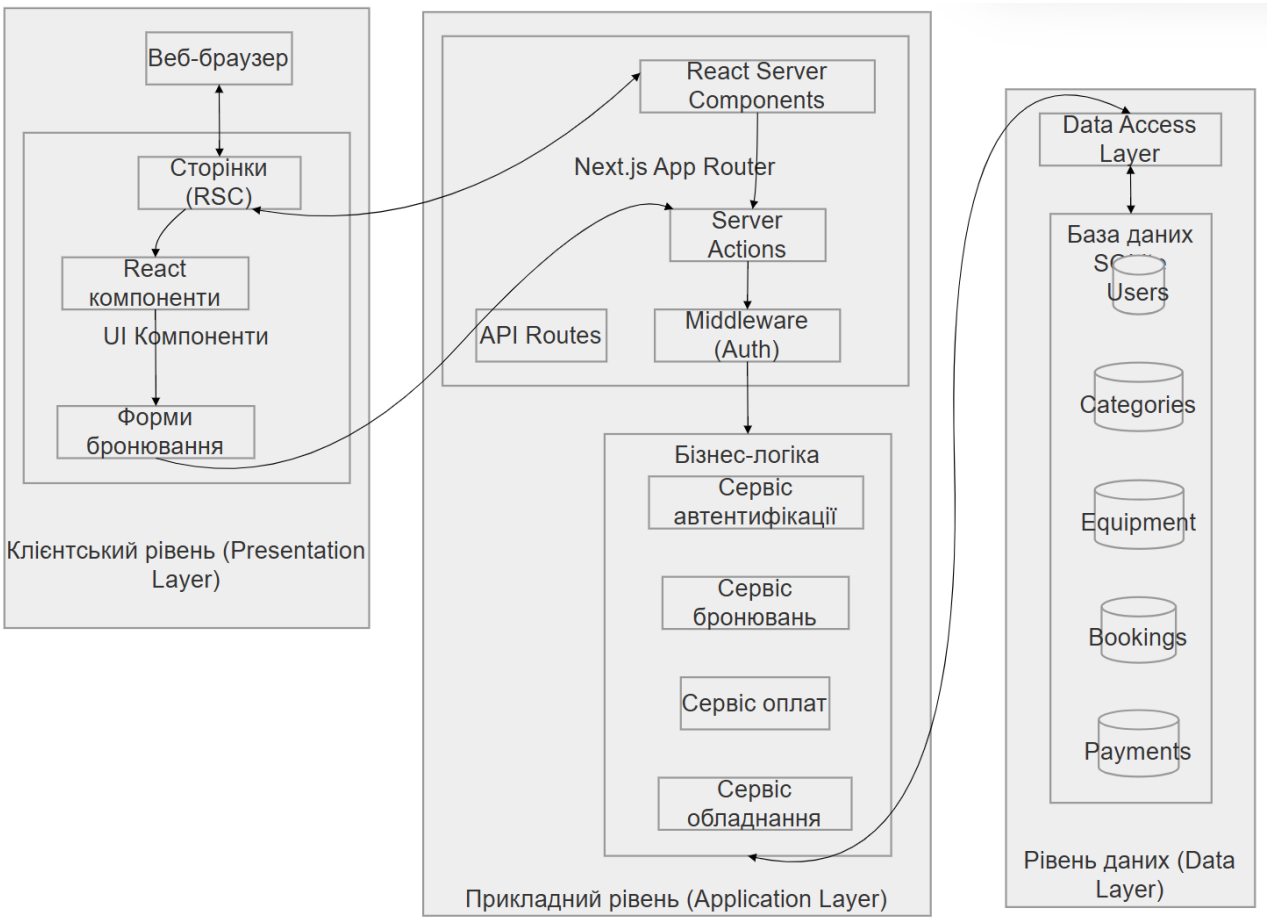


Рисунок 2.2 – Архітектура системи

Компонентна архітектура (рис. 2.3) деталізує структуру фронтенд-частини системи та описує повторно використовувані елементи інтерфейсу.

На діаграмі показано, що фронтенд побудований на основі модульних React-компонентів. Центральним елементом є спільні компоненти, такі як форма бронювання, картки обладнання та таблиці бронювань, які використовуються в різних частинах системи.

Окремо виділено публічні сторінки (головна, каталог, деталі товару, авторизація та реєстрація), захищені сторінки для авторизованих користувачів, а також адміністративну панель, яка включає модулі управління бронюваннями, обладнанням і користувачами.

Така компонентна структура дозволяє повторно використовувати код, спрощує підтримку системи та забезпечує масштабованість інтерфейсу при подальшому розвитку вебзастосунку.

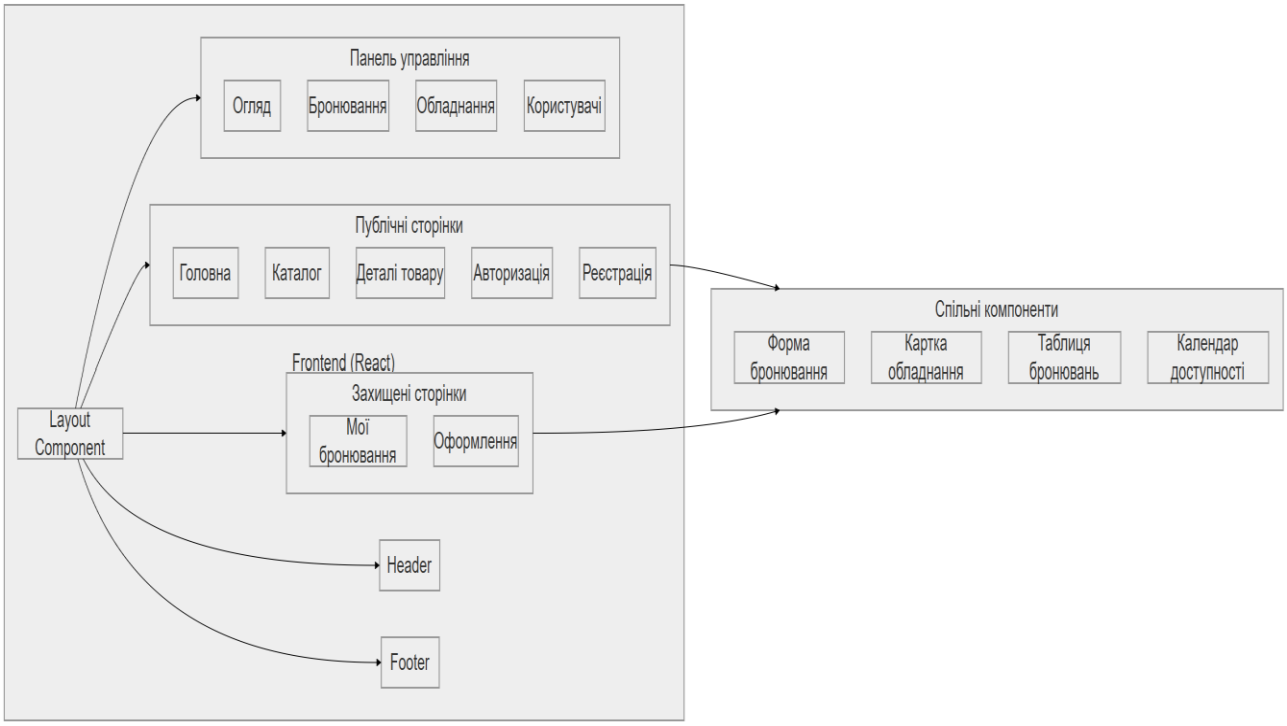


Рисунок 2.3 – Компонентна архітектура

Отже, у межах розділу було визначено функціональні та нефункціональні вимоги до вебзастосунку оренди та бронювання обладнання, сформовано модель ролей користувачів і описано їх взаємодію із системою. Також побудовано трирівневу архітектуру системи та компонентну структуру фронтенду.

Отримані результати дозволяють перейти до етапу реалізації програмного продукту, оскільки забезпечують чітке розуміння структури системи, її обмежень і логіки роботи.

2.2 Проєктування бази даних та логіки управління бронюваннями й оплатами

Проєктування бази даних для вебзастосунку оренди та бронювання обладнання є ключовим етапом, оскільки саме воно визначає структуру зберігання даних, зв’язки між сутностями та ефективність роботи всієї системи. Основною метою даного етапу є створення такої моделі даних, яка забезпечує

цілісність інформації, мінімізує надлишковість та підтримує логіку процесів бронювання і оплати в межах багаторівневої архітектури системи.

На концептуальному рівні система базується на п’яти основних сутностях: USERS, CATEGORIES, EQUIPMENT, BOOKINGS та PAYMENTS (рис. 2.4). Вони відображають ключові бізнес-об’єкти предметної області – користувачів системи, категорії обладнання, самі одиниці обладнання, бронювання та платежі.

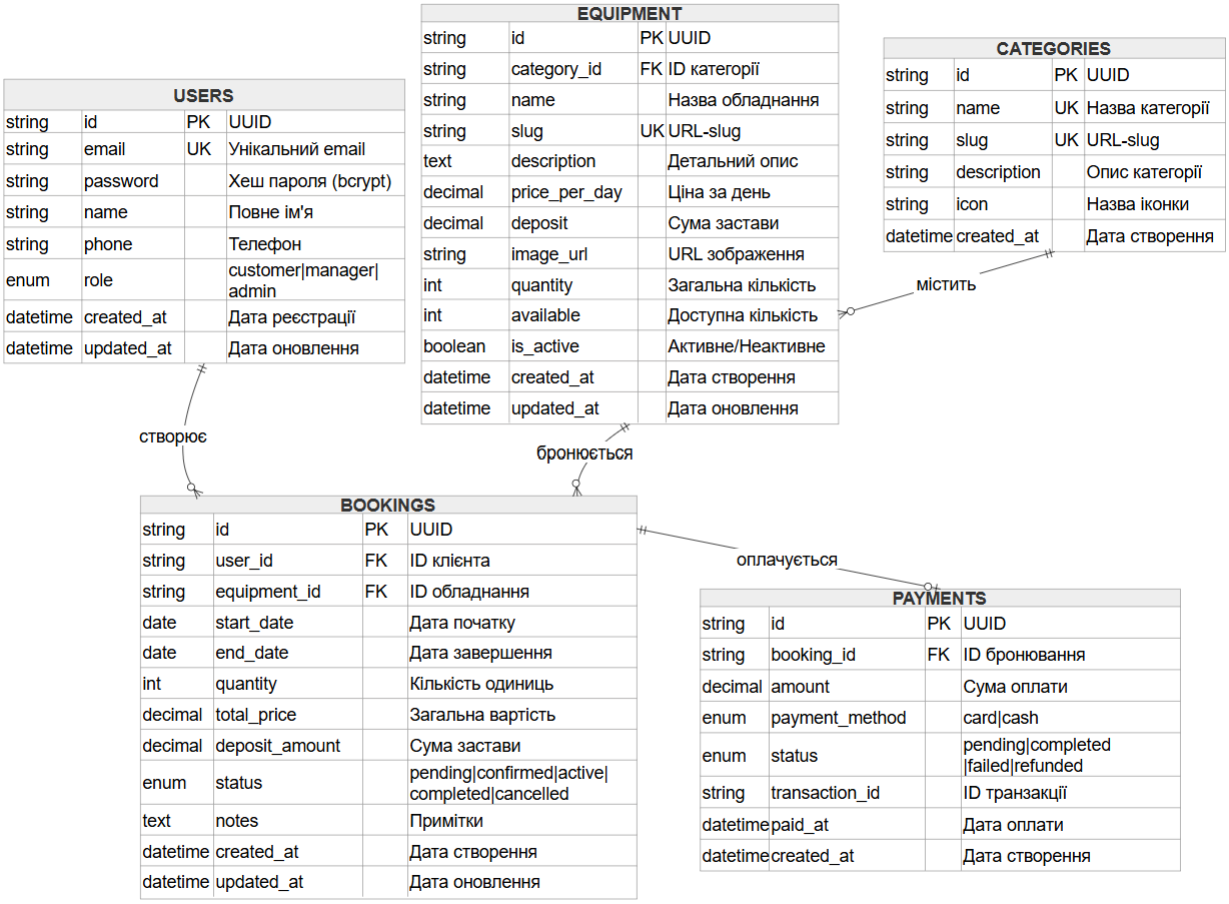


Рисунок 2.4 – Концептуальна модель даних

Як видно з ER-діаграми, центральною сутністю є BOOKINGS, яка зв’язує користувача (USERS) з конкретним обладнанням (EQUIPMENT). Один користувач може створювати багато бронювань, що формує зв’язок типу «один-до-багатьох». Аналогічно, кожна одиниця обладнання може бути пов’язана з великою кількістю бронювань у різні періоди часу.

Сутність EQUIPMENT, у свою чергу, належить до конкретної категорії (CATEGORIES), що дозволяє структурувати каталог та забезпечити зручну фільтрацію. Сутність PAYMENTS пов'язана з BOOKINGS зв'язком «один-до-одного» або «один-до-багатьох» (залежно від бізнес-логіки), оскільки кожне бронювання може мати один або декілька платіжних записів (наприклад, часткова оплата або повернення коштів).

Таким чином, ER-модель забезпечує логічну цілісність системи та відображає повний цикл життєвого процесу бронювання: від вибору обладнання до підтвердження оплати.

Фізична модель бази даних реалізована у вигляді реляційної структури, де кожна сутність представлена окремою таблицею з чітко визначеними типами даних, первинними та зовнішніми ключами.

Таблиця USERS (табл. 2.3) містить інформацію про користувачів системи та використовується для автентифікації, авторизації та персоналізації взаємодії.

Вона включає унікальний ідентифікатор користувача (id), електронну пошту (email), хешований пароль (password), персональні дані (name, phone) та роль користувача (role), яка визначає рівень доступу в системі – клієнт, менеджер або адміністратор. Додатково зберігаються часові мітки створення та оновлення записів.

Таблиця 2.3 – USERS

Поле	Тип даних	Обмеження	Опис
id	TEXT	PRIMARY KEY	Унікальний ідентифікатор (UUID)
email	TEXT	NOT NULL, UNIQUE	Електронна пошта
password	TEXT	NOT NULL	Хеш пароля (bcrypt)
name	TEXT	NOT NULL	Повне ім'я
phone	TEXT	NULL	Контактний телефон
role	TEXT	NOT NULL	customer / manager / admin
created_at	TEXT	NOT NULL	Дата створення
updated_at	TEXT	NOT NULL	Дата оновлення

Таблиця CATEGORIES (табл. 2.4) забезпечує логічну класифікацію обладнання та дозволяє реалізувати структурований каталог.

Вона містить назву категорії, унікальний slug для URL-навігації, опис та іконку для візуального представлення в інтерфейсі.

Таблиця 2.4 – CATEGORIES

Поле	Тип даних	Обмеження	Опис
id	TEXT	PRIMARY KEY	Унікальний ідентифікатор
name	TEXT	NOT NULL	Назва категорії
slug	TEXT	UNIQUE	URL-ідентифікатор
description	TEXT	NULL	Опис категорії
icon	TEXT	NULL	Іконка
created_at	TEXT	NOT NULL	Дата створення

Таблиця EQUIPMENT (табл. 2.5) є центральною для предметної області, оскільки містить дані про доступне обладнання для оренди.

Вона включає фінансові параметри (price_per_day, deposit), інформацію про доступність (quantity, available), а також статус активності запису. Це дозволяє системі контролювати актуальність пропозицій та запобігати бронюванню недоступних позицій.

Таблиця 2.5 – EQUIPMENT

Поле	Тип даних	Обмеження	Опис
id	TEXT	PRIMARY KEY	Унікальний ідентифікатор
category_id	TEXT	FOREIGN KEY	Посилання на категорію
name	TEXT	NOT NULL	Назва обладнання
slug	TEXT	UNIQUE	URL-ідентифікатор
description	TEXT	NULL	Опис
price_per_day	REAL	NOT NULL	Ціна за добу
deposit	REAL	NOT NULL	Застава
image_url	TEXT	NULL	Зображення
quantity	INTEGER	NOT NULL	Загальна кількість
available	INTEGER	NOT NULL	Доступна кількість
is_active	INTEGER	NOT NULL	Активність
created_at	TEXT	NOT NULL	Створення
updated_at	TEXT	NOT NULL	Оновлення

Таблиця BOOKINGS (табл. 2.6) реалізує логіку бронювання та є ключовою транзакційною сутністю системи. Саме тут фіксуються всі орендні операції користувачів.

Кожне бронювання містить посилання на користувача та обладнання, період оренди, кількість одиниць, загальну вартість та суму застави. Поле status дозволяє відслідковувати життєвий цикл бронювання.

