Вінницький національний технічний університет
(повне найменування вищого навчального закладу)
Факультет інформаційних технологій та комп'ютерної інженерії
(повне найменування інституту)
Кафедра обчислювальної техніки
(повна назва кафедри)

Бакалаврська кваліфікаційна робота на тему:

«ВЕБЗАСТОСУНОК ДЛЯ БРОНЮВАННЯ РОБОЧИХ МІСЦЬ У КОВОРКІНГАХ»

Виконав: студент 2 курсу, групи 1КІ-23мс
спеціальності
123 Комп'ютерна інженерія
(шифр і назва напряму підготовки, спеціальності)
освітня програма «Комп'ютерна інженерія»
Шевчук В.В.
(прізвище та ініціали)

Керівник: д. ф., ст. викл. каф. ОТ
Обертюх М.Р.
(прізвище та ініціали)

Рецензент:
Ракитянська Г. Б.
(прізвище та ініціали)

Допущено до захисту
Завідувач кафедри ОТ
д.т.н., проф. Азаров О. Д.
«___» червня 2025 р.

Вінниця ВНТУ 2025

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки
Освітньо—кваліфікаційний рівень бакалвр
Галузь знань — 12 Інформаційні технології
Спеціальність — 123 Комп'ютерна інженерія
Освітня програма — Комп'ютерна інженерія

ЗАТВЕРДЖУЮ
Завідувач кафедри ОТ
д.т.н.проф._____Азаров О. Д.
«21» березня 2025 року

З А В Д А Н Н Я

НА БАКАЛАВРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Шевчуку Владиславу Вікторовичу

1 Тема роботи: «Вебзастосунок для бронювання робочих місць у коворкінгах», керівник роботи Обертюх М.Р. д.,ф. ст. викл. кафедри ОТ затверджені наказом ВНТУ від «20» березня 2025 року № 97.

2 Термін подання студентом роботи 13.05.2025

3 Вихідні дані до роботи: особисті дані клієнтів; список бронювань; аналітика по коворкінгам; сучасні технології розробки клієнтської та серверної частин, середовище розробки — Visual Studio Code.

4 Зміст розрахунково-пояснювальної записки: вступ, аналіз аналогів вебзастосунків для бронювання робочих місць у коворкінгах, проєктування вебдодатку для бронювання робочих місць у коворкінгах, програмна реалізація вебдодатку для бронювання робочих місць у коворкінгах, тестування вебдодатку для бронювання робочих місць у коворкінгах.

5 Перелік графічного матеріалу: структурна схема архітектурних рівнів системи, структурна схема клієнтської частини, структурна схема серверної частини. Перелік ілюстраційного матеріалу: скріншоти графічного інтерфейсу системи.

6 Консультанти розділів роботи приведені в таблиці 1.

Таблиця 1 — Консультанти роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1 - 4	д. ф., ст. викл. кафедри ОТ Обертюх М.Р.	21.03.25	13.05.25
Нормоконтроль	ас. каф. ОТ Швець С. І.	14.05.25	30.05.25

7 Дата видачі завдання 21.03.2025

8 Календарний план приведений в таблиці 2.

Таблиця 2 — Календарний план

№ з/п	Назва етапів дипломної роботи	Строк виконання	Підпис
1.	Постановка задачі роботи	21.03.25	
2.	Аналіз аналогів вебдодатків для бронювання робочих місць у коворкінгах	21.03.25—30.03.25	
3.	Проєктування вебдодатку для бронювання робочих місць у коворкінгах	21.03.25—30.03.25	
4.	Програмна реалізація онлайн системи	31.03.25—17.04.25	
5.	Тестування онлайн системи	18.04.25—27.04.25	
6.	Оформлення пояснювальної записки та ілюстративного матеріалу	28.04.25—11.05.25	
7.	Перевірка якості виконання бакалаврської кваліфікаційної роботи та усунення недоліків	13.05.2025	

Студент Владислав ШЕВЧУК

Керівник роботи д. ф., ст. викл. Максим ОБЕРТЮХ

АНОТАЦІЯ

УДК 004.42

Шевчук В. В. Вебдодаток для бронювання робочих місць у коворкінгах. Бакалаврська кваліфікаційна робота (проєкт) зі спеціальності 123 «Комп'ютерна інженерія», освітня програма «Комп'ютерна інженерія». Вінниця: ВНТУ, 2022. 83 с.

На укр. мові. Бібліогр.: 18 назв; рис.: 16; табл. 3.

У бакалаврській кваліфікаційній роботі розроблено онлайн-систему для бронювання робочих місць у коворкінгах, яка дозволяє автоматизувати процес резервування, покращити управління простором та забезпечити зручну взаємодію між користувачами та адміністраторами. У роботі проведено порівняльний аналіз існуючих рішень у сфері онлайн-бронювання, а також досліджено сучасні технології розробки вебзастосунків. Спроєктовано архітектуру системи та реалізовано вебінтерфейс, який надає функціональні можливості для бронювання місць, перегляду доступних опцій, ведення історії бронювань, фільтрації за параметрами, а також управління користувацькими даними.

Ключові слова: комп'ютерна система, React, lowDB, Node.js облік коворкінгів, бронювання, аналітика місць, коворкінг.

ABSTRACT

Shevchuk V. V. Web application for booking workplaces in coworking spaces. Bachelor's qualification work (project) in specialty 123 "Computer Engineering", educational program "Computer Engineering". Vinnytsia: VNTU, 2022. 83 p.

In Ukrainian. Bibliography: 18 titles; fig.: 16; table. 3.

In the bachelor's qualification work, an online system for booking workplaces in coworking spaces has been developed, which allows you to automate the reservation process, improve space management and ensure convenient interaction between users and administrators. The work provides a comparative analysis of existing solutions in the field of online booking, and also explores modern technologies for developing web applications. The system architecture was designed and a web interface was implemented, which provides functionality for booking seats, viewing available options, maintaining reservation history, filtering by parameters, and managing user data.

Keywords: computer system, React, lowDB, Node.js, coworking accounting, booking, seat analytics, coworking.

ЗМІСТ

ВСТУП

У сучасному світі, що швидко змінюється під впливом розвитку інформаційних технологій та трансформації форм зайнятості, все більше людей обирають гнучкі варіанти роботи. Одним із таких варіантів є коворкінги — спільні робочі простори, які надають змогу працювати в зручному середовищі поза межами традиційного офісу або дому. Застарілі методи бронювання — через дзвінки, електронну пошту або особисте відвідування — часто є малоефективними, займають багато часу і можуть спричиняти помилки.

Актуальність даної роботи обумовлена стрімким зростанням популярності коворкінг-просторів, що створює нагальну потребу у розробці зручних і ефективних засобів для організації процесу бронювання робочих місць у таких просторах.

У цьому контексті розробка автоматизованого вебзастосунку для онлайн-бронювання робочих місць постає як важливий крок у цифровій трансформації даної сфери. Такий застосунок дозволить покращити якість обслуговування клієнтів, оптимізувати використання ресурсів та загалом удосконалити функціонування коворкінгів.

Основна **мета** роботи — створити вебзастосунок, який дасть змогу користувачам швидко та зручно бронювати робочі місця в коворкінгах, а адміністраторам — ефективно керувати цими бронюваннями, аналізувати завантаженість та підвищувати якість сервісу.

Для досягнення поставленої мети необхідно реалізувати такі **завдання**:

— провести аналіз існуючих рішень для бронювання місць у коворкінгах;

— визначити вимоги до функціональності вебзастосунку з урахуванням потреб користувачів та адміністраторів;

— спроєктувати архітектуру системи, що забезпечить її надійність та масштабованість;

— реалізувати користувацький інтерфейс з можливістю перегляду вільних місць, оформлення та управління бронюваннями;

— розробити адміністративну панель для керування бронюваннями, календарем доступності, користувачами та статистичними даними;

— здійснити тестування створеної системи та її оптимізацію.

Об’єкт дослідження — вебзастосунок як інструмент автоматизації процесу бронювання робочих місць у коворкінг-просторах. Це охоплює розробку, проектування та впровадження вебзастосунку, що виконує функцію цифрового посередника між відвідувачами коворкінгу та адміністрацією, забезпечуючи ефективну взаємодію між усіма учасниками процесу.

Предмет дослідження — це вебзастосунок, який виступає засобом автоматизації процесу бронювання робочих місць у коворкінг-просторах. Основна увага приділяється трьом ключовим аспектам: зручності користування (user experience), ефективному керюванню та технічній реалізації функціональних можливостей системи.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Сучасні вимоги до вебзастосунків для бронювання робочих місць у коворкінгах

Вебзастосунок для бронювання робочих місць у коворкінгах має відповідати сучасним стандартам як з технічної точки зору, так і з погляду зручності використання. Стрімке зростання популярності коворкінг-просторів, пов’язане з розвитком гнучких форм зайнятості та зростаючими потребами мобільних працівників, створює попит на інноваційні цифрові рішення, що дозволяють спростити управління цими просторами. У цьому контексті вебзастосунок виступає як ключовий інструмент автоматизації основних бізнес-процесів, спрямований на покращення взаємодії з клієнтами, зменшення адміністративного навантаження та підвищення ефективності використання ресурсів [1].

Одним із критично важливих аспектів для успішного функціонування такого застосунку є забезпечення його високої продуктивності та здатності до масштабування. У пікові години — наприклад, зранку чи ввечері в будні дні — кількість користувачів системи може значно зрости, що вимагає від програмного забезпечення здатності безперебійно обробляти велику кількість запитів. Це досягається завдяки впровадженню відповідних технологічних рішень, таких як масштабування інфраструктури, балансування навантаження на сервери, використання хмарних сервісів і розподілених баз даних. Забезпечення стабільної роботи системи незалежно від навантаження є важливою передумовою позитивного користувацького досвіду.

Не менш важливим аспектом є забезпечення зручності та інтуїтивної зрозумілості користувацького інтерфейсу. Враховуючи, що значна частина користувачів взаємодіє із системою з мобільних пристроїв, вебінтерфейс повинен бути гнучким і коректно відображатися на екранах різних розмірів — від смартфонів до десктопів. Адаптивність інтерфейсу дозволяє забезпечити комфортне використання сервісу незалежно від типу пристрою, що є необхідною умовою сучасної веброзробки. Користувач повинен мати змогу без труднощів

знайти вільне робоче місце, ознайомитися з розкладом, здійснити бронювання за кілька кліків, а також легко орієнтуватися у структурі застосунку завдяки логічній навігації та зрозумілому розташуванню елементів [2].

Користувачі дедалі частіше очікують персоналізованого підходу, що виражається у здатності системи враховувати їхні індивідуальні вподобання та історію попередніх дій. Тому важливо впроваджувати функціонал, що дозволяє зберігати історію бронювань, пропонувати найбільш релевантні місця на основі минулих виборів, а також реалізовувати програми лояльності для постійних клієнтів [3].

Окремої уваги потребує безпека системи, оскільки вебзастосунок працює з конфіденційними даними користувачів, включно з особистою інформацією та платіжними реквізитами. Для забезпечення належного рівня захисту необхідно реалізувати шифрування трафіку через протокол HTTPS, впровадити захист від типових вебзагроз, таких як SQL-ін’єкції, міжсайтові скрипти (XSS), міжсайтові запити (CSRF) тощо. Крім того, вебзастосунок повинен відповідати актуальним міжнародним стандартам захисту персональних даних, зокрема таким як GDPR, що регулює збір, зберігання та обробку даних громадян Європейського Союзу. Також важливо впроваджувати надійні механізми автентифікації користувачів, зокрема з підтримкою двофакторної перевірки особи.

Ще одним напрямом, який суттєво підвищує цінність застосунку, є інтеграція з зовнішніми цифровими сервісами. Наприклад, підключення до платіжних платформ, таких як Stripe або PayPal, дозволяє користувачам швидко здійснювати оплату без необхідності залишати межі вебзастосунку. Синхронізація з календарями (Google Calendar, Microsoft Outlook) спрощує організацію часу та нагадування про бронювання. Водночас підключення до сервісів аналітики, таких як Google Analytics, дає змогу адміністраторам отримувати цінну інформацію про поведінку користувачів, конверсію та ефективність окремих функцій.

Функції аналітики також відіграють важливу роль у внутрішньому управлінні простором. Керівницька частина системи повинна містити модулі для візуалізації статистики завантаженості, оцінки популярності певних зон, аналізу

пікових періодів активності, а також формування звітів за обраними параметрами. Це дозволяє приймати обґрунтовані рішення щодо вдосконалення сервісу, зміни цінової політики або оптимізації планування простору.

Нарешті, важливою вимогою до вебзастосунку є його гнучкість та розширюваність. У зв’язку з постійним розвитком цифрових технологій та змінністю потреб користувачів, застосунок має дозволяти швидке впровадження нових функцій або зміну вже існуючих без необхідності кардинального переписування коду. Для цього доцільно використовувати сучасні архітектурні підходи, зокрема мікросервісну модель або headless-архітектуру, які дозволяють масштабувати окремі модулі системи незалежно один від одного.

Таким чином, вебзастосунок для бронювання робочих місць у коворкінгах є складною, багатокомпонентною інформаційною системою, яка має відповідати високим вимогам до зручності, надійності, безпеки, адаптивності та продуктивності. Його створення передбачає не лише якісну технічну реалізацію, а й глибоке розуміння бізнесових потреб, пов’язаних із управлінням просторами спільної роботи, з метою забезпечення максимальної ефективності та конкурентоспроможності на ринку.

1.2 Порівняльний аналіз аналогів бронювань місць у коворкінгах

Для ефективної розробки вебзастосунку з бронювання робочих місць у коворкінгах необхідно врахувати досвід існуючих рішень. Проведення порівняльного аналізу аналогічних систем дозволяє виявити їхні сильні та слабкі сторони, визначити поширені функціональні підходи, а також сформувати уявлення про очікування користувачів. Такий аналіз слугує основою для обґрунтованого проєктування власного застосунку, орієнтованого на ефективність, зручність та інноваційність.

Одним із прикладів схожих систем є міжнародна онлайн-платформа AllBooked (skedda.com) — сервіс, що надає можливість знаходити та бронювати коворкінги в різних країнах світу (рис. 1.1). Цей ресурс насамперед орієнтований на фрилансерів, цифрових кочівників і віддалених співробітників, які потребують

комфортного робочого простору під час перебування в іншому місті чи за кордоном [4].

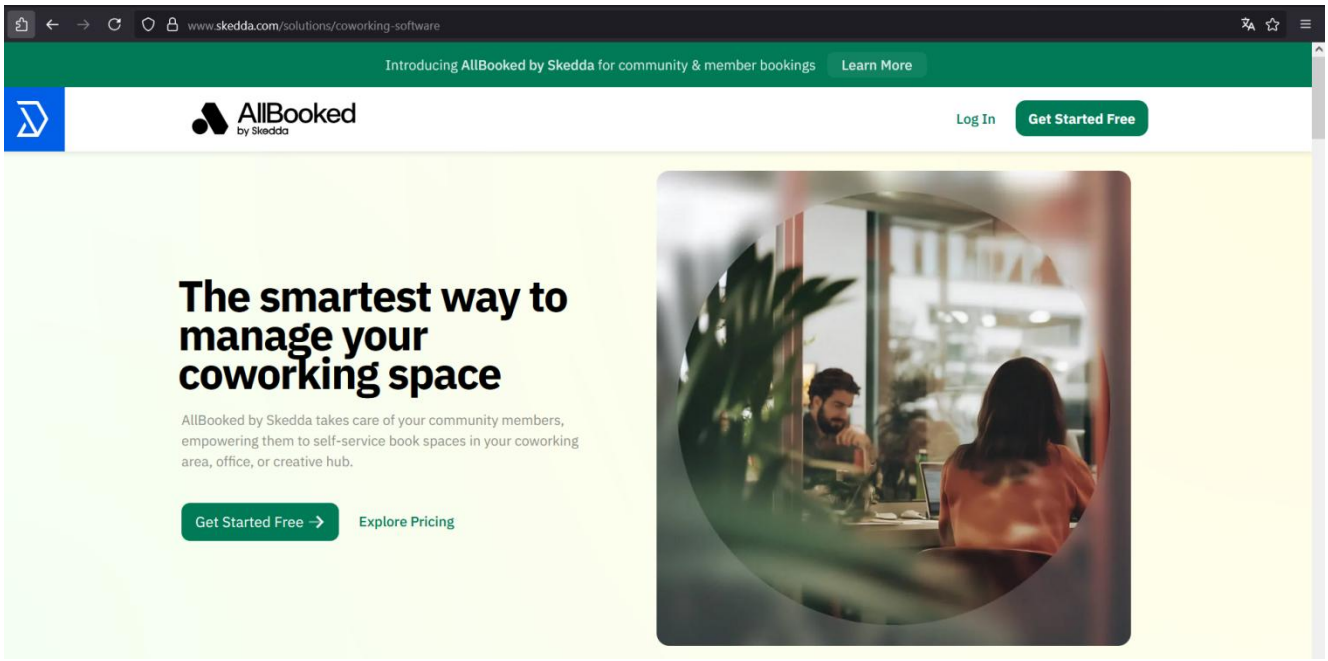


Рисунок 1.1 — Головна сторінка платформи AllBooked

Платформа має низку ключових особливостей, які роблять її зручною для користувачів. Вона пропонує інтерактивну карту, що дозволяє шукати коворкінги за країнами та містами, а також використовувати систему фільтрації за параметрами — типом простору (відкритий офіс, приватний кабінет, конференц-зал), наявністю Wi-Fi, кухні, принтера тощо. Кожен об’єкт має рейтинг і відгуки, що допомагає користувачам приймати обґрунтовані рішення. Також доступне онлайн-бронювання: користувачі можуть зарезервувати місце безпосередньо через сайт або перейти на зовнішній ресурс для завершення процедури. Крім того, інтерфейс підтримує декілька мов, що робить сервіс зручним для міжнародної аудиторії.

Серед недоліків платформи — не всі локації підтримують миттєве бронювання, оскільки частина з них просто перенаправляє користувача на сторонні сайти; також спостерігається обмежена підтримка українських міст, а мобільного застосунку наразі немає.

Ще один приклад — вебсервіс Deskpass (deskpass.com), який працює за

моделлю підписки і дає доступ до сотень коворкінгів у містах США, Канади та Австралії (рис. 1.2). Це сучасна платформа, орієнтована на гнучкий стиль роботи та діджитал-номадів [5].

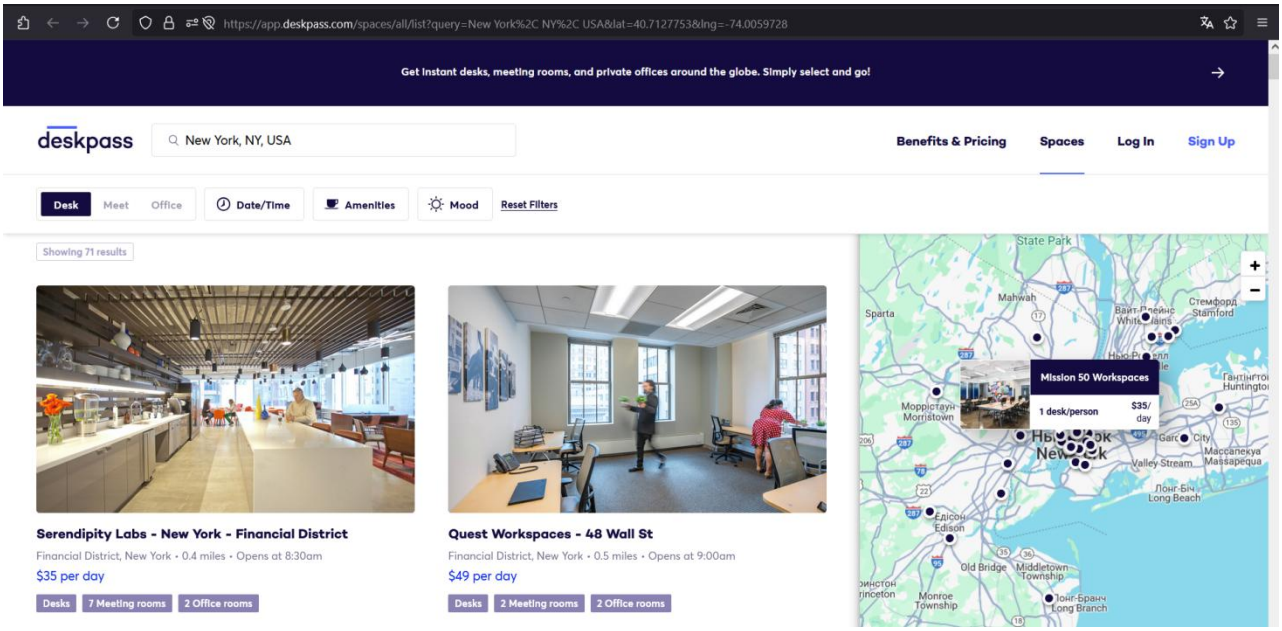


Рисунок 1.2 — Головна сторінка платформи Deskpass

Сервіс вирізняється кількома ключовими особливостями. Він працює за підписною моделлю: користувачі оформлюють абонемент, який дозволяє відвідувати різні коворкінги в межах обраного тарифного плану. Система підтримує повністю автоматизоване бронювання — місця, кімнати або конференц-зали можна зарезервувати онлайн як через вебсайт, так і через мобільний застосунок. Мобільний додаток забезпечує зручний доступ до функціоналу, дозволяючи швидко знаходити найближчі доступні простори, перевіряти наявність і здійснювати бронювання. Крім того, сервіс пропонує корпоративні рішення: компанії можуть придбати підписки для працівників, які працюють віддалено або в гібридному форматі.

До основних недоліків можна віднести те, що платформа не охоплює Україну та країни Східної Європи. Також система потребує регулярної оплати за підписку, що може бути невигідно для користувачів, які рідко користуються послугами. Деякі функції доступні виключно в межах платних тарифів.

LiquidSpace — це міжнародний сервіс, який надає можливість бронювання коворкінгів, переговорних кімнат та офісів у режимі реального часу (рис. 1.3). Платформа орієнтована на корпоративних клієнтів, стартапи, віддалених працівників і мандрівників [6].

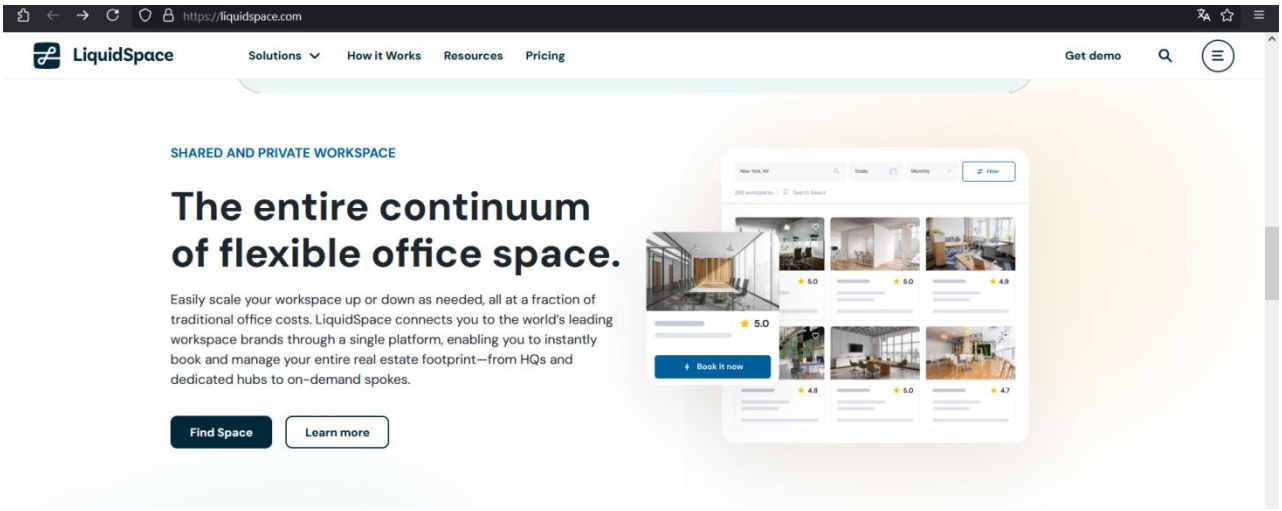


Рисунок 1.3 — Головна сторінка платформи LiquidSpace

Платформа має низку важливих функцій. Зокрема, вона дозволяє здійснювати миттєве бронювання робочих місць у режимі онлайн лише одним кліком. У її розпорядженні — широка база перевірених коворкінг-просторів у різних країнах світу. Сервіс також підтримує інтеграцію з корпоративними системами бронювання, що зручно для бізнес-користувачів. Доступні різні варіанти оренди: погодинна, на день або довгострокова. Крім того, існує зручний мобільний додаток для пристроїв на Android та iOS.

Серед недоліків варто зазначити, що платформа переважно орієнтована на ринки США, Канади та Західної Європи. Вона не має підтримки української мови, а вартість бронювання зазвичай вища, ніж у місцевих аналогів.

Усі існуючі системи для бронювання місць у коворкінгах демонструють загальну тенденцію до автоматизації процесів, персоналізації взаємодії з користувачем та гнучкого масштабування функціоналу. Платформи забезпечують зручний пошук, бронювання та оплату через інтерфейс, а також надають можливості для адаптації під потреби користувачів.

Таким чином, аналіз функціональності та можливостей існуючих онлайн-систем для бронювання коворкінгів дозволяє зробити висновки про ефективність використаних компонентів, виявити слабкі місця та сформулювати рекомендації для побудови більш вдосконаленої системи.

2 ВИБІР ТА ОБҐРУНТУВАННЯ ІНСТРУМЕНТІВ ДЛЯ РОЗРОБКИ ВЕБЗАСТОСУНКУ ДЛЯ БРОНЮВАННЯ РОБОЧИХ МІСЦЬ

Для створення надійного та функціонального вебзастосунку важливо обрати відповідні інструменти розробки, які забезпечать зручність роботи, масштабованість, безпеку та легкість подальшого розширення системи. У цьому розділі буде здійснено вибір технологічного стеку та обґрунтовано доцільність використання конкретних мов програмування, фреймворків, баз даних та допоміжних сервісів відповідно до вимог проєкту.

2.1 Архітектура та структура веб-сайту як складова інформаційної системи онлайн-магазину комп’ютерної техніки

Вебзастосунок для бронювання робочих місць у коворкінгах є не лише інструментом для управління бронюваннями, а й невід’ємною частиною масштабної інформаційної системи. Ця система охоплює всі ключові аспекти управління коворкінгом — від візуального відображення доступних місць до інтеграції з платіжними платформами, календарями та аналітичними сервісами.

Одним із головних елементів системи виступає інтерфейс користувача. Він є тією частиною вебзастосунку, з якою безпосередньо взаємодіє користувач. До його функцій належить система фільтрів для пошуку місць за певними критеріями, такими як наявність вікна, розетки чи доступ до конференц-залів. Крім того, інтерфейс має бути адаптивним і забезпечувати зручне користування як зі смартфонів і планшетів, так і з настільних комп’ютерів.

Серверна логіка та бази даних формують бекенд-середовище системи. Серверна частина відповідає за обробку запитів користувачів, взаємодію з базами даних і виконання логіки бізнес-процесів. Для зберігання інформації про бронювання, користувачів і ресурси застосовується реляційна база даних, зокрема LowDB.

Ще одним важливим елементом є інтеграційний модуль. Він забезпечує зв’язок вебзастосунку з зовнішніми сервісами. Наприклад, підключення платіжних шлюзів (таких як Stripe або PayPal) дозволяє здійснювати оплату

безпосередньо через застосунок. Синхронізація з календарями (Google Calendar, Outlook) дає змогу додавати бронювання до особистих розкладів користувачів, а підтримка CRM-систем сприяє ефективному управлінню клієнтськими даними та маркетинговими активностями.

Існують також різні типи структур, які застосовуються у вебзастосунках для коворкінгів. Деревоподібна структура є ієрархічною і дозволяє легко орієнтуватися між коворкінг-зонами, приміщеннями та доступними місцями. Компонентна архітектура базується на модульному підході, який включає окремі функціональні частини, зокрема модулі бронювання, пошуку, особистого кабінету користувача, а також аналітики.

Функціональні зони застосунку розділяються відповідно до своїх задач. Інформаційна зона містить перелік доступних коворкінгів і довідкові матеріали для нових користувачів. Керівницька зона надає інструменти для адміністрування простору, включно зі створенням нових робочих місць. Аналітична зона збирає дані про завантаженість, активність користувачів і пікові години відвідуваності, що дає змогу приймати обґрунтовані управлінські рішення.

Окрему увагу слід приділяти доступності вебзастосунку. Він повинен забезпечувати швидкий доступ із будь-яких пристроїв, а також мати інструменти для користувачів із вадами зору, наприклад, контрастний режим або текстові альтернативи зображенням.

Добре продумана структура вебзастосунку сприяє покращенню користувацького досвіду, полегшує технічну підтримку та масштабування системи, а також дозволяє без ускладнень інтегрувати нові функції. Загалом, архітектура та структура є основою для ефективної реалізації системи бронювання, що відповідає сучасним вимогам щодо функціональності, зручності та безпеки.

2.2 Функціональні можливості вебзастосунку для бронювання робочих місць у коворкінгах

Вебзастосунок для бронювання робочих місць у коворкінгах виконує

важливу роль у автоматизації ключових процесів, пов’язаних із вибором, бронюванням і подальшим управлінням робочими просторами. Це суттєво підвищує ефективність роботи не лише самих користувачів, які шукають комфортне та відповідне їхнім потребам місце, але й керівників, які контролюють зайнятість приміщень і ресурси. Основний функціонал вебзастосунку спроектований таким чином, щоб забезпечити максимально зручний, інтерактивний та безпечний досвід для всіх учасників процесу.

Однією з центральних функцій є управління робочими місцями, що реалізується через можливість користувачів переглядати актуальний перелік доступних місць. При цьому передбачена система фільтрів, яка допомагає швидко знаходити робочі зони за різними параметрами — наприклад, типом робочого простору, наявністю додаткового обладнання, ціною чи іншими важливими характеристиками. Користувач отримує детальну інформацію про кожне місце: доступність розеток, рівень освітлення, розташування біля вікна чи поруч із певними зручностями, що дозволяє індивідуалізувати вибір під особисті вподобання. Важливою складовою є інтерактивна карта приміщення, яка надає візуальне відображення вільних місць у реальному часі, що значно спрощує процес пошуку та прийняття рішення.

Процес бронювання організований послідовно і зрозуміло: спочатку користувач обирає робоче місце, потім вказує бажану дату і час, після чого підтверджує бронювання та, за потреби, здійснює оплату. Вебзастосунок автоматично перевіряє наявність обраного місця, резервує його для клієнта і відправляє підтвердження бронювання у вигляді повідомлення на email або в месенджери. У разі необхідності користувач має змогу легко скасувати чи перенести бронювання через особистий кабінет, що додає гнучкості та зручності в користуванні сервісом.

Особистий кабінет є важливим елементом інтерфейсу користувача, де зберігається історія всіх бронювань, а також збережені вподобання і персональні налаштування. Це дозволяє клієнтам не тільки переглядати попередні дії, але й швидко повторювати попередні бронювання, що значно спрощує та пришвидшує

процес повторного замовлення. Користувачі можуть також коригувати свої дані та параметри, що підвищує рівень персоналізації і комфорту.

З боку керівників функціональність вебзастосунку реалізується через керівницьку панель, яка надає можливості для ефективного управління коворкінг-простором. Власники коворкінгів можуть додавати нові робочі місця, змінювати їх статус (наприклад, доступність або технічний стан), переглядати статистику використання і обробляти запити клієнтів. Крім того, панель підтримує інструменти для налаштування цін, запуску акцій і спеціальних пропозицій, що допомагає оптимізувати комерційні процеси.

Інтеграція із зовнішніми сервісами є важливою складовою для розширення можливостей застосунку. Зокрема, підключення платіжних систем, таких як PayPal чи Stripe, забезпечує швидку, безпечну і зручну оплату безпосередньо в системі. Це підвищує комфорт користувачів і сприяє зростанню рівня довіри до сервісу.

Безпека персональних даних і фінансових транзакцій — один із пріоритетів вебзастосунку. Для цього застосовуються сучасні методи шифрування інформації, а також дотримуються міжнародні стандарти безпеки, зокрема GDPR, що гарантує захист конфіденційної інформації користувачів і мінімізує ризики витоку даних.

Аналітичні можливості системи дають змогу керівникам отримувати детальну інформацію про популярність робочих місць, час їхнього використання та інші ключові показники. На основі цих даних адміністратори можуть приймати обґрунтовані рішення щодо оптимізації розміщення робочих зон, коригування цінової політики та розробки маркетингових стратегій, що підвищують ефективність роботи коворкінгу.

Важливим елементом є також система персоналізації. Алгоритми аналізують поведінку користувачів, їхню історію бронювань і вподобання, щоб пропонувати найбільш релевантні варіанти робочих місць. Такий підхід підвищує рівень задоволеності клієнтів і створює додаткову цінність для сервісу.

Таким чином, вебзастосунок для бронювання робочих місць у коворкінгах є

комплексним інструментом, який поєднує автоматизацію, зручність, безпеку та персоналізований підхід. Завдяки цьому він забезпечує всебічну підтримку процесів бронювання і управління, відповідно до сучасних вимог ринку і потреб користувачів.

2.3 Технологічне забезпечення вебзастосунку для бронювання робочих місць у коворкінгах

Розробка вебзастосунку для бронювання робочих місць у коворкінгах вимагає дуже ретельного підбору технологічного стека, адже саме від нього залежить стабільність роботи системи, її безпека, інтерактивність і здатність масштабуватися відповідно до зростаючих навантажень. Враховуючи, що застосунок має підтримувати інтенсивну взаємодію користувачів із робочими місцями, інтегруватися з різноманітними зовнішніми сервісами, а також обробляти значний обсяг даних, вибір технологій стає надзвичайно важливим і визначальним для успішності проєкту.

Клієнтська частина вебзастосунку, або фронтенд, створюється за допомогою сучасних технологій, що дозволяють побудувати адаптивний і максимально інтуїтивний інтерфейс. Він має коректно працювати на різних типах пристроїв — від смартфонів і планшетів до десктопних комп’ютерів, забезпечуючи однаково комфортний користувацький досвід. Основу фронтенду складають HTML, CSS та JavaScript. HTML формує структуру сторінок, CSS відповідає за зовнішній вигляд і стилі, при цьому особлива увага приділяється адаптивному дизайну, щоб інтерфейс автоматично підлаштовувався під розмір екрану користувача. JavaScript же надає динамічність, дозволяючи оновлювати інформацію, наприклад, статус доступності робочих місць у режимі реального часу, без необхідності перезавантажувати сторінку, що значно покращує взаємодію з користувачем [7 - 9].

Серверна частина, або бекенд, відповідає за логіку обробки запитів, управління базою даних, аутентифікацію та авторизацію користувачів, ведення бронювань, а також за зв’язок із зовнішніми сервісами. Для цих цілей було обрано

Node.js — платформу, що працює на основі асинхронної моделі, що дозволяє ефективно справлятися з великою кількістю одночасних запитів без втрати продуктивності. Бекенд реалізує REST API, яке забезпечує чіткий і стандартизований обмін даними між фронтендом і сервером, дозволяючи користувачам отримувати актуальну інформацію про доступні робочі місця, робити бронювання, редагувати їх або скасовувати [10].

Важливою складовою є база даних, де зберігається вся інформація про робочі місця, користувачів, бронювання, тарифи і інші необхідні дані для функціонування системи. Для роботи з такими структурованими даними застосовується реляційна система управління базами даних (СУБД), наприклад LowDB, що підтримує виконання SQL-запитів і транзакції, що гарантує цілісність та надійність інформації навіть при високому навантаженні [11].

Для організації спільної роботи розробників у проєкті використовується система контролю версій Git [12]. Це дозволяє відстежувати всі зміни у коді, ефективно координувати роботу команди, створювати нові введення в іншій гілці, не втручаючись у головну стабільну версію, а потім при потребі об’єднати їх з головною гілкою та швидко повертатися до попередніх версій у разі потреби. Автоматизація процесів тестування, розгортання (деплою) та моніторингу системи реалізується за допомогою CI/CD-пайплайнів, наприклад, через сервіс GitHub Actions. Нові функції перед впровадженням обов’язково проходять тестування у спеціальному staging-середовищі, що допомагає уникнути помилок та збоїв у робочій (продакшн) версії застосунку.

Таким чином, технологічний стек вебзастосунку включає в себе інтеграцію frontend і backend технологій, систему управління базою даних, а також інфраструктурні та DevOps-інструменти, які разом забезпечують швидку, безпечну і масштабовану роботу системи. Вдалий вибір та гармонійне поєднання цих компонентів є фундаментом для успішної реалізації проєкту, що відповідає сучасним вимогам ринку коворкінгів та очікуванням користувачів щодо зручності та функціональності сервісу [13].

2.4 Безпека вебзастосунку для бронювання робочих місць у коворкінгах

Інформаційна безпека є надзвичайно важливою складовою при розробці вебзастосунків, особливо тих, що працюють з конфіденційними даними користувачів — такими як персональна інформація, дані бронювання робочих місць та платіжні реквізити. Потенційний витік або компрометація цих даних може призвести не лише до втрати довіри з боку клієнтів, а й до суттєвих фінансових збитків, а також викликати юридичні проблеми, що негативно вплинуть на репутацію компанії. Тому для забезпечення надійного захисту вебзастосунку застосовуються комплексні заходи, які охоплюють програмний, автентифікаційний, мережевий та організаційний рівні безпеки.

На програмному рівні система захищена від найбільш поширених видів атак, таких як SQL-ін’єкції, міжсайтове скриптування (XSS) та підробка міжсайтових запитів (CSRF). Для цього реалізується сувора валідація всіх вхідних даних на стороні сервера — усі форми, що приймають інформацію від користувачів (реєстрація, авторизація, пошук робочих місць), мають вбудовані механізми перевірки коректності введених даних. Окрім цього, застосовуються методи фільтрації та санітизації даних, а також використання параметризованих SQL-запитів, що значно знижує ризик впровадження шкідливого коду у базу даних. Сучасні системи безпеки також можуть застосовувати поведінкову аналітику для виявлення підозрілої активності, наприклад, багаторазових невдалих спроб входу або спроб вставки шкідливих скриптів.

Автентифікація і авторизація користувачів організовані через розмежування ролей, що дозволяє контролювати рівень доступу до різних частин системи. Наприклад, зареєстрований користувач має розширені права, зокрема керування своїми бронюваннями, тоді як відвідувач без реєстрації бачить лише обмежений обсяг інформації. Використовується підхід Zero Trust, який передбачає перевірку кожного запиту незалежно, що мінімізує ризики несанкціонованого доступу.

Усі дані, що передаються між клієнтським пристроєм і сервером, шифруються за допомогою протоколу HTTPS із застосуванням SSL/TLS-сертифікатів, що забезпечує конфіденційність та цілісність інформації під час

передачі. Паролі користувачів перед збереженням у базі даних хешуються із застосуванням надійних алгоритмів, таких як bcrypt, що унеможливлює відновлення оригінальних паролів навіть у разі компрометації серверу. Для збереження чутливих файлів, наприклад підтверджень бронювання, застосовується серверне шифрування, яке захищає інформацію від несанкціонованого доступу без наявності відповідного ключа.

Для захисту від атак типу DDoS (розподілених атак відмови в обслуговуванні) в системі впроваджено обмеження частоти запитів (rate limiting), що запобігає надмірному навантаженню. Крім того, підтримується інтеграція з зовнішніми сервісами, наприклад, Cloudflare, які забезпечують додатковий рівень мережевого захисту. Системи виявлення вторгнень (IDS/IPS) аналізують мережевий трафік для виявлення підозрілих дій, а моніторинг логів дає змогу оперативно реагувати на потенційні загрози.

Організаційні заходи безпеки включають регулярне резервне копіювання даних, що забезпечує їх збереження у випадку збою або атаки. Крім того, проводиться аудит активності користувачів і системи для виявлення аномалій, а також навчання персоналу основам кібербезпеки. Зокрема, адміністратори отримують тренінги щодо розпізнавання фішингових атак, а доступ до найбільш чутливих даних суворо обмежений і надається лише необхідним співробітникам.

Вебзастосунок відповідає вимогам міжнародних стандартів захисту персональних даних, таких як GDPR, що гарантує користувачам право контролювати свої особисті дані — доступ до них, видалення чи обмеження обробки. Політика конфіденційності є прозорою, а передача даних третім особам відбувається лише у разі необхідності, наприклад, при інтеграції з платіжними шлюзами, такими як Stripe, що мають власні високі стандарти безпеки.

Таким чином, комплексний підхід до інформаційної безпеки, який включає програмні методи захисту, контроль доступу, мережеві технології і організаційні практики, забезпечує високий рівень захисту даних користувачів, сприяє формуванню довіри клієнтів і гарантує стабільність та безперебійність роботи вебзастосунку для бронювання робочих місць у коворкінгах.

2.5 Масштабування вебзастосунку для бронювання робочих місць у коворкінгах

Масштабування вебзастосунку для бронювання робочих місць у коворкінгах є важливим для забезпечення його стабільності, продуктивності та здатності обробляти зростаючі обсяги користувачів і запитів. Особливо це актуально для періодів пікового навантаження, наприклад, під час проведення акцій або запуску нових коворкінгів. Масштабування охоплює як апаратні, так і програмні рішення.

Вертикальне масштабування передбачає збільшення апаратних ресурсів серверів, таких як процесорна потужність, оперативна пам’ять чи обсяг сховища. Це може бути корисно для невеликих коворкінгів, де кількість запитів не перевищує певного порогу. Наприклад, для обробки збільшеної кількості бронювань можна оновити сервер без змін у архітектурі. Проте цей підхід має обмеження, оскільки апаратні ресурси мають свій фізичний максимум, і вартість збільшення стає значною.

Горизонтальне масштабування передбачає додавання нових серверів у систему, що дозволяє розподілити навантаження між кількома машинами. Наприклад, якщо кількість користувачів, які одночасно переглядають доступні місця, значно зростає, додаткові сервери забезпечують обробку більшої кількості запитів. Синхронізація між серверами реалізується за допомогою розподілених баз даних, таких як lowDB із реплікацією, щоб забезпечити актуальність даних у реальному часі.

Використання мікросервісної архітектури розділяє вебзастосунок на незалежні модулі, такі як управління бронюваннями, авторизація користувачів та аналітика. Кожен модуль масштабується окремо, залежно від потреб. Наприклад, під час пікових навантажень модуль бронювання може бути масштабований без необхідності змін у роботі інших модулів. Оркестрація мікросервісів здійснюється за допомогою Kubernetes, який автоматизує процеси розгортання та відновлення.

Хмарні платформи, такі як AWS, Azure або Google Cloud, дозволяють динамічно масштабувати ресурси залежно від поточного навантаження.

Наприклад, під час зростання трафіку AWS Auto Scaling може автоматично додати нові віртуальні машини для обробки додаткових запитів. Використання мереж доставки контенту (CDN), таких як Cloudflare, прискорюватиме завантаження статичних файлів, наприклад, зображень коворкінгів, для користувачів з різних регіонів [14].

Моніторинг є важливим елементом масштабування. Інструменти, такі як Prometheus і Grafana, відстежують ключові метрики, включаючи час відповіді сервера, кількість запитів на секунду та використання ресурсів. Наприклад, якщо модуль авторизації починає працювати повільно, система автоматично сповіщатиме адміністратора, який може додати додаткові ресурси для вирішення проблеми.

Процес масштабування може супроводжуватися технічними викликами, такими як складність управління мікросервісами, збільшення витрат на хмарні ресурси або потенційні затримки у синхронізації баз даних. Для мінімізації цих ризиків використовуються автоматизовані тести, системи резервного копіювання та постійний моніторинг.

Масштабування вебзастосунку для бронювання робочих місць у коворкінгах є комплексним процесом, який включає вертикальні та горизонтальні рішення, балансування навантаження, мікросервісну архітектуру та хмарні технології. Правильний підхід до масштабування дозволяє забезпечити стабільність, продуктивність і гнучкість системи, відповідаючи зростаючим потребам користувачів.

3 РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЛЯ ВЕБЗАСТОСУНКУ ПО БРОНЮВАННЮ МІСЦЬ У КОВОРКІНГАХ

3.1 Структура та загальна організація вебзастосунку для бронювання робочих місць у коворкінгах

Вебзастосунок для бронювання робочих місць у коворкінгах розроблений як сучасний веб-додаток, який базується на динамічному оновленні контенту. Основна мета системи полягає в забезпеченні швидкої, інтуїтивно зрозумілої та зручної взаємодії користувачів із сервісом за рахунок автоматизації процесів вибору, бронювання та управління робочими місцями. Використання сучасних веб-технологій, таких як HTML5, CSS3 і JavaScript (ES6+), дозволяє створити адаптивний, функціональний і ефективний інтерфейс, який працює стабільно на різних пристроях.

Структура вебзастосунку побудована за модульним принципом, що забезпечує гнучкість, масштабованість і зручність у подальшій підтримці та розвитку системи. Одним із ключових модулів є каталог робочих місць, який відображає доступні опції з докладною інформацією. Тут користувачі можуть ознайомитися з типом місця (наприклад, окрема кімната або відкрита зона), наявністю додаткових послуг, таких як розетки чи кондиціонер, а також з ціною. Каталог інтегрований із пошуковою системою і фільтрами, що допомагає швидко знаходити потрібні варіанти [15].

Пошук реалізує можливість швидкого знаходження місць за ключовими словами, наприклад, «кімната для переговорів» або «робоче місце з видом на вікно». Результати пошуку відображаються динамічно у вигляді спливаючого вікна, що робить процес дуже зручним і економить час користувача.

Модуль фільтрації дає змогу налаштовувати відображення робочих місць за різними критеріями, такими як ціна, тип місця або наявність певного обладнання. Фільтри формуються динамічно, залежно від обраної категорії, що робить систему адаптивною і дозволяє користувачу швидко відсіяти непотрібні варіанти.

Бронювання — це окремий модуль, який дозволяє вибирати конкретний час і дату для резервування робочого місця. Процес бронювання включає заповнення

простої форми з особистими даними користувача (ім’я, телефон, email) і підтвердження замовлення. Це робить користування сервісом зрозумілим і безперешкодним.

Для зручності користувачів у системі реалізовано модуль списку обраних місць. Цей функціонал дає змогу зберігати вподобані варіанти для подальшого бронювання або порівняння. Особливо це корисно для тих, хто планує свої бронювання заздалегідь і хоче мати швидкий доступ до улюблених робочих місць.

Навігація у вебзастосунку побудована так, щоб забезпечити інтуїтивний перехід між основними розділами — «Головна», «Каталог місць», «Про нас» та «Контакти». Використання навігаційного меню і «хлібних крихт» значно полегшує орієнтацію користувача в системі та дозволяє швидко знаходити потрібну інформацію без зайвих зусиль. Головну сторінку каталогу бронювань наведено на рисунку 3.1.

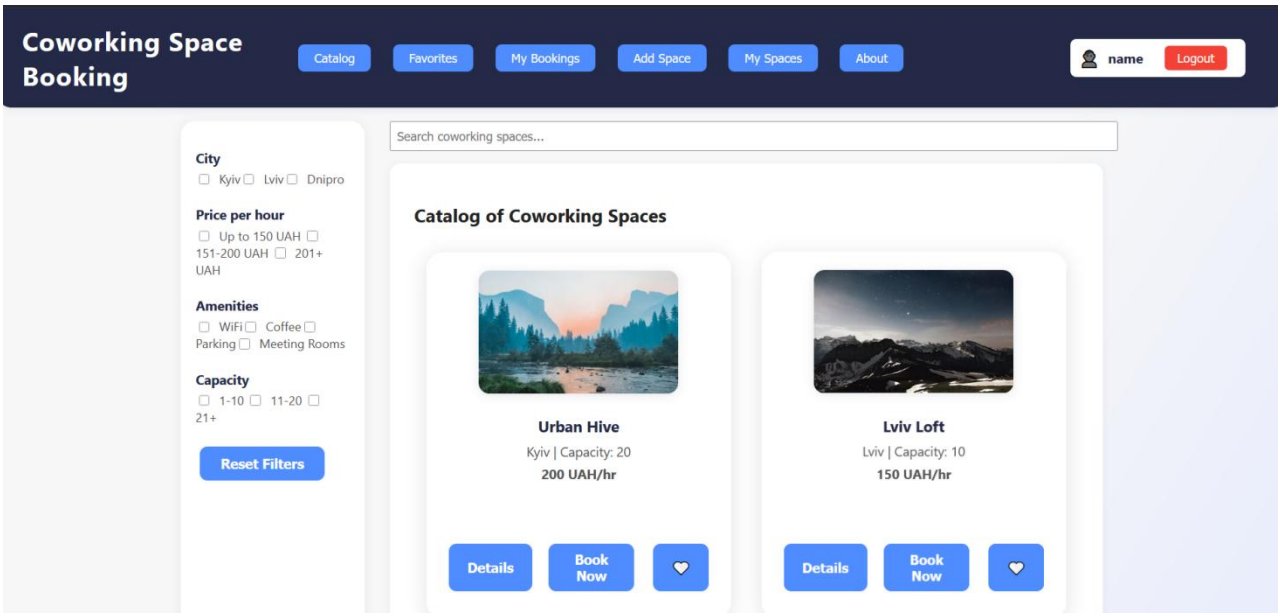


Рисунок 3.1 — Головна сторінка доступних коворкінгів

Основна архітектура вебзастосунку базується на динамічному оновленні DOM-дерева за допомогою JavaScript. Усі компоненти інтегровані в єдину структуру, що спрощує управління та забезпечує швидкість роботи системи. Наприклад, при виборі категорії "Переговорні кімнати" викликається функція

`renderCatalog("Переговорні кімнати")`, яка динамічно генерує список доступних місць із відповідними характеристиками.

Дані про місця зберігаються у вигляді об'єкта `workspaces`, який містить ієрархічну структуру категорій, опис робочих місць і їх атрибути, такі як `id`, `name`, `price`, `features` та `image`. Це дозволяє легко додавати нові місця чи категорії без значних змін у коді.

Для збереження даних між сеансами, таких як список обраних місць або дані про поточне бронювання, використовується `localStorage`. Наприклад, функція `saveFavorites()` зберігає список вподобаних місць у форматі JSON, що дозволяє користувачам продовжити роботу навіть після закриття браузера.

Інтерфейс вебзастосунку адаптований для роботи на різних пристроях: від смартфонів до настільних комп’ютерів. Використання CSS3-медіазапитів дозволяє автоматично налаштовувати макет залежно від розміру екрана. Для забезпечення доступності застосовуються атрибути `aria-label`, що робить систему зручною для користувачів із обмеженими можливостями.

Система спроектована з урахуванням можливості подальшого розвитку. Наприклад, для інтеграції з платіжними шлюзами чи сервісами автоматизації бронювання достатньо додати відповідні модулі без значних змін у існуючій структурі. Це забезпечує гнучкість і готовність до нових функціональних вимог.

Отже, структура вебзастосунку для бронювання робочих місць у коворкінгах забезпечує зручність, швидкість і гнучкість взаємодії з користувачами, роблячи систему ефективним інструментом для управління робочими місцями.

3.2 Стилізація та адаптація інтерфейсу вебзастосунку для бронювання робочих місць у коворкінгах

Оформлення та адаптивний дизайн інтерфейсу вебзастосунку для бронювання робочих місць у коворкінгах є визначальними чинниками для забезпечення зручності користування, візуальної привабливості та ефективної взаємодії з системою. Основне завдання стилізації — створення інтерфейсу, який

буде інтуїтивно зрозумілим, гармонійно виглядатиме на різних пристроях та відповідатиме сучасним вимогам до UX/UI-дизайну. Це сприяє покращенню користувацького досвіду і підвищує задоволеність від використання сервісу [16].

Для реалізації стилізації застосовується мова каскадних таблиць стилів CSS3. Вона дозволяє створювати адаптивну структуру завдяки використанню таких інструментів, як Flexbox і Grid-сітки. Вони забезпечують гнучке компонування елементів на сторінці, дозволяючи інтерфейсу коректно масштабуватися під будь-який розмір екрана — від мобільного телефону до великого монітора.

Анімації та ефекти переходу додають інтерактивності: кнопки, картки робочих місць і спливаючі вікна реагують на дії користувача плавними візуальними змінами. Завдяки цьому система виглядає живою, сучасною та зручною у користуванні. Медіазапити (media queries) забезпечують адаптацію інтерфейсу під конкретні типи пристроїв, змінюючи розміщення, розміри або навіть відображення окремих елементів відповідно до ширини екрану.

Особлива увага приділяється типографіці: у проєкті використовуються сучасні шрифти без засічок, наприклад Roboto. Це сприяє кращій читабельності тексту, що є важливим для зручного сприйняття інформації користувачами.

Ще одним ключовим принципом стилізації є дотримання єдиної кольорової гами. В основі дизайну лежить поєднання світлих нейтральних тонів (білий — #FFFFFF, світло-сірий — #F5F5F5) як фонового оформлення і насичених акцентних кольорів (наприклад, синього #007bff і зеленого #00cc99) для елементів управління. Це поєднання робить інтерфейс візуально привабливим, не перевантажуючи його кольорами. Для прикладу, кнопка «Забронювати» стилізується з використанням градієнта (background: linear-gradient(90deg, #007bff, #00cc99)), що створює ефект динаміки та привертає увагу користувача до ключових дій.

```
#content {
   flex: 1;
   background: #fff;
   padding: 2rem;
   border-radius: 14px;
```

```
    box-shadow: 0 2px 8px rgba(0,0,0,0.05);
}
.catalog-grid {
    display: grid;
    grid-template-columns: repeat(auto-fill, minmax(320px, 1fr));
    gap: 2.5rem;
    margin-top: 2rem;
    margin-bottom: 2rem;
    padding: 0 1rem;
}
.space-item {
    background: #fff;
    border-radius: 22px;
    box-shadow: 0 4px 24px rgba(35,41,70,0.13);
    padding: 1.5rem 1.5rem 2rem 1.5rem;
    text-align: center;
    transition: box-shadow 0.25s, transform 0.18s;
    position: relative;
    display: flex;
    flex-direction: column;
    align-items: center;
    min-height: 420px;
}
```

У вебінтерфейсі для бронювання робочих місць у коворкінгах використовується сучасна типографіка, яка базується на шрифтах без засічок — зокрема, Roboto та Open Sans. Для забезпечення чіткого візуального поділу інформації застосовуються різні ваги шрифтів: основний текст має стандартну товщину (вага 400), а заголовки — виділену (вага 700). Розміри шрифтів задаються в одиницях rem, що дозволяє їм масштабуватись відповідно до роздільної здатності та налаштувань пристрою користувача. Наприклад, основний текст встановлено на рівні 1rem (еквівалент 16 пікселів), тоді як заголовки — на 1.25rem (близько 20 пікселів), що забезпечує хорошу читабельність на будь-якому екрані.

Структура інтерфейсу побудована на основі гнучких макетів. Для елементів навігаційного меню та фільтрації використовується display: flex, що дає змогу легко вирівнювати вміст. Візуальне відображення робочих місць реалізовано через CSS Grid — на великих екранах картки розміщуються у три колонки за допомогою grid-template-columns: repeat(3, 1fr), а на мобільних пристроях ця

структура перебудовується в одну колонку завдяки медіазапитам, що забезпечує зручність перегляду.

Для покращення взаємодії користувача з інтерфейсом передбачено динамічні візуальні ефекти. Наприклад, кнопки мають плавні переходи кольору (transition: background-color 0.3s ease), а картки робочих місць збільшуються при наведенні (transform: scale(1.05)), що створює ефект живої взаємодії. Інтерфейс також спроектовано з урахуванням принципів доступності (A11y). Для покращення контрастності дотримано мінімального співвідношення 4.5:1 між кольором тексту та фону, що дозволяє комфортно користуватись сайтом людям із вадами зору. Інтерактивні елементи мають атрибути aria-label, наприклад, aria-label="Пошук місць", що допомагає користувачам із допоміжними технологіями краще орієнтуватись у функціоналі сайту. Розміри кнопок оптимізовані для взаємодії на сенсорних пристроях — вони не менші за 44x44 пікселі, що відповідає рекомендаціям щодо ергономіки.

Завдяки медіазапитам інтерфейс адаптується під різні розміри екранів. На пристроях із шириною до 768 пікселів стандартне меню приховується, а натомість відображається бургер-іконка, що відкриває вертикальну навігацію. Пошуковий рядок при цьому розтягується до 80% ширини екрана, а панель фільтрів перебудовується у вертикальне розташування. На ще менших екранах — до 480 пікселів — розмір шрифтів зменшується до 14 пікселів, відступи оптимізуються до 8 пікселів, а кнопки стають більшими для зручнішого натискання пальцем.

Інтерфейс вебзастосунку складається з кількох ключових елементів, що забезпечують зручність та ефективність взаємодії користувача із системою. Одним із основних компонентів є навігаційне меню, яке містить швидкі посилання на головні розділи сайту — такі як «Головна» та «Каталог місць». Це дозволяє користувачам швидко орієнтуватися та переходити між різними сторінками.

Картки робочих місць є центральним елементом контенту. Кожна з них включає зображення, назву, короткий опис та кнопку «Забронювати», що дає змогу легко переглядати варіанти і швидко оформлювати бронювання. Для

покращення пошуку потрібного місця передбачена система фільтрації — користувачі можуть відсортувати результати за ціною, типом простору або наявним обладнанням. Додаткові дії, як-от перегляд кошика або підтвердження бронювання, відображаються у вигляді спливаючих вікон, що з'являються поверх основного інтерфейсу.

Вигляд інтерфейсу на мобільних пристроях ілюстровано на рисунку 3.1.

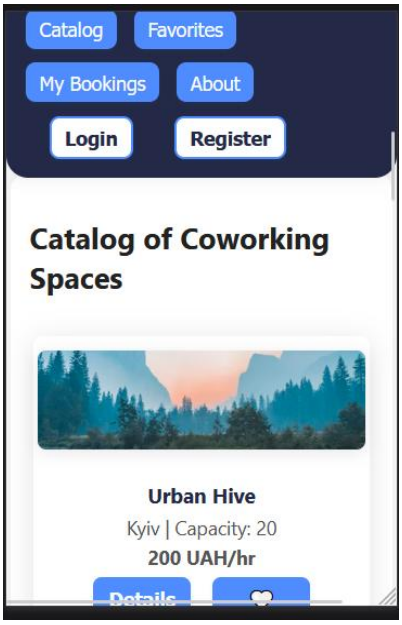


Рисунок 3.2 — Скріншот списку бронювань на мобільному пристрої

З метою покращення швидкодії вебзастосунку реалізовано низку технічних оптимізацій. CSS-стилі мінімізовано: застосування спільних класів зменшує обсяг дубльованого коду. Анімації реалізовано за допомогою властивостей transform та opacity, що є менш ресурсоємними порівняно зі зміною ширини чи висоти елементів. Крім того, для економії трафіку та пришвидшення завантаження сторінки впроваджено ліниве завантаження зображень — через атрибут loading="lazy".

Система має гнучку архітектуру та підтримує розширення функціональності. Наприклад, реалізація темної теми можлива завдяки використанню CSS-змінних — таких як --primary-color, що спрощує зміну кольорової схеми інтерфейсу. Додатково передбачена підтримка багатомовності, включно з мовами, що

читаються справа наліво (RTL), що реалізується за допомогою властивості direction: rtl.

Особливу увагу також приділено мікроанімаціям — легкі ефекти, такі як пульсація кнопок або плавне розгортання фільтрів, роблять взаємодію з інтерфейсом живішою та приємнішою.

Загалом, продумана стилізація та адаптивний дизайн забезпечують привабливість, функціональність і доступність вебзастосунку на різних пристроях. Завдяки використанню сучасних можливостей CSS — таких як Flexbox, Grid, анімації та медіазапити — система має потужну основу для подальшого вдосконалення та впровадження нових функцій.

3.3 Реалізація логіки взаємодії за допомогою JavaScript

Логіка взаємодії у вебзастосунку для бронювання робочих місць у коворкінгах побудована з використанням JavaScript (ES6+), що забезпечує динамічну роботу інтерфейсу, інтерактивність і зручність для користувачів. Основна мета цього підпункту — описати, як JavaScript реалізує ключові функції системи: відображення каталогу місць, пошук, фільтрацію, бронювання та збереження обраних місць. Логіка системи базується на принципах Single Page Application (SPA), де всі дії користувачів виконуються без перезавантаження сторінки.

JavaScript використовується у вебзастосунку для обробки подій, що є фундаментальним аспектом інтерактивної взаємодії з інтерфейсом. Завдяки обробникам подій система реагує на дії користувачів — такі як кліки по кнопках, введення тексту в поля пошуку або зміна стану чекбоксів у фільтрах. Це дозволяє миттєво запускати відповідні функції, наприклад, застосування фільтрів чи додавання місця до списку обраних, що значно покращує зручність користування.

Ще однією важливою функцією є рендеринг вмісту, який забезпечує динамічне оновлення DOM (Document Object Model). Коли користувач переглядає каталог, обирає фільтри або переходить до сторінки з деталями місця, JavaScript оперативно змінює вміст сторінки без її повного перезавантаження. Такий підхід

дозволяє значно зменшити час очікування та створити відчуття безперервної взаємодії.

JavaScript також використовується для збереження стану між сеансами користувача. Застосування механізмів локального сховища (localStorage) дозволяє зберігати обрані місця, застосовані фільтри чи попередні пошукові запити навіть після оновлення сторінки або повторного входу користувача. Це створює більш персоналізований досвід і зберігає контекст дій користувача.

Важливою перевагою є модульна структура застосунку. Уся логіка поділена на окремі файли та функціональні блоки, які відповідають за конкретні завдання — наприклад, рендеринг карток місць, обробку пошуку або роботу з фільтрами. Такий підхід покращує читабельність коду, полегшує його тестування та дає змогу швидко вносити зміни або додавати нові функції.

Загалом використання JavaScript забезпечує гнучкість та масштабованість вебзастосунку. Завдяки добре структурованому коду і принципам SPA (Single Page Application), система легко адаптується до змін у бізнес-логіці, зростання кількості користувачів або розширення функціоналу, зберігаючи при цьому високу швидкість і стабільність роботи.

3.4 Оформлення бронювання та валідація введених даних

Оформлення бронювання разом із перевіркою введених даних є фінальним кроком у взаємодії користувача з вебзастосунком для резервування робочих місць у коворкінгу. Користувач переходить до цього етапу після натискання кнопки «Забронювати» на картці вибраного місця, що ініціює функцію navigate('booking') і відкриває форму бронювання в основному контейнері сайту. У цій формі користувач заповнює персональні дані, обирає дату та час через інтерактивні елементи й підтверджує бронювання.

Уся перевірка заповнених даних виконується на клієнтській стороні. Використовуються можливості HTML5, як-от обов’язковість полів, типи email і tel, а також атрибути pattern для перевірки формату. Додатково застосовується JavaScript-функція validateBookingForm(), яка проводить більш точну валідацію з

використанням регулярних виразів: наприклад, ім’я перевіряється на наявність лише літер і пробілів, а email — на відповідність стандартному формату адреси. У разі помилок користувачу негайно відображаються підказки поруч із полями, а самі поля виділяються червоною рамкою.

Після успішної перевірки створюється об’єкт booking, у якому зберігаються дані користувача, вибране місце, дата, час та згенерований унікальний номер бронювання. Цей об’єкт записується у локальне сховище браузера (localStorage), а користувач бачить повідомлення про підтвердження операції — у вигляді модального вікна або спливаючого елементу.

Інтерфейс форми побудований з урахуванням сучасних принципів UX/UI. Всі елементи логічно згруповані, мають чіткі підписи й реагують на дії користувача. Наприклад, список часу автоматично оновлюється згідно з обраною датою, а кнопка підтвердження має привабливе візуальне оформлення з градієнтом і анімацією натискання. Простота структури форми — лише ключові поля без зайвих запитів — дозволяє заповнити її швидко та без зусиль.

Особлива увага приділяється доступності: усі поля мають атрибути aria-label, що підтримують роботу з екранними читачами, а повідомлення про помилки пов’язані з відповідними полями через aria-describedby. Таким чином, оформлення бронювання реалізоване з урахуванням потреб усіх користувачів і відповідає сучасним стандартам зручності, доступності та технічної надійності.

3.5 Реалізація персоналізованого функціоналу: рекомендації, обрані місця та збереження стану

Індивідуалізація інтерфейсу значно покращує зручність користування вебзастосунком, адаптуючи його відповідно до особистих вподобань користувача. У сервісі для бронювання робочих місць у коворкінгах передбачено декілька механізмів персоналізації: модуль рекомендацій, система збережених (обраних) місць, фіксація обраних фільтрів і пошукових параметрів, а також інтерактивні елементи, що сприяють легкому та зрозумілому користувацькому досвіду.

Ці функції реалізовано із використанням JavaScript (на базі сучасного стандарту ES6+), HTML5, CSS3, а також засобів локального зберігання (localStorage), що забезпечує швидку клієнтську обробку та мінімізує затримки під час взаємодії. У цьому розділі розглядається технічне впровадження персоналізації, її вплив на UX, продуктивність системи та можливості подальшого розвитку — з прикладами коду та візуальними фрагментами інтерфейсу.

Блок рекомендацій відображається на головній сторінці в елементі під заголовком «Рекомендовані місця», а також на сторінці конкретного об’єкта у секції . На головній сторінці функція renderHomeRecommendations() вибірково формує список із максимум десяти робочих місць, які обираються випадково з масиву workspaces, формуючи картки зі зображенням, назвою, вартістю та кнопкою переходу до детального перегляду. На сторінці окремого місця функція renderRecommendationsForWorkspace() генерує до чотирьох альтернативних місць тієї ж категорії, виключаючи поточне.

javascript name=recommendations.js

```
function renderHomeRecommendations() {
  const recommendationsContainer = document.getElementById("home-recommendations");
  if (!recommendationsContainer) return;
  const randomRecommendations = workspaces.sort(() => 0.5 - Math.random()).slice(0, 10);
  recommendationsContainer.innerHTML = randomRecommendations.map(place => `
    <div class="recommendation-item" onclick="openWorkspace('${place.id}')">
      <img src="${place.image}" alt="${place.name}">
      <h3>${place.name}</h3>
      <p>${place.price} грн/год</p>
    </div>
  `).join('');
}
```

Механізм збереження стану забезпечує безперервність взаємодії користувача з вебзастосунком. Обрані параметри фільтрації, зокрема тип місця або діапазон цін, фіксуються у localStorage, що дозволяє зберегти їх навіть після

оновлення сторінки чи повторного відкриття сайту. Крім того, останній введений пошуковий запит також зберігається, завдяки чому користувач може миттєво відновити попередній пошук під час наступного сеансу.

```
javascript name=state.js
function updateBook() {
   const bookItemsContainer = document.getElementById("book-items");
   const total =book.reduce((sum, item) => sum + item.price * item.hours, 0);
   bookItemsContainer.innerHTML = book.length > 0
      ? book.map(item => `
         <li>
            <img src="${item.image}" alt="${item.name}" style="width: 50px; height: 50px;">
            <span>${item.name} - ${item.price} грн/год</span>
            <button onclick="removeFromBook('${item.id}')">❌</button>
         </li>
      `).join('')
     localStorage.setBook('bookt', JSON.stringify(book));
}
```

Реалізація персоналізованих функцій, таких як рекомендації, список обраних місць, історія бронювань та збереження стану допомагає забезпечити зручність і інтерактивність роботи користувачів із системою. Використання сучасних веб-технологій дозволяє досягнути швидкої взаємодії та створити основу для подальшого вдосконалення та розширення інформаційної системи.

3.6 Реалізація інтерактивної взаємодії з користувачем: пошук, фільтрація, навігація

Інтерактивна взаємодія з користувачем є критичною для системи бронювання робочих місць у коворкінгах. У вебзастосунку реалізовані пошук місць, фільтрація за параметрами та інтуїтивна навігація, що спрощує користування системою. Ці функції базуються на JavaScript (ES6+), HTML5, CSS3 та localStorage, забезпечуючи зручність, швидкість і адаптивність. Розділ описує технічну реалізацію кожної функції, аспекти UX, продуктивність та перспективи вдосконалення.

Пошукова система надає користувачам зручний спосіб швидко знаходити доступні робочі місця. У центрі цієї функціональності знаходиться поле для введення запиту, яке розміщене в блоці з класом search-container і має

ідентифікатор search-box. Коли користувач вводить текст у це поле, автоматично активується функція searchWorkspaces(), яка порівнює введений запит із назвами доступних місць. Порівняння здійснюється у нижньому регістрі за допомогою методу includes().

Результати пошуку динамічно відображаються в елементі з ідентифікатором search-results. Кожен результат подається у вигляді списку, що містить зображення, назву робочого місця, його ціну та кнопку або посилання для переходу до детальної інформації про нього. Такий підхід забезпечує інтерактивність і полегшує користувачу процес пошуку та вибору відповідного варіанту.

```
javascript name=search.js
function searchWorkspaces() {
   const searchBox = document.getElementById("search-box");
   const searchResults = document.getElementById("search-results");
   const query = searchBox.value.toLowerCase();
   if (query.length === 0) {
      searchResults.style.display = "none";
      return;
   }
   const results = workspaces.filter(workspace => workspace.name.toLowerCase().includes(query));
   if (results.length > 0) {
      searchResults.innerHTML = results.map(result => `
         <div class="search-result-item" onclick="openWorkspace('${result.id}')">
            <img src="${result.image}" alt="${result.name}" style="width: 50px; height: 50px;">
            <p>${result.name} - ${result.price} грн/год</p>
         </div>
      `).join('');
      searchResults.style.display = "block";
   } else {
      searchResults.innerHTML = "<p>Нічого не знайдено</p>";
      searchResults.style.display = "block";
   }
}
```

Функція фільтрації дозволяє користувачам налаштовувати відображення робочих місць відповідно до власних уподобань — зокрема за типом місця, ціновим діапазоном чи наявністю обладнання. Інтерфейс фільтрів реалізовано у бічній панелі з ідентифікатором filter, яка містить набір чекбоксів для вибору відповідних параметрів.

Коли користувач змінює налаштування фільтрів, активується функція applyFilters(). Вона збирає значення обраних чекбоксів, після чого застосовує метод filter() до масиву workspaces, залишаючи лише ті об'єкти, які відповідають критеріям. Відфільтровані дані передаються до функції renderFilteredPlaces(), яка оновлює список місць на сторінці.

Для зручності користувача та збереження вибору між сеансами, поточний стан фільтрів синхронізується з localStorage. Увесь код, що відповідає за цю логіку, розміщено у файлі filters.js.

```
function applyFilters() {
    const filteredPlaces = workspaces.filter(place => {
        const matchesType = filters.type.length === 0 || filters.type.includes(place.type);
        const matchesPrice = filters.price.length === 0 || filters.price.some(range => {
            const [min, max] = range.split('-').map(Number);
            return place.price >= min && (max ? place.price <= max : true);
        });
        return matchesType && matchesPrice;
    });
    renderFilteredPlaces(filteredPlaces);
}

function renderFilteredPlaces(filteredPlaces) {
    const content = document.getElementById("content");
    content.innerHTML = `<div class="place-grid">` +
        filteredPlaces.map(place => `
            <div class="place-item">
                <img src="${place.image}" alt="${place.name}" style="width: 100px; height: 100px;">
                <h3>${place.name}</h3>
                <p>${place.price} грн/год</p>
                <button onclick="openWorkspace('${place.id}')">Деталі</button>
            </div>
        `).join('') + `</div>`;
}
```

Навігація у вебзастосунку реалізована таким чином, щоб забезпечити користувачеві інтуїтивний та зручний перехід між основними розділами. Головне меню розташоване всередині елемента <nav>, що входить до складу шапки сайту (<header>), і містить основні пункти — «Головна», «Каталог» та «Контакти».

Додатково для орієнтації в структурі сайту використовується навігаційний ланцюжок, або так звані «хлібні крихти», що реалізований у блоці з ідентифікатором breadcrumbs. Він відображає поточний маршрут користувача,

наприклад: «Головна > Каталог > Переговорні кімнати», допомагаючи швидко зрозуміти, де саме в ієрархії сайту він знаходиться.

Хлібна крихта або слід хлібних крихт — це графічний елемент керування, який використовується як допоміжний засіб навігації в інтерфейсах користувача та на веб-сторінках. Він дозволяє користувачам відстежувати та підтримувати усвідомлення свого місцезнаходження в програмах, документах або на веб-сайтах. Приклад подібного елементу наведено на рисунку 3.3 .

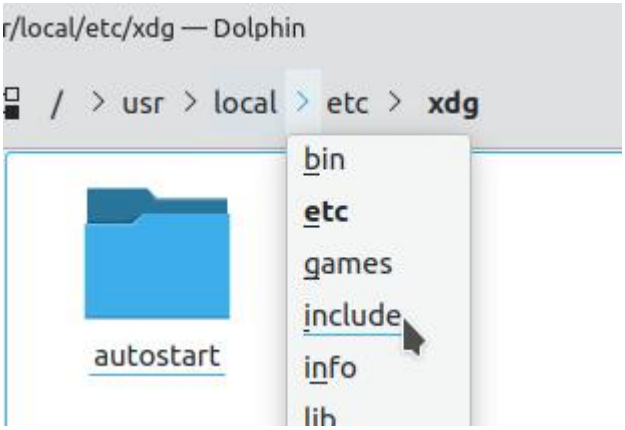


Рисунок 3.3 — Скріншот списку бронювань на мобільному пристрої

За керування переходами між сторінками відповідає функція navigate(). Вона оновлює вміст головного блоку <main id="content"> відповідно до вибраного розділу та викликає відповідні функції рендерингу для динамічного відображення даних. Такий підхід забезпечує плавну навігацію без перезавантаження сторінки.

```
javascript name=navigation.js
function navigate(page) {
   const content = document.getElementById("content");
   if (page === "home") {
      breadcrumbs = [{ name: "Головна", page: "home" }];
      content.innerHTML = `
         <h2>Вітаємо у коворкінгу!</h2>
         <div class="recommendations-container">
            <h2>Рекомендації для вас</h2>
            <div class="recommendations" id="home-recommendations"></div>
         </div>`;
      renderHomeRecommendations();
   } else if (page === "catalog") {
      breadcrumbs.push({ name: "Каталог", page: "catalog" });
      content.innerHTML = `
         <h2>Каталог місць</h2>
         <div class="category-grid">
```

```
                ${categories.map(category => `
                  <div class="category-card" onclick="navigate('${category}')">
                    <h3>${category}</h3>
                  </div>
                `).join('')}
              </div>`;
          }
          updateBreadcrumbs();
        }

        function updateBreadcrumbs() {
          const breadcrumbsNav = document.getElementById("breadcrumbs");
          breadcrumbsNav.innerHTML = breadcrumbs
            .map((crumb, index) => `<a href="#" onclick="navigate('${crumb.page}')">${crumb.name}</a>${index < breadcrumbs.length - 1 ? ' > ' : ''}`)
            .join("");
        }
```

3.7 Апаратна частина та технічна інфраструктура

Апаратна частина та технічна інфраструктура вебзастосунку для бронювання робочих місць у коворкінгах є ключовими для забезпечення стабільності, масштабованості та високої продуктивності системи. У цьому підрозділі описано вимоги до серверного та клієнтського обладнання, технічну архітектуру для розгортання, тестування та експлуатації системи. Реалізація враховує сучасні стандарти хмарних технологій, контейнеризації та клієнтських пристроїв, що забезпечує швидкий доступ, безперебійну роботу та захист даних.

Для забезпечення стабільної роботи вебзастосунку використовується хмарна інфраструктура, яка дає змогу гнучко масштабувати ресурси відповідно до рівня навантаження. Це особливо важливо в умовах змінного трафіку та великої кількості одночасних запитів.

Серверна частина повинна відповідати кільком технічним вимогам. Перш за все, необхідний багатоядерний серверний процесор — наприклад, Intel Xeon або AMD EPYC — з частотою не менше 2.5 ГГц. Для обробки пікового навантаження рекомендується мінімум 8 ядер, що дозволяє ефективно виконувати запити до бази даних і обробляти серверну логіку. Оперативна пам’ять має становити не менше 16 ГБ для підтримки багатозадачності, кешування (наприклад, через Redis)

і взаємодії з базою даних PostgreSQL. За умов високого трафіку бажано мати 32 ГБ або більше.

Щодо сховища, рекомендується використовувати SSD-диски обсягом від 500 ГБ, що забезпечує швидкий доступ до даних, включно з бронюваннями та медіафайлами. Для підвищення надійності даних застосовується RAID-конфігурація (наприклад, RAID 5 або RAID 10). Мережевий інтерфейс має підтримувати Gigabit Ethernet (1 Гбіт/с) для ефективного обміну даними між користувачами та сервером.

Для масштабованості впроваджується хмарна платформа, така як Amazon Web Services (AWS), із використанням контейнеризації через Docker та оркестрації за допомогою Kubernetes. Окремі мікросервіси, наприклад модуль бронювання, ізолюються для підвищення рівня безпеки та продуктивності.

З боку користувача вебзастосунок доступний через браузер на різних пристроях. Мінімальні вимоги до клієнтського обладнання включають двоядерний процесор із частотою 1.5 ГГц (наприклад, Intel Core i3 або ARM-аналоги для мобільних пристроїв), 4 ГБ оперативної пам’яті для комфортної роботи, а також екран із роздільною здатністю від 320×568 пікселів (мобільні) до 1920×1080 пікселів (настільні комп’ютери). Адаптивний дизайн забезпечується через CSS-медіазапити. Мінімальна рекомендована швидкість інтернет-з’єднання — 5 Мбіт/с, що дозволяє швидко завантажувати сторінки та зображення.

Система підтримує сучасні браузери (Google Chrome, Mozilla Firefox, Safari, Microsoft Edge) з увімкненим JavaScript. Для оптимізації продуктивності застосовується ліниве завантаження зображень (loading="lazy") та кешування ресурсів.

Розробка системи здійснювалася у середовищі Visual Studio Code із плагінами для JavaScript (ESLint, Prettier), CSS (Stylelint) та інтеграцією з Git.

Основними інструментами, що використовуються у розробці вебзастосунку, є JavaScript (ES6+) із нативним маніпулюванням DOM. Це дозволяє створювати динамічний і чутливий інтерфейс, який оперативно реагує на дії користувача без необхідності повного перезавантаження сторінки. Використання чистого

JavaScript забезпечує гнучкість у роботі з елементами сторінки та оптимальну продуктивність.

Для ефективної обробки даних застосовується бібліотека Lodash, яка надає широкий набір утиліт для роботи з масивами, об’єктами та функціями. Це значно спрощує виконання складних операцій, таких як сортування, фільтрація або клонування даних, що є важливим при роботі з каталогом робочих місць і фільтрами.

На серверній стороні використовується середовище Node.js разом із фреймворком Express для реалізації REST API. Це дозволяє організувати обмін даними між клієнтом і сервером у вигляді чітких HTTP-запитів, забезпечуючи швидке і надійне отримання та оновлення інформації, такої як бронювання або дані користувачів.

Для збереження даних застосовується база lowDB, яка відповідає за зберігання інформації про користувачів, бронювання та інші сутності системи. Додатково використовується Redis для кешування сесій і запитів, що підвищує швидкодію застосунку та зменшує навантаження на основну базу даних, особливо при великій кількості одночасних користувачів.

Для забезпечення якості коду і стабільності роботи реалізовано комплексне тестування. Модульне тестування основних функцій, таких як searchWorkspaces і applyFilters, проводиться за допомогою Jest, що дозволяє швидко виявляти помилки на рівні окремих компонентів. Крім того, для тестування повного циклу взаємодії користувача із застосунком застосовується Cypress, який перевіряє роботу системи від початку до кінця, забезпечуючи надійність функціоналу у реальних умовах.

Для забезпечення безпеки передачі даних у вебзастосунку застосовується протокол HTTPS із SSL-сертифікатами, наприклад, Let’s Encrypt. Це гарантує шифрування трафіку між клієнтом і сервером, що захищає інформацію від перехоплення або підробки під час передачі.

Ще одним важливим заходом безпеки є використання сервісу Cloudflare, який допомагає запобігати DDoS-атакам — масовим атакам на сервери з метою

перевантаження. Крім того, Cloudflare інтегрує веб-аплікаційний файрвол (WAF), що блокує спроби SQL-ін’єкцій та інші види атак на рівні додатку, підвищуючи загальну захищеність системи.

Для підвищення швидкості завантаження сайту використовується CDN (Content Delivery Network), яка кешує статичні ресурси, такі як зображення, стилі та скрипти, на серверах, розташованих у різних географічних регіонах. Це значно знижує затримки при завантаженні сторінок і покращує користувацький досвід, особливо для віддалених користувачів.

Для забезпечення відмовостійкості здійснювалось резервне копіювання. На випадок потреби попередня копія даних може бути реплікованою. Здіййснюється регулярне резервне копіювання бази даних (щоденно) в AWS S3.

Реплікація бази даних здійснюється для швидкого відновлення у разі збою. Кожен день створюються снапшоти контейнерів для автоматичного відновлення у разі виникнення критичних помилок.,

4 ТЕСТУВАННЯ ВЕБЗАСТОСУНКУ ДЛЯ БРОНЮВАННЯ РОБОЧИХ МІСЦЬ У КОВОРКІНГАХ

4.1 Тестування можливостей клієнтської частини вебзастосунку для бронювання робочих місць у коворкінгах

З метою забезпечення стабільної, надійної та зручної роботи вебзастосунку було проведено тестування клієнтської частини, яке охоплює ключові аспекти взаємодії користувача з інтерфейсом і логікою системи. У процесі розробки застосовувались типові методи тестування, що широко використовуються у веброзробці: функціональне, інтерфейсне, рольове, інтеграційне, а також ручне тестування.

Функціональне тестування дозволило перевірити, чи відповідає вебзастосунок вимогам, передбаченим технічним завданням. Було перевірено сценарії авторизації, створення та скасування бронювання, перевірка доступності місць у реальному часі, обробка помилок тощо.

Інтерфейсне тестування зосереджувалося на перевірці правильності відображення елементів UI — карт приміщень, форм пошуку, фільтрів, повідомлень, кнопок тощо. Тестувалась зручність навігації та логіка взаємодії користувача із системою. Зокрема, перевірено інформативність повідомлень про успішне чи неуспішне бронювання, зручність у використанні мобільної версії та адаптивність сторінок.

Рольове тестування дало змогу оцінити правильність розмежування доступу для різних категорій користувачів. Було перевірено, що відвідувач бачить лише свою інформацію та доступні йому функції, тоді як зареєстрований користувач має розширені права — перегляд та редагування бронювань, управління просторами, доступ до аналітики та налаштувань.

Інтеграційне тестування включало перевірку взаємодії між клієнтською частиною, сервером і базою даних. Було перевірено, як React-інтерфейс обробляє відповіді API, чи коректно відображаються повідомлення про помилки сервера, чи здійснюється синхронізація станів при оновленні даних.

Оскільки клієнтська частина реалізована як односторінковий застосунок (SPA) на базі React, особливу увагу приділялося динамічному оновленню даних, маршрутизації, обробці подій, формам бронювання, інтерактивній карті робочих місць та таблицям з фільтрацією.

Було проведено ручне тестування з опрацюванням конкретних сценаріїв, результати якого узагальнено в таблиці 4.1.

Модуль	**Сценарій**	**Результат**
Авторизація	Вхід, збереження токена, маршрутизація по сторінках	Успішно
Перевірка доступу	Відображення функцій за тим чи зареєстрований користувач	Успішно
Список коворкінгів	Завантаження, пошук, фільтрація, перегляд інформації	Успішно
Форма бронювання	Валідація, вибір коворкінгу та дати, обробка обмежень	Успішно
Список бронювань	Завантаження обраних коворкінгів, кольори, інформація по кліку	Успішно
Створення коворкінгу	Створення, перегляд, прикріплення файлу	Успішно
Робота з Google Drive	Завантаження файлу, генерація посилання, перегляд	Успішно
Звіти та аналітика	Фільтрація, експорт, вивід інформації	Успішно

У процесі тестування було виявлено низку незначних недоліків. Наприклад, після редагування бронювання не оновлювались дані в таблиці — це було усунено шляхом повторного запиту з використанням хука useEffect. Також було додано валідацію для запобігання створенню порожнього бронювання. Для захисту адміністративних сторінок впроваджено перевірку ролі на клієнтському рівні.

Загалом тестування клієнтської частини вебзастосунку підтвердило її відповідність функціональним вимогам, стабільність роботи та зручність у використанні для різних категорій користувачів.

4.2 Оцінка продуктивності клієнтської частини вебзастосунку для бронювання робочих місць у коворкінгах

Метою оцінки продуктивності клієнтської частини вебзастосунку є визначення її швидкодії, стабільності та ефективності роботи в умовах навантаження, великого обсягу даних або складних сценаріїв використання. Це, зокрема, включає перевірку роботи при фільтрації великих списків робочих місць, швидкому перемиканні між сторінками, а також взаємодії з формами бронювання.

Особливу увагу було приділено часу завантаження сторінки, швидкості реакції на дії користувача та плавності інтерфейсу під час типових сценаріїв роботи користувачів — зокрема, адміністраторів коворкінгів, менеджерів локацій і користувачів, які здійснюють бронювання.

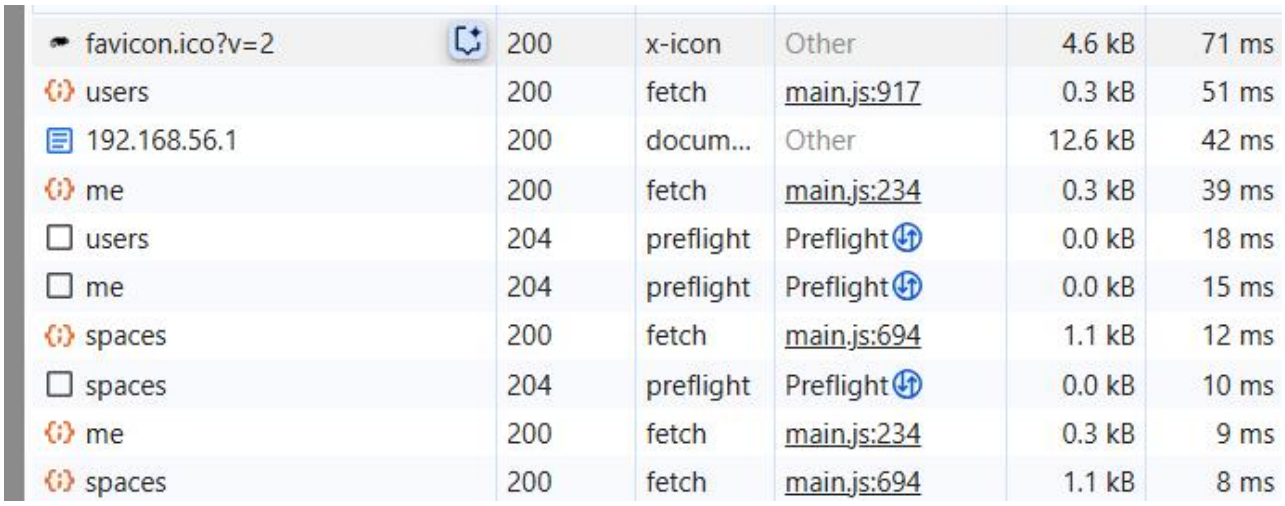
Для оцінки продуктивності було використано інструменти Chrome DevTools та Lighthouse.

Chrome DevTools — це вбудований у Google Chrome набір інструментів, що дозволяє розробникам відстежувати продуктивність, аналізувати запити до сервера, швидкість відображення компонентів, навантаження на DOM та інші важливі показники [17]. Було задіяно вкладки Performance та Network. За допомогою Network встановлено, що всі основні елементи інтерфейсу (включаючи зображення та стилі) завантажуються оперативно, без затримок.

Показник часу завантаження елементів (колонка Time) не перевищував 1000 мс, що свідчить про хорошу оптимізацію завантаження.

Інструмент Performance показав, що найскладніші компоненти сторінки рендеряться в межах 0.51 секунди, що є відмінним результатом для інтерактивного інтерфейсу. Також було зафіксовано невеликий сукупний зсув макету (Cumulative Layout Shift = 0,24), але він виникає внаслідок реакції навігації на прокрутку сторінки, яка рухається разом з прокрутом сторінки. Загалом дані результати вказують на стабільне й прогнозоване відображення сторінки без зрушення важливих елементів під час завантаження. Це позитивно впливає на користувацький досвід, зменшуючи ймовірність випадкових натискань або візуального дискомфорту.

Результат тестування зображено на рисунку 4.1



favicon.ico?v=2	200	x-icon	Other	4.6 kB	71 ms
users	200	fetch	main.js:917	0.3 kB	51 ms
192.168.56.1	200	docum...	Other	12.6 kB	42 ms
me	200	fetch	main.js:234	0.3 kB	39 ms
users	204	preflight	Preflight	0.0 kB	18 ms
me	204	preflight	Preflight	0.0 kB	15 ms
spaces	200	fetch	main.js:694	1.1 kB	12 ms
spaces	204	preflight	Preflight	0.0 kB	10 ms
me	200	fetch	main.js:234	0.3 kB	9 ms
spaces	200	fetch	main.js:694	1.1 kB	8 ms

Рисунок 4.1 — Результати тестування часу завантаження елементів в Network

Lighthouse — автоматизований інструмент аудиту якості вебсторінок, який оцінює продуктивність, доступність, дотримання стандартів PWA, оптимізацію під пошукові системи (SEO) тощо [18]. Під час тестування головної сторінки вебзастосунку, що містить інтерактивну карту з позначеними робочими місцями, було отримано оцінку 82 бала за продуктивність. Цей результат знаходиться в діапазоні 50–89, що вважається задовільним з огляду на складність інтерфейсу та обсяг візуального контенту.

Зниження балів пояснюється присутністю зміною макета на сторінці, коли навігаційне меню рухається разом з прокручуванням сторінки, що потребує більшого обсягу даних і ресурсів для завантаження, однак в загальному система продемонструвала стабільну та ефективну роботу. Це свідчить про відповідність клієнтської частини сучасним вимогам до продуктивності вебзастосунків.

У вебзастосунку для бронювання робочих місць у коворкінгах великі компоненти, такі як сторінка з аналітикою чи зображення коворкінгів, завантажуються лише за потреби, що зменшує початкове навантаження на клієнта. Також ці зображення кешуються, що ще більше зменшує час на їхнє завантаження. Усі мережеві запити реалізовані через axiosInstance, який містить централізовану обробку помилок, що дозволяє забезпечити єдину логіку обробки збоїв і

повідомлень для користувача.

Результати тестування в Lighthouse показано на рисунку 4.2.

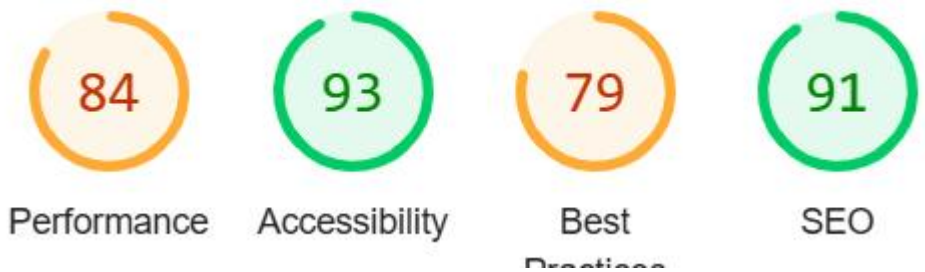


Рисунок 4.2 — Результати тестування інструменту часу завантаження елементів в Performance

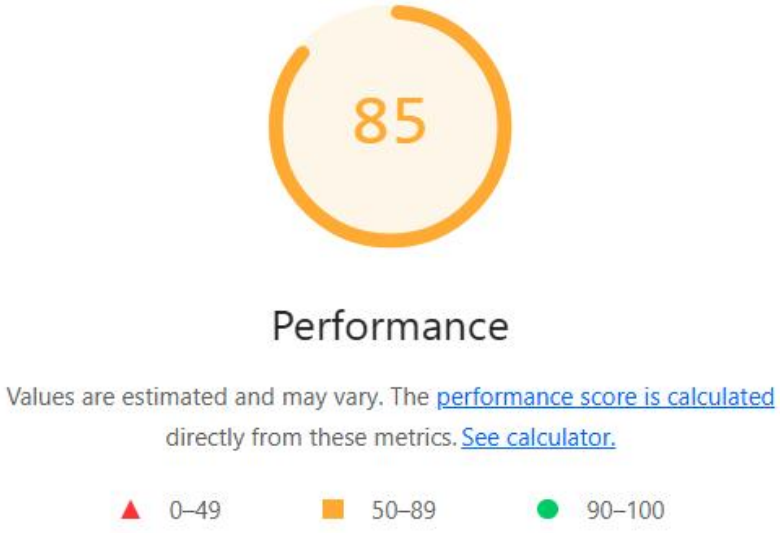


Рисунок 4.3 — Результати тестування головної сторінки

Під час аудиту вебзастосунку за допомогою інструменту Lighthouse було виявлено, що показник у категорії Best Practices становить 79 балів. Основною причиною зниження цього показника є наявність сторонніх cookie-файлів, які встановлюються через зовнішні зображення, зокрема через логотип платіжної системи Visa у форматі PNG. Зображення visa.png, що розміщене у футері сайту для позначення підтримуваних способів оплати, завантажується з зовнішнього ресурсу cdn.payment-logos.com.

Під час завантаження цього логотипу браузер користувача встановлює cookie з цього стороннього домену. Це є потенційною загрозою з точки зору

приватності користувачів, адже такі cookie можуть бути використані для відстеження поведінки користувача без його згоди. Lighthouse інтерпретує це як порушення сучасних вебпрактик, зокрема щодо безпечної взаємодії зі сторонніми ресурсами, що й призводить до зниження оцінки у відповідній категорії.

Щоб усунути проблему та підвищити рейтинг у Lighthouse, було замінено стороннє зображення visa.png на його локальну копію, збережену та обслуговувану з власного сервера. Це дозволило уникнути встановлення сторонніх cookie та покращило як безпеку, так і довіру користувачів до вебзастосунку.

Після усунення проблеми зі сторонніми cookie-файлами, які встановлювалися під час завантаження зовнішнього логотипу visa.png, показник у категорії Best Practices в інструменті Lighthouse значно покращився. Замість підключення зображення з зовнішнього ресурсу, яке ініціювало небажане зберігання cookie, було впроваджено локальну версію цього логотипу, що розміщується безпосередньо на сервері вебзастосунку.

Ця зміна дозволила усунути небезпечну взаємодію з третім джерелом і уникнути автоматичного встановлення сторонніх cookie без згоди користувача. Як результат, під час повторного тестування, показник Best Practices зріс з 78 до 96 балів, як показано на рисунку 4.4. Це свідчить про набагато вищу відповідність сучасним вимогам безпеки, приватності та якості реалізації вебзастосунку.

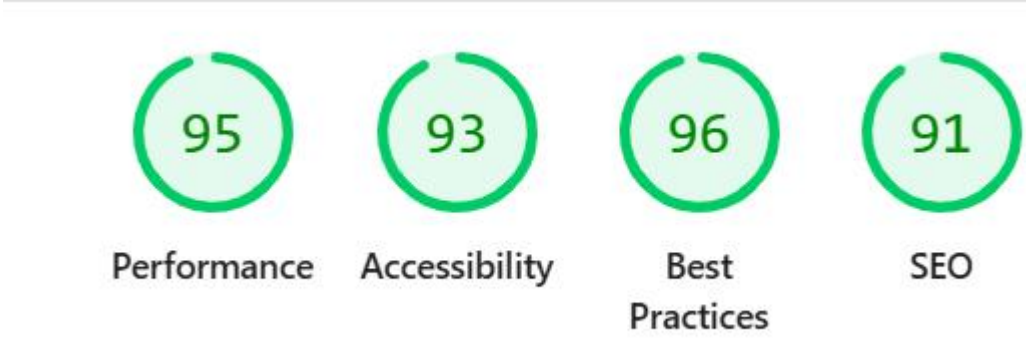


Рисунок 4.4 — Результати тестування головної сторінки після виключення використання сторонніх cookie

Завдяки цьому покращенню система стала не лише технічно надійнішою,

але й більш прозорою для користувачів, що особливо важливо для сервісів, які працюють з особистими даними або фінансовими операціями. Це також позитивно впливає на довіру до платформи з боку користувачів та потенційних партнерів.

У рамках оптимізації роботи вебзастосунку реалізовано кешування деяких результатів запитів на стороні клієнта. Зокрема, це стосується запиту до списку доступних коворкінгів, який завантажується під час першого відвідування відповідного розділу або сторінки. Збереження цих даних у клієнтському кеші (наприклад, у стані програми або локальному сховищі браузера) дозволяє уникнути повторного звернення до серверу при кожному перегляді сторінки або переході між підрозділами інтерфейсу, що суттєво зменшує навантаження на мережу та підвищує швидкість відгуку інтерфейсу.

Кешовані дані залишаються актуальними доти, доки користувач не змінює параметри фільтрації — наприклад, місто, кількість місць або наявність додаткових послуг — або доки не відбувається примусове оновлення контенту. У таких випадках виконується новий запит до серверу, щоб отримати найсвіжішу інформацію. Такий підхід дозволяє досягти балансу між ефективністю роботи застосунку та актуальністю відображуваних даних, забезпечуючи комфортне користувацьке враження навіть при повільному інтернет-з’єднанні.

Крім того, це рішення сприяє економії серверних ресурсів і зменшує кількість дубльованих запитів, особливо у випадках, коли багато користувачів одночасно переглядають популярні коворкінги. У перспективі кешування може бути доповнене інтелектуальним механізмом інвалідації або синхронізацією з сервером за допомогою WebSocket або технологій типу stale-while-revalidate, щоб ще точніше підтримувати актуальність інформації без шкоди для продуктивності.

У результаті проведеної оптимізації клієнтська частина вебзастосунку демонструє високу стабільність і надійність роботи навіть на пристроях із середніми або низькими технічними характеристиками, такими як менш потужні процесори, обмежений обсяг оперативної пам’яті або повільніше інтернет-

з’єднання. Завдяки впровадженим заходам оптимізації, таким як зменшення обсягу завантажуваних ресурсів, асинхронне завантаження даних та кешування, застосунок забезпечує швидке завантаження основних сторінок. Це значно покращує користувацький досвід, оскільки користувачі можуть миттєво переходити між розділами без довгих затримок.

Плавна навігація між інтерфейсними елементами і розділами дозволяє уникнути “зависань” або раптових фризів, що особливо важливо при взаємодії на мобільних пристроях або в умовах нестабільного з’єднання. Мінімальні затримки під час запитів до серверної частини гарантують, що користувачі отримують актуальну інформацію та швидкий відгук на свої дії, що, у свою чергу, підвищує загальну ефективність і привабливість застосунку. Такий підхід до оптимізації позитивно впливає на продуктивність, доступність і задоволеність користувачів, що є ключовими факторами успішності вебзастосунку.

4.3 Розгортання розробленого вебзастосунку для бронювання робочих місць у коворкінгах

Система вебзастосунку для бронювання робочих місць у коворкінгах була розгорнута у тестовому режимі на платформах Vercel. Було змодельовано повноцінний цикл взаємодії користувача із системою: вибір і бронювання робочого місця, зміна бронювання, його відміна. А також зміна його опису власником коворкінгу.

Клієнтська частина застосунку розгорнута на платформі Vercel, яка забезпечує безкоштовне розгортання додатків. Vercel також надає швидке кешування статичних ресурсів, підтримку HTTPS за замовчуванням і гарантує стабільну та надійну роботу інтерфейсу користувача.

Серверна частина розгорнута на хмарній платформі Heroku.com, що надає середовище для розміщення Express-серверів із підтримкою автоматичного перезапуску та логування. Водночас безкоштовний тарифний план Vercel накладає певні обмеження — ліміт у 500 МБ оперативної пам’яті та зупинку сервера після 15 хвилин бездіяльності, що може спричиняти невеликі затримки

під час першого запиту після простою. Це змусило оптимізувати використання ресурсів для забезпечення безперервної роботи застосунку.

Під час тестування встановлено, що раніше процес бронювання робочого місця міг займати значний час через ручне оформлення та реєстрацію інформації, тоді як у новій системі він скорочений до декількох секунд із повною автоматизацією. Аналогічно, пошук вільних місць або інформації про конкретне робоче місце тепер виконується майже миттєво, що підвищує зручність використання. Система також забезпечує мобільний доступ — користувачі можуть швидко переглядати бронювання і вносити зміни зі смартфонів, без необхідності звертатись до адміністрації.

Для зберігання даних у застосунку використовується lowDB — легковага файлова база даних, що зберігає інформацію у форматі JSON. Для забезпечення надійності та доступності дані синхронізуються з хмарним сховищем, що дозволяє централізовано зберігати актуальні файли бази та забезпечує резервне копіювання. Через обмеження розміру файлу в lowDB необхідно регулярно контролювати обсяг даних і видаляти застарілі записи для підтримки оптимальної роботи системи.

Для забезпечення високої продуктивності застосунку реалізовано оптимізацію роботи з файлами бази даних, обмеження розміру завантажуваних файлів, а також кешування відповідей сервера. Це дозволяє значно скоротити час обробки запитів і покращити швидкодію системи навіть при великій кількості користувачів.

Впровадження системи суттєво знизило кількість помилок, пов’язаних із дублюванням або втратами даних, оскільки вся інформація централізовано зберігається у базі даних із вбудованими механізмами валідації. Крім того, інтеграція з Google Drive дозволяє надійно зберігати та оперативно отримувати всі супровідні документи, що підвищує прозорість і контроль за процесом бронювання.

Усі дані про користувачів, забронювані робочі місця та історію змін об’єднані в єдину інформаційну систему, яка дає змогу аналізувати заповненість коворкінгу, виявляти активних користувачів і контролювати дотримання правил.

Вебзастосунок розміщено на платформі Vercel, а посилання на нього інтегровано на офіційному сайті коворкінгу, що забезпечує легкий доступ для користувачів. Завдяки цьому забезпечується зручний перехід на сторінку бронювання без зайвих кроків.

Отже, впровадження вебзастосунку створює умови для якісної цифрової трансформації процесу управління бронюванням робочих місць у коворкінгах, знижує навантаження на адміністрацію, зменшує кількість помилок, пришвидшує обробку замовлень та робить всі необхідні дані доступними в режимі онлайн.

Перспективи подальшого розвитку включають інтеграцію із зовнішніми системами управління, розширення аналітичних можливостей для керівників коворкінгів, масштабування на інші локації та створення мобільного застосунку на базі React Native для сповіщень та зручнішого доступу користувачів.

Оцінка продуктивності системи підтвердила високу швидкодію та стабільність роботи завдяки застосуванню оптимізацій як на клієнтській частині, так і при обробці серверних запитів. Інтеграція з Google Drive забезпечує централізоване, безпечне зберігання супровідних даних, що підвищує прозорість бронювань.

Отже, на підставі проведеного етапу тестування можна впевнено констатувати успішну реалізацію ключових функціональних та технічних аспектів вебзастосунку для бронювання робочих місць у коворкінгах. В ході тестування було перевірено всі основні сценарії взаємодії користувача із системою — від реєстрації й авторизації до вибору, бронювання, зміни та скасування робочих місць. Кожен із цих етапів пройшов без збоїв, що свідчить про стабільність роботи як клієнтської, так і серверної частини застосунку.

Крім того, результати навантажувального тестування показали, що система здатна ефективно обробляти одночасні запити від великої кількості користувачів без істотного погіршення швидкодії чи затримок. Впроваджені оптимізації — як

на рівні запитів до бази даних, так і в роботі інтерфейсу — дозволили досягти високої продуктивності, що є важливим для забезпечення комфортного користувацького досвіду в реальних умовах інтенсивного використання.

Також успішно пройшли перевірку механізми безпеки, зокрема захист від дублювання бронювань, контроль коректності введених даних, а також централізоване зберігання інформації з резервним копіюванням, що гарантує збереження даних навіть у разі несподіваних збоїв або втрати зв’язку.

Інтеграція із зовнішніми сервісами, зокрема хмарним сховищем була реалізована коректно, що додатково підвищує надійність і зручність використання системи. Завдяки цьому користувачі мають постійний доступ до актуальної інформації, а керівники можуть оперативно контролювати та аналізувати стан бронювань і активність користувачів.

Таким чином, всі проведені випробування підтвердили, що розроблений вебзастосунок є готовим до впровадження і використання у реальних умовах коворкінг-просторів. Його функціональність, продуктивність і безпека відповідають сучасним вимогам, що дозволяє очікувати позитивний вплив на організацію робочого простору, підвищення ефективності управління бронюваннями та покращення загального користувацького досвіду.

ВИСНОВКИ

Цю бакалаврську кваліфікаційну роботу було присвячено створенню та впровадженню вебзастосунку для бронювання робочих місць у коворкінгах. Основна мета полягала в розробці інтерактивної системи, яка спрощує процес вибору, бронювання та управління робочими місцями, а також забезпечує комфортну взаємодію користувача із системою. Розроблений вебзастосунок орієнтований на клієнтів, які шукають зручні умови для роботи, і дозволяє швидко знайти потрібне місце, здійснити пошук, налаштувати фільтри, додати місця до обраних та оформити бронювання.

У процесі роботи було проаналізовано сучасні вебтехнології, які дозволяють створювати адаптивні, зручні та функціональні вебзастосунки. Для реалізації клієнтської частини використовувались HTML, CSS і JavaScript. Інструменти стилізації, зокрема Flexbox і Grid, забезпечили побудову чіткого, інтуїтивного і привабливого інтерфейсу. Було враховано принципи UX/UI-дизайну, що сприяє зручності використання системи. Додатково було застосовано адаптивний підхід до дизайну, що дозволяє коректно відображати інтерфейс на настільних ПК, планшетах і смартфонах.

У межах реалізації було створено логічну структуру вебзастосунку, яка включає головну сторінку, каталог робочих місць, модуль пошуку, фільтрацію, форму бронювання та список обраних місць. Особливу увагу приділено реалізації персоналізованого функціоналу, як-от модуль рекомендацій, який показує клієнту варіанти місць на основі його попередньої активності. Такий підхід підвищує зручність використання та сприяє зростанню зацікавленості.

Додатково було реалізовано елементи збереження стану, як-от збереження обраних місць, параметрів пошуку та фільтрів у `localStorage`, що забезпечує безперервність роботи користувача навіть після перезавантаження сторінки. Також були застосовані заходи для оптимізації продуктивності: ліниве завантаження зображень, мінімізація JavaScript та CSS, а також кешування даних для швидшого відгуку системи.

Усі поставлені завдання були успішно виконані. Розроблений вебзастосунок

відповідає функціональним, візуальним і структурним вимогам, визначеним на етапі постановки задачі. Система може бути використана як основа для реального коворкінгу з можливістю подальшого розвитку, інтеграції з серверною частиною, базами даних, платіжними системами та іншими зовнішніми сервісами. Перспективи масштабування включають впровадження додаткових функцій, таких як аналітика поведінки користувачів та інтеграція з мобільними додатками, що зробить систему ще більш зручною та ефективною.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Гороховський О.І., Азаров О.Д., Трояновська Т.І. Інформаційна технологія доставки контенту у системі комп'ютеризованої підготовки спеціалістів: монографія. Вінниця: ВНТУ, 2016. 160 с.

2. Етапи створення вебзастосунків. URL: https://my-master.net.ua/ua/etapi-stvorennya-sajtu-z-nulya/

3. Методи розробки вебзастосунків. URL: https://webstudio2u.net/ua/webdesign/354-site-developmethods.html

4. AllBooked. URL: https://www.allbooked.com/

5. Deskpass. URL: https://www.deskpass.com/

6. LiquidSpace. URL: https://liquidspace.com/

7. CSS Documentation. URL: https://developer.mozilla.org/en-US/docs/Web/CSS

8. HTML Documentation. URL: https://developer.mozilla.org/en-US/docs/Web/HTML

9. JavaScript Documentation. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript

10. Node.js. URL: https://nodejs.org/en

11. lowDB Documentation. URL: https://www.npmjs.com/package/lowdb/v/2.1.0

12. Git. URL: https://git-scm.com/

13. Адаптивний дизайн: основи та принципи. URL: https://uxplanet.org/adaptive-design-principles

14. Хмарні сервіси для вебзастосунків: AWS. URL: https://aws.amazon.com/

15. Типи архітектури ПЗ. URL: https://www.artofba.com/uk/post/main-types-of-software-architecture

16. Основні принципи UI/UX дизайну. URL: https://blog.ithillel.ua/articles/basic-

principles-of-visual-design-in-ux

17. ChromDevTools. URL: https://dou.ua/lenta/articles/chrome-dev-tools-guide/

18. Introduction to Chrome Lighthouse. URL: https://www.freecodecamp.org/news/introduction-to-chrome-lighthouse/

ДОДАТОК А

Технічне завдання

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки

ЗАТВЕРДЖУЮ
Завідувач кафедри ОТ
проф., д.т.н.. Азаров О.Д.
21 березня 2025 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання бакалаврської кваліфікаційної роботи

«Комп'ютерна онлайн система обліку, рейтингування та контролю проживання студентів у гуртожитку»

08-54.БКР.013.00.000 ТЗ

Науковий керівник: д. ф., ст. викл.
_____ Обертюх М.Р.

Студент групи 1КІ-23мс
_____ Шевчук В.В.

м. Вінниця — 2025

1 Підстава для використання бакалаврської кваліфікаційної роботи (БКР)

1.1У сучасному світі, де цифровізації освітніх процесів приділяють особливу увагу, важливим є використанням нових технологій. Однією з ключових сфер, яка потребує систематизації та оптимізації, є управління проживанням студентів у гуртожитках.

1.2 Наказ про затвердження теми кваліфікаційної роботи.

2 Мета БКР і призначення розробки

2.1 Мета роботи – розширення функціональних можливостей наявної системи обліку для гуртожитку шляхом створення комп'ютерної онлайн системи, яка забезпечить можливість формування рейтингу, контролю дисциплінарних заходів, ведення звітності за окремим студентом та збереження супровідних документів.

2.2 Призначення розробки — онлайн система обліку, рейтингування та контролю проживання студентів у гуртожитку.

3 Вихідні дані для виконання БКР

3.1 Проведення аналізу існуючих систем та технологій обліку, рейтингування та контролю проживання студентів у гуртожитку;

3.2 Розробка структури та функціональної схеми програмно- апаратного комплексу для онлайн системи обліку, рейтингування та контролю проживання студентів у гуртожитку;

3.3 Розробка клієнтської частини веб-додатку.

3.4 Розробка серверної частини веб-додатку.

3.5 Тестування клієнтської частини веб- додатку.

3.6 Впровадження онлайн системи контролю проживання студентів у гуртожитках;

4 Вимоги до виконання БКР

Головна вимога – створити систему, яка матиме практичне застосування в

контролі проживання студентів у гуртожитках.

5 Етапи БКР та очікувані результати

Етапи роботи та очікувані результати приведено в Таблиці А.1.

Таблиця А.1 — Етапи БКР

№ етапу	Назва етапу	Термін виконання		Очікувані результати
		початок	кінець	
1	Огляд і аналіз сучасних аналогів вебдодатків для бронювання робочих місць у коворкінгах	21.03.25	30.03.25	Аналітичний огляд літературних джерел
2	Проєктування вебдодатку для бронювання робочих місць у коворкінгах	21.03.25	30.03.25	Розділ 2
3	Програмна реалізація вебдодатку для бронювання робочих місць у коворкінгах	31.03.25	13.04.25	Розділ 3
4	Тестування вебдодатку для бронювання робочих місць у коворкінгах	14.04.25	27.04.25	Розділ 4
5	Апробація результатів дослідження	28.04.25	11.05.2025	Тези доповідей
6	Оформлення пояснювальної записки, графічного матеріалу і презентації	28.04.25	11.05.2025	пояснювальна записка, графічний матеріал і/або презентація

6 Матеріали, що подаються до захисту БКР

До захисту подаються: пояснювальна записка БКР, графічні і ілюстративні матеріали, протокол попереднього захисту БКР на кафедрі, відгук наукового керівника, відгук опонента, протоколи складання державних екзаменів, анотації до БКР українською та іноземною мовами, довідка про відповідність оформлення БКР діючим вимогам.

7 Порядок контролю виконання та захисту БКР

Виконання етапів графічної та розрахункової документації БКР

контролюється науковим керівником згідно зі встановленими термінами. Захист БКР відбувається на засіданні Екзаменаційної комісії, затвердженої наказом ректора.

8 Вимоги до оформлювання та порядок виконання БКР

8.1 При оформлюванні БКР використовуються:

— ДСТУ 3008 : 2015 «Звіти в сфері науки і техніки. Структура та правила оформлювання»;

— ДСТУ 8302 : 2015 «Бібліографічні посилання. Загальні положення та правила складання»;

— методичні вказівки до виконання бакалаврських кваліфікаційних робіт для студентів спеціальності 123 «Комп'ютерна інженерія» (освітня програма «Комп'ютерна інженерія») — кафедра обчислювальної техніки, ВНТУ 2023.

8.2 Порядок виконання БКР викладено в «Положення про кваліфікаційні роботи на першому (бакалаврському) рівні вищої освіти СУЯ ВНТУ-03.02.02-П.001.01:21»

ДОДАТОК Б

ПРОТОКОЛ
ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи:
Вебзастосунок для бронювання робочих місць у коворкінгах

Тип роботи: Бакалаврська кваліфікаційна робота
(БДР,МКР)

Підрозділ кафедра обчислювальної техніки
(кафедра,факультет)

Показники звіту подібності StrikePlagiarism

Оригінальність 97,5 Схожість 2,5%

Аналіз звіту подібності (відмітити потрібне):

✓ Запозичення, виявлення у роботі, оформлені коректно і не містять ознак плагіату.
☐ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.
☐ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ Захарченко С.М.
(підпис) (прізвище,ініціали)

Ознайомлені з повним звітом подібності, який був згенерований системою StrikePlagiarism щодо роботи.

Автор роботи _____ Шевчук В.В.
(підпис) (прізвище,ініціали)

Керівник роботи _____ Обертюх М.Р.
(підпис) (прізвище,ініціали)

ДОДАТОК Г

ІЛЮСТРАТИВНА ЧАСТИНА

ВЕБЗАСТОСУНОК ДЛЯ БРОНЮВАННЯ РОБОЧИХ МІСЦЬ У КОВОРКІНГАХ

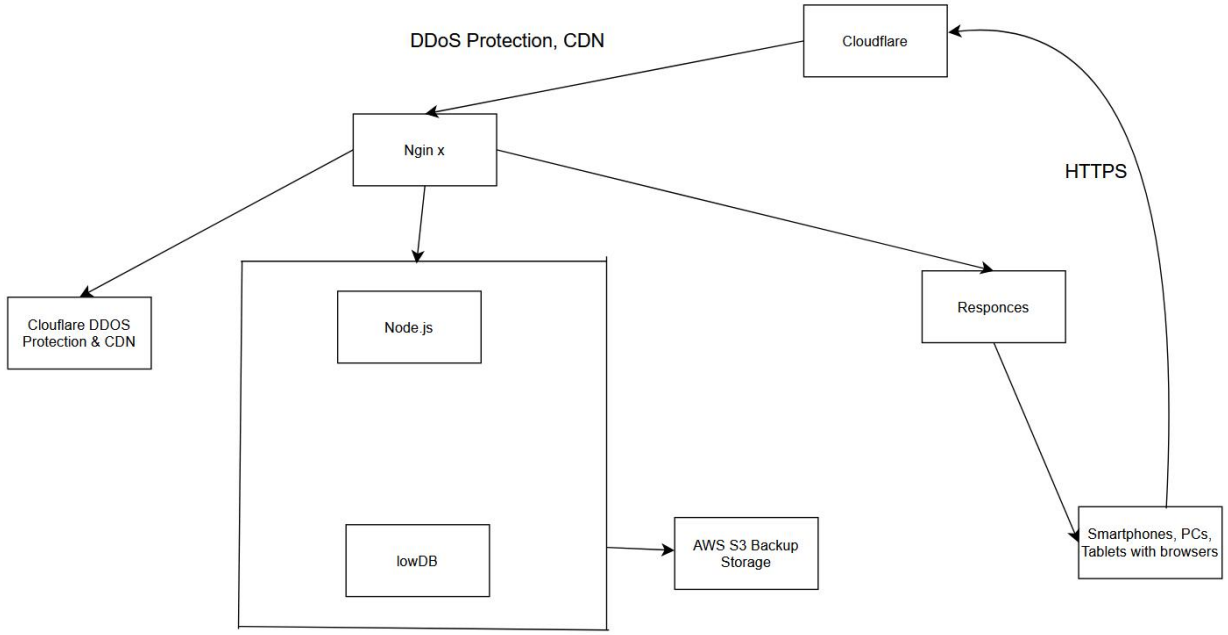


Рисунок В1 — Схема технічного середовища інформаційної системи

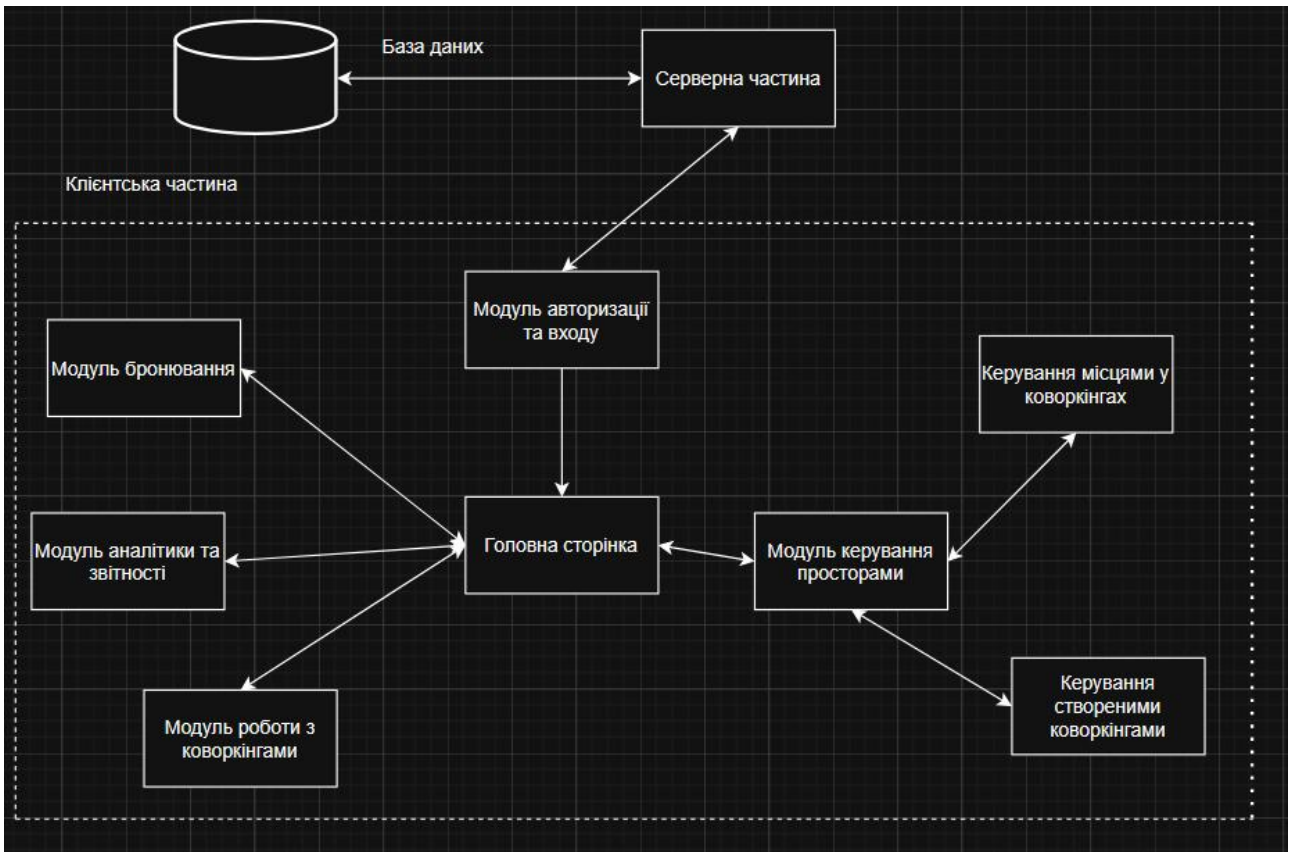


Рисунок В2— Загальна схема взаємодії користувачів з інформаційною системою

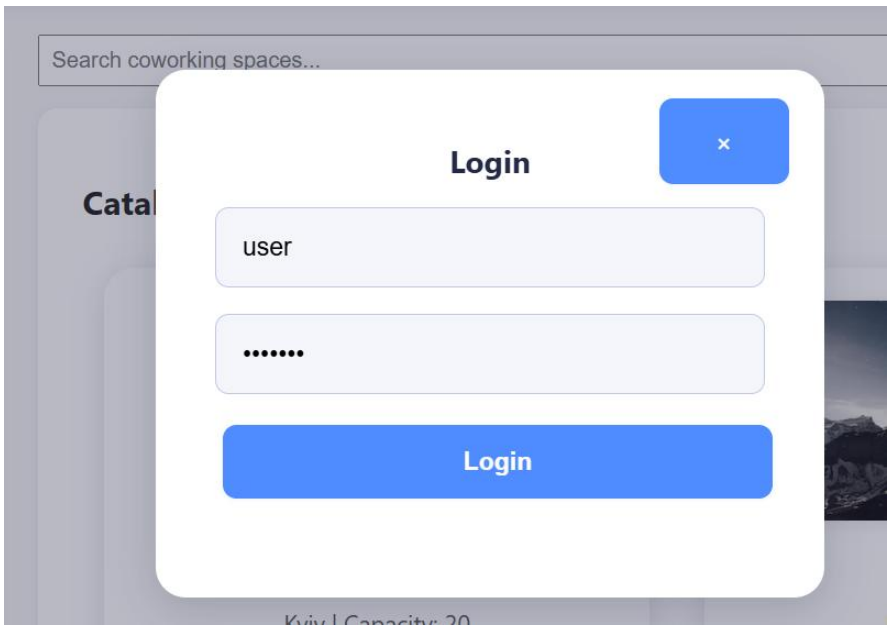


Рисунок В.3 Графічний інтерфейс входу в програмний модуль реєстрації для бронювання коворкнгів

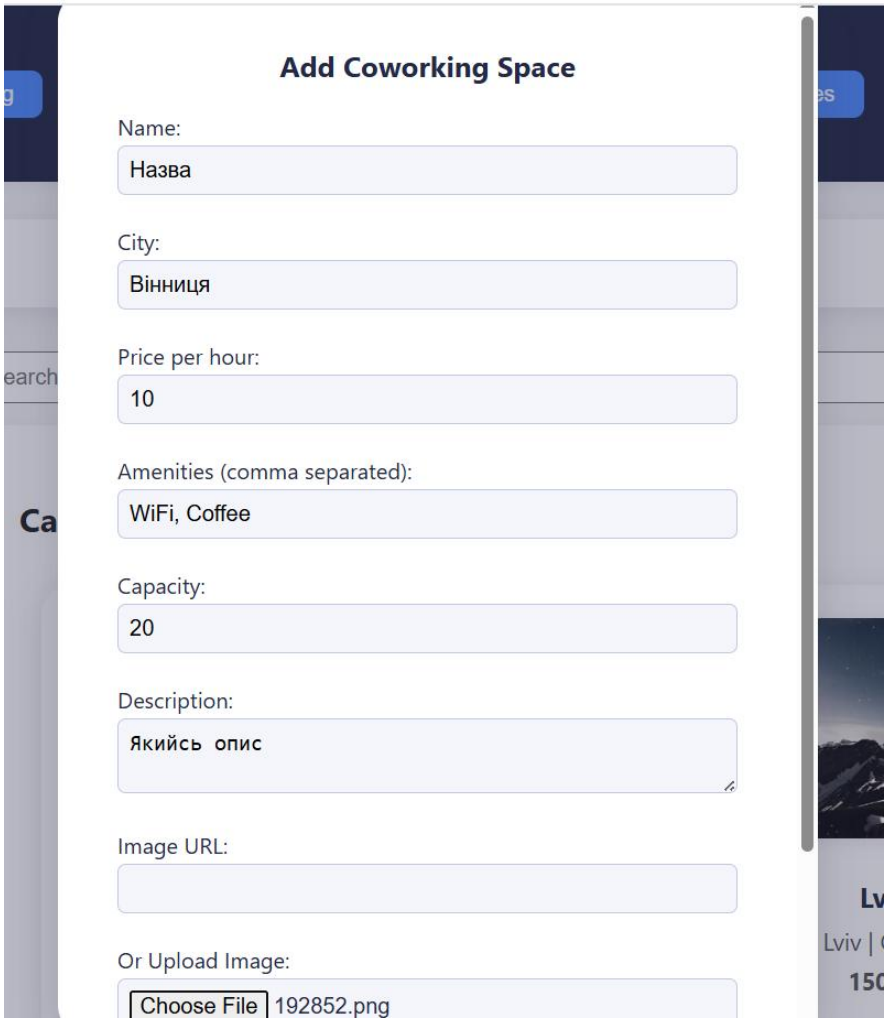


Рисунок В.4 Графічний інтерфейс форми створення реєстрації місць у коворкінгу

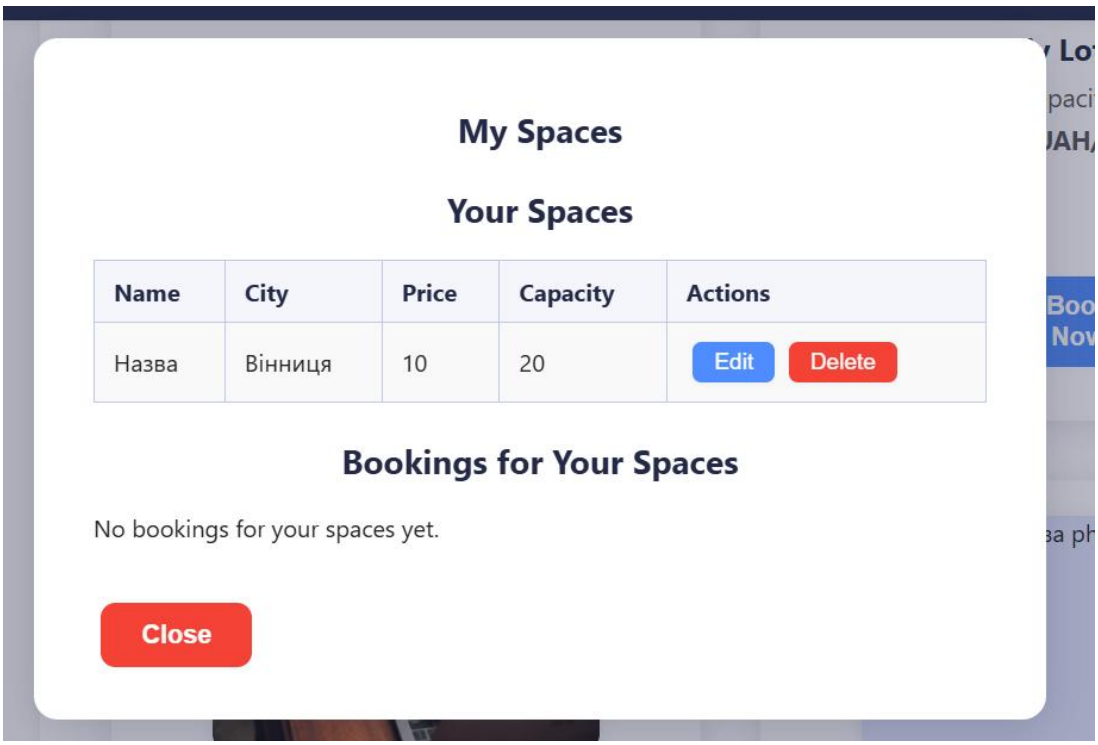


Рисунок В.5 Графічний інтерфейс форми керування створеним коворкінгом

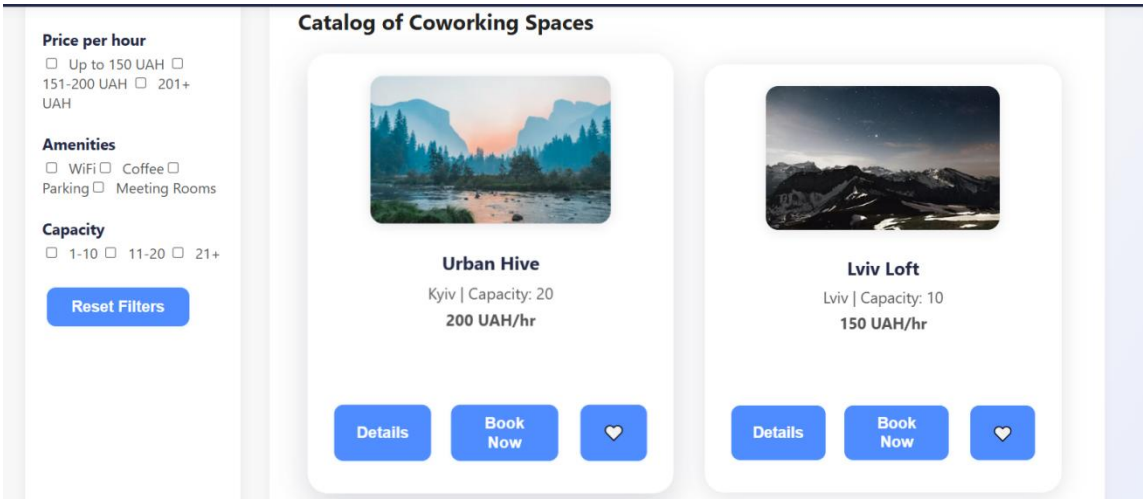


Рисунок В.6 Графічний інтерфейс сторінки «Каталог робочих місць у коворкінгах»

ДОДАТОК Д

Лістинг програмного коду файлу HomePage.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Coworking Space Booking</title>
  <link rel="stylesheet" href="styles.css">
  <link rel="icon" type="image/x-icon" href="favicon.ico?v=2">
</head>
<body>
  <header>
    <h1>Coworking Space Booking</h1>
    <nav>
      <button onclick="navigate('catalog')">Catalog</button>
      <button onclick="toggleFavorites()">Favorites</button>
      <button onclick="navigate('my-bookings')">My Bookings</button>
      <button id="add-space-btn" class="hidden" onclick="showAddSpaceModal()">Add Space</button>
      <button id="admin-panel-btn" class="hidden" onclick="showAdminPanel()">Admin Panel</button>
      <button id="my-spaces-btn" class="hidden" onclick="showMySpaces()">My Spaces</button>
      <button onclick="showAboutModal()" id="about-btn">About</button>
      <span class="nav-spacer"></span>
      <span id="nav-user-info" class="hidden"></span>
      <button id="show-login-btn" type="button" onclick="showLogin()">Login</button>
      <button id="show-register-btn" type="button" onclick="showRegister()">Register</button>
    </nav>
  </header>
  <nav id="breadcrumbs-nav" aria-label="Breadcrumb"></nav>
  <div id="main">
    <aside id="filter"></aside>
    <section style="width:100%">
      <div style="margin-bottom:1rem;">
        <input type="text" id="search-box" placeholder="Search coworking spaces..." style="width:100%;padding:0.5rem;font-size:1rem;">
      </div>
      <div id="content"></div>
    </section>
  </div>
  <div id="favorites-modal" class="modal hidden" role="dialog" aria-modal="true" aria-labelledby="favorites-modal-title">
    <div class="modal-content">
      <span class="close" onclick="toggleFavorites()" aria-label="Close Favorites Modal">&times;</span>
      <h2 id="favorites-modal-title">Favorites</h2>
      <ul id="favorites-items"></ul>
    </div>
  </div>
```

```
  <div id="toast" aria-live="polite" aria-atomic="true"></div>
  <div id="payment-modal" class="modal hidden" role="dialog" aria-modal="true" aria-labelledby="payment-modal-title">
    <div class="modal-content stripe-payment">
      <span class="close" onclick="closePaymentModal()" aria-label="Close Payment Modal">&times;</span>
      <h2 id="payment-modal-title">Payment</h2>
      <form id="payment-form" aria-label="Payment form">
        <div class="stripe-card-icons" aria-label="Accepted card brands">
          <img src="https://upload.wikimedia.org/wikipedia/commons/4/41/Visa_Logo.png" alt="Visa" style="height:32px;width:auto;margin-right:1.1em;vertical-align:middle;filter:drop-shadow(0 1px 2px #bfc8e6);background:#fff;border-radius:6px;">
          <img src="https://upload.wikimedia.org/wikipedia/commons/0/04/Mastercard-logo.png" alt="Mastercard" style="height:32px;width:auto;margin-right:1.1em;vertical-align:middle;filter:drop-shadow(0 1px 2px #bfc8e6);background:#fff;border-radius:6px;">
        </div>
        <label>Card Number: <input type="text" name="card" maxlength="19" placeholder="1234 5678 9012 3456" required autocomplete="cc-number" aria-label="Card Number"></label>
        <label>Expiry: <input type="text" name="expiry" maxlength="5" placeholder="MM/YY" required autocomplete="cc-exp" aria-label="Expiry"></label>
        <label>CVC: <input type="text" name="cvc" maxlength="4" placeholder="123" required autocomplete="cc-csc" aria-label="CVC"></label>
        <button type="submit" aria-label="Pay">Pay</button>
        <button type="button" class="cancel" onclick="closePaymentModal()" aria-label="Cancel Payment">Cancel</button>
      </form>
      <div id="payment-confirmation" class="hidden" style="text-align:center;">
        <p style="font-size:1.15em;">Are you sure you want to pay for this booking?</p>
        <button id="confirm-payment-btn" style="margin-right:1em;" aria-label="Confirm Payment">Confirm Payment</button>
        <button class="cancel" onclick="closePaymentModal()" aria-label="Cancel Payment">Cancel</button>
      </div>
    </div>
  </div>
  <div id="add-space-modal" class="modal hidden" role="dialog" aria-modal="true" aria-labelledby="add-space-modal-title">
    <div class="modal-content">
      <h2 id="add-space-modal-title">Add Coworking Space</h2>
      <form id="add-space-form" aria-label="Add Coworking Space form">
        <label>Name: <input type="text" name="name" required aria-label="Space Name"></label>
        <label>City: <input type="text" name="city" required aria-label="City"></label>
        <label>Price per hour: <input type="number" name="price" min="1" required aria-label="Price per hour"></label>
        <label>Amenities (comma separated): <input type="text" name="amenities" aria-label="Amenities"></label>
        <label>Capacity: <input type="number" name="capacity" min="1" required aria-label="Capacity"></label>
        <label>Description: <textarea name="description" rows="2" aria-label="Description"></textarea></label>
        <label>Image URL: <input type="text" name="image" aria-label="Image URL"></label>
```

```
        <label>Or Upload Image: <input type="file" name="imageFile" accept="image/*" aria-label="Upload Image"></label>
        <button type="submit" aria-label="Add Space">Add Space</button>
        <button type="button" class="cancel" onclick="closeAddSpaceModal()" aria-label="Cancel">Cancel</button>
      </form>
    </div>
  </div>
  <div id="admin-panel-modal" class="modal hidden" role="dialog" aria-modal="true" aria-labelledby="admin-panel-modal-title">
    <div class="modal-content" style="min-width: 600px; max-width: 98vw;">
      <h2 id="admin-panel-modal-title">Admin Panel</h2>
      <div id="admin-panel-tabs">
        <button onclick="showAdminTab('users')" id="admin-tab-users" aria-label="Users Tab">Users</button>
        <button onclick="showAdminTab('spaces')" id="admin-tab-spaces" aria-label="Spaces Tab">Spaces</button>
        <button onclick="showAdminTab('bookings')" id="admin-tab-bookings" aria-label="Bookings Tab">Bookings</button>
        <button style="float:right;" class="cancel" onclick="closeAdminPanel()" aria-label="Close Admin Panel">Close</button>
      </div>
      <div id="admin-panel-content" style="margin-top:1.5rem;"></div>
    </div>
  </div>
  <div id="my-spaces-modal" class="modal hidden" role="dialog" aria-modal="true" aria-labelledby="my-spaces-modal-title">
    <div class="modal-content" style="min-width: 600px; max-width: 98vw;">
      <h2 id="my-spaces-modal-title">My Spaces</h2>
      <div id="my-spaces-content" style="margin-top:1.5rem;"></div>
      <button style="margin-top:1.5rem;" class="cancel" onclick="closeMySpaces()" aria-label="Close My Spaces">Close</button>
    </div>
  </div>
  <div id="edit-space-modal" class="modal hidden" role="dialog" aria-modal="true" aria-labelledby="edit-space-modal-title">
    <div class="modal-content" style="min-width: 400px; max-width: 95vw;">
      <h2 id="edit-space-modal-title">Edit Space</h2>
      <form id="edit-space-form" aria-label="Edit Space form">
        <input type="hidden" name="id">
        <label>Name: <input type="text" name="name" required aria-label="Space Name"></label>
        <label>City: <input type="text" name="city" required aria-label="City"></label>
        <label>Price per hour: <input type="number" name="price" min="1" required aria-label="Price per hour"></label>
        <label>Amenities (comma separated): <input type="text" name="amenities" aria-label="Amenities"></label>
        <label>Capacity: <input type="number" name="capacity" min="1" required aria-label="Capacity"></label>
        <label>Description: <textarea name="description" rows="2" aria-label="Description"></textarea></label>
        <label>Image URL: <input type="text" name="image" aria-label="Image URL"></label>
```

```
          <label>Or Upload Image: <input type="file" name="imageFile" accept="image/*" aria-label="Upload Image"></label>
          <button type="submit" aria-label="Save Changes">Save Changes</button>
          <button type="button" class="cancel" onclick="closeEditSpaceModal()" aria-label="Cancel">Cancel</button>
        </form>
      </div>
    </div>
    <div id="modal-overlay" class="modal-overlay hidden"></div>
    <!-- Add login modal -->
    <div id="login-modal" class="modal hidden" role="dialog" aria-modal="true" aria-labelledby="login-modal-title">
      <div class="modal-content" style="max-width: 400px;">
        <span class="close" onclick="closeLoginModal()" aria-label="Close Login Modal">&times;</span>
        <h3 id="login-modal-title">Login</h3>
        <form id="login-form" aria-label="Login form">
          <input type="text" name="username" placeholder="Username" required aria-label="Username">
          <input type="password" name="password" placeholder="Password" required aria-label="Password">
          <button type="submit" aria-label="Login">Login</button>
        </form>
        <div id="login-message"></div>
      </div>
    </div>
    <!-- Add register modal -->
    <div id="register-modal" class="modal hidden" role="dialog" aria-modal="true" aria-labelledby="register-modal-title">
      <div class="modal-content" style="max-width: 400px;">
        <span class="close" onclick="closeRegisterModal()" aria-label="Close Register Modal">&times;</span>
        <h3 id="register-modal-title">Register</h3>
        <form id="register-form" class="form-modern" aria-label="Register form">
          <div class="form-icon"><span>  </span></div>
          <input type="text" name="username" placeholder="Username" required aria-label="Username">
          <input type="password" name="password" placeholder="Password" required aria-label="Password">
          <button type="submit" aria-label="Register">Register</button>
        </form>
        <div id="register-message"></div>
      </div>
    </div>
    <div id="about-modal" class="modal hidden" role="dialog" aria-modal="true" aria-labelledby="about-modal-title">
      <div class="modal-content" style="max-width: 600px;">
        <span class="close" onclick="closeAboutModal()" aria-label="Close About Modal">&times;</span>
        <h2 id="about-modal-title">About This App</h2>
        <div style="font-size:1.13rem;line-height:1.7;color:#232946;">
          <p><b>Coworking Space Booking</b> is a modern, peer-to-peer marketplace for
```

```
discovering, listing, and booking coworking spaces.</p>
            <ul style="margin:1em 0 1em 1.2em;">
               <li>Find and book unique coworking spaces in your city.</li>
               <li>List your own space and manage bookings easily.</li>
               <li>Enjoy a beautiful, accessible, and user-friendly experience.</li>
               <li>All payments are simulated for demo purposes.</li>
            </ul>
            <p style="margin-top:1.5em;">This project was built as a showcase of modern web development best practices:<br>
            <span style="color:#4f8cff;">HTML</span>, <span style="color:#4f8cff;">CSS</span>, <span style="color:#4f8cff;">JavaScript</span>, <span style="color:#4f8cff;">Node.js</span>, and <span style="color:#4f8cff;">lowdb</span>.<br>
            <span style="font-size:1.1em;">Thank you for visiting!    </span></p>
         </div>
      </div>
   </div>
   <script src="main.js"></script>
</body>
</html>
```