Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет комп'ютерних наук
(повна назва)

Кафедра програмної інженерії
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

Програмна система для бронювання коворкінгів

(тема)

Виконала:
студентка 4 курсу, групи ПЗПІ-20-3

Мельник К.В.
(прізвище, ініціали)

Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Програмна інженерія
(повна назва освітньої програми)

Керівник доц. кафедри ПІ Побіженко І.О.
(посада, прізвище, ініціали)

Допускається до захисту
Зав. кафедри _____ З.В.Дудар
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет комп’ютерних наук
Кафедра програмної інженерії
Рівень вищої освіти перший (бакалаврський)
Спеціальність 121 – Інженерія програмного забезпечення
Тип програми Освітньо-професійна
Освітня програма Програмна Інженерія
(шифр і назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«___» _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентці Мельник Катерині Вадимовні
(прізвище, ім’я, по батькові)

1. Тема роботи Програмна система для бронювання коворкінгів
Затверджена наказом по університету від 20.05.2024р. № 471 Ст
2. Термін подання студентом роботи до екзаменаційної комісії 14.06.2024
3. Вихідні дані до роботи
Розробити програмну систему для бронювання коворкінгів, яка складатиметься з серверної та клієнтської частин, а також мобільного додатку. Для розробки використовуватимуться мови програмування JavaScript та TypeScript, програмна платформа Node.js та бібліотека React, а також фреймворк React Native.
4. Перелік питань, що потрібно опрацювати в роботі
Вступ, аналіз предметної галузі, формування вимог до програмної системи, архітектура та проектування програмного забезпечення, опис прийнятих програмних рішень, тестування розробленого програмного забезпечення, висновки, додатки.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз предметної галузі	15.04.2024	*виконано*
2	Створення специфікації ПЗ	18.05.2024	*виконано*
3	Проектування ПЗ	23.04.2024	*виконано*
4	Розробка ПЗ	10.05.2024	*виконано*
5	Тестування ПЗ	30.05.2024	*виконано*
6	Оформлення пояснювальної записки	05.06.2024	*виконано*
7	Підготовка презентації та доповіді	06.06.2024	*виконано*
8	Оцінка роботи рецензентом, отримання відзиву від керівника кваліфікаційної роботи, попередні захист роботи та проходження норм-контролю	08.06.2024 - 15.06.2024	*виконано*
9	Здачі роботи у електронний архів, допуск роботи до захисту завідувачем кафедри	15.06.2024	*виконано*
10	Захист кваліфікаційної роботи	18.06.2024	*виконано*

Дата видачі завдання 08 травня 2024р.

Студент (ка) _____ Мельник К.В.
(підпис)

Керівник роботи _____ доц. кафедри ПІ Побіженко І.О.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ / ABSTRACT

Пояснювальна записка до кваліфікаційної роботи бакалавра, 123 стор., 52 рис., 12 табл.,10 джерел.

БРОНЮВАННЯ КОВОРКІНГІВ, КЛІЄНТСЬКА ЧАСТИНА, КОВОРКІНГИ, МОБІЛЬНИЙ ЗАСТОСУНОК, СЕРВЕРНА ЧАСТИНА, JAVASCRIPT, NODE.JS, REACT, REACT NATIVE, TYPESCRIPT

Об’єкт роботи – програмна система для бронювання коворкінгів.

Мета роботи – розробка програмної системи для бронювання коворкінгів, застосування якої надає можливість спростувати, оптимізувати процес пошуку та бронювання робочих місць у коворкінгах.

Метод рішення – мови програмування JavaScript та TypeScript, програмна платформа Node.js та бібліотека React, а також фреймворк React Native.

У результаті розробки створено програмну систему для бронювання коворкінгів, що вирішує проблему полегшення процесу пошуку, бронювання і управління робочими місцями у коворкінгах для користувачів та власників.

BOOKING COWORKING SPACES, CLIENT-SIDE, COWORKING SPACES, MOBILE APPLICATION, SERVER-SIDE, JAVASCRIPT, NODE.JS, REACT, REACT NATIVE, TYPESCRIPT

The object of development is a software system for booking coworking spaces.

The purpose of the development is to develop and implement a software system for booking coworking spaces in order to simplify and optimise the process of searching for and booking workplaces in coworking spaces.

The solution method is JavaScript and TypeScript programming languages, the Node.js software platform and the React library, as well as the React Native framework.

The development resulted in a software system for booking coworking spaces that solves the problem of facilitating the process of searching, booking and managing workplaces in coworking spaces for users and owners.

Я, Мельник Катерина Вадимівна, студентка гр. ПЗПІ-20-3, здобувач вищої освіти на першому (бакалаврському) рівні кафедри «Програмна інженерія», заявляю: моя кваліфікаційна робота на тему «Програмна система для бронювання коворкінгів», що буде представлена до екзаменаційної комісії для публічного захисту, виконана самостійно, в ній не містяться елементи плагіату і вона може бути опублікована в електронному архіві відкритого доступу ElAr KhNURE. Усі запозичення з друкованих та електронних джерел мають відповідні посилання.

Я ознайомлена з діючим положенням «Про протидію академічному плагіату в ХНУРЕ», згідно з яким виявлення плагіату є підставою для відмови до допуску кваліфікаційної роботи до захисту та застосування дисциплінарних заходів.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

CEO – Chief Executive Officer

CRUD – Create, Read, Update, Delete

ПЗ – Програмне Забезпечення

БД – База Даних

ВСТУП

Актуальність розробки програмної системи для бронювання коворкінгів наразі набуває особливого значення через зростання частоти перебоїв з електроенергією, які часто трапляються в реаліях сучасного життя в Україні. У контексті цих перебоїв коворкінги стають не лише місцями для праці, а й центрами стабільності та безпеки для тих, хто працює дистанційно. Важливою рисою цих просторів є їх здатність забезпечувати безперебійну роботу завдяки наявності електрогенераторів, павербанків та інших засобів живлення. У таких умовах, розробка програмної системи, яка дозволяє швидко та зручно знаходити та бронювати робочі місця в коворкінгах, стає настільки актуальною, що вона може вирішити реальні потреби користувачів, які залежать від стабільного доступу до робочого середовища в умовах електропостачання.

Темою кваліфікаційної роботи є програмна система для бронювання коворкінгів. Завдання програмної системи для бронювання коворкінгів – спростити та оптимізувати процес пошуку та бронювання робочих місць у коворкінгах.

Метою роботи є розробка програмної системи, яка складається з серверної частини, клієнтської частини та мобільного додатку. Для розробки використовуватимуться мови програмування JavaScript та TypeScript, програмна платформа Node.js та бібліотека React, а також фреймворк React Native.

Ця програмна система може бути корисною як для індивідуальних відвідувачів коворкінгу, так і для груп, допомагаючи організовувати робочі місця. Адміністратори коворкінгів зможуть автоматизувати процес бронювання та управління простором. Мобільний додаток дозволить легко знаходити доступні робочі місця, що є важливим у сучасному ритмі життя. Розробка цієї системи відповідає вимогам ринку та вирішує актуальні завдання у сфері робочого середовища.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Аналіз предметної галузі

Програмна система є великим набором програмних компонентів, який включає в себе серверну та клієнтську частини, а також мобільний застосунок.

Слово «коворкінг», втілює поняття, яке стосується моделі організації роботи людей з різним типом зайнятості у єдиному робочому просторі. Водночас, це назва конкретного офісу, що має доцільно організований простір для колективного перебування, роботи та відпочинку. Попит на такі офіси – індивідуальні і колективні – на основі фрілансу та аутсорсингу, також приваблює користувачів нетрадиційною формою оренди, яка є більш ефективною, оскільки пропонує гнучкий підхід до строку використання та розміру орендованих площ, дозволяє швидко змінювати форм дозволяє швидко змінювати формат роботи у сучасному бізнес середовищі [1].

Програмна система для бронювання коворкінгів є великою сукупністю програмних елементів, що включає серверну та клієнтську складові, а також мобільний додаток. Узгоджена робота всіх цих програмних компонентів формує повноцінну систему, яка дозволяє ефективно управляти коворкінгами та їх бронюваннями від імені адміністраторів, а також спрощує процес перегляду та бронювання для відвідувачів.

Клієнтська складова програмна система для бронювання коворкінгів має включати окремі інтерфейси для клієнтів та адміністраторів, щоб забезпечити правильне функціонування системи в цілому.

Адміністратор – це особа, яка відповідає за управління та контроль різними аспектами системи. У випадку з програмною системою для бронювання коворкінгів, адміністратор відповідає за керування коворкінгами та їх бронюваннями.

Відвідувач – це особа, яка користується послугами системи, в даному випадку, це програмна система для бронювання коворкінгів. Відвідувачі використовують систему для перегляду доступних коворкінгів, бронювання

робочих місць. Їм надається можливість зручного та ефективного вибору для обирання бронювання необхідних послуг через мобільний додаток або вебінтерфейс.

Для визначення мети програмної системи необхідно врахувати цільову аудиторію. У випадку програми для бронювання коворкінгів, ця аудиторія включає адміністраторів коворкінгів, яким важливо мати зручний інструмент для управління бронюваннями та доступ до інформації про коворкінг, а також відвідувачів, які шукають коворкінги, переглядають інформацію про них та роблять бронювання.

Прогнозується, що більшість користувачів програми будуть використовувати коворкінги для роботи, тому вони будуть повнолітні.

Отже, цільова аудиторія включає як адміністраторів, так і відвідувачів коворкінгів, що розширюється на користувачів, зацікавлених у керуванні та використанні коворкінгових просторів.

Мета програмної системи для бронювання коворкінгів полягає в реалізації ефективного управління коворкінгами та їх бронюваннями з боку адміністраторів, а також у спрощенні процесу перегляду та бронювання для відвідувачів. Конкретні цілі системи включають:

– надання адміністраторам коворкінгів зручного інструменту для ефективного управління бронюваннями робочих місць та доступу до важливої інформації про коворкінг;

– забезпечення відвідувачів зручною платформою для пошуку доступних коворкінгів, перегляду інформації про них та зручного процесу бронювання робочих місць;

– оптимізація робочих процесів для обох сторін, зменшення часу, необхідного для здійснення бронювань та управління коворкінгами;

– підвищення задоволеності користувачів шляхом забезпечення швидкого доступу до потрібної інформації та максимального комфорту під час користування системою.

Фрагменти предметної галузі з роботи програмної системи включають:

– виконання завдань, пов’язаних із заповненням і зміною даних про коворкінги (для адміністратора);

– виконання завдань, пов’язаних із обробкою заявок на бронювання місць у коворкінгах (для адміністратора);

– виконання завдань, пов’язаних із переглядом інформації у коворкінгах (для відвідувачів);

– виконання завдань, пов’язаних із створенням заявок на бронювання місць у коворкінгах (для відвідувачів).

Фрагменти предметної галузі з роботи програмної системи включають виконання завдань, пов’язаних із управлінням даними про коворкінги та обробкою заявок на їх бронювання для адміністраторів. Для відвідувачів система забезпечує доступ до інформації про коворкінги та можливість створення заявок на бронювання місць у них.