Таблиця 2.6 – BOOKINGS

Поле	Тип даних	Опис
id	TEXT	Унікальний ідентифікатор
user_id	TEXT	Користувач
equipment_id	TEXT	Обладнання
start_date	TEXT	Початок оренди
end_date	TEXT	Кінець оренди
quantity	INTEGER	Кількість
total_price	REAL	Вартість
deposit_amount	REAL	Застава
status	TEXT	pending / confirmed / active / completed / cancelled
notes	TEXT	Примітки
created_at	TEXT	Створення
updated_at	TEXT	Оновлення

Таблиця PAYMENTS забезпечує фінансову складову системи та фіксує всі операції оплати, пов’язані з бронюваннями. Вона дозволяє підтримувати різні методи оплати, контролювати статус транзакцій та забезпечує можливість повернення коштів.

Діаграма станів бронювання (рис. 2.5) відображає життєвий цикл сутності BOOKINGS від моменту створення до завершення або скасування. Основними станами є pending, confirmed, active, completed, cancelled та overdue.

Початковий стан pending означає, що бронювання створене, але ще не оплачено або не підтверджено. Після успішної оплати система переводить бронювання у стан confirmed, що означає резервування обладнання за користувачем. Далі, після фактичної видачі обладнання, статус змінюється на active, що означає початок періоду оренди.

У разі успішного завершення оренди бронювання переходить у стан completed, де фіксується повернення обладнання та закриття фінансових зобов’язань. Якщо користувач скасовує бронювання до видачі або протягом таймауту оплати (24 години), статус змінюється на cancelled. Окремо виділяється

стан overdue, який застосовується у випадку прострочення повернення обладнання.

Таким чином, діаграма станів відображає повний життєвий цикл бронювання та забезпечує контроль бізнес-процесів у системі.

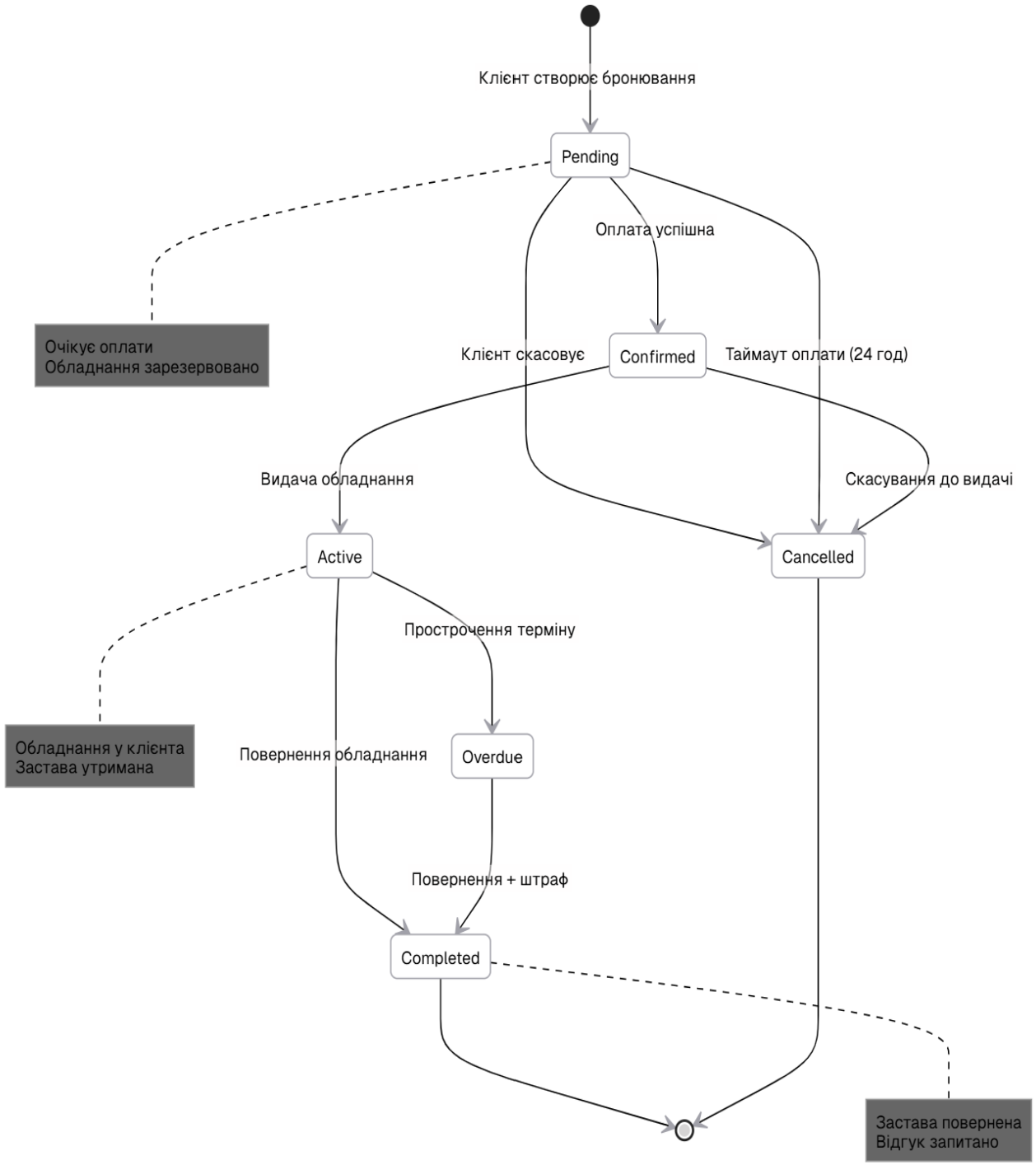


Рисунок 2.5 – Діаграма станів бронювання

Алгоритм перевірки доступності (рис. 2.6) є критично важливим етапом перед створенням бронювання. Він базується на аналізі перетинів дат та доступної кількості одиниць обладнання.

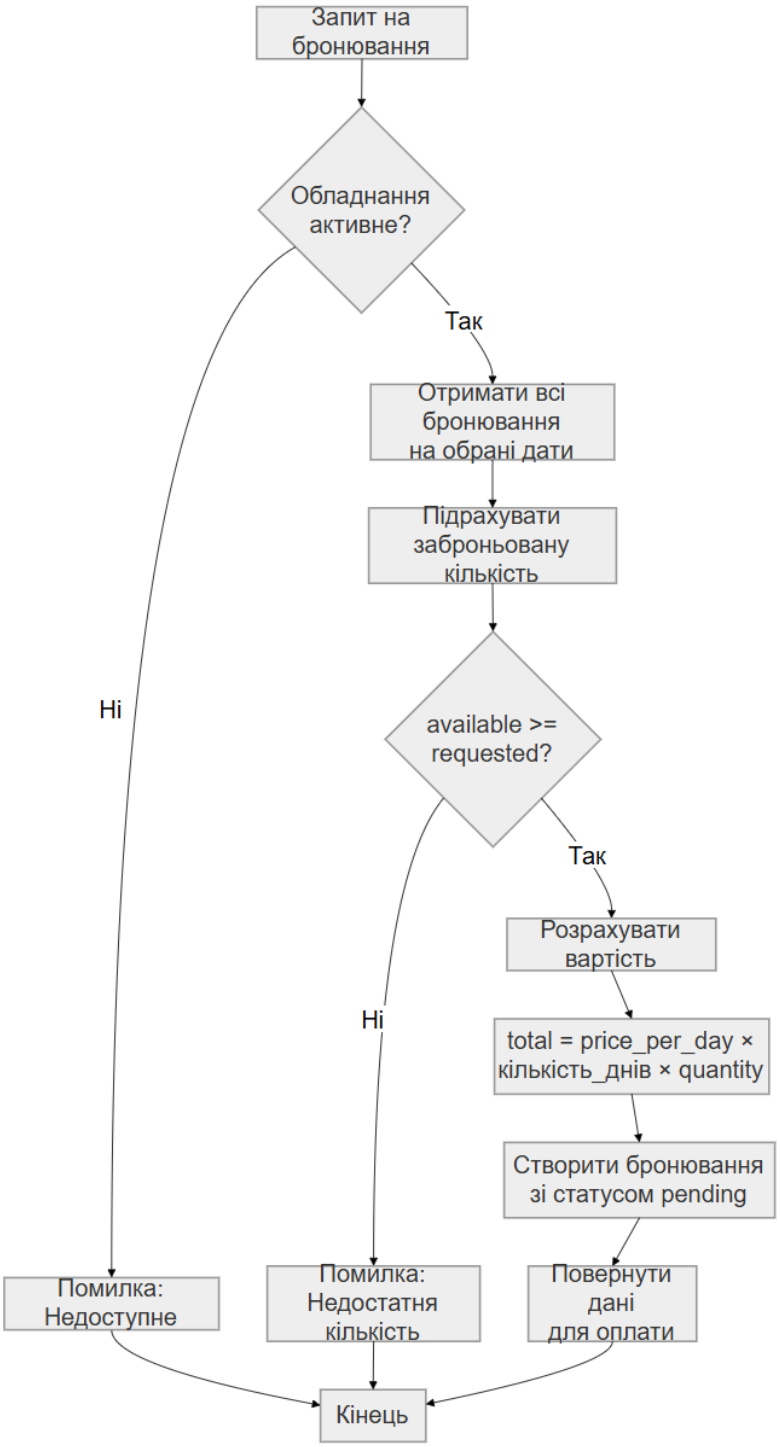


Рисунок 2.6 – Алгоритм перевірки доступності обладнання

Спочатку система перевіряє, чи є обладнання активним. Якщо поле is_active має значення false, процес завершується з помилкою недоступності. Далі виконується вибірка всіх бронювань, що перетинаються з запитуваним періодом.

Після цього обчислюється сумарна кількість вже забронюваного обладнання. Якщо доступна кількість (available) менша за запитувану кількість, система повертає помилку про недостатню кількість одиниць.

У разі успішної перевірки виконується розрахунок вартості оренди, після чого створюється запис у таблиці BOOKINGS зі статусом pending. Таким чином, алгоритм забезпечує уникнення конфліктів бронювання та перевищення доступних ресурсів.

Розрахунок вартості оренди базується на комбінації базової ціни, тривалості оренди, кількості одиниць та системи знижок. Кількість днів визначається як різниця між датами завершення та початку оренди з додаванням одного дня для коректного врахування повного періоду.

Базова вартість обчислюється як добова ставка, помножена на кількість днів та кількість одиниць обладнання. Додатково застосовується система знижок залежно від тривалості оренди: від 7 днів – 10%, від 14 днів – 15%, від 30 днів – 20%.

Окремо розраховується сума застави як фіксована величина, помножена на кількість одиниць обладнання. Підсумкова сума до оплати формується як різниця між базовою вартістю та знижкою з додаванням застави.

Такий підхід дозволяє гнучко масштабувати цінову політику та забезпечує прозорість фінансових розрахунків для користувача.

Діаграма послідовності (рис. 2.7) ілюструє повний сценарій взаємодії користувача із системою під час створення бронювання та здійснення оплати. У процесі беруть участь чотири основні компоненти: інтерфейс користувача, сервер застосунку, база даних та платіжна система.

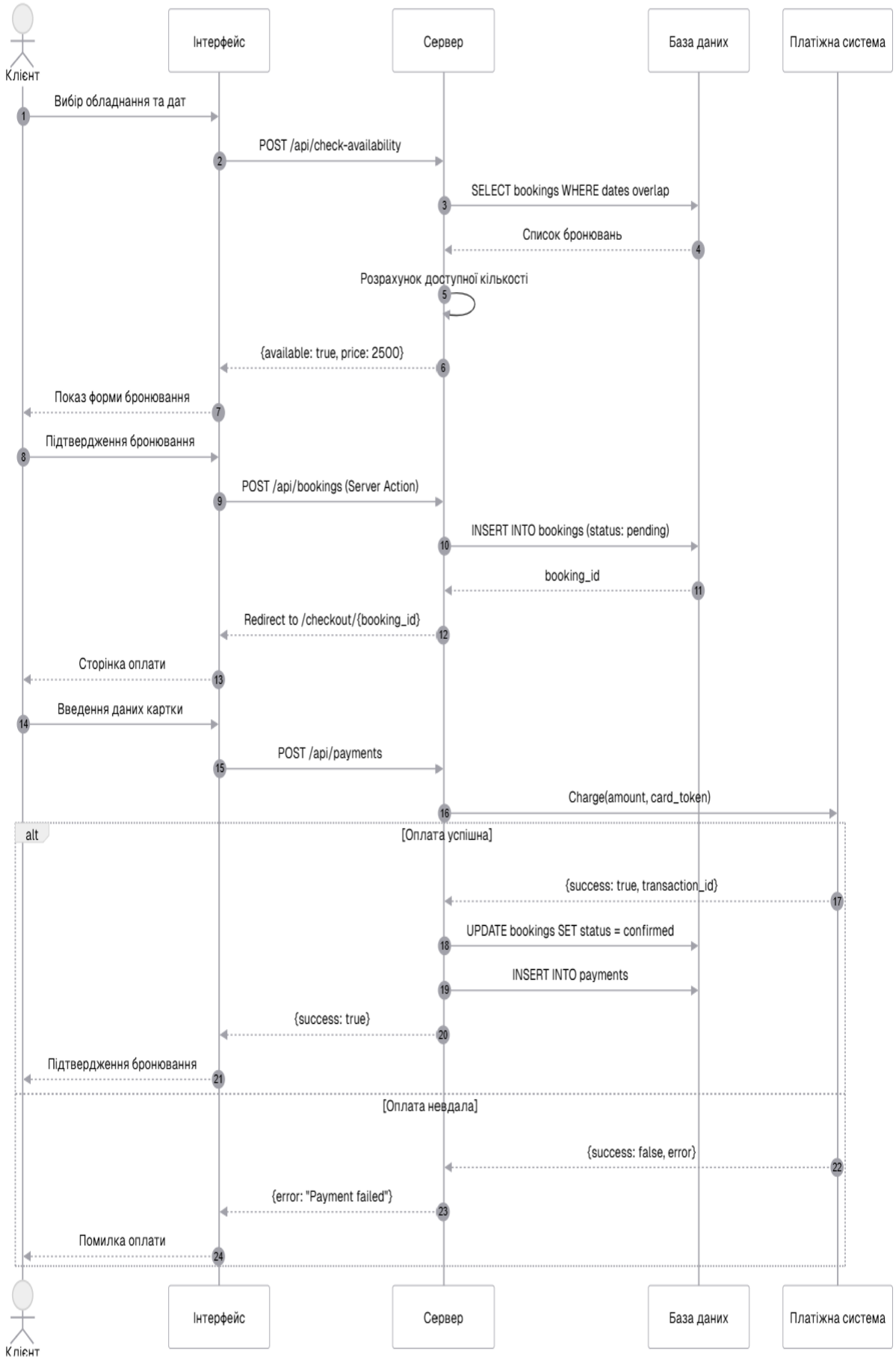


Рисунок 2.7 – Діаграма послідовності процесу бронювання

Процес починається з вибору обладнання та дат оренди. Інтерфейс надсилає запит на перевірку доступності, після чого сервер виконує запити до бази даних для отримання інформації про перетин бронювань.

У разі успішної перевірки користувачу повертається розрахована вартість і пропонується форма підтвердження бронювання. Після підтвердження створюється запис у таблиці BOOKINGS зі статусом pending, а користувач перенаправляється на сторінку оплати.

Далі відбувається взаємодія з платіжною системою: після введення платіжних даних виконується транзакція. У разі успіху система оновлює статус бронювання на confirmed та створює запис у таблиці PAYMENTS.

У випадку помилки оплати бронювання не підтверджується, а користувач отримує повідомлення про невдалу транзакцію. Така послідовність забезпечує цілісність даних і виключає ситуації неконсистентних бронювань.

Отже, спроєктовано базу даних та логіку управління бронюваннями й оплатами вебзастосунку оренди обладнання. Визначено основні сутності системи та їх взаємозв’язки, що забезпечують цілісність і узгодженість даних.

Розроблено фізичну структуру таблиць із ключами та обмеженнями, що мінімізує дублювання інформації та підвищує надійність зберігання даних. Описано життєвий цикл бронювання, алгоритм перевірки доступності та правила розрахунку вартості оренди. Загалом запропонована модель забезпечує коректну обробку бронювань, безпечну інтеграцію оплат і стабільну роботу системи.

2.3 Проєктування користувацького інтерфейсу та вибір технологій реалізації

Проєктування користувацького інтерфейсу (UI) у межах даної системи оренди та бронювання обладнання спрямоване на створення інтуїтивно зрозумілого, мінімалістичного та функціонального середовища взаємодії користувача з платформою. Основний акцент зроблено на швидку навігацію між

каталогом, карткою товару, процесом бронювання та оплатою, а також на розділення ролей користувачів (клієнт, менеджер, адміністратор) із відповідними рівнями доступу.

Інформаційна архітектура системи (рис. 2.8) відображає логічну структуру сторінок та зв’язки між ними. Вона поділена на публічну частину, захищений розділ користувача та адміністративну панель.

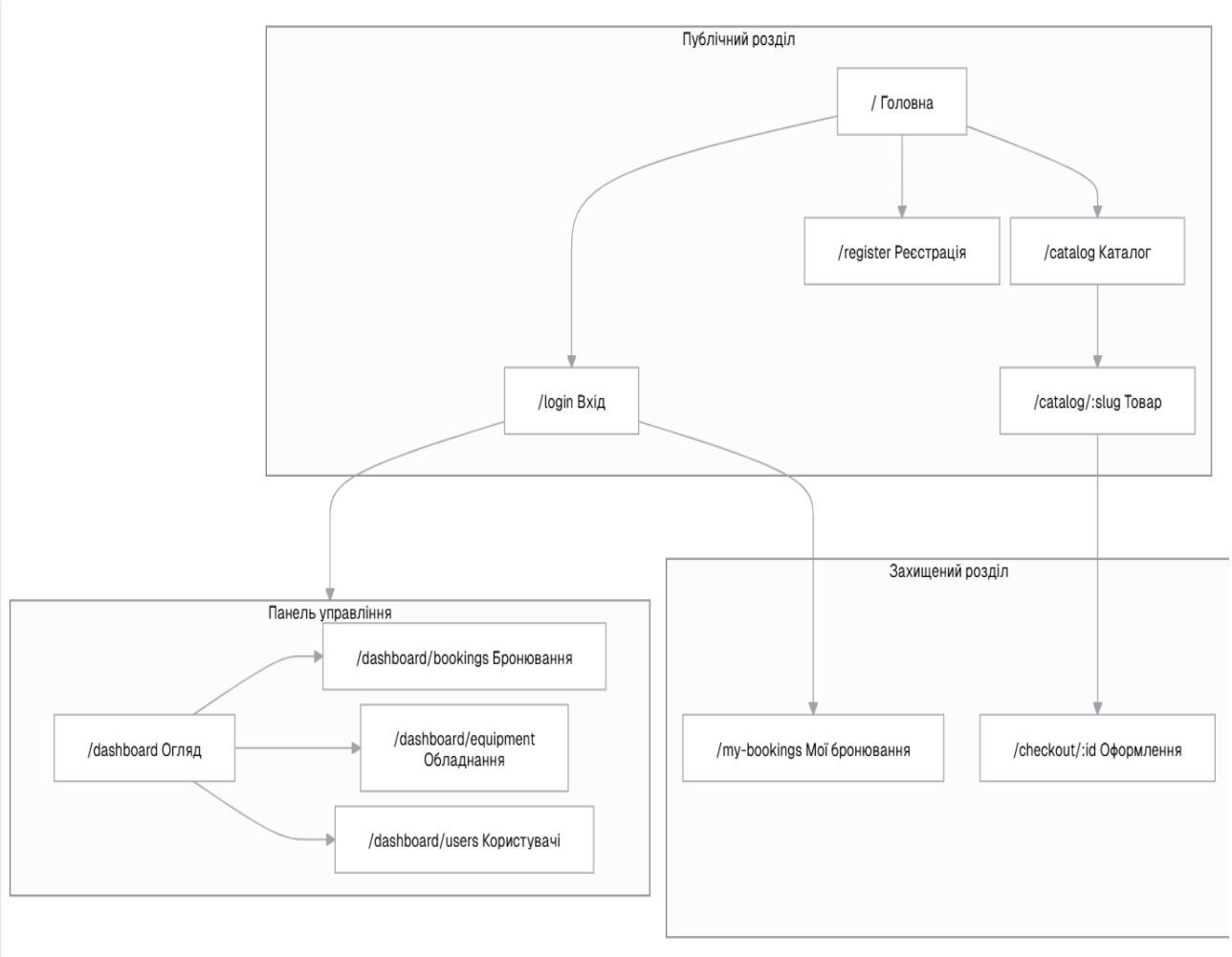


Рисунок 2.8 – Інформаційна архітектура

Публічний розділ включає головну сторінку, каталог обладнання, сторінку окремого товару, а також сторінки авторизації та реєстрації. Саме цей блок забезпечує первинну взаємодію користувача з системою та формує основний потік перегляду й вибору обладнання.

Захищений розділ призначений для авторизованих користувачів і містить функціонал перегляду власних бронювань, оформлення замовлення та проведення оплати. Це дозволяє користувачу контролювати всі етапи життєвого циклу бронювання в одному інтерфейсі.

Адміністративна панель (dashboard) призначена для менеджерів та адміністраторів і включає модулі керування бронюваннями, обладнанням та користувачами, а також аналітичний огляд системи. Така структура забезпечує розмежування доступу та зручне адміністрування бізнес-процесів.

Проєктування інтерфейсу базується на компонентному підході, де кожна сторінка складається з набору повторно використовуваних UI-блоків.

Головна сторінка виконує роль вхідної точки та містить hero-секцію для привернення уваги, блок категорій для швидкої навігації, а також добірку популярного обладнання. Така структура спрямована на підвищення конверсії переходів у каталог.

Сторінка каталогу реалізує механізми пошуку та фільтрації. Вона включає панель фільтрів, рядок пошуку, сітку карток товарів і пагінацію. Це забезпечує ефективну роботу з великим обсягом даних.

Сторінка деталей товару є ключовою для прийняття рішення про бронювання. Вона містить галерею зображень, інформаційний блок, календар доступності та форму бронювання, що дозволяє користувачу одразу ініціювати процес оренди.

Сторінка оформлення замовлення об’єднує підсумок бронювання, форму введення платіжних даних і підтвердження умов. Її основна задача – завершення транзакційного процесу.

Адміністративний dashboard містить аналітичні картки, графіки, таблицю останніх бронювань та швидкі дії, що забезпечує контроль ключових бізнес-показників.

Система дизайну побудована на основі design tokens, що забезпечує єдині правила стилізації інтерфейсу в межах усього застосунку.

Кольорова палітра розроблена з підтримкою світлої та темної теми, що підвищує зручність використання в різних умовах освітлення. Основним акцентним кольором виступає синій відтінок, який використовується для інтерактивних елементів, кнопок та посилань. Нейтральні кольори застосовуються для фону, тексту та допоміжних елементів, що дозволяє створити візуальну ієрархію та зменшити навантаження на користувача.

Типографіка побудована на шрифті Inter, який забезпечує високу читабельність у вебсередовищі. Ієрархія заголовків (H1–H3), основного тексту та допоміжних підписів дозволяє чітко структурувати контент і підвищує сприйняття інформації.

Адаптивність інтерфейсу реалізовано за принципом mobile-first, що передбачає початкову оптимізацію для мобільних пристроїв із подальшим розширенням для планшетів і десктопів.

На мобільних пристроях використовується одноколонкова структура з прихованою бічною панеллю, що відкривається через hamburger-меню. Це дозволяє максимально ефективно використовувати обмежений простір екрана.

На планшетах інтерфейс переходить до двоколонкової або трьохколонкової сітки, а навігація стає частково розгорнутою. Це покращує баланс між щільністю інформації та зручністю взаємодії.

На десктопах застосовується повноцінна багатоколонкова сітка (3–4 колонки), розширена навігація та активна бічна панель. Такий підхід забезпечує максимальну продуктивність роботи з системою.

Вибір технологічного стеку здійснювався з урахуванням вимог до продуктивності, масштабованості та швидкості розробки.

Для frontend-частини обрано Next.js [16] як основний фреймворк завдяки підтримці серверного рендерингу (SSR), статичної генерації (SSG) та Server Actions. Це дозволяє створювати високопродуктивні вебзастосунки з оптимізованим SEO та швидким завантаженням сторінок.

У якості бази даних обрано SQLite [17] як легке та просте рішення для локального зберігання даних. Вона забезпечує швидке розгортання системи та достатній рівень продуктивності для навчального та середнього навантаження.

Порівняльний аналіз показав, що Next.js має переваги над альтернативами (Nuxt.js, Remix, SvelteKit) завдяки глибокій інтеграції з React-екосистемою, підтримці App Router та Server Actions. Аналогічно, SQLite було обрано через простоту розгортання та мінімальні вимоги до інфраструктури.

Фінальний стек технологій формує єдину архітектурну екосистему, що забезпечує повний цикл розробки вебзастосунку – від інтерфейсу до серверної логіки.

Frontend реалізовано на базі Next.js 15 із використанням React 19 [18] та TypeScript [19], що забезпечує типобезпечну розробку та сучасний компонентний підхід. Для стилізації використовується Tailwind CSS [20], який дозволяє швидко створювати адаптивні інтерфейси.

UI-компоненти побудовані на основі shadcn/ui, що забезпечує консистентність дизайну та повторне використання елементів. Для покращення UX застосовуються бібліотеки Sonner (повідомлення), Lucide React (іконки) та date-fns (робота з датами) [21].

На серверному рівні використовується Node.js [22] середовище з API Routes та Server Actions. Автентифікація реалізована через bcryptjs [23], що забезпечує безпечне хешування паролів.

Таким чином, обраний стек дозволяє створити сучасний, масштабований і продуктивний вебзастосунок із чітким поділом відповідальностей між клієнтською та серверною частинами.

3 РОЗРОБКА ТА ТЕСТУВАННЯ ВЕБЗАСТОСУНКУ

3.1 Реалізація функціоналу системи (реєстрація, оренда, бронювання, оплата)

У межах розробки вебзастосунку реалізовано комплексний функціонал для забезпечення повного циклу взаємодії користувача із системою оренди обладнання. Основними функціональними модулями є реєстрація та автентифікація користувачів, перегляд каталогу обладнання, створення бронювань, розрахунок вартості оренди та проведення оплати. Реалізація здійснювалася із використанням сучасних можливостей платформи Next.js 15, зокрема Server Actions, серверних компонентів React та інтеграції з локальною базою даних SQLite.

Система автентифікації забезпечує безпечну реєстрацію нових користувачів, авторизацію в системі та підтримку активних сесій. Основною вимогою під час реалізації даного модуля було забезпечення належного рівня захисту персональних даних та облікових записів користувачів.

Процес реєстрації починається із заповнення користувачем спеціальної форми, яка містить поля електронної пошти, пароля та імені. Після надсилання форми дані передаються до серверної функції, де виконується їх первинна перевірка. Зокрема контролюється наявність усіх обов’язкових полів, коректність введених значень та відповідність пароля мінімальним вимогам безпеки.

Після успішної валідації система виконує перевірку наявності користувача з аналогічною електронною адресою. Якщо запис відсутній, пароль проходить процедуру хешування за допомогою алгоритму bcrypt із використанням десяти раундів генерації salt. Такий підхід дозволяє значно підвищити стійкість системи до атак типу brute force та rainbow table.

Після створення облікового запису формується унікальний ідентифікатор сесії, який зберігається в cookie-файлі браузера. Для забезпечення додаткового захисту використовуються параметри httpOnly, secure та sameSite, що мінімізують ризики викрадення сесій через XSS- та CSRF-атаки.

Логіка реєстрації користувача реалізована у вигляді серверної функції, фрагмент якої наведено в додатку А.1.

Авторизація користувача відбувається за аналогічним принципом. Після введення електронної пошти та пароля система виконує пошук відповідного запису в базі даних та здійснює порівняння введеного пароля із збереженим хешем за допомогою функції bcrypt.compare(). У випадку успішної перевірки створюється нова сесія користувача та формується токен доступу.

Діаграма архітектури модуля автентифікації (рис. 3.1) демонструє послідовність обробки даних від моменту реєстрації або входу користувача до створення та перевірки активної сесії. Особливістю реалізованого рішення є виконання всієї критично важливої логіки на сервері, що мінімізує ризики компрометації конфіденційної інформації.

Модуль бронювання є центральним елементом системи, оскільки забезпечує основний бізнес-процес – оформлення оренди обладнання на визначений період часу.

Процес починається з вибору користувачем необхідного обладнання в каталозі та переходу на сторінку його детального опису. На даній сторінці користувач має можливість ознайомитися з характеристиками обладнання, переглянути фотографії, обрати період оренди та вказати необхідну кількість одиниць.

Після введення параметрів оренди система виконує перевірку доступності обладнання. Для цього здійснюється аналіз існуючих бронювань, які перетинаються з вибраним часовим інтервалом. Якщо кількість доступних одиниць є достатньою, користувач може продовжити оформлення замовлення.



Рисунок 3.1 – Архітектура модуля автентифікації

Наступним етапом є автоматичний розрахунок вартості оренди. Система визначає тривалість оренди у днях, обчислює базову вартість відповідно до встановленого тарифу та застосовує можливі знижки залежно від тривалості користування обладнанням. Додатково враховується сума застави, яка необхідна для забезпечення збереження майна.

Після завершення всіх розрахунків створюється запис бронювання в базі даних. Нове бронювання отримує статус pending, що означає очікування оплати. Одночасно формується унікальний номер замовлення для подальшої ідентифікації операції.

Реалізація логіки створення бронювання наведена в додатку А.2.

Особливістю реалізованого алгоритму є багаторівнева перевірка коректності даних перед створенням бронювання. Система контролює:

– наявність активної сесії користувача;

– існування вибраного обладнання;

– актуальність статусу обладнання;

– достатню кількість доступних одиниць;

– коректність дат оренди;

– правильність розрахунку вартості.

Діаграма процесу бронювання (рис. 3.2) відображає повний сценарій взаємодії користувача із системою – від вибору обладнання до створення бронювання та переходу на сторінку оплати.

У разі відсутності необхідної кількості обладнання система повідомляє користувача про неможливість оформлення оренди та пропонує змінити дати або обрати інший товар. Такий підхід дозволяє уникнути конфліктів бронювання та забезпечує актуальність інформації про доступність обладнання.

Після успішного створення бронювання користувач автоматично перенаправляється до модуля оплати, де завершується процес оформлення замовлення.

Модуль оплати є завершальним етапом процесу бронювання обладнання та забезпечує переведення бронювання зі статусу очікування оплати (pending) до статусу підтвердженого замовлення (confirmed). Для реалізації функціоналу використано клієнтські компоненти React для взаємодії з користувачем та серверні функції Next.js Server Actions для обробки транзакцій на серверній стороні.

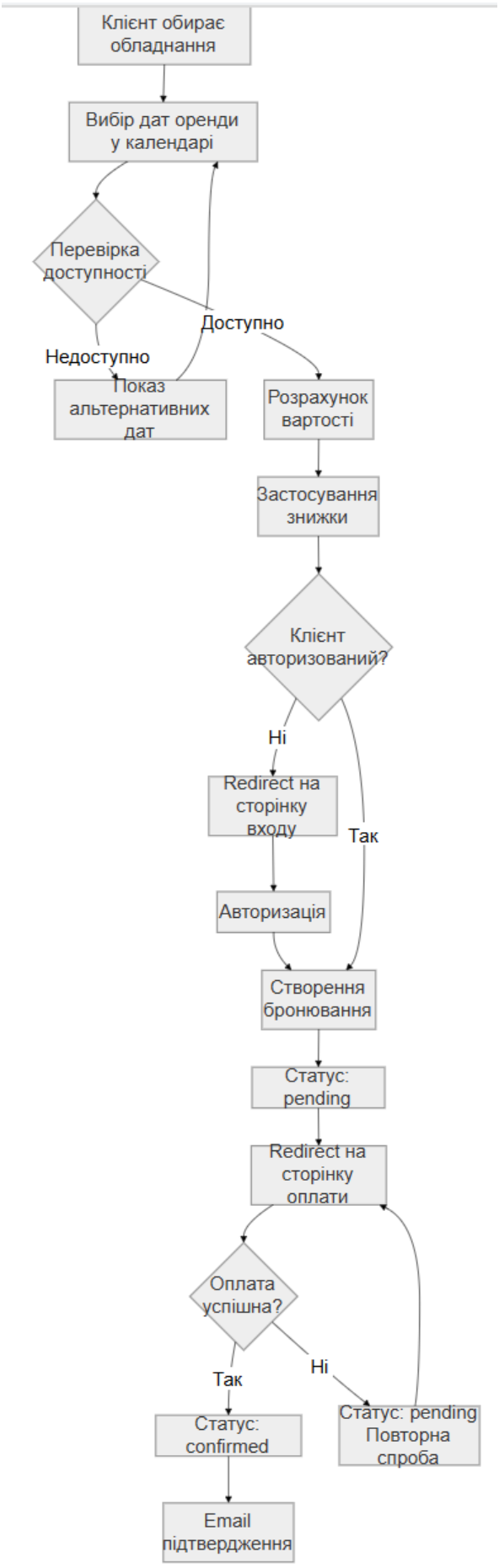


Рисунок 3.2 – Діаграма процесу бронювання

Архітектура модуля розроблена таким чином, щоб у майбутньому забезпечити можливість інтеграції з реальними платіжними сервісами, такими як Stripe, LiqPay або Portmone. У поточній реалізації використовується емуляція платіжного шлюзу, що дозволяє протестувати повний цикл оформлення замовлення без підключення зовнішніх фінансових сервісів.

Інтерфейс сторінки оплати формується після успішного створення бронювання та містить зведену інформацію про замовлення. Користувач отримує можливість переглянути тривалість оренди, вартість послуги, суму застави та загальну суму до оплати. Такий підхід забезпечує прозорість фінансових розрахунків і дозволяє уникнути помилок під час підтвердження замовлення.

Реалізацію клієнтської форми оплати наведено у додатку А.3. Форма містить поля для введення номера банківської картки, терміну її дії, CVV-коду та імені власника. Для покращення користувацького досвіду застосовується автоматичне форматування введених даних та миттєва перевірка коректності заповнення полів.

Перед передачею платіжних даних на сервер виконується первинна валідація номера картки за алгоритмом Луна (Luhn Algorithm), який використовується більшістю сучасних платіжних систем для перевірки правильності структури номера картки. У випадку виявлення помилки користувач отримує відповідне повідомлення без надсилання запиту на сервер.

Після успішного проходження клієнтської перевірки дані передаються до серверної функції обробки платежу. Реалізацію серверної логіки наведено у додатку А.4. На першому етапі сервер перевіряє існування бронювання та його поточний статус. Оплата може бути виконана лише для бронювань зі статусом pending. Це запобігає повторному списанню коштів та дублюванню транзакцій.

У процесі обробки платежу створюється окремий запис у таблиці платежів, який містить унікальний ідентифікатор транзакції, суму платежу, спосіб оплати, останні чотири цифри картки та часову позначку виконання операції. Після

успішного завершення транзакції статус бронювання автоматично змінюється на confirmed, що підтверджує резервування обладнання на обраний період.

Для демонстраційних цілей у системі реалізовано механізм імітації платіжного шлюзу з випадковим визначенням результату операції. Такий підхід дозволяє протестувати обробку як успішних, так і помилкових сценаріїв оплати, включаючи відображення повідомлень про помилки та можливість повторної спроби здійснення платежу.

Особливу увагу приділено забезпеченню надійності процесу оплати. Під час виконання транзакції інтерфейс блокує повторне надсилання форми, а користувачеві відображається індикатор виконання операції. Це виключає можливість випадкового дублювання платежів через багаторазове натискання кнопки підтвердження.

Для уніфікації процесів управління платежами в системі використовується набір статусів, що відображають поточний стан фінансової операції. Характеристика основних статусів наведена в таблиці 3.1.

Таблиця 3.1 – Статуси платежів та їх обробка

Статус	**Опис**	**Дія системи**	**Дія користувача**
pending	Очікує оплати	Резервує обладнання на визначений період та очікує підтвердження платежу	Завершити оплату замовлення
processing	Платіж перебуває в обробці	Виконує перевірку та обробку транзакції, блокує повторне надсилання запиту	Очікувати завершення операції
completed	Платіж успішно виконано	Змінює статус бронювання на підтверджений, створює запис про оплату та надсилає підтвердження	Отримати підтвердження бронювання
failed	Платіж не виконано	Фіксує помилку та зберігає інформацію про невдалу спробу оплати	Повторити спробу оплати
refunded	Кошти повернено користувачу	Виконує повернення платежу та оновлює статус операції	Отримати повернуті кошти

Таблиця 3.1 демонструє повний життєвий цикл платежу – від створення бронювання до успішного завершення операції або повернення коштів. Використання статусної моделі дозволяє забезпечити контроль усіх фінансових

процесів та створює основу для подальшої інтеграції з реальними платіжними платформами.

У ході реалізації функціоналу системи було розроблено основні модулі, необхідні для роботи сервісу оренди обладнання. Реалізовано механізми реєстрації та авторизації користувачів із використанням безпечного хешування паролів, систему створення та управління бронюваннями, а також модуль обробки платежів. Забезпечено автоматичну перевірку доступності обладнання, розрахунок вартості оренди з урахуванням знижок і застави, зміну статусів бронювань та платежів відповідно до бізнес-логіки системи. У результаті створено цілісний функціональний комплекс, який підтримує повний цикл взаємодії користувача із сервісом – від реєстрації до підтвердження оплаченого бронювання.

3.2 Розробка модуля відстеження доступності обладнання в реальному часі

Модуль відстеження доступності обладнання (додаток Б.1) є одним із ключових компонентів системи бронювання, оскільки забезпечує актуальну інформацію про наявність ресурсів у режимі, близькому до реального часу. Його основне завдання полягає у запобіганні конфліктам бронювань, перевантаженню обладнання та забезпеченні коректного планування оренди.

Архітектурно модуль побудований на основі серверної логіки (Server-side queries) із використанням централізованої бази даних бронювань. Обчислення доступності виконується динамічно при кожному запиті користувача, що дозволяє уникнути застарілих даних та підвищує точність відображення стану обладнання.

Алгоритм визначення доступної кількості обладнання базується на аналізі всіх активних бронювань, що перетинаються з обраним часовим проміжком. До активних відносяться бронювання зі статусами: pending, confirmed, active.

Спочатку система отримує повний список бронювань для конкретного обладнання. Далі формується діапазон дат користувацького запиту, який розбивається на окремі календарні дні. Для кожного дня виконується підрахунок кількості вже зайнятих одиниць обладнання.

Після цього обчислюється залишкова доступна кількість як різниця між загальною кількістю одиниць обладнання та сумою зайнятих одиниць у кожен конкретний день. Кінцевим результатом є мінімальне значення доступності за весь період, що гарантує коректність бронювання навіть у випадку часткових накладень дат.

Якщо виявляється, що доступність дорівнює нулю або є недостатньою для запитаної кількості, система додатково формує альтернативні часові проміжки, у яких обладнання є доступним.

Логіка процесу реалізована як послідовність етапів: отримання бронювань → фільтрація активних записів → побудова календарного діапазону → аналіз перетинів дат → обчислення зайнятості → визначення доступності.

Реалізація алгоритму перевірки доступності виконується на рівні серверної логіки через окрему функцію запитів (додаток Б.2). Функція приймає ідентифікатор обладнання та часовий проміжок, після чого виконує обчислення доступних одиниць.

Спочатку відбувається перевірка існування обладнання в базі даних. У випадку відсутності запису система повертає помилку. Далі вибираються всі бронювання, що належать до цього обладнання та мають активний статус.

Після цього генерується список дат у межах заданого періоду. Для кожної дати виконується перевірка перетину з існуючими бронюваннями. Якщо дата входить у проміжок бронювання, кількість зайнятих одиниць збільшується відповідно до кількості забронюваного обладнання.

Отримані значення використовуються для формування масиву щоденної доступності, де для кожного дня визначається кількість доступних одиниць. На основі цього масиву обчислюється мінімальне значення доступності за весь період, що і визначає фінальний результат.

У випадку повної або часткової недоступності система ініціює пошук альтернативних періодів, що дозволяє користувачу обрати інші доступні дати без необхідності повторного ручного пошуку.

Для покращення взаємодії з користувачем реалізовано інтерактивний календар, який відображає стан доступності обладнання у вигляді кольорової індикації (додаток Б.3). Компонент працює на стороні клієнта та отримує дані з API модуля доступності.

Після завантаження сторінки виконується запит, який отримує інформацію про доступність на горизонті до 60 днів. Отримані дані зберігаються у структурі Map для швидкого доступу до значень за конкретною датою.

Кожна дата календаря відображається з відповідним стилем залежно від рівня доступності: повністю доступні дні, дні з обмеженою кількістю одиниць та повністю зайняті дати. Минулі дати автоматично блокуються для вибору.

Користувач може обрати діапазон дат, після чого система автоматично передає вибраний період до функції бронювання. Це забезпечує тісну інтеграцію між модулем доступності та модулем створення бронювань.

Таким чином, календар виступає не лише візуальним інтерфейсом, а й інтерактивним інструментом планування, що підвищує точність вибору дат та зменшує ймовірність конфліктів бронювання.

Для забезпечення роботи модуля відстеження доступності в реальному часі реалізовано окремий API-ендпоінт /api/availability, який відповідає за отримання актуальної інформації про стан обладнання на заданий період. Даний підхід дозволяє відокремити бізнес-логіку розрахунку доступності від клієнтського інтерфейсу та забезпечує повторне використання функціоналу у різних частинах системи (календар, сторінка товару, адміністративна панель).

API реалізовано у вигляді GET-ендпоінта, який приймає три обов’язкові параметри:

- equipmentId – ідентифікатор обладнання;
- startDate – дата початку перевірки доступності;
- endDate – дата завершення періоду.

На першому етапі виконується перевірка наявності всіх параметрів. У випадку відсутності хоча б одного з них система повертає HTTP 400 із повідомленням про помилку. Це дозволяє уникнути некоректних запитів ще до виконання бізнес-логіки.

Далі здійснюється валідація формату дат за допомогою регулярного виразу YYYY-MM-DD. Такий підхід гарантує уніфікованість вхідних даних і запобігає помилкам при перетворенні рядків у об’єкти дати.

Після цього виконується перевірка логічної коректності діапазону: дата початку не може бути більшою за дату завершення. У разі порушення цього правила API повертає помилку з відповідним поясненням.

Після успішної валідації параметрів API передає дані до функції розрахунку доступності checkAvailability(), яка виконує основну обчислювальну роботу:

- отримання обладнання з бази даних;
- аналіз активних бронювань;
- розрахунок зайнятості по кожній даті;
- визначення мінімальної доступної кількості.

Таким чином, API виконує роль лише транспортного шару, тоді як вся логіка знаходиться у виділеному сервісному модулі.

Результат роботи функції доповнюється полем checkedAt, яке фіксує час виконання запиту. Це дозволяє відслідковувати актуальність даних і може використовуватись для кешування або діагностики.

У випадку виникнення будь-яких виключень під час виконання запиту (наприклад, помилки бази даних або некоректні дані), API повертає HTTP 500 із узагальненим повідомленням. Такий підхід забезпечує безпеку системи, оскільки не розкриває внутрішні деталі реалізації клієнтській стороні.

Відповідь API має чітко визначену структуру, що дозволяє уніфіковано використовувати її у фронтенд-компонентах системи.

- available – мінімальна кількість доступного обладнання у вибраному періоді;

- totalQuantity – загальна кількість одиниць обладнання в системі;
- bookedDates – масив інформації по кожній даті, що включає зайняті та доступні одиниці;
- bookedDates.date – конкретна дата у форматі ISO;
- bookedDates.bookedQuantity – кількість зайнятих одиниць на дату;
- bookedDates.availableQuantity – кількість доступних одиниць;
- alternativePeriods – список альтернативних періодів у разі відсутності доступності;
- checkedAt – час останнього розрахунку доступності.

Така структура забезпечує гнучкість використання API як для календаря, так і для сторінок бронювання та адміністративної аналітики.

Взаємодія між користувачем, клієнтським інтерфейсом, API та базою даних реалізована у вигляді послідовного процесу, який забезпечує отримання актуальних даних у реальному часі (рис. 3.3).

Спочатку користувач відкриває сторінку товару або календар бронювання. Після цього клієнтський компонент надсилає HTTP GET-запит до /api/availability із параметрами обраного обладнання та діапазону дат.

API-ендпоінт передає запит до модуля бізнес-логіки checkAvailability(), який звертається до бази даних та отримує всі бронювання для конкретного обладнання. Після цього виконується розрахунок доступності по кожній даті.

Далі користувач обирає діапазон дат, і система виконує валідацію вибору. У разі успішної перевірки користувач переходить до етапу бронювання, а у випадку недоступності система пропонує альтернативні періоди.

У межах підрозділу було розроблено модуль відстеження доступності обладнання в реальному часі, який забезпечує точний розрахунок вільних одиниць ресурсів на основі активних бронювань та заданого часовго періоду. Реалізовано алгоритм аналізу перетину дат, серверну функцію обчислення доступності та API-ендпоінт для отримання актуальних даних. Додатково впроваджено клієнтську візуалізацію у вигляді інтерактивного календаря, що дозволяє користувачу зручно оцінювати доступність та обирати оптимальні дати

бронювання. Модуль підвищує точність планування та зменшує ризик конфліктів при оренді обладнання.

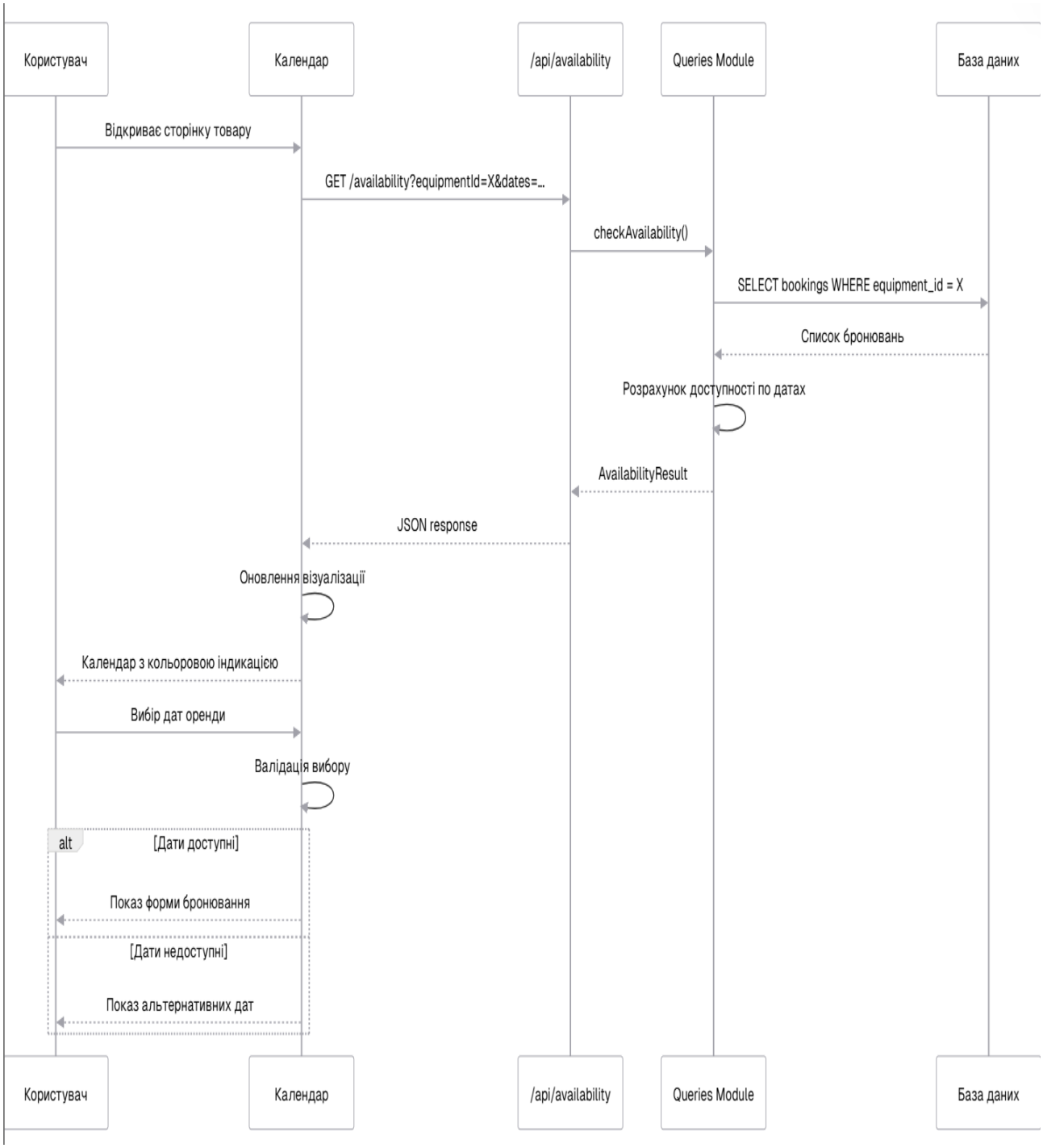


Рисунок 3.3 – Діаграма взаємодії компонентів модуля доступності

Отриманий результат повертається у форматі JSON до API, який, у свою чергу, передає його клієнтському календарю. Після отримання даних інтерфейс

оновлюється та відображає кольорову індикацію доступності: доступні, обмежені та недоступні дати.

3.3 Тестування вебзастосунку та аналіз результатів роботи системи

На етапі тестування вебзастосунку було реалізовано комплексну перевірку всіх ключових функціональних модулів системи, зокрема логіки бронювання, розрахунку доступності обладнання, роботи бази даних та допоміжних утилітарних функцій. Основною метою тестування було підтвердження коректності бізнес-логіки, стійкості системи до некоректних вхідних даних та відповідності очікуваній поведінці в різних сценаріях використання.

Для реалізації тестування було використано фреймворк Jest, який дозволяє виконувати модульні (unit) тести з автоматичною перевіркою результатів через assert-вирази (expect). Усі тести були згруповані за функціональними блоками, що відповідають архітектурі застосунку: модуль перевірки доступності обладнання, модуль роботи з базою даних та набір утилітарних функцій, які відповідають за розрахунки та валідацію даних користувача.

Особливу увагу було приділено тестуванню логіки доступності обладнання, оскільки саме цей компонент є критично важливим для системи оренди. У межах цього блоку перевірялася функція checkAvailability, яка аналізує активні бронювання, виключає скасовані та завершені записи, а також визначає перетин дат між існуючими та новими замовленнями. Додатково тестувалася функція getAvailabilityByDay, що формує календарну доступність по кожному дню. Тести охоплювали різні сценарії: повну доступність без бронювань, часткове резервування обладнання, повну відсутність вільних одиниць, а також ситуації, коли бронювання не перетинаються по датах. Це дозволило переконатися, що алгоритм коректно враховує часові інтервали та кількість доступного ресурсу.

Окремий блок тестів було присвячено генерації діапазону дат через функцію generateDateRange. Вона перевірялася на коректність формування одного дня, кількох днів та довших періодів, зокрема тижневих інтервалів. Це є важливим для подальшої побудови календаря доступності та відображення даних у користувацькому інтерфейсі.

Другий великий блок тестування стосувався модуля InMemoryDB, який імітує роботу бази даних. Тут перевірялися CRUD-операції для користувачів, категорій, обладнання та бронювань. Зокрема, тестувалося створення нових записів, пошук за email та slug, оновлення даних (наприклад, зміна ролі користувача або ціни обладнання), а також видалення записів. Додатково перевірялася логіка фільтрації бронювань за користувачем і діапазоном дат, що є критично важливим для роботи системи оренди. Ці тести дозволили підтвердити стабільність внутрішньої структури даних і коректність взаємодії між сутностями.

Третій блок тестів охоплював утилітарні функції системи, які відповідають за розрахунок вартості оренди, знижок, валідацію введених даних та форматування. Зокрема, перевірялася функція розрахунку кількості днів оренди, яка враховує включно початкову та кінцеву дату, а також система знижок, що залежить від тривалості оренди (10%, 15% та 20% для різних періодів). Окремо тестувалися функції перевірки email та телефонних номерів, валідації діапазону дат, генерації slug для URL-адрес, а також форматування цін у гривнях. Важливим елементом також стала перевірка алгоритму Луна для валідації банківських карток, що використовується у модулі оплати.

Результати виконання тестів (рис. 3.4) показали повну коректність реалізації логіки системи. Усі тестові сценарії були успішно пройдені без помилок.

Отриманий результат свідчить про стабільну роботу всіх основних модулів системи та відсутність критичних помилок у бізнес-логіці. Час виконання тестів (менше 10 секунд) також вказує на оптимальну продуктивність та ефективність реалізації алгоритмів.

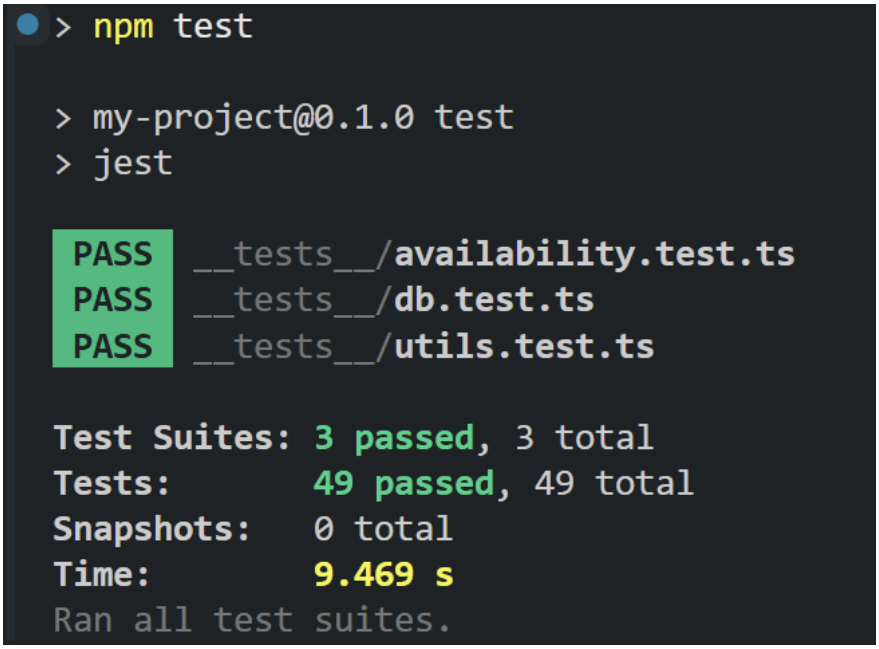


Рисунок 3.4 – Результати тестування

Таким чином, проведене тестування підтвердило, що вебзастосунок є функціонально коректним, стабільним і готовим до подальшого використання або розгортання в реальному середовищі.

3.4 Інструкція користувача

Розроблений вебзастосунок для онлайн-оренди обладнання та інструментів призначений для зручної взаємодії між користувачем і системою з метою пошуку, бронювання та оплати необхідного обладнання, а також контролю його доступності в реальному часі. Робота із системою починається з реєстрації або авторизації користувача. Для створення облікового запису користувач заповнює базову форму, що містить електронну пошту, пароль, ім’я та контактний номер телефону. Після успішної реєстрації користувач отримує доступ до особистого кабінету, де може переглядати свої бронювання, історію замовлень та актуальний статус оренди.

Після входу в систему користувач переходить до каталогу обладнання, який структурований за категоріями для спрощення навігації. Кожна позиція містить детальний опис, фотографію, технічні характеристики, вартість оренди за добу та інформацію про доступну кількість одиниць. Система автоматично відображає актуальну доступність обладнання на обрані дати, що дозволяє уникнути конфліктів бронювання ще на етапі вибору.

Для оформлення бронювання користувач обирає потрібне обладнання, вказує період оренди та кількість одиниць. Після цього система виконує автоматичну перевірку доступності з урахуванням уже існуючих бронювань, їх статусів та перетину дат. У разі наявності достатньої кількості ресурсів користувач переходить до підтвердження замовлення та оплати. Якщо доступної кількості недостатньо, система інформує користувача про обмеження та пропонує скоригувати параметри бронювання.

Оплата здійснюється через інтегрований механізм фінансових операцій, після чого бронювання отримує відповідний статус (наприклад, «підтверджено»). Усі операції фіксуються в системі та доступні для перегляду в особистому кабінеті користувача. Додатково користувач має можливість скасувати бронювання відповідно до правил платформи.

Адміністратор системи має розширені права доступу, що дозволяє керувати категоріями обладнання, додавати нові позиції, змінювати їх характеристики, а також контролювати всі бронювання та користувацькі замовлення. Це забезпечує повний цикл управління системою та підтримує її коректне функціонування.

Завдяки реалізованій логіці перевірки доступності в реальному часі, система забезпечує актуальність даних навіть при одночасній роботі великої кількості користувачів. Це підвищує надійність сервісу та мінімізує ризик подвійного бронювання одного й того самого обладнання.

ВИСНОВКИ

У межах виконання бакалаврської роботи було розглянуто та реалізовано комплексне рішення для створення вебзастосунку онлайн-оренди обладнання та інструментів, що поєднує теоретичні засади побудови подібних систем із практичною реалізацією повноцінного функціонального прототипу. У першому розділі роботи було проаналізовано понятійний апарат вебзастосунків для систем оренди та бронювання ресурсів, визначено їх ключові особливості, зокрема необхідність обробки динамічних даних, управління доступністю ресурсів у реальному часі та забезпечення коректної роботи механізмів бронювання. Окрему увагу було приділено аналізу існуючих рішень на ринку, що дозволило виявити їхні переваги та недоліки, а також сформувати вимоги до власної системи, орієнтованої на зручність користувача, точність розрахунків та масштабованість.

Другий розділ бакалаврської роботи був присвячений детальному проєктуванню вебзастосунку для онлайн-оренди обладнання та інструментів як цілісної інформаційної системи. На цьому етапі було сформовано комплекс вимог до майбутньої системи, які охоплюють як функціональну складову, так і нефункціональні характеристики. Зокрема, до функціональних вимог було віднесено можливість реєстрації та авторизації користувачів, перегляду каталогу обладнання, створення та керування бронюваннями, а також обробку інформації про доступність ресурсів і розрахунок вартості оренди. Нефункціональні вимоги включали забезпечення швидкодії системи, масштабованості, надійності збереження даних, зручності користувацького інтерфейсу та захищеності персональної інформації користувачів.

У межах проєктування архітектури системи було визначено основні компоненти вебзастосунку та їх взаємодію, що дозволило сформувати логічну структуру поділу відповідальності між клієнтською та серверною частинами. Такий підхід забезпечує гнучкість розвитку системи, спрощує її супровід та

дозволяє незалежно масштабувати окремі модулі. Окрему увагу було приділено побудові потоків даних між компонентами системи, що є критично важливим для коректної обробки бронювань у реальному часі та уникнення конфліктів при одночасному доступі до ресурсів.

Особливо важливим етапом другого розділу стало проєктування бази даних, яка є ядром усієї системи. Було визначено основні сутності, зокрема користувачів, категорії обладнання, самі одиниці обладнання, бронювання та платежі. Для кожної сутності було продумано набір атрибутів, що забезпечують повне відображення предметної області, а також встановлено зв’язки між таблицями, які гарантують цілісність і узгодженість даних. Наприклад, зв’язок між користувачами та бронюваннями дозволяє відстежувати історію оренд, а зв’язок між обладнанням і бронюваннями – контролювати доступну кількість ресурсів у різні проміжки часу. Окремо було враховано необхідність обробки статусів бронювань, що дає змогу відрізняти активні замовлення від скасованих або завершених і коректно розраховувати доступність.

Також у межах проєктування було розроблено логіку управління бронюваннями, яка базується на аналізі перетину дат та перевірці доступної кількості обладнання. Ця логіка є критичною для запобігання перевищенню допустимої кількості орендованих одиниць і забезпечує коректне функціонування системи в умовах паралельних запитів. Окрім цього, було сформовано підхід до реалізації користувацького інтерфейсу, орієнтованого на інтуїтивну взаємодію користувача із системою, а також обрано технологічний стек, який забезпечує баланс між продуктивністю, простотою розробки та можливістю подальшого масштабування проєкту.

Третій розділ роботи був спрямований на практичну реалізацію розробленої концепції та проєктних рішень. У процесі розробки було реалізовано ключові модулі вебзастосунку, які формують його основну функціональність. Серед них – система реєстрації та авторизації користувачів, що забезпечує розмежування прав доступу, модуль управління бронюваннями, який дозволяє створювати, оновлювати та відстежувати замовлення, а також

функціонал розрахунку вартості оренди з урахуванням тривалості використання та кількості одиниць обладнання.

Окрему увагу було приділено реалізації механізму перевірки доступності обладнання в реальному часі. Даний модуль аналізує поточні бронювання, враховує їх статуси, виключає неактивні записи та визначає фактичну кількість доступних одиниць обладнання на заданий проміжок часу. Завдяки цьому забезпечується коректна робота системи навіть у випадках одночасного створення кількох бронювань різними користувачами. Реалізована логіка дозволяє уникнути конфліктів у даних та гарантує, що користувач отримає актуальну інформацію про доступність ресурсів у момент оформлення замовлення.

У результаті поєднання проєктних рішень і практичної реалізації було створено функціонально завершений вебзастосунок, який відповідає сучасним вимогам до систем онлайн-оренди та може бути використаний як основа для подальшого розвитку і масштабування.

Окремим етапом стало тестування вебзастосунку, що відіграло ключову роль у перевірці правильності реалізованої логіки. Було проведено модульне тестування основних функцій системи, включаючи перевірку доступності обладнання, роботу бази даних та утилітарних функцій розрахунку. Тести охоплювали різні сценарії використання: повну доступність ресурсів, часткову зайнятість, повне блокування обладнання, ігнорування скасованих бронювань, а також перевірку коректності обробки дат і фінансових розрахунків. Окремо було протестовано функції валідації користувацьких даних, таких як email, телефонні номери та банківські картки, що забезпечує додатковий рівень надійності системи.

Результати тестування показали, що всі 49 тестів були успішно виконані, при цьому не було виявлено жодного проваленого тесту, що свідчить про стабільність і коректність реалізованої логіки. Загальний час виконання тестів склав 9.469 секунд, а всі три тестові набори – перевірка доступності обладнання, робота бази даних та утилітарні функції – пройшли успішно. Це підтверджує, що

система коректно обробляє бізнес-логіку, правильно розраховує доступність ресурсів та забезпечує цілісність даних при різних сценаріях використання.

Таким чином, у ході виконання бакалаврської роботи було розроблено повноцінний вебзастосунок для онлайн-оренди обладнання, який відповідає сучасним вимогам до подібних систем, забезпечує точність розрахунків, надійність обробки даних та зручність користування. Отримані результати можуть бути використані як основа для подальшого розширення функціоналу системи, зокрема інтеграції платіжних сервісів у реальному часі, впровадження аналітики попиту та масштабування до багатокористувацької платформи з високим навантаженням.

ПЕРЕЛІК ПОСИЛАНЬ

1. Raisinghani M. S. 1.1 Introduction to information systems - Foundations of Information Systems | OpenStax. *OpenStax*. 2025. URL : https://openstax.org /books/foundations-information-systems/pages/1-1-introduction-to-information-systems (дата звернення : 04.05.2026).

2. Wiesman F., Hasman A., Van Den Herik H. Information retrieval : an overview of system characteristics. *International Journal of Medical Informatics*. 1997. Vol. 47, no. 1–2. P. 5–26. DOI : https://doi.org/10.1016/s1386-5056(97)00094-4.

3. Клієнт-серверна архітектура. *QATestLab*. 2024. URL : https://training.qatestlab.com/blog/technical-articles/client-server-architecture/ (дата звернення : 04.05.2026).

4. Тарасюк В. Р. Вебзастосунок для бронювання житлових та нежитлових приміщень : пояснювальна записка до кваліфікаційної роботи здобувача СВО «Магістр» / кер. С. В. Шаров. Запоріжжя, 2025. 92 с.

5. Програмне забезпечення для бронювання житла для відпочинку. *Anolla*. URL : https://anolla.com/uk/programne-zabezpecennia-dlia-orendi-zitla-na-cas-vidpustki (дата звернення : 04.05.2026).

6. Tool & Equipment Rental in the US – Industry Report. *IBISWorld*. 2025. URL : https://www.ibisworld.com/united-states/industry/tool-equipment-rental/1376/ (дата звернення : 04.05.2026).

7. Equipment Rental Software Market Size, Growth – Forecast [2033]. *Market Reports World*. 2024. URL : https://www.marketreportsworld.com/market-reports/equipment-rental-software-market-14720435 (дата звернення : 04.05.2026).

8. Tool Rental Apps Market Size – CAGR of 14.6%. *Market.us*. 2025. URL : https://market.us/report/tool-rental-apps-market/ (дата звернення : 04.05.2026).

9. Fat Llama – How It Works. *Business Model Canvas Template*. 2026. URL : https://businessmodelcanvastemplate.com/blogs/how-it-works/fat-llama-how-it-works (дата звернення : 04.05.2026).

10. United Rentals' Digital Transformation: How AI and AR Are Reshaping Industrial Efficiency and ROI. *AInvest*. 2025. URL : https://www.ainvest.com/news/united-rentals-digital-transformation-ai-ar-reshaping-industrial-efficiency-roi-2508/ (дата звернення : 04.05.2026).

11. United Rentals Expands Digital Platform with Smart Suggestions and Equipment Fit AR Capabilities. *United Rentals*. 2025. URL : https://investors.unitedrentals.com/press-releases/press-releases-details/2025/United-Rentals-Expands-Digital-Platform-with-Smart-Suggestions-and-Equipment-Fit-Augmented-Reality-Capabilities/default.aspx (дата звернення : 04.05.2026).

12. Sunbelt Rentals expands Trackunit partnership. *Trackunit*. 2025. URL : https://trackunit.com/press/sunbelt-trackunit-partnership/ (дата звернення : 04.05.2026).

13. Sunbelt Rentals transforms B2B sales experience with Adobe. *Adobe Business*. 2023. URL : https://business.adobe.com/customer-success-stories/sunbelt-rentals-case-study.html (дата звернення : 04.05.2026).

14. P2P Rental Apps Market Size & Share 2025–2034. *GMInsights*. 2024. URL : https://www.gminsights.com/industry-analysis/p2p-rental-apps-market (дата звернення : 04.05.2026).

15. Future of Rental Asset Management Software: AI, IoT, and Automation Trends. *SuiteWorks* Tech. 2025. URL : https://suiteworkstech.com/future-rental-asset-management-software-ai-iot/ (дата звернення : 04.05.2026).

16. Next.js Documentation. *Vercel*. URL : https://nextjs.org/ (дата звернення : 04.05.2026).

17. SQLite Documentation. URL : https://sqlite.org/docs.html (дата звернення : 04.05.2026).

18. React Documentation. URL : https://react.dev/ (дата звернення : 04.05.2026).

19. TypeScript Documentation. URL : https://www.typescriptlang.org/docs/handbook/typescript-in-5-minutes.html (дата звернення : 04.05.2026).

20. Tailwind CSS Documentation. URL : https://tailwindcss.com/ (дата звернення : 04.05.2026).

21. shadcn/ui Documentation. URL : https://ui.shadcn.com/ (дата звернення : 04.05.2026).

22. Node.js Documentation. URL : https://nodejs.org/en (дата звернення : 04.05.2026).

23. bcryptjs package. npm. URL : https://www.npmjs.com/package/bcryptjs (дата звернення : 04.05.2026).

ДОДАТОК А

Коди для реалізації функціоналу системи

Лістинг А.1 Реалізація Server Action для реєстрації користувача

```
// lib/actions.ts
'use server'


import bcrypt from 'bcryptjs'
import { cookies } from 'next/headers'
import { redirect } from 'next/navigation'
import { db } from '@/lib/db'

export async function registerUser(formData: FormData) {
  const email    = formData.get('email')    as string
  const password = formData.get('password') as string
  const name     = formData.get('name')     as string

  if (!email || !password || !name) {
    return { error: "Всі поля обов'язкові" }
  }

  if (password.length < 6) {
    return { error: 'Пароль має містити мінімум 6 символів' }
  }

  // Перевірка існування користувача через SQLite
```

```
const existingUser = db
  .prepare('SELECT id FROM users WHERE email = ?')
  .get(email)

if (existingUser) {
  return { error: 'Користувач з такою поштою вже існує' }
}

const hashedPassword = await bcrypt.hash(password, 10)
const userId         = crypto.randomUUID()
const now            = new Date().toISOString()

// Збереження нового користувача в SQLite
db.prepare(`
  INSERT INTO users (id, email, password, name, role, created_at, updated_at)
  VALUES (?, ?, ?, ?, 'customer', ?, ?)
`).run(userId, email, hashedPassword, name, now, now)

// Створення сесії та збереження токена в SQLite
const sessionToken = crypto.randomUUID()
const expiresAt    = new Date(Date.now() + 7 * 24 * 60 * 60 *
1000).toISOString()

db.prepare(`
  INSERT INTO sessions (token, user_id, expires_at, created_at)
  VALUES (?, ?, ?, ?)
`).run(sessionToken, userId, expiresAt, now)

// Запис токена у httpOnly cookie
const cookieStore = await cookies()
```

```
  cookieStore.set('session', sessionToken, {
    httpOnly: true,
    secure:   process.env.NODE_ENV === 'production',
    sameSite: 'lax',
    maxAge:   60 * 60 * 24 * 7,
  })

  redirect('/')
}
```

Лістинг А.2 Server Action для створення бронювання

```
// lib/actions.ts

'use server'

import { redirect } from 'next/navigation'
import { db } from '@/lib/db'
import { getSession } from '@/lib/session'
import { checkAvailability } from '@/lib/queries'

export async function createBooking(formData: FormData) {
  const session = await getSession()
  if (!session) {
    return { error: 'Необхідна авторизація' }
  }

  const equipmentId = formData.get('equipmentId') as string
  const startDate   = formData.get('startDate')   as string
```

```
const endDate     = formData.get('endDate')     as string
const quantity    = parseInt(formData.get('quantity') as string) || 1

// Отримання обладнання з SQLite
const equipment = db
  .prepare('SELECT * FROM equipment WHERE id = ? AND is_active = 1')
  .get(equipmentId) as Equipment | undefined

if (!equipment) {
  return { error: 'Обладнання недоступне' }
}

// Перевірка доступності через SQLite
const availability = checkAvailability(equipmentId, startDate, endDate)
if (availability.available < quantity) {
  return {
    error: `Доступно лише ${availability.available} од. на обрані дати`,
  }
}

// Розрахунок вартості
const days          = calculateDays(startDate, endDate)
const basePrice     = equipment.price_per_day * days * quantity
const discount      = calculateDiscount(days, basePrice)
const totalPrice    = basePrice - discount
const depositAmount = equipment.deposit * quantity

const bookingId = crypto.randomUUID()
const orderId   = generateOrderId()
const now       = new Date().toISOString()
```

```
  // Збереження бронювання в SQLite
  db.prepare(`
    INSERT INTO bookings
      (id, order_id, user_id, equipment_id, start_date, end_date,
       quantity, total_price, deposit_amount, status, created_at, updated_at)
    VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, 'pending', ?, ?)
  `).run(
    bookingId, orderId, session.userId, equipmentId,
    startDate, endDate, quantity, totalPrice, depositAmount, now, now,
  )

  redirect(`/checkout/${bookingId}`)
}

function calculateDays(startDate: string, endDate: string): number {
  const start = new Date(`${startDate}T00:00:00Z`)
  const end   = new Date(`${endDate}T00:00:00Z`)
  return Math.ceil((end.getTime() - start.getTime()) / (1000 * 60 * 60 * 24)) + 1
}

function calculateDiscount(days: number, basePrice: number): number {
  if (days >= 30) return basePrice * 0.20
  if (days >= 14) return basePrice * 0.15
  if (days >= 7)  return basePrice * 0.10
  return 0
}

function generateOrderId(): string {
  const year = new Date().getFullYear()
```

```
  const rand = Math.floor(1000 + Math.random() * 9000)
  return `RG-${year}-${rand}`
}
```

Лістинг А.3 Компонент форми оплати

```
// components/checkout-form.tsx
'use client'

import { useState } from 'react'
import { useRouter } from 'next/navigation'
import { processPayment } from '@/lib/actions/payment'
import {
  Card, CardContent, CardHeader, CardTitle,
} from '@/components/ui/card'
import { Button }    from '@/components/ui/button'
import { Input }     from '@/components/ui/input'
import { Label }     from '@/components/ui/label'
import { Separator } from '@/components/ui/separator'
import { Loader2, CreditCard } from 'lucide-react'
import { toast } from 'sonner'
import { validateCardNumber, formatCardNumber, formatExpiry } from
'@/lib/card-utils'

interface CheckoutFormProps {
  booking:   Booking
  equipment: Equipment
}
```

```
export function CheckoutForm({ booking, equipment }: CheckoutFormProps) {
  const router = useRouter()
  const [isProcessing, setIsProcessing] = useState(false)
  const [cardData, setCardData] = useState({
    number: '',
    expiry: '',
    cvv:    '',
    name:   '',
  })

  const totalAmount = booking.total_price + booking.deposit_amount

  async function handleSubmit(e: React.FormEvent) {
    e.preventDefault()
    setIsProcessing(true)

    try {
      if (!validateCardNumber(cardData.number)) {
        throw new Error('Невірний номер картки')
      }

      const result = await processPayment({
        bookingId:     booking.id,
        amount:        totalAmount,
        cardLast4:     cardData.number.slice(-4),
        paymentMethod: 'card',
        // токенізована картка передається на сервер (PCI DSS)
        cardToken:     cardData.number,
      })
```

```
    if (result.success) {
      router.push('/my-bookings?success=true')
    } else {
      throw new Error(result.error)
    }
  } catch (error: any) {
    toast.error(error.message)
  } finally {
    setIsProcessing(false)
  }
}

return (
  <form onSubmit={handleSubmit} className="space-y-6">
    {/* Сума до оплати */}
    <Card>
      <CardHeader>
        <CardTitle>Деталі замовлення</CardTitle>
      </CardHeader>
      <CardContent className="space-y-2">
        <div className="flex justify-between">
          <span>Оренда ({booking.days} днів)</span>
          <span>{booking.total_price} грн</span>
        </div>
        <div className="flex justify-between">
          <span>Застава (повертається)</span>
          <span>{booking.deposit_amount} грн</span>
        </div>
        <Separator />
        <div className="flex justify-between font-bold text-lg">
```

```
      <span>До оплати</span>
      <span>{totalAmount} грн</span>
    </div>
  </CardContent>
</Card>

{/* Форма картки */}
<Card>
  <CardHeader>
    <CardTitle>Платіжні дані</CardTitle>
  </CardHeader>
  <CardContent className="space-y-4">
    <div>
      <Label>Номер картки</Label>
      <Input
        placeholder="0000 0000 0000 0000"
        value={formatCardNumber(cardData.number)}
        onChange={(e) =>
          setCardData({ ...cardData, number: e.target.value.replace(/\s/g, '') })
        }
        maxLength={19}
      />
    </div>

    <div className="grid grid-cols-2 gap-4">
      <div>
        <Label>Термін дії</Label>
        <Input
          placeholder="MM/YY"
          value={cardData.expiry}
```

```
      onChange={(e) =>
        setCardData({ ...cardData, expiry: formatExpiry(e.target.value) })
      }
      maxLength={5}
    />
  </div>
  <div>
    <Label>CVV</Label>
    <Input
      type="password"
      placeholder="***"
      value={cardData.cvv}
      onChange={(e) =>
        setCardData({ ...cardData, cvv: e.target.value.slice(0, 3) })
      }
      maxLength={3}
    />
  </div>
</div>

<div>
  <Label>Ім&amp;apos;я власника картки</Label>
  <Input
    placeholder="IVAN PETRENKO"
    value={cardData.name}
    onChange={(e) =>
      setCardData({ ...cardData, name: e.target.value.toUpperCase() })
    }
  />
</div>
```

```
        </CardContent>
      </Card>

      <Button
        type="submit"
        className="w-full"
        size="lg"
        disabled={isProcessing}
      >
        {isProcessing ? (
          <>
            <Loader2 className="animate-spin mr-2" />
            Обробка платежу...
          </>
        ) : (
          <>
            <CreditCard className="mr-2" />
            Оплатити {totalAmount} грн
          </>
        )}
      </Button>
    </form>
  )
}
```

Лістинг А.4 Server Action для обробки платежу

```
// lib/actions.ts
```

```
'use server'

import crypto from 'crypto'
import { db } from '@/lib/db'

interface PaymentData {
  bookingId:     string
  amount:        number
  cardLast4:     string
  paymentMethod: string
  cardToken:     string
}

// Формує підписаний запит до LiqPay
function buildLiqPayRequest(params: Record<string, string | number>): string
{
  const privateKey = process.env.LIQPAY_PRIVATE_KEY!
  const publicKey  = process.env.LIQPAY_PUBLIC_KEY!

  const data = Buffer.from(
    JSON.stringify({ ...params, public_key: publicKey }),
  ).toString('base64')

  const signature = crypto
    .createHash('sha1')
    .update(privateKey + data + privateKey)
    .digest('base64')

  return new URLSearchParams({ data, signature }).toString()
}
```

```
export async function processPayment(data: PaymentData) {
  // Отримання бронювання з SQLite
  const booking = db
    .prepare('SELECT * FROM bookings WHERE id = ?')
    .get(data.bookingId) as Booking | undefined

  if (!booking) {
    return { success: false, error: 'Бронювання не знайдено' }
  }

  if (booking.status !== 'pending') {
    return { success: false, error: 'Бронювання вже оплачено' }
  }

  // Виклик реального платіжного шлюзу LiqPay
  let gatewayTransactionId: string

  try {
    const response = await fetch('https://www.liqpay.ua/api/request', {
      method:  'POST',
      headers: { 'Content-Type': 'application/x-www-form-urlencoded' },
      body:    buildLiqPayRequest({
        action:      'pay',
        version:     '3',
        amount:      data.amount,
        currency:    'UAH',
        description: `Оренда обладнання, бронювання ${data.bookingId}`,
        order_id:    data.bookingId,
        card:        data.cardToken,
```

```
        }),
      })

      const result = await response.json()

      if (result.status !== 'success') {
        return {
          success: false,
          error:      result.err_description ?? 'Платіж відхилено. Перевірте дані картки.',
        }
      }

      gatewayTransactionId = result.transaction_id as string
    } catch {
      return { success: false, error: 'Помилка з\'єднання з платіжним шлюзом' }
    }

    const now       = new Date().toISOString()
    const paymentId = crypto.randomUUID()

    // Транзакція: запис оплати + оновлення статусу бронювання атомарно
    const savePayment = db.transaction(() => {
      db.prepare(`
        INSERT INTO payments
          (id, booking_id, amount, payment_method, status,
           transaction_id, card_last4, paid_at, created_at)
        VALUES (?, ?, ?, ?, 'completed', ?, ?, ?, ?)
      `).run(
        paymentId, booking.id, data.amount, data.paymentMethod,
```

```
        gatewayTransactionId, data.cardLast4, now, now,
      )

      db.prepare(`
        UPDATE bookings SET status = 'confirmed', updated_at = ? WHERE id =
?
      `).run(now, booking.id)
    })

    savePayment()

    return { success: true, transactionId: gatewayTransactionId }
  }
```

ДОДАТОК Б

Лістинг для розробки модуля відстеження доступності обладнання в реальному часі

Лістинг Б.1 Функція перевірки доступності

```
// lib/queries.ts

import { db } from '@/lib/db'

interface AvailabilityResult {
  available:          number
  totalQuantity:      number
  bookedDates:        BookedDate[]
  alternativePeriods: Period[]
}

interface BookedDate {
  date:             string
  bookedQuantity:   number
  availableQuantity: number
}

interface Period {
  startDate: string
  endDate:   string
}
```

```
export function checkAvailability(
  equipmentId: string,
  startDate:   string,   // YYYY-MM-DD
  endDate:     string,
): AvailabilityResult {
  // Отримання обладнання з SQLite
  const equipment = db
    .prepare('SELECT * FROM equipment WHERE id = ?')
    .get(equipmentId) as Equipment | undefined

  if (!equipment) throw new Error('Обладнання не знайдено')

  // Активні бронювання, що перетинаються з вибраним діапазоном (SQLite)
  const activeBookings = db.prepare(`
    SELECT start_date, end_date, quantity
    FROM   bookings
    WHERE  equipment_id = ?
      AND  status IN ('pending', 'confirmed', 'active')
      AND  start_date <= ?
      AND  end_date   >= ?
  `).all(equipmentId, endDate, startDate) as {
    start_date: string
    end_date:   string
    quantity:   number
  }[]

  // Генерація дат через UTC, щоб уникнути зсуву через локальний timezone
  const requestedDates = generateDateRangeUTC(startDate, endDate)
```

```
    const bookedDates: BookedDate[] = requestedDates.map(date => {
      const bookedQuantity = activeBookings
        .filter(b => date >= b.start_date && date <= b.end_date)
        .reduce((sum, b) => sum + b.quantity, 0)

      return {
        date,
        bookedQuantity,
        availableQuantity: equipment.quantity - bookedQuantity,
      }
    })

    const minAvailable = Math.min(...bookedDates.map(d => d.availableQuantity))

    const alternativePeriods =
      minAvailable <= 0
        ? findAlternativePeriods(equipmentId, startDate, endDate)
        : []

    return {
      available: Math.max(0, minAvailable),
      totalQuantity: equipment.quantity,
      bookedDates,
      alternativePeriods,
    }
  }

  // UTC-генерація дат: setUTCDate замість setDate
  function generateDateRangeUTC(start: string, end: string): string[] {
```

```
  const dates:   string[] = []
  const current = new Date(`${start}T00:00:00Z`)
  const endDay  = new Date(`${end}T00:00:00Z`)

  while (current <= endDay) {
    dates.push(current.toISOString().slice(0, 10))
    current.setUTCDate(current.getUTCDate() + 1)
  }

  return dates
}

function findAlternativePeriods(
  equipmentId: string,
  startDate:   string,
  endDate:     string,
): Period[] {
  const durationDays =
    (new Date(`${endDate}T00:00:00Z`).getTime() -
     new Date(`${startDate}T00:00:00Z`).getTime()) /
    (1000 * 60 * 60 * 24)

  // Шукаємо найближчі вільні вікна тієї ж тривалості (SQLite)
  const busyRanges = db.prepare(`
    SELECT start_date, end_date
    FROM   bookings
    WHERE  equipment_id = ?
      AND  status IN ('pending', 'confirmed', 'active')
    ORDER  BY start_date
  `).all(equipmentId) as { start_date: string; end_date: string }[]
```

```
const equipment = db
  .prepare('SELECT quantity FROM equipment WHERE id = ?')
  .get(equipmentId) as { quantity: number }

const alternatives: Period[] = []
let candidateStart = new Date(`${endDate}T00:00:00Z`)
candidateStart.setUTCDate(candidateStart.getUTCDate() + 1)

while (alternatives.length < 3) {
  const candidateEnd = new Date(candidateStart)
  candidateEnd.setUTCDate(candidateEnd.getUTCDate() + durationDays)

  const cs = candidateStart.toISOString().slice(0, 10)
  const ce = candidateEnd.toISOString().slice(0, 10)

  const conflict = busyRanges.some(
    r => r.start_date <= ce && r.end_date >= cs,
  )

  if (!conflict) {
    alternatives.push({ startDate: cs, endDate: ce })
  }

  candidateStart.setUTCDate(candidateStart.getUTCDate() + 1)

  // Не шукати далі ніж 6 місяців
  const limit = new Date()
  limit.setUTCMonth(limit.getUTCMonth() + 6)
  if (candidateStart > limit) break
```

```
  }

  return alternatives
}
```

Лістинг Б.2 Компонент календаря доступності

```
// components/availability-calendar.tsx
'use client'

import { useState, useEffect, useRef } from 'react'
import { Calendar }                    from '@/components/ui/calendar'
import { Skeleton }                    from '@/components/ui/skeleton'
import { uk }                          from 'date-fns/locale'

interface AvailabilityCalendarProps {
  equipmentId:       string
  quantity:          number
  onDateRangeSelect: (range: DateRange) => void
}

export function AvailabilityCalendar({
  equipmentId,
  quantity,
  onDateRangeSelect,
}: AvailabilityCalendarProps) {
  const [availability, setAvailability] =
    useState<Map<string, number>>(new Map())
  const [dateRange, setDateRange] = useState<DateRange>()
```

```
const [isLoading, setIsLoading] = useState(true)
const eventSourceRef            = useRef<EventSource | null>(null)

useEffect(() => {
  const today  = new Date().toISOString().slice(0, 10)
  const future = (() => {
    const d = new Date()
    d.setUTCDate(d.getUTCDate() + 60)
    return d.toISOString().slice(0, 10)
  })()

  // Початкове завантаження через HTTP GET
  fetch(`/api/availability?${new URLSearchParams({
    equipmentId,
    startDate: today,
    endDate:   future,
  })}`)
    .then(r => r.json())
    .then((data: { bookedDates: BookedDate[] }) => {
      const map = new Map<string, number>()
      data.bookedDates.forEach(d => map.set(d.date, d.availableQuantity))
      setAvailability(map)
      setIsLoading(false)
    })

  // SSE: підписка на зміни у реальному часі
  const es = new EventSource(
    `/api/availability/stream?equipmentId=${equipmentId}`,
  )
  eventSourceRef.current = es
```

```
  es.onmessage = (event: MessageEvent) => {
    const update = JSON.parse(event.data) as BookedDate
    setAvailability(prev => {
      const next = new Map(prev)
      next.set(update.date, update.availableQuantity)
      return next
    })
  }

  es.onerror = () => {
    // EventSource автоматично перепідключається
    console.warn('SSE: втрата з\'єднання, перепідключення...')
  }

  return () => {
    es.close()
  }
}, [equipmentId])

const isDateDisabled = (date: Date): boolean => {
  if (date < new Date()) return true
  const dateStr   = date.toISOString().slice(0, 10)
  const available = availability.get(dateStr) ?? quantity
  return available === 0
}

return (
  <div className="space-y-4">
    {/* Легенда */}
```

```
<div className="flex items-center gap-4 text-sm">
  <div className="flex items-center gap-2">
    <div className="size-4 rounded bg-primary/20" />
    <span>Доступно</span>
  </div>
  <div className="flex items-center gap-2">
    <div className="size-4 rounded bg-warning/20" />
    <span>Обмежено</span>
  </div>
  <div className="flex items-center gap-2">
    <div className="size-4 rounded bg-destructive/20" />
    <span>Зайнято</span>
  </div>
</div>

{isLoading ? (
  <Skeleton className="h-[300px] w-full" />
) : (
  <Calendar
    mode="range"
    selected={dateRange}
    onSelect={(range) => {
      setDateRange(range)
      if (range?.from && range?.to) onDateRangeSelect(range)
    }}
    locale={uk}
    disabled={isDateDisabled}
    modifiers={{
      available: (date) => {
        const s = date.toISOString().slice(0, 10)
```

```
            return (availability.get(s) ?? quantity) >= quantity
          },
          limited: (date) => {
            const s = date.toISOString().slice(0, 10)
            const a = availability.get(s) ?? quantity
            return a > 0 && a < quantity
          },
          unavailable: (date) => {
            const s = date.toISOString().slice(0, 10)
            return (availability.get(s) ?? quantity) === 0
          },
        }}
        modifiersClassNames={{
          available:   'bg-primary/10',
          limited:     'bg-warning/20',
          unavailable: 'bg-destructive/20 line-through',
        }}
        numberOfMonths={2}
        className="rounded-md border"
      />
    )}
  </div>
 )
}
```

Лістинг Б.3 API Route для перевірки доступності

```
// app/api/availability/route.ts
```

```
import { NextRequest, NextResponse } from 'next/server'
import { checkAvailability } from '@/lib/queries'

export async function GET(request: NextRequest) {
  const p           = request.nextUrl.searchParams
  const equipmentId = p.get('equipmentId')
  const startDate   = p.get('startDate')
  const endDate     = p.get('endDate')

  if (!equipmentId || !startDate || !endDate) {
    return NextResponse.json(
      { error: "Відсутні обов'язкові параметри" },
      { status: 400 },
    )
  }

  const dateRegex = /^\d{4}-\d{2}-\d{2}$/
  if (!dateRegex.test(startDate) || !dateRegex.test(endDate)) {
    return NextResponse.json(
      { error: 'Невірний формат дати. Використовуйте YYYY-MM-DD' },
      { status: 400 },
    )
  }

  if (new Date(`${startDate}T00:00:00Z`) > new Date(`${endDate}T00:00:00Z`)) {
    return NextResponse.json(
      { error: 'Дата початку не може бути пізніше дати завершення' },
      { status: 400 },
    )
```

```
  }

  try {
    const availability = checkAvailability(equipmentId, startDate, endDate)
    return NextResponse.json({
      ...availability,
      checkedAt: new Date().toISOString(),
    })
  } catch {
    return NextResponse.json(
      { error: 'Помилка перевірки доступності' },
      { status: 500 },
    )
  }
}


//
================================================================
// Лістинг Б.4 SSE Route для оновлень доступності у реальному часі
// app/api/availability/stream/route.ts
//
================================================================

import { NextRequest } from 'next/server'
import { checkAvailability } from '@/lib/queries'

export const dynamic = 'force-dynamic'

export async function GET(request: NextRequest) {
```

```
const equipmentId = request.nextUrl.searchParams.get('equipmentId')

if (!equipmentId) {
  return new Response('equipmentId required', { status: 400 })
}

const encoder = new TextEncoder()

const stream = new ReadableStream({
  start(controller) {
    // Надсилаємо поточний стан одразу при підключенні
    pushAvailability(controller, encoder, equipmentId)

    // Оновлення кожні 5 секунд
    const interval = setInterval(() => {
      pushAvailability(controller, encoder, equipmentId)
    }, 5_000)

    // Закриття при від'єднанні клієнта
    request.signal.addEventListener('abort', () => {
      clearInterval(interval)
      controller.close()
    })
  },
})

return new Response(stream, {
  headers: {
    'Content-Type':      'text/event-stream',
    'Cache-Control':     'no-cache',
```

```
      'Connection':        'keep-alive',
      'X-Accel-Buffering': 'no',
    },
  })
}

function pushAvailability(
  controller:  ReadableStreamDefaultController,
  encoder:     TextEncoder,
  equipmentId: string,
): void {
  try {
    const today  = new Date().toISOString().slice(0, 10)
    const future = (() => {
      const d = new Date()
      d.setUTCDate(d.getUTCDate() + 60)
      return d.toISOString().slice(0, 10)
    })()

    const { bookedDates } = checkAvailability(equipmentId, today, future)

    for (const d of bookedDates) {
      controller.enqueue(encoder.encode(`data: ${JSON.stringify(d)}\n\n`))
    }
  } catch (err) {
    console.error('SSE помилка:', err)
  }
}
```

ДОДАТОК В

Інтерфейс розробленого вебзастосунку

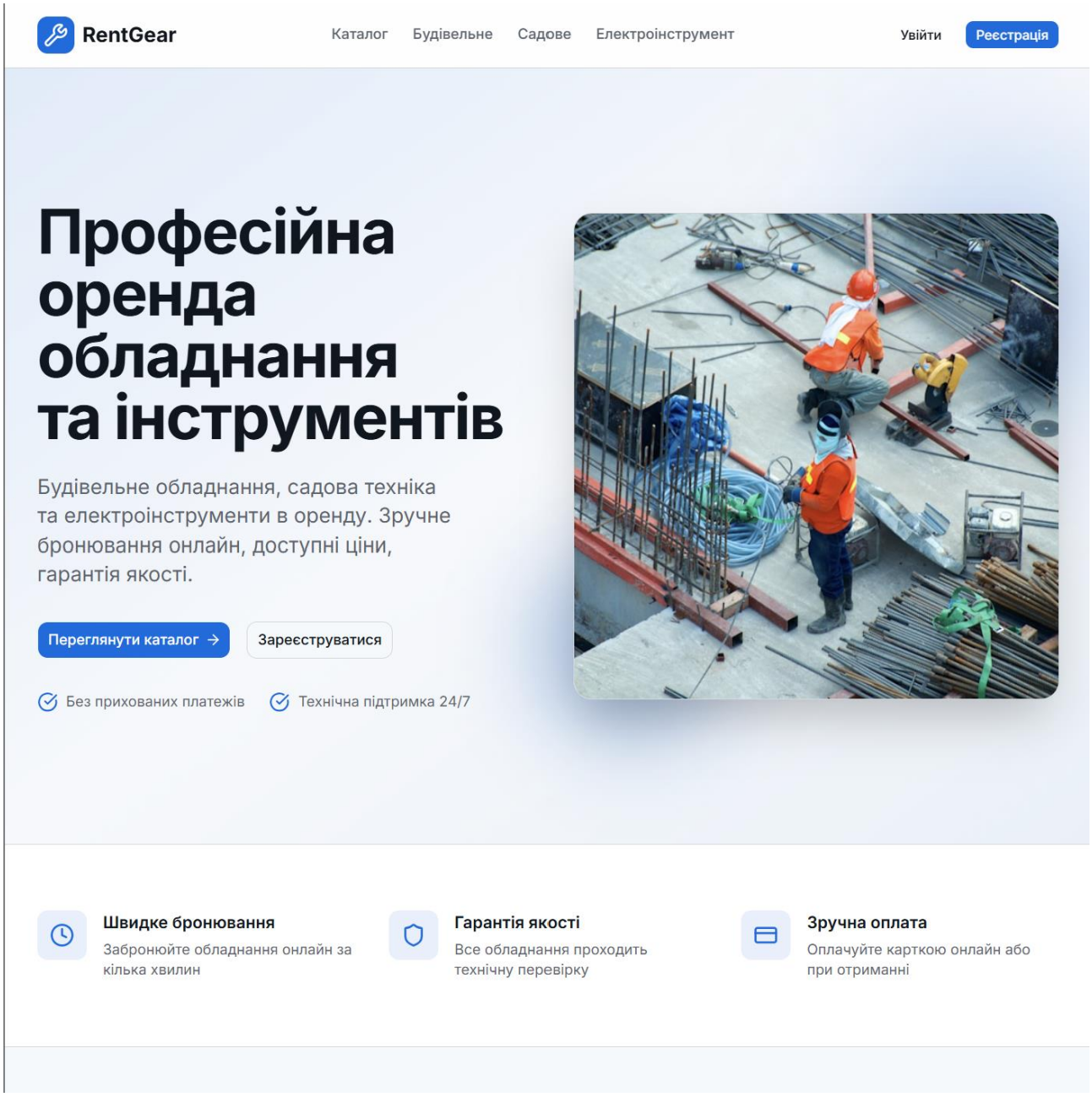


Рисунок В.1 – Головна сторінка

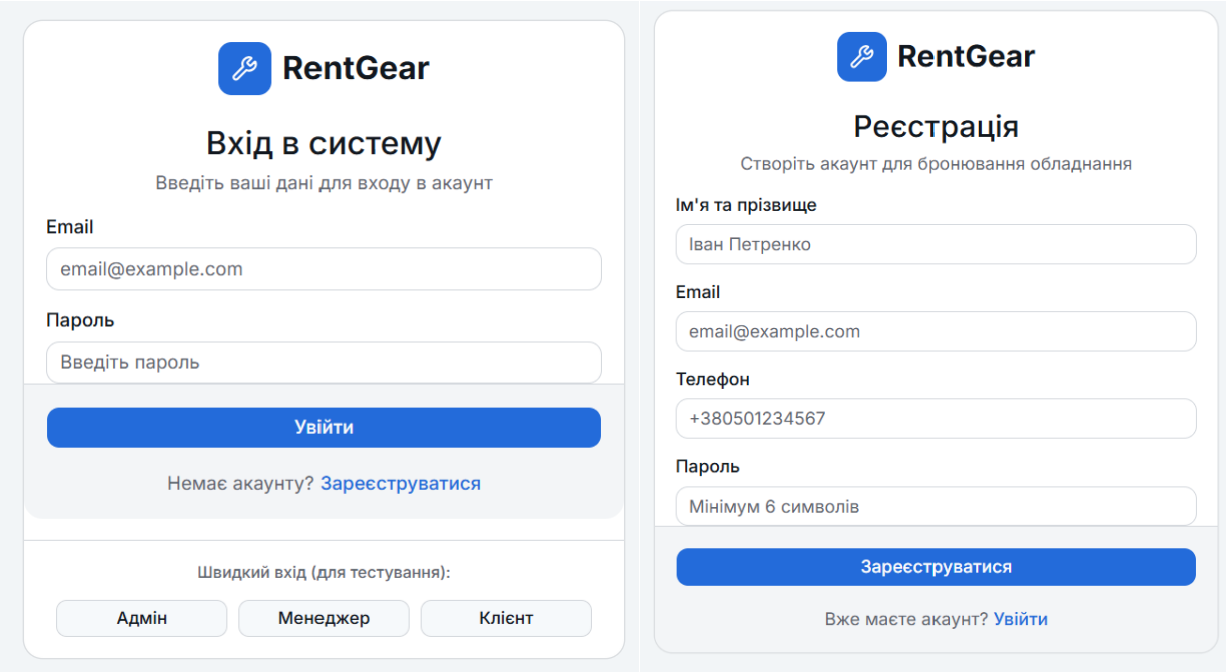


Рисунок В.2 – Сторінка входу та рестрації

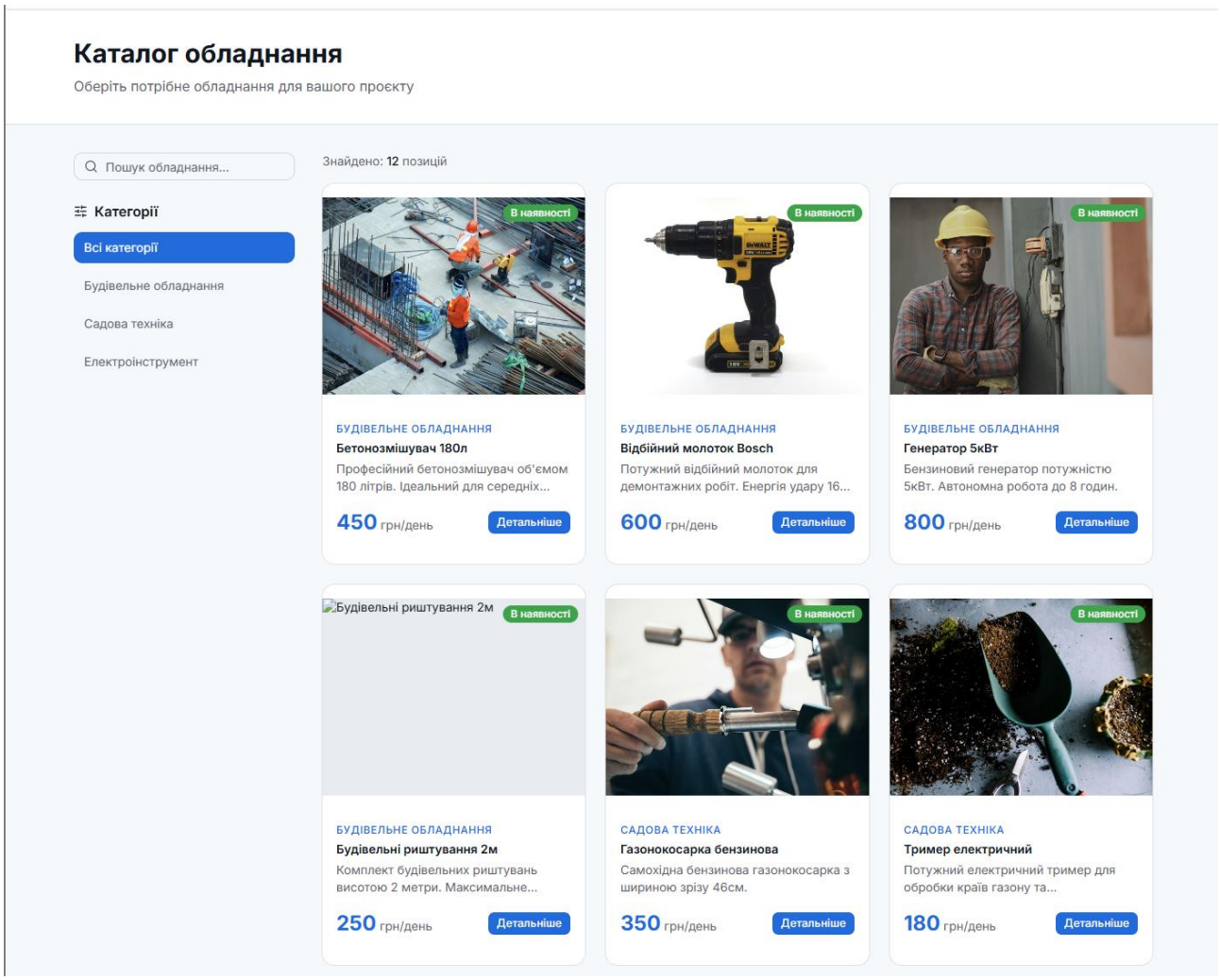


Рисунок В.3 – Каталог товарів

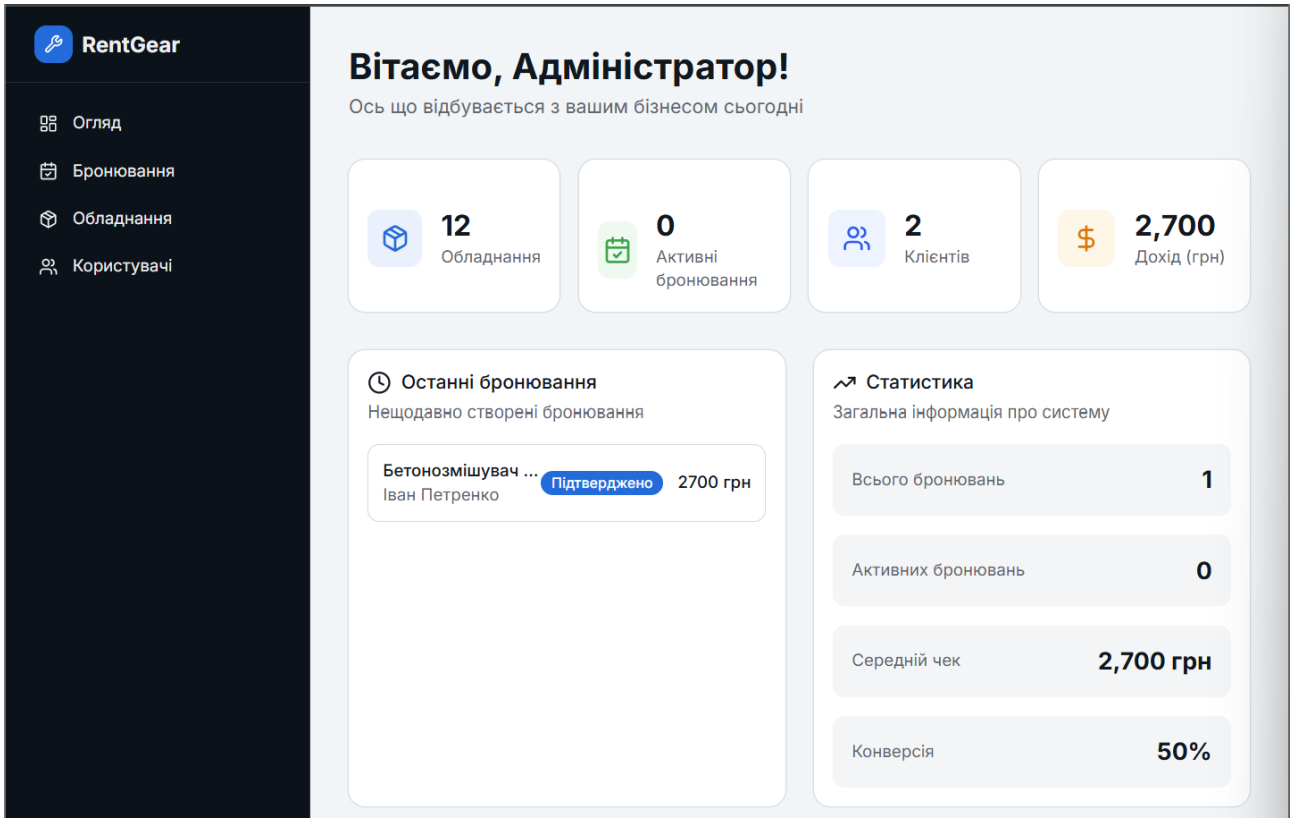


Рисунок В.4 – Дашборд адміністратора

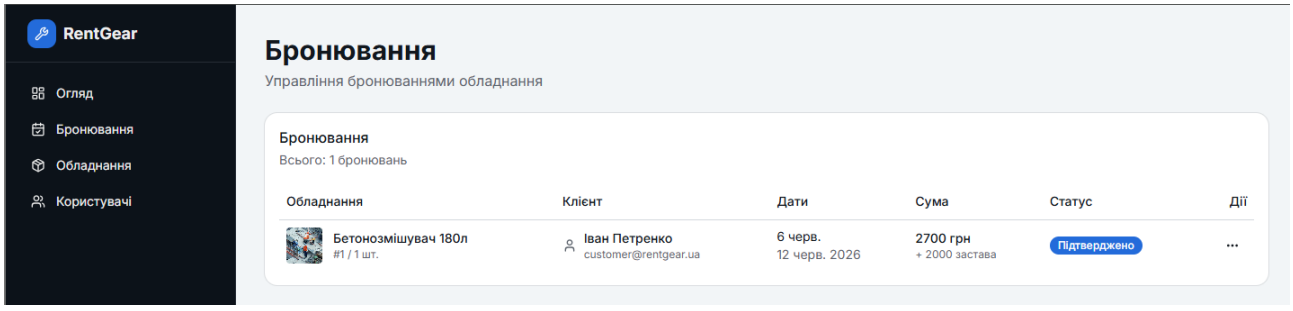


Рисунок В.5 – Сторінка бронювання в адміністратора / менеджера

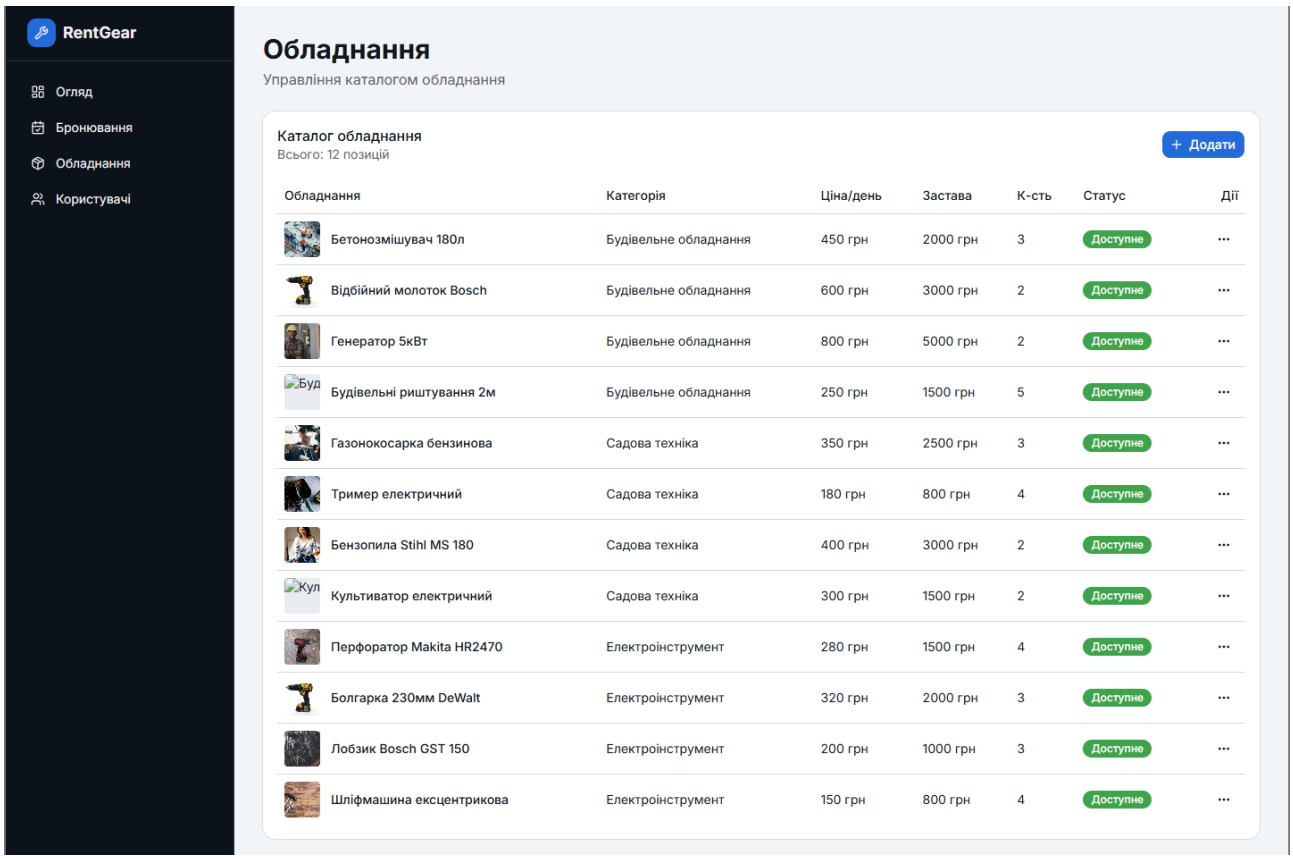


Рисунок В.6 – Сторінка обладнення в адміністратора

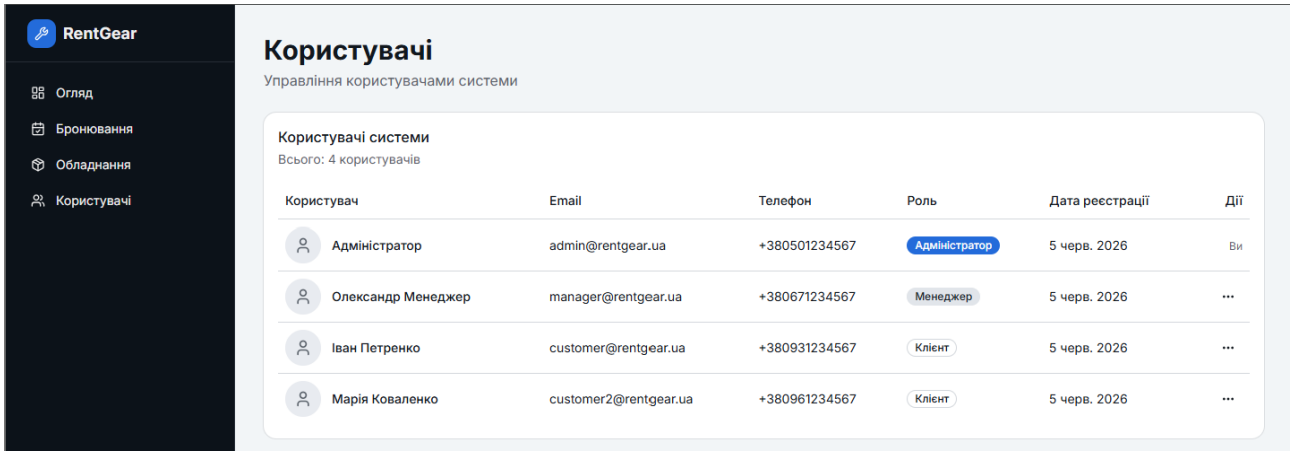


Рисунок В.7 – Сторінка користувачі в адміністратора

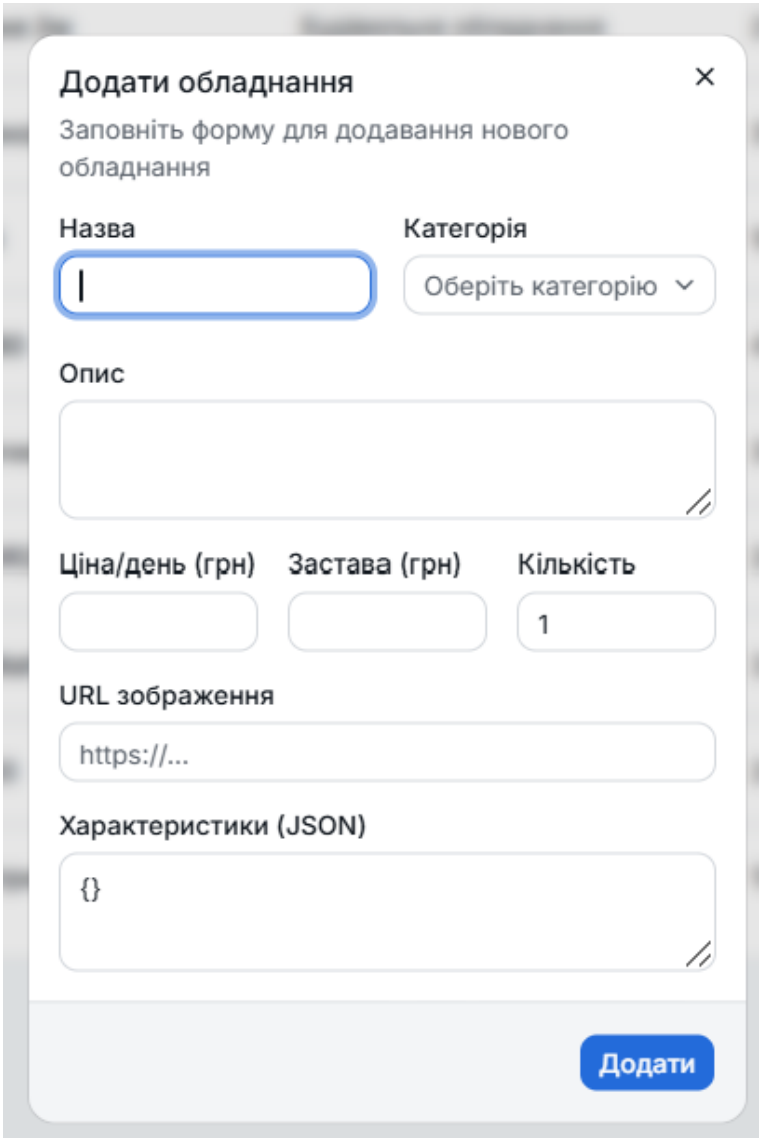


Рисунок В.8 – Вікно додавання товару