Інформаційні потреби програмної системи для бронювання коворкінгів охоплюють наступне:

– відвідувачі коворкінгів отримують інформацію про перелік доступних коворкінгів через інтерфейс клієнтського веб-застосунку або мобільного додатку;

– відвідувачі коворкінгів виконують бронювання наявних місць у коворкінгах через інтерфейс клієнтського веб-застосунку або мобільного додатку;

– адміністратори коворкінгів оброблюють заявки на бронювання наявних місць у коворкінгах через інтерфейс клієнтського веб-застосунку;

– адміністратори коворкінгів виконують заповнення і зміни даних про коворкінги через інтерфейс клієнтського веб-застосунку.

Інформаційні потреби програмної системи для бронювання коворкінгів узгоджують зручний доступ до переліку доступних коворкінгів та забезпечують швидкий та ефективний процес бронювання для відвідувачів. Також, система спрощує обробку заявок на бронювання та управління даними для адміністраторів коворкінгів, забезпечуючи надійну базу для ефективної роботи.

Для подальшого аналізу предметної галузі проведемо аналіз існуючих аналогів.

Розглянуто веб-сайт https://coworking.ua/. Перше, що можна побачити на початковій сторінці (див. рис. 1.1), це банер з надписом «У НАС Є ТЕРМІНАЛИ STARLINK ДРУГОГО ПОКОЛІННЯ ТА ПОТУЖНІ ГЕНЕРАТОРИ». Оскільки цей веб-сайт обслуговує коворкінги в Україні, де наразі відбуваються часті перебої з електропостачанням, такий надпис виявляється доречним та привабливим для відвідувачів, може заохочувати їх відвідати коворкінг та скористатися цією перевагою. Крім того, «хедер» веб-сайту містить навігаційні посилання на всі необхідні сторінки, які можуть зацікавити відвідувача, такі як локації, сервіс, ціни, відгуки та контакти. Додатково, прямо в «хедері» наведені контактні номери телефонів, за якими можна уточнити необхідну інформацію. Також на початковій сторінці одразу відображено назва зі слоганом компанії та кнопка, яка ініціює заповнення заяви на бронювання.

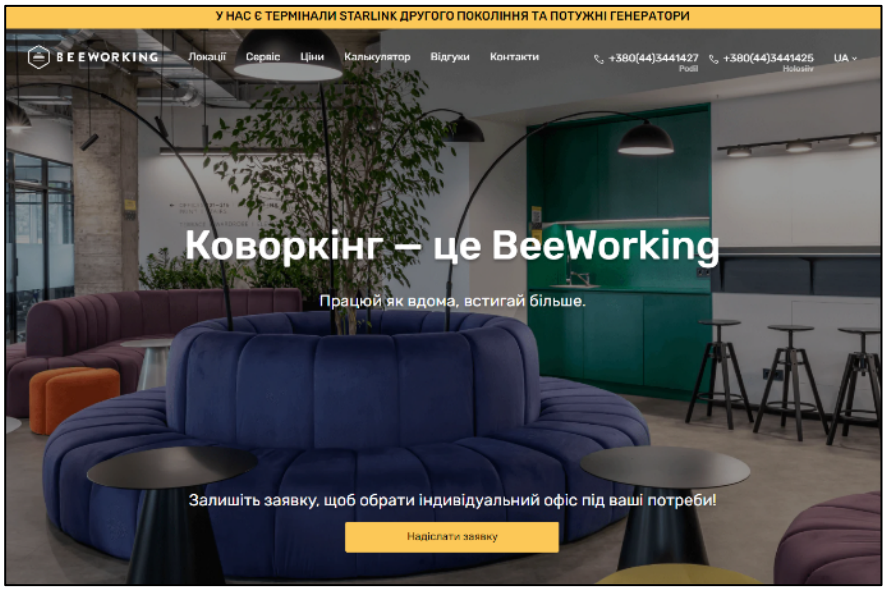


Рисунок 1.1 – Початкова сторінка веб-сайту https://coworking.ua/ (рисунок виконаний самостійно)

Є окремий блок, який має назву «Локації» (див. рис. 1.2), де показані різні локації коворкінгів, у якому включений гасло від CEO та слайдер з фотографіями різних коворкінгів, супроводжуваними підписами.

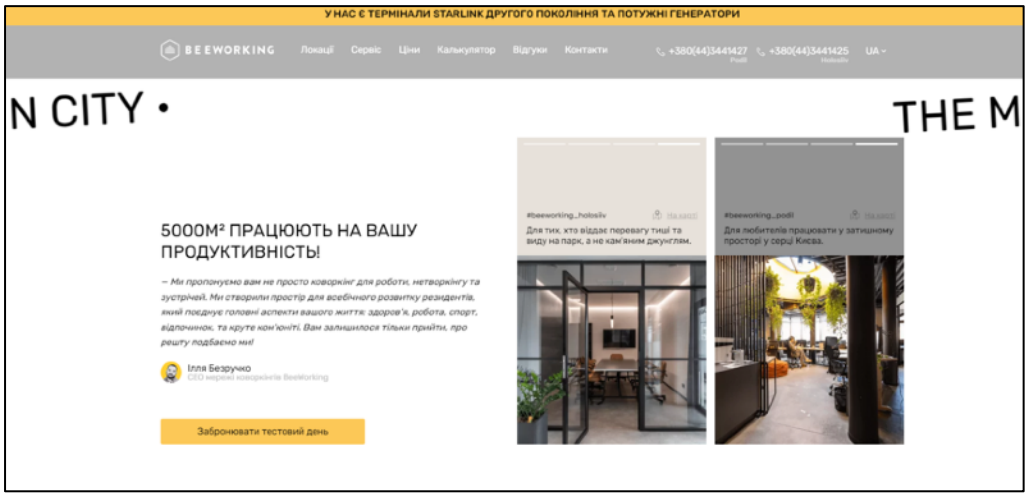


Рисунок 1.2 – Інформація про коворкінги на веб-сайті https://coworking.ua/ (рисунок виконаний самостійно)

Додатково, під гаслом розташована кнопка «Забронювати тестовий день», яка відкриває вікно з анкетою на бронювання (див. рис. 1.3), де доступний певний набір опцій, необхідних для здійснення бронювання.

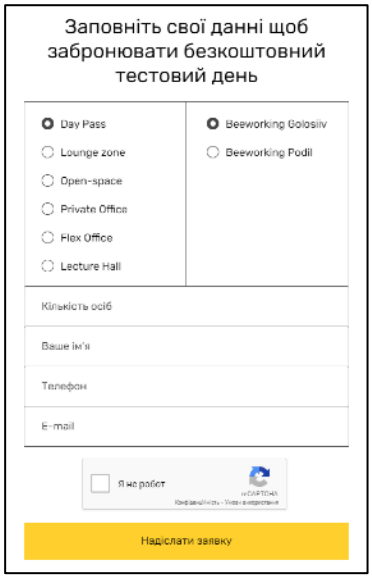


Рисунок 1.3 – Анкета для бронювання коворкінгу на веб-сайті https://coworking.ua/ (рисунок виконаний самостійно)

До переваг цього веб-сайту можна віднести наступне:

– він слідкує за поточною ситуацією в країні та пропонує сервіси відповідно до поточних трендів;

– він має приємний UI;

– наявна деяка інформація про коворкінги, яка надає користувачеві певне уявлення про них.

До недоліків цього веб-сайту можна віднести наступне:

– недосконалий UX: після закриття анкети завжди піднімається екран до початку сторінки, відсутня смуга прокрутки, деякі елементи мають ефект курсору для кліку, але самі по собі вони не клікабельні;

– немає автоматизації: заявку на бронювання можна подати в будь-який час та будь-який день з будь-якими даними, але все перевіряється вручну адміністраторами, які потім звіряються з заявниками по телефону.

Розглянуто веб-додаток https://app.natcoworking.com/. Тут присутня можливість переглядати доступні робочі місця після введення дати та кількості місць (див. рис. 1.4).

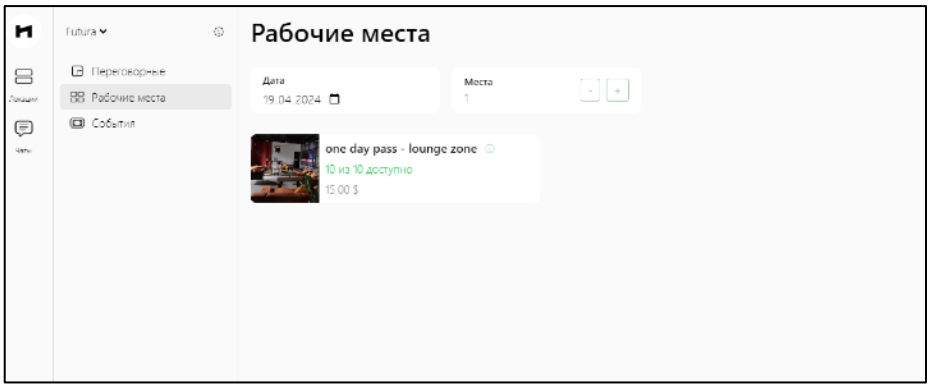


Рисунок 1.4 – Доступні робочі місця у коворкінгу у веб-додатку https://app.natcoworking.com/ (рисунок виконаний самостійно)

Також, коли були знайдені бажані локації для бронювання, їх можна обрати та, заповнивши анкету (див. рис. 1.5), забронювати їх.

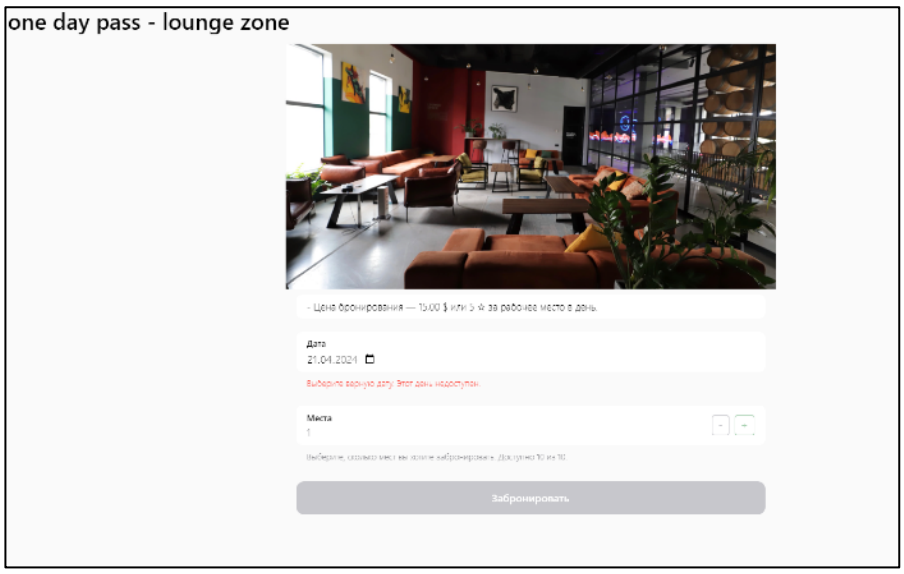


Рисунок 1.5 – Бронювання місця у коворкінгу через у веб-додаток https://app.natcoworking.com/ (рисунок виконаний самостійно)

До переваг цього веб-додатку можна віднести наступне:

– наявність автоматизації: програмне забезпечення, крім того, що воно відображає лише доступні робочі місця після застосування фільтрації та перевіряє введені дані в анкеті для бронювання, також забороняє внесення нового бронювання, якщо всі місця в коворкінгу вже заброньовані;

– наявність підказок при валідації введених даних.

До недоліків цього веб-сайту можна віднести наступне:

– деякий текст прозорий і його важко побачити;

– дуже мінімалістичний дизайн, ніяким чином не запам’ятовується.

Розглянуто веб-сайт https://coworkingclub.com.ua/. Він дуже схожий за функціоналом на перший розглянутий приклад, але тут відсутня навіть анкета. Тобто всі бронювання відбуваються шляхом зв’язку за номером телефона, який вказаний на сайті. Єдине, що варто відзначити – UX кращий, ніж у першому прикладі. Всі інші недоліки та переваги залишаються однаковими.

Після аналізу існуючих аналогів можна зробити висновок, що для того, щоб програмна система була конкурентоспроможною, необхідно поєднати в собі переваги всіх розглянутих прикладів та виправити їх недоліки.

1.2 Виявлення та вирішення проблем

В Україні, де є сильним підприємницький дух, постійно з’являються нові стартапи й розвивається бізнес, коворкінги є одним із найзручніших офісних рішень [2]. За останнє десятиліття Україна випередила практично всі країни світу щодо приросту кількості коворкінгів. У коворкінги перетворювалися колишні пожежні частини, міські поштові відділення, навіть стайні. Компанії застосовують принципи коворкінгу у своїх офісах і роблять їх частиною корпоративної культури [3].

З підвищенням популярності коворкінгів зростає потреба у програмних системах, придатних як для відвідувачів, так і для працівників коворкінгів. Наявність швидкої, зручної та функціональної системи для бронювання робочих місць у коворкінгах спрощує роботу адміністраторів та збільшує швидкість обробки бронювань, що підвищує конкурентоспроможність. Крім того, це зменшує витрати на додатковий персонал, чиї функції може виконати програмне забезпечення.

Така система також допомагає вирішувати проблеми, пов’язані з неефективним управлінням бронюваннями та обробкою заявок, забезпечуючи зручний та надійний інструмент для обох сторін – адміністраторів та відвідувачів.

1.3 Постановка задачі

Основними задачами роботи є розробка програмної системи для бронювання коворкінгів, застосування якої надає можливість спростувати, оптимізувати процес пошуку та бронювання робочих місць у коворкінгах.

Система для бронювання коворкінгів має включати три основні програмні компоненти: клієнтський веб-застосунок, серверну частину та мобільний додаток.

Клієнтський веб-застосунок призначений для адміністраторів коворкінгів і відвідувачів. Адміністратори повинні мати можливість додавати, переглядати та редагувати інформацію про свої коворкінги, а також приймати та обробляти запити на бронювання від відвідувачів. Відвідувачі з свого боку повинні мати можливість шукати та переглядати коворкінги, бронювати робочі місця та залишати відгуки. Все це повинно відбуватися за допомогою зручного адміністративного інтерфейсу. Окрім цього, клієнтський веб-застосунок повинен мати окремі видимі частини для відвідувачів і невидимі для адміністраторів для правильного функціонування системи в цілому.

Клієнтський веб-застосунок повинен бути розроблений з використанням мови програмування JavaScript та бібліотеки React.

Мобільний додаток призначений для відвідувачів коворкінгу. Вони повинні шукати й переглядати коворкінги, бронювати в них місця та залишати відгуки.

Мобільний додаток повинен бути розроблений з використанням мови програмування JavaScript та фреймворку React Native.

Серверна частина повинна обробляти дані, які були відправлені з клієнтського веб-застосунку та мобільного додатку. Вона розроблена з використанням мови програмування TypeScript та програмної платформи Node.js.

Програмна система повинна бути доступною з використанням будь-якого пристрою з доступом до Інтернету та веб-браузера для взаємодії з вебсторінками.

2 ФОРМУВАННЯ ВИМОГ ДО ПРОГРАМНОЇ СИСТЕМИ

Основною метою цієї системи є розробка програмного забезпечення, що надає зручний та ефективний інструмент для пошуку, перегляду та бронювання робочих місць у коворкінгах. Система спрощує процес вибору коворкінгу, забезпечуючи користувачам швидкий доступ до потрібної інформації, можливість бронювання в реальному часі та отримання підтвердження або скасування талону на електронну пошту. Такий підхід робить пошук та бронювання робочих місць більш зручним та ефективним для користувачів.

Програмна система повинна вирішувати наступні завдання:

а) для користувачів:

1) перегляд інформації про коворкінги;
2) бронювання робочих місць у коворкінгах;
3) залишення відгуків та оцінок;
4) пошук та фільтрація коворкінгів;

б) для адміністраторів (власників коворкінгів):

1) надання можливості управління інформацією про власні коворкінги;
2) перегляд інформації про власні створені коворкінги;
3) обробка запитів на бронювання робочих місць від користувачів;
4) отримання зворотного зв’язку від користувачів.

Програмна система для бронювання коворкінгів має наступні функціональні вимоги:

а) перегляд інформації про коворкінги:

1) відображення детальної інформації про кожен коворкінг, включаючи назву, фотографії, опис, послуги, час роботи, за номером телефону, адреса, кількість робочих місць, тариф та відгуки користувачів;
2) перегляд списку доступних коворкінгів;

б) бронювання робочих місць:

1) заповнення інформації для бронювання робочого місця в обраному коворкінгу, включаючи ім’я, номер телефону, електронну пошту, дату, час та кількість робочих місць;

2) визначення доступності місць на обрану дату та обраний час;

3) отримання талону на скасування або підтвердження бронювання через електронну пошту;

в) залишення відгуків та оцінок:

1) залишення відгуків та оцінок про коворкінги;

2) відображення середньої оцінки кожного коворкінгу на основі отриманих відгуків;

г) пошук та фільтрація коворкінгів:

1) пошук коворкінгів за містом або назвою;

2) фільтрація коворкінгів за ціною та рейтингом;

д) управління коворкінгами (для адміністраторів):

1) перегляд, додавання, редагування та видалення інформації про власні коворкінги;

2) пошук та фільтрація запитів на бронювання робочих місць за назвою коворкінгу та статусом;

3) обробка запитів на бронювання робочих місць від користувачів;

4) отримання зворотного зв’язку від користувачів через систему;

е) доступ до системи:

1) забезпечення можливості користувачам використовувати систему як через мобільний додаток, так і через веб-браузер.

Програмна система для бронювання коворкінгів також має наступні нефункціональні вимоги:

а) продуктивність:

1) система повинна забезпечувати швидку реакцію на запити користувачів, з максимально можливою ефективністю;

2) максимальний час відповіді на запити не повинен перевищувати трьох секунд навіть при великому навантаженні на систему;

б) безпека:

1) система повинна забезпечувати захист даних користувачів, використовуючи механізми шифрування та автентифікації;

в) масштабованість:

1) система повинна бути здатна масштабуватися для впорядкування зростаючого обсягу користувачів та кількості коворкінгів, забезпечуючи стабільну роботу без відчутних втрат продуктивності;

2) архітектура системи повинна бути доволі гнучкою та легко масштабованою, щоб дозволити додавання нового функціоналу та розширення функціональності без серйозних перешкод;

г) зручність використання:

1) інтерфейс користувача повинен бути інтуїтивно зрозумілим та легким у використанні;

д) надійність:

1) система повинна бути стійкою до відмов та забезпечувати неперервну доступність для користувачів, навіть у випадку виникнення непередбачених ситуацій або помилок;

е) сумісність:

1) система повинна бути сумісною з різними веб-браузерами та мобільними пристроями для забезпечення доступу користувачам з різних платформ.

Клієнтська частина буде розроблена як веб-застосунок з використанням React.js та мобільний застосунок для операційних систем Android та iOS – з використанням React Native. Серверна частина програми буде реалізована за допомогою Node.js та TypeScript. База даних сервера буде побудована на MongoDB. Технології, які будуть використовуватися: React, React Native, Node.js, Express.js, MongoDB, HTML, Sass, Mongoose, JavaScript та JSON.

Розробка програми буде здійснюватися в кілька етапів з розподілом ресурсів та тимчасових рамок таким чином:

а) підготовчий етап:

1) визначення та затвердження вимог до системи;

2) розробка архітектури програми;

3) підготовка оточення розробки та налаштування інструментів;

б) розробка серверної частини:

1) проєктування та реалізація бази даних MongoDB;

2) написання серверного коду на Node.js за допомогою Express.js;

в) розробка клієнтської частини:

1) створення UI/UX у Figma;

2) створення користувальницького інтерфейсу веб-застосунку за допомогою React.js;

3) розробка мобільного додатка для операційних систем Android та iOS з використанням React Native;

г) тестування та оптимізація:

1) проведення ручного тестування;

2) оптимізація продуктивності та усунення виявлених проблем.

Етапи розробки будуть взаємопов’язані та здійснюватимуться паралельно, щоб забезпечити своєчасне завершення проєкту у межах встановлених термінів.

3 АРХІТЕКТУРА ТА ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 UML проєктування ПЗ

Для розроблюваної серверної частини системи було створено діаграму розгортання (див. рис. 3.1), діаграму прецедентів (див. рис. 3.2), а також діаграму станів (див. рис. 3.3 та рис. 3.4).

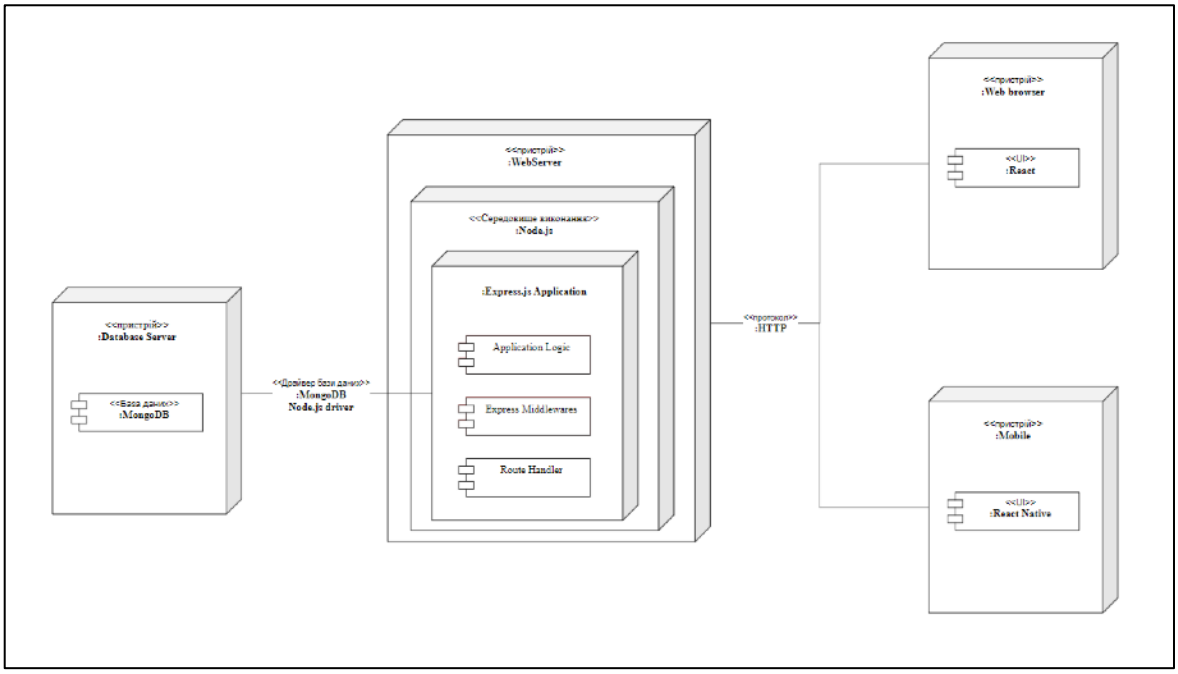


Рисунок 3.1 – UML діаграма розгортання серверної частини проєкту (рисунок виконаний самостійно)

Система складається з чотирьох ключових компонентів, а саме: база даних, веб-сервер, веб-застосунок і мобільний додаток. База даних є центральним сховищем інформації, в якому зберігаються всі необхідні дані. Веб-сервер відповідає за обробку та маршрутизацію запитів користувачів. Веб-застосунок забезпечує інтерфейс для взаємодії з системою через веб-браузер, забезпечуючи зручність управління та доступ до інформації. Мобільний додаток розширює доступність системи, надаючи користувачам зручний засіб взаємодії через їх мобільні пристрої.

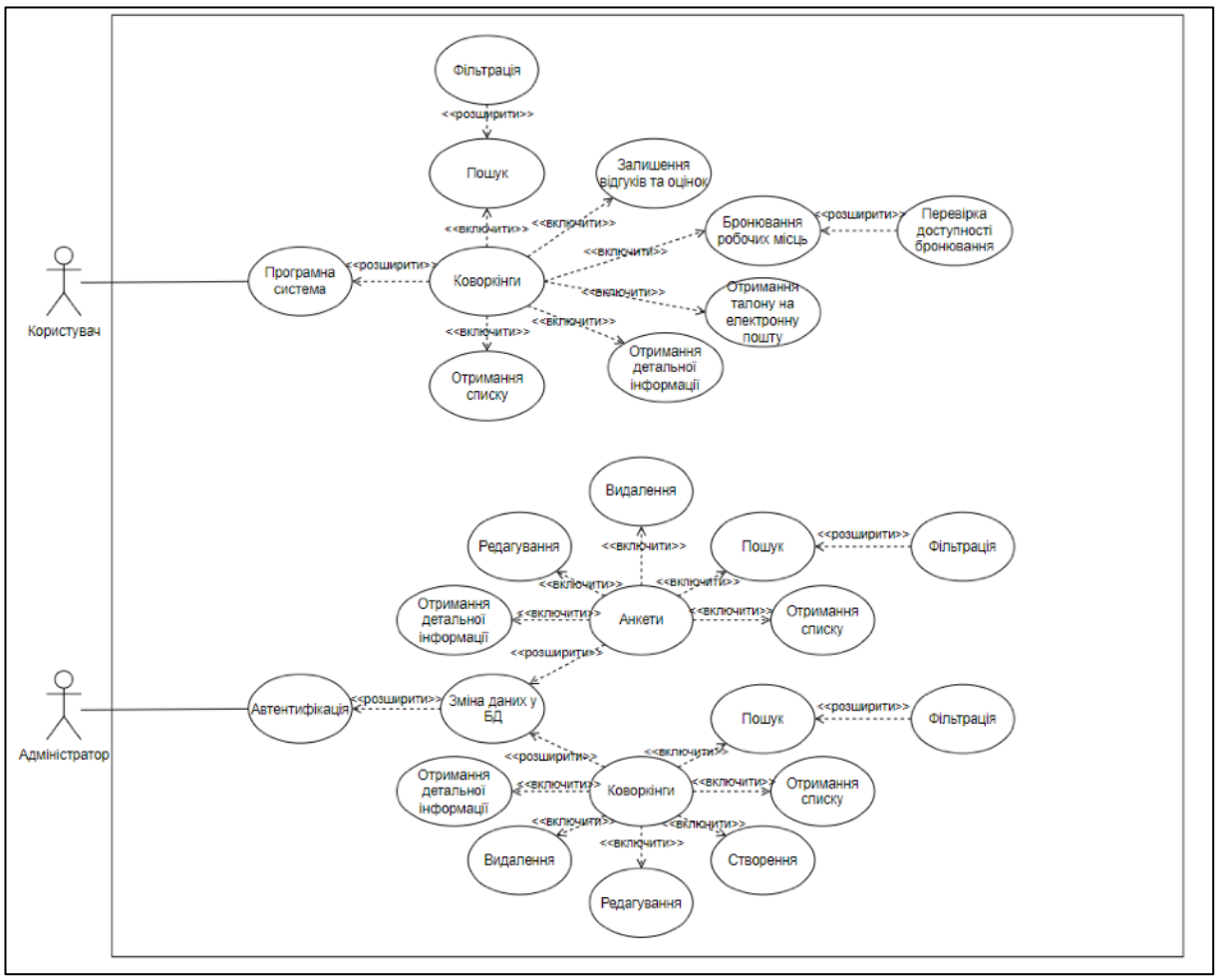


Рисунок 3.2 – UML діаграма прецедентів серверної частини проєкту для двох акторів: користувач та адміністратор (рисунок виконаний самостійно)

Користувач може виконувати наступні дії у додатку:

– отримати список коворкінгів;
– здійснити пошук та фільтрацію коворкінгів;
– отримати детальну інформацію про коворкінги;
– залишити оцінки та відгуки про коворкінги;
– переглянути доступність бронювання;
– здійснити бронювання робочих місць на обрану дату та обраний час;
– отримати талон на електронну пошту.

Адміністратор може виконувати наступні дії у додатку:

– здійснити автентифікацію;

– виконати CRUD-операції для коворкінгів;

– виконати CRUD-операції для анкет.

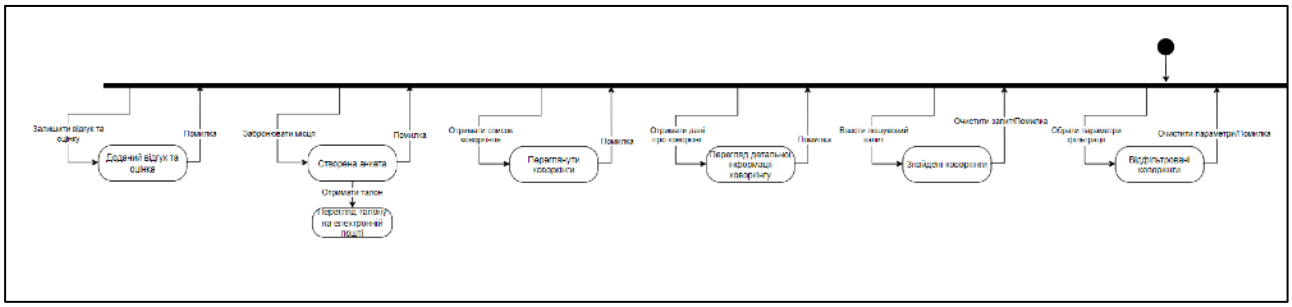


Рисунок 3.3 – Частина UML діаграми станів серверної частини проєкту для користувачів (рисунок виконаний самостійно)

Перша частина діаграми відображає доступні функції для користувача, а саме: перегляд усіх коворкінгів, перегляд детальної інформації про коворкінги, додавання відгуків та оцінок, створення анкет, отримання талону на власну електронну пошту, а також пошук та фільтрація коворкінгів.

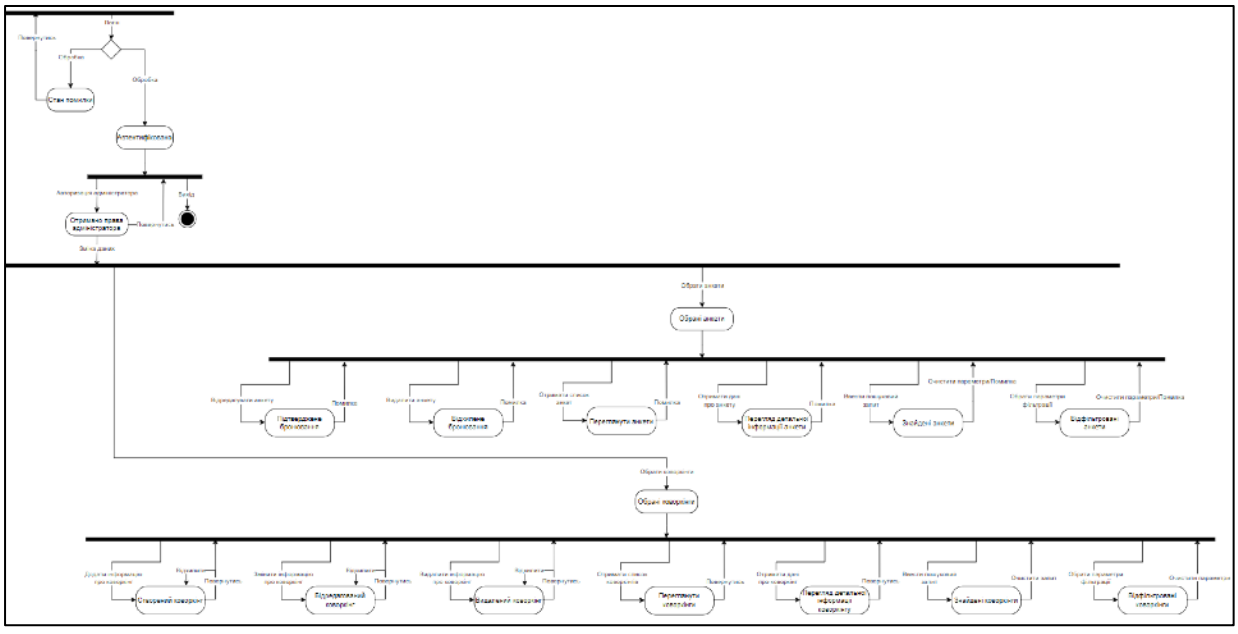


Рисунок 3.4 – Частина UML діаграми станів серверної частини проєкту для адміністраторів (рисунок виконаний самостійно)

Друга частина діаграми відображає доступні функції для адміністраторів системи, а саме: автентифікація в системі, вихід з неї, а також виконання CRUDоперацій для коворкінгів та анкет.

Для розроблюваної клієнтської частини системи було створено діаграму прецедентів (див. рис. 3.5), діаграму компонентів (див. рис. 3.6), діаграму пакетів (див. рис. 3.7), а також діаграму станів (див. рис. 3.8 та див. рис. 3.9).

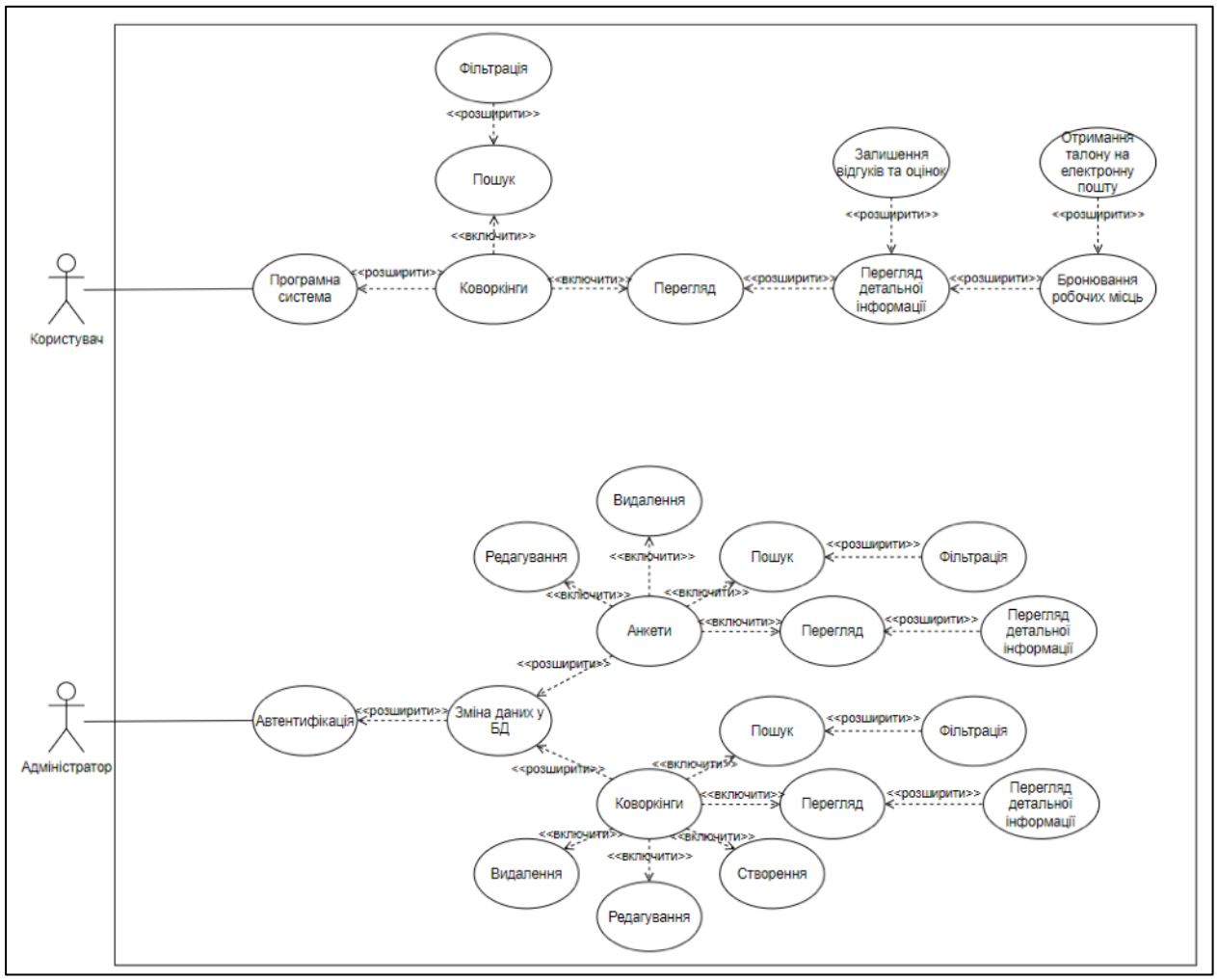


Рисунок 3.5 – UML діаграма прецедентів клієнтської частини проєкту (рисунок виконаний самостійно)

Користувач може виконувати наступні дії у додатку:

– переглянути всі коворкінги;
– здійснити пошук та фільтрацію коворкінгів;
– переглянути детальну інформацію про коворкінги;

– залишити оцінки та відгуки про коворкінги;
– забронювати робочі місця на доступну дату та час;
– отримати талон на електронну пошту.

Адміністратор може виконувати наступні дії у додатку:

– здійснити автентифікацію;
– здійснити пошук та фільтрацію анкет;
– переглянути анкети;
– відредагувати анкети;
– видалити анкети;
– здійснити пошук та фільтрацію коворкінгів;
– переглянути коворкінги;
– створити коворкінги;
– відредагувати коворкінги;
– видалити коворкінги.

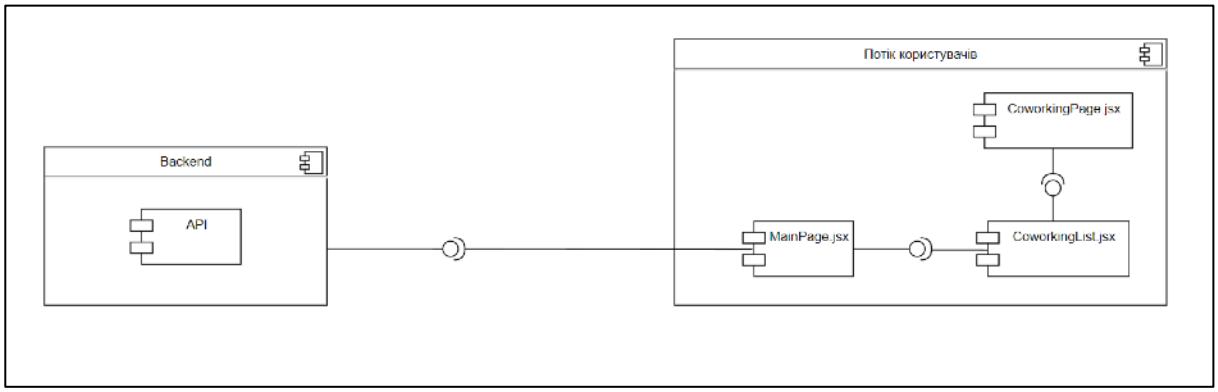


Рисунок 3.6 – UML діаграма прецедентів клієнтської частини проєкту (рисунок виконаний самостійно)

Діаграма компонентів клієнтської частини відображає підключені API розробленої програмної системи та доступні сторінки. Для користувачів доступна головна сторінка, сторінка зі списком усіх коворкінгів та окрема сторінка для кожного коворкінгу.

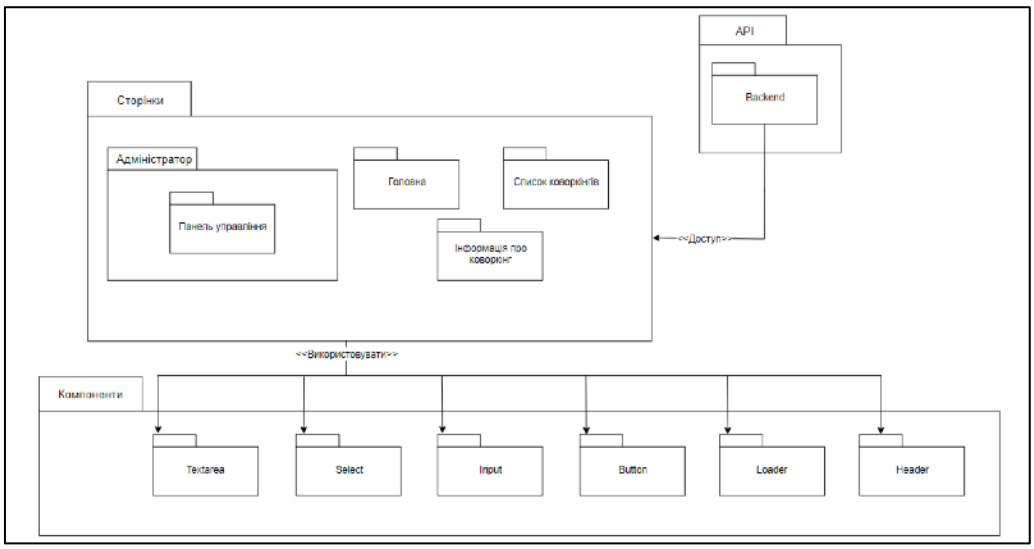


Рисунок 3.7 – UML діаграма пакетів клієнтської частини проєкту (рисунок виконаний самостійно)

Діаграма пакетів клієнтської частини відображає підключені API розробленої програмної системи та доступні сторінки для користувачів в залежності від їх ролі у системі. Для звичайних користувачів доступні головна сторінка, сторінка зі списком усіх коворкінгів та окрема сторінка для кожного коворкінгу. Авторизований адміністратор має доступ до адміністративної панелі для керування даними системи.

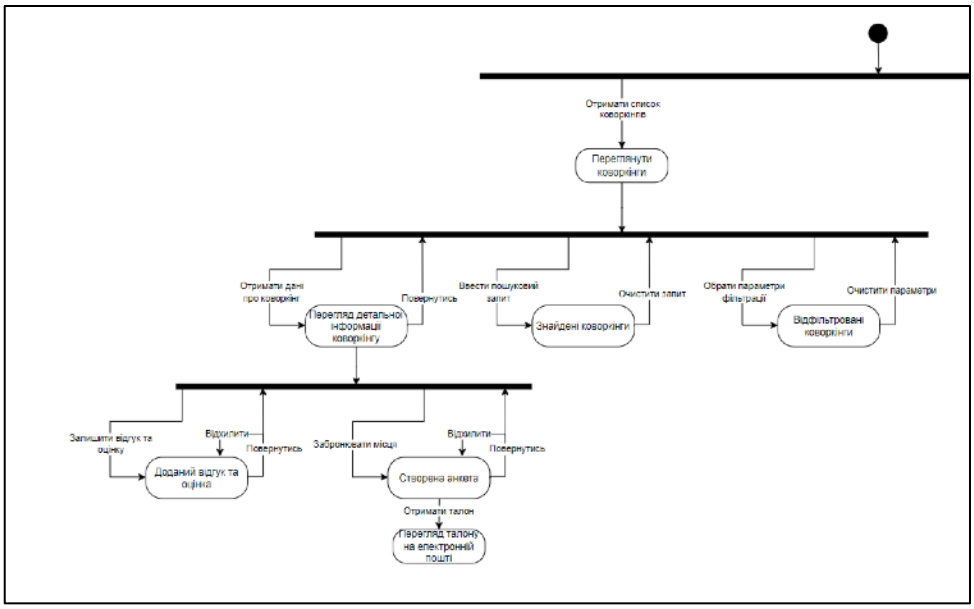


Рисунок 3.8 – Частина UML діаграми станів клієнтської частини проєкту для користувачів (рисунок виконаний самостійно)

Перший блок діаграми відображає функціонал, доступний для користувачів, включаючи перегляд списку коворкінгів, отримання детальної інформації про них, можливість залишати відгуки та оцінки, створення анкет, отримання талону на електронну пошту, а також можливість проведення пошуку та застосування фільтрів для коворкінгів.

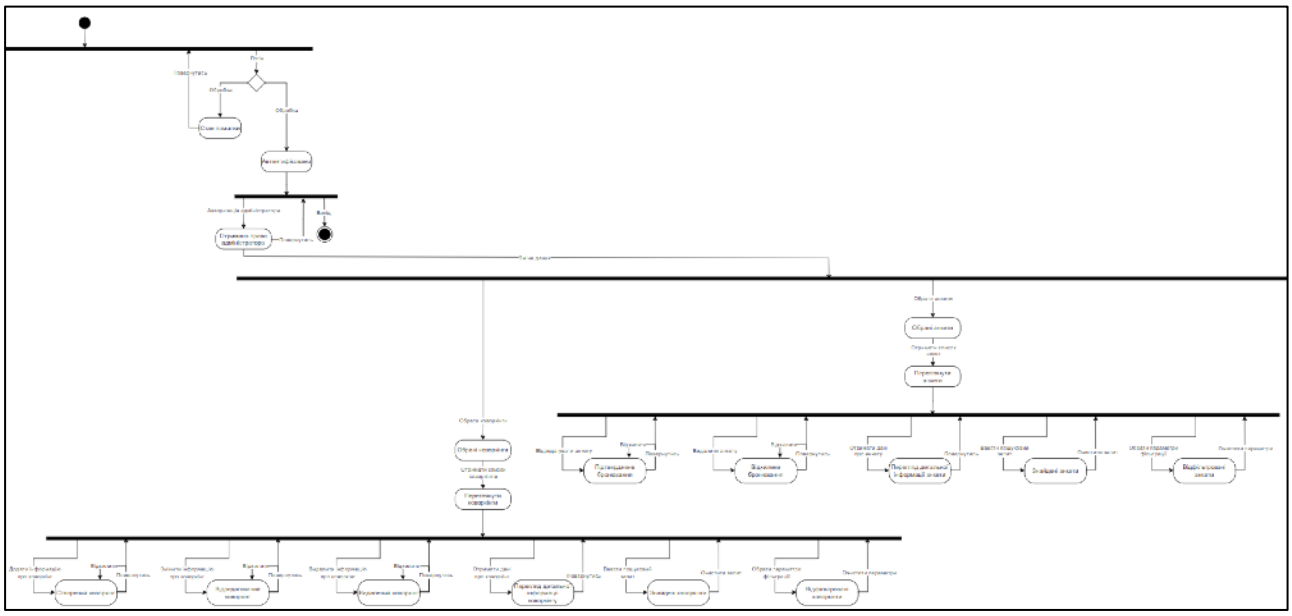


Рисунок 3.9 – Частина UML діаграми станів клієнтської частини проєкту для адміністраторів (рисунок виконаний самостійно)

Друга частина діаграми відображає доступні функції для адміністраторів системи, такі як автентифікація в системі та вихід з неї. Крім того, адміністратор має можливість створювати нові коворкінги, а також переглядати, шукати, застосовувати фільтри, редагувати та видаляти анкети та інформацію про коворкінги.

Для розроблюваного мобільного програмного застосунку було створено діаграму прецедентів (див. рис. 3.10), діаграму компонентів (див. рис. 3.11), діаграму пакетів (див. рис. 3.12), а також діаграму станів (див. рис. 3.13).

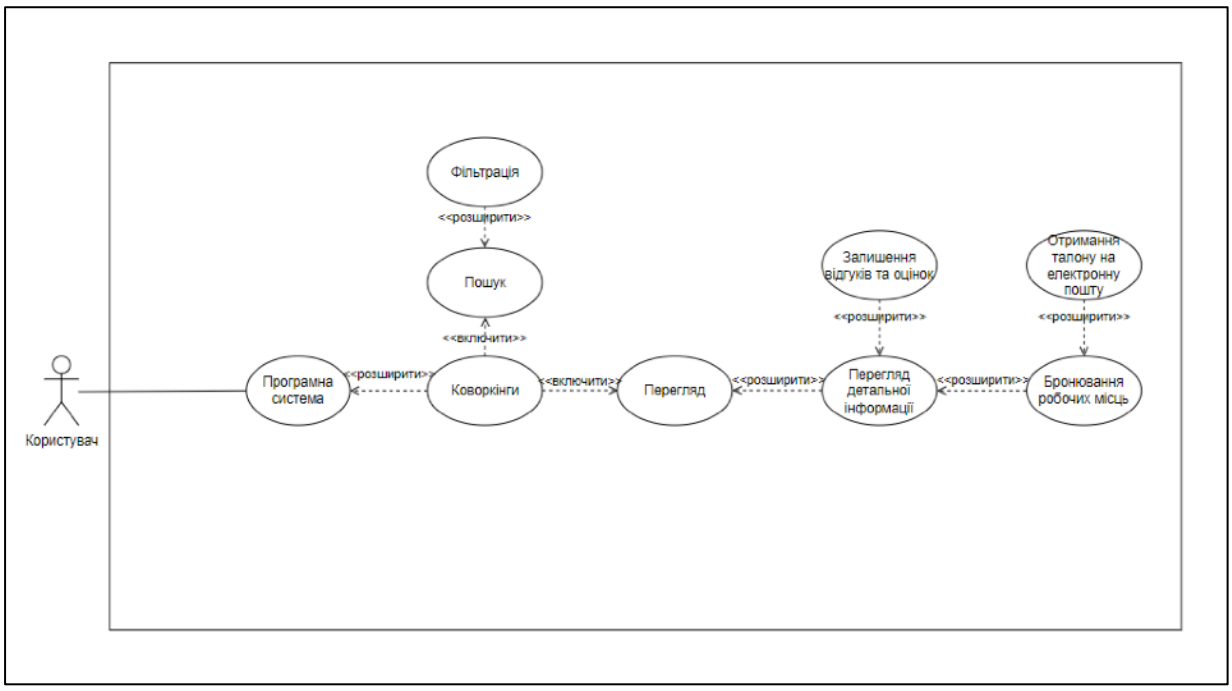


Рисунок 3.10 – UML діаграма прецедентів мобільного програмного застосунку (рисунок виконаний самостійно)

Користувач може виконувати наступні дії у програмного застосунку:

– переглянути всі коворкінги;
– здійснити пошук та фільтрацію коворкінгів;
– переглянути детальну інформацію про коворкінги;
– залишити оцінки та відгуки про коворкінги;
– забронювати робочі місця на доступну дату та час;
– отримати талон на електронну пошту.

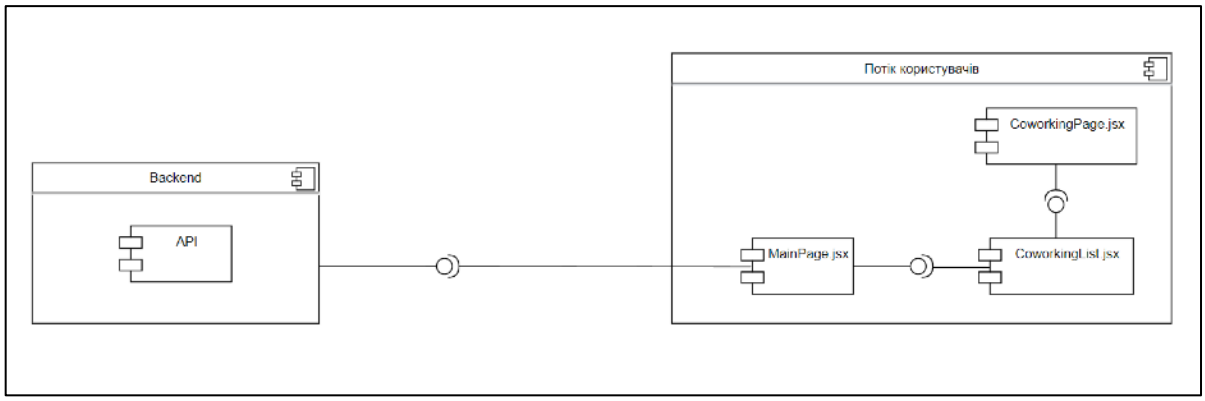


Рисунок 3.11 – UML діаграма компонентів мобільного програмного застосунку (рисунок виконаний самостійно)

Діаграма компонентів клієнтської частини системи ілюструє, які API використовуються у програмному забезпеченні та які сторінки доступні для користувачів. Підключені API визначають, які функції може виконувати програма. До сторінок, які доступні для користувачів, включаються головна сторінка, сторінка зі списком всіх коворкінгів та окремі сторінки для кожного з них.

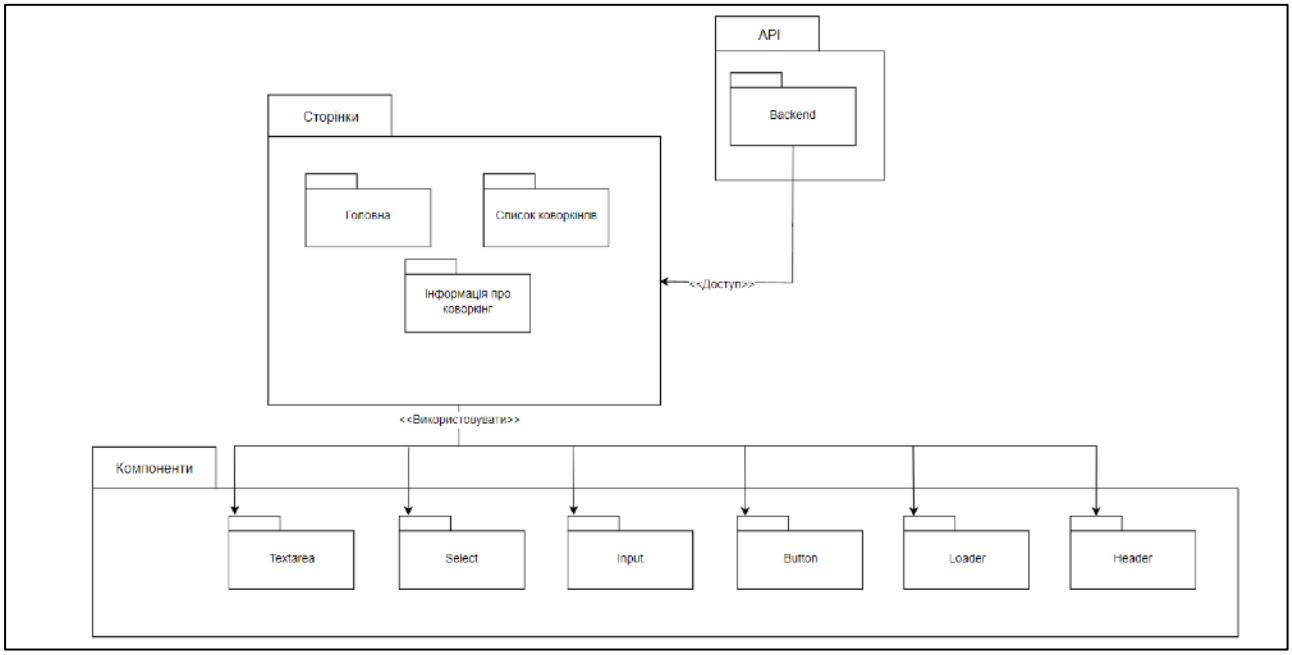


Рисунок 3.12 – UML діаграма пакетів мобільного програмного застосунку (рисунок виконаний самостійно)

У діаграмі пакетів клієнтської частини системи зображено, які API використовуються у розробленій програмній системі, а також які сторінки доступні для користувачів, в залежності від їх ролі у системі. Звичайним користувачам надається доступ до головної сторінки, сторінки зі списком усіх коворкінгів, а також окремої сторінки для кожного коворкінгу.

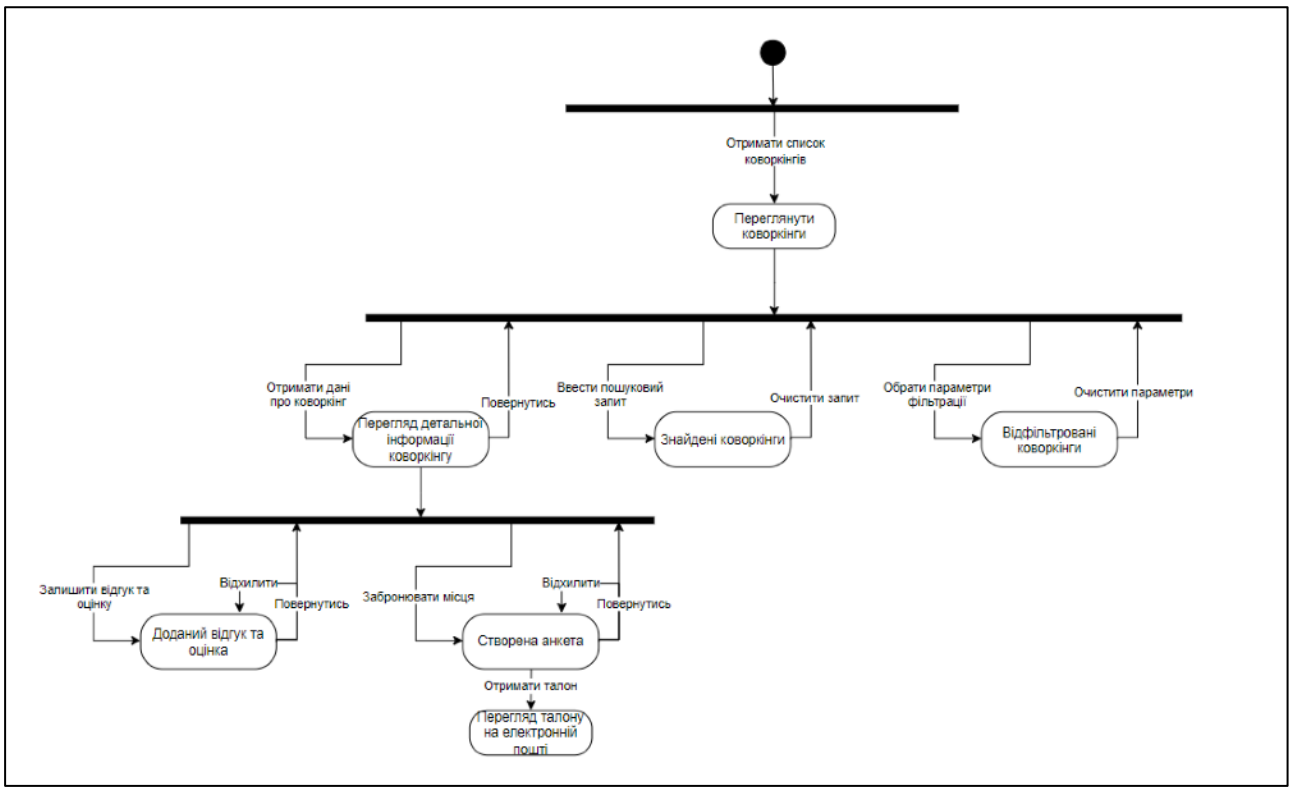


Рисунок 3.13 – UML діаграма станів мобільного програмного застосунку (рисунок виконаний самостійно)

Діаграма показує, що користувачі можуть виконувати різноманітні дії, такі як перегляд списку коворкінгів, отримання детальної інформації про них, залишення відгуків та оцінок, створення анкет, отримання талону на електронну пошту, а також здійснювати пошук та використовувати фільтри для коворкінгів.

3.2 Проєктування архітектури ПЗ

При проєктування архітектури програмного забезпечення було обрано трирівневу архітектуру (див. рис. 3.14), яка дозволяє зберігати код додатку організованим та легко зрозумілим способом. Кожен рівень має свою відповідальність та чітко визначені обов'язки, сприяючи модульності, перевикористанню коду та підтримці довгострокової розширюваності додатку.

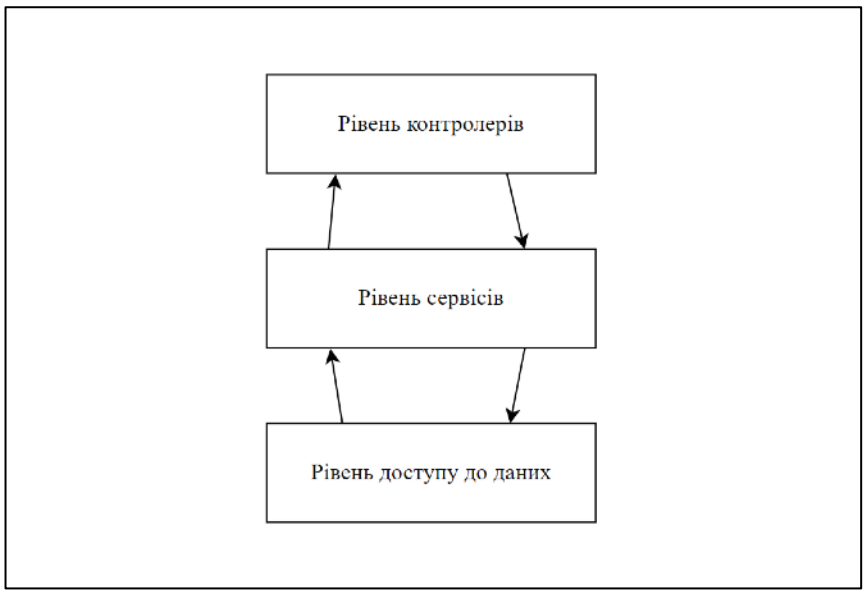


Рисунок 3.14 – Архітектура програмного забезпечення (рисунок виконаний самостійно)

Обрана архітектура має таку структуру:

– рівень контролерів: на цьому рівні знаходяться контролери, які обробляють запити, відповідають за маршрутизацію та взаємодію з клієнтською частиною; контролери приймають вхідні запити, обробляють їх та викликають відповідні функції сервісів для виконання бізнес-логіки; також вони формують відповіді для клієнтської частини, включаючи статус-коди, заголовки та дані;

– рівень сервісів: на цьому рівні розташовані компоненти, що відповідають за бізнес-логіку програми; ці сервіси отримують дані від контролерів та виконують з ними різноманітні операції, такі як валідація, обчислення, взаємодія з базою даних та інші; вони можуть містити бізнесправила та логіку обробки даних, які не прив’язані безпосередньо до конкретних інструментів доступу до даних;

– рівень доступу до даних: на даному рівні здійснюється доступ до даних через модулі або класи, що взаємодіють з базою даних; у контексті використання архітектури з «mongoose», цей рівень включає схеми та моделі «mongoose», які визначають структуру даних та методи для роботи з базою даних «MongoDB» [4]. Його функцією є збереження, оновлення, видалення та отримання даних з бази даних.

Структура серверної частини має такий вигляд (див. рис. 3.15):

– у директорії «middlewares» розташовані проміжкові функції, що керують різними аспектами обробки запитів; вони охоплюють функції для автентифікації користувача, обробки помилок та перевірки введених даних у запитах;

– у директорії «resources» знаходяться всі складові, що стосуються обробки запитів; тут є контролери, які спілкуються з клієнтською частиною й здійснюють обробку запитів, інтерфейси для взаємодії між компонентами, моделі колекцій для роботи з базою даних, а також логіка бізнес-процесів та валідація даних;

– у директорії «utils» розташовані допоміжні функції, які застосовуються в інших частинах серверної частини; сюди входять функції, що відносяться до створення та перевірки JWT токенів, або інші загальні функції, які можуть використовуватися в різних частинах проєкту.

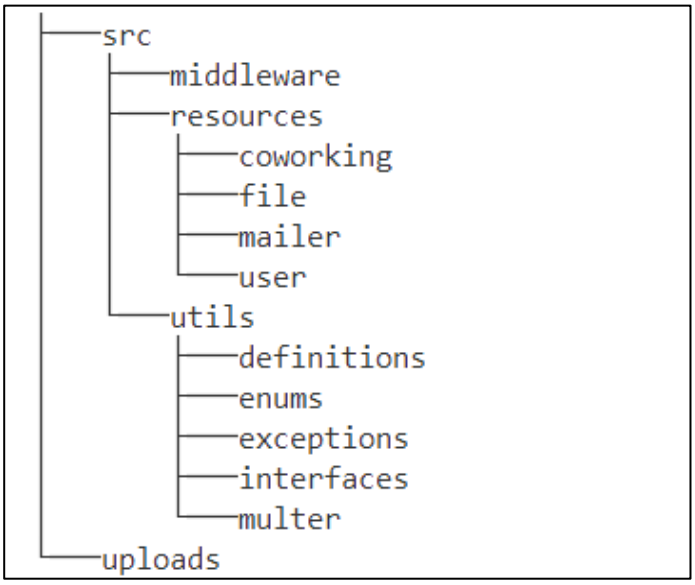


Рисунок 3.15 – Структура програмного забезпечення (рисунок виконаний самостійно)

Ця структура забезпечує логічне розділення функціональності, що полегшує розуміння та підтримку коду. Кожна директорія має чітко визначену роль, що сприяє організованості проєкту.

3.3 Проєктування структури зберігання даних

Було створено ER-модель даних (див. рис. 3.16), що включає 7 таблиць: користувачі, анкети, коворкінги, послуги, оцінки, графіки роботи, фотографії.

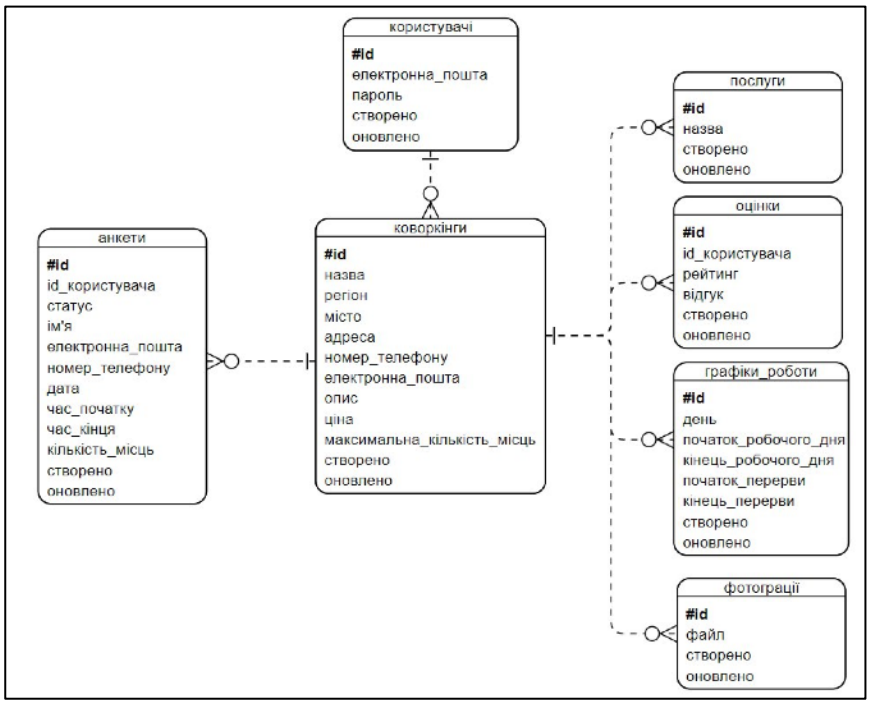


Рисунок 3.16 – Структура бази даних у вигляді ER-моделі даних (рисунок виконаний самостійно)

Між сутностями були визначені наступні взаємозв’язки:

- користувачі – коворкінги (1:M);
- коворкінги – анкети (1:M);
- коворкінги – послуги (1:M);
- коворкінги – оцінки (1:M);
- коворкінги – графіки роботи (1:M);
- коворкінги – фотографії (1:M).

3.4 Приклади найцікавіших алгоритмів та методів

Для отримання доступних часів для бронювання робочих місць у коворкінгвх на обрану дату та кількість місць була реалізована функція «getAvailableAppointmentTimes» (див. додаток В). Ця функція виконує наступні кроки:

– на вхід отримується ідентифікатор коворкінгу, дата, для якої потрібно отримати доступні часи, і кількість місць, необхідна для бронювання;

– за допомогою ідентифікатору знаходиться інформація про коворкінг;

– перевіряється наявність коворкінгу, якщо його немає, генерується помилка;

– за допомогою переданої дати ініціалізується змінна «dayOfWeek», що представляє день тижня у форматі «повна назва дня»;

– фільтрується список анкет коворкінгу за вказаною датою;

– для кожної години перевіряється, чи є збіги з існуючими записами;

– проводиться ітерація з 0 до 23, щоб визначити доступні години для запису;

– розраховується загальна кількість зайнятих місць для кожної години;

– встановлюється прапорець «disable» для кожної години в залежності від максимальної кількості місць та робочого часу;

– повертається масив «appointmentTimes», який містить інформацію про доступні години разом з прапорцем «disable».

Таким чином, цей код ефективно організовує процес отримання доступних годин для запису в коворкінги, з урахуванням різних обмежень, таких як кількість доступних місць і робочий графік.

Для відправлення підтвердження або скасування талону на електрону пошту було створено клас «MailerService» (див. додаток Г), який виконує такі кроки:

– клас має метод «sendMail», який приймає ряд параметрів для формування інформації про бронювання та відправлення листа: статус

бронювання, назву коворкінгу, електронну адресу отримувача, ім’я отримувача, номер телефону отримувача, дату бронювання, час початку бронювання, час завершення бронювання, кількість забронюваних місць, розраховану ціну та унікальний код анкети;

– створюється новий об’єкт транспортера Nodemailer за допомогою «nodemailer.createTransport» для параметрів підключення («host», «port», «secure», «user», «password»);

– створюється HTML-рядок для вмісту листа на основі переданого статусу бронювання та інших параметрів. У випадку успішного бронювання відправляється лист з підтвердженням, у протилежному випадку – лист зі скасуванням;

– відправляється лист за допомогою методу «transporter.sendMail»;

– результат відправлення листа повертається як об’єкт «SendMail»;

– у разі помилки відправлення листа виникає виключення, яке обробляється в блоках «catch», де викидається нове виключення з повідомленням про помилку.

Таким чином, цей код дозволяє відправляти електронні листи з детальною інформацією про бронювання робочих місць у коворкінгах.

Для автоматичного оновлення доступних часів бронювання була реалізована функція «updateAvailableTime» з використанням хука «useEffect» (див. додаток Д). Цей код виконує наступні кроки:

– функція перевіряє, чи передана дата співпадає з поточною датою. Якщо так, вона продовжує виконання, в іншому випадку повертає поточні блоки часу;

– функція перебирає всі блоки часу і встановлює прапорець «disable» у «true» для тих, що вже минули на момент виконання функції;

– у кінці функція знову ініціалізує змінні «selectedBlocks», «firstBlock» та «secondBlock», і повертає оновлені блоки часу;

– хук використовується для виклику функції при зміні дати бронювання або кількості місць для бронювання;

– при виклику функції спочатку оновлюються доступні часи для бронювання;

– обчислюється кількість мілісекунд до наступної години (часу, коли оновлення часів стане актуальним);

– за допомогою «setTimeout» встановлюється затримка перед наступним оновленням доступних часів;

– якщо значення дати бронювання і кількості місць для бронювання не пусті, тоді викликається функція «updateAvailableTimeHandler».

Таким чином, цей код автоматично оновлює доступні часи для бронювання кожну годину в залежності від поточного часу і дати бронювання.

3.5 Створення UI / UX або іншого дизайну системи

На рисунку 3.17 зображена початкова сторінка застосунку, через яку можно відкрити сторінку коворкінгу або перейти до пошуку.

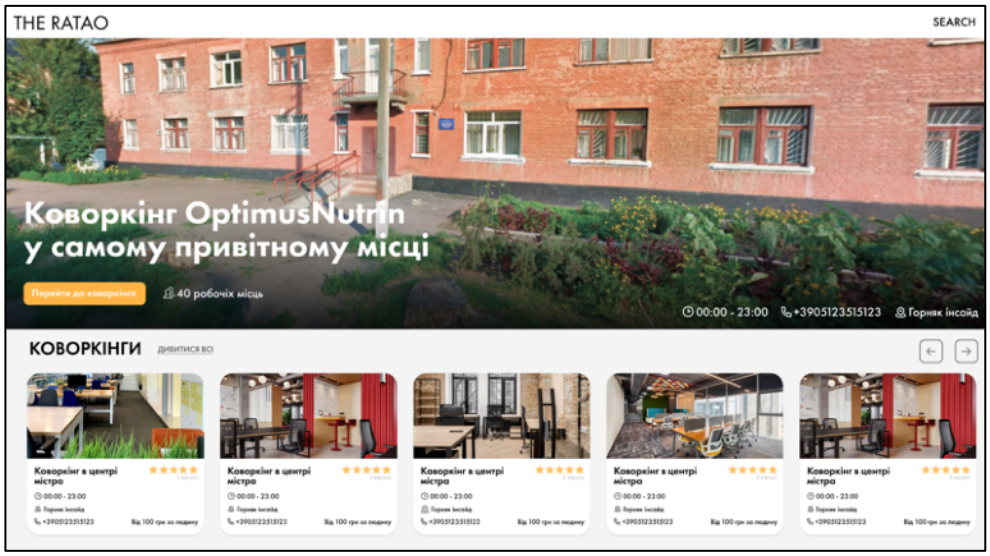


Рисунок 3.17 – Початкова сторінка застосунку (рисунок виконаний самостійно)

На рисунку 3.18 зображена сторінка пошуку всіх коворкінгів, через яку можна перейти до обраного коворкінгу, або натиснути кнпоку фільтрації та перейти до форми, яка фільтрує усі елементи пошуку.

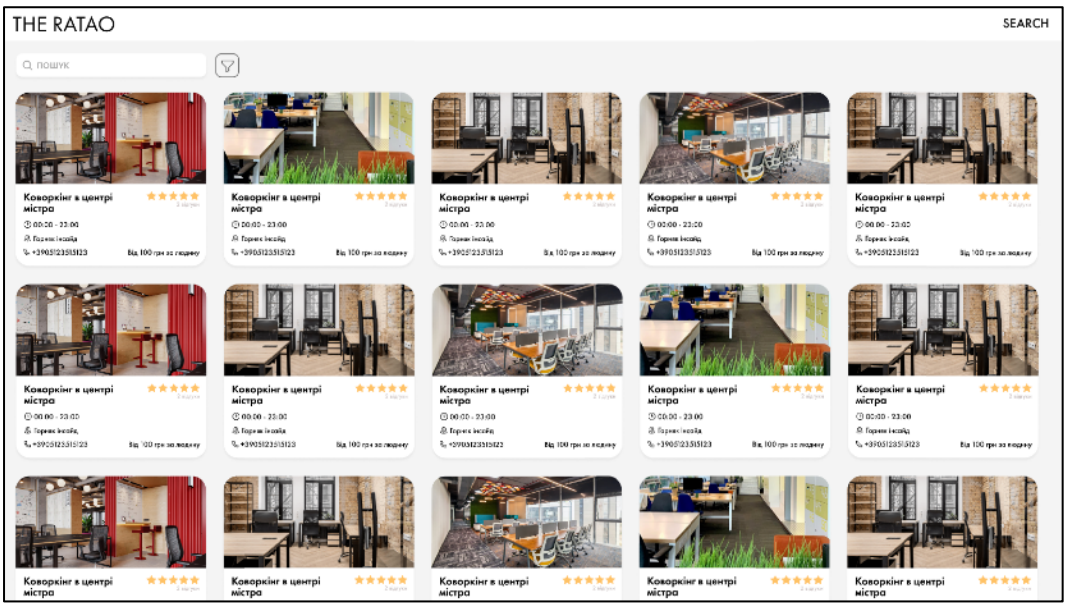


Рисунок 3.18 – Сторінка пошуку та фільтрації коворкінгів (рисунок виконаний самостійно)

На рисунку 3.19 зображена форма фільтрації коворкінгів за оцінкою та проміжком цінового інтервалу.

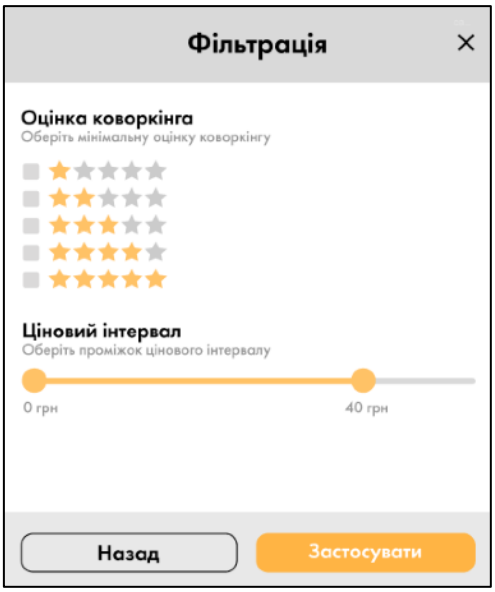


Рисунок 3.19 – Форма фільтрації (рисунок виконаний самостійно)

На рисунку 3.20 зображена перга частина сторінки коворкінгу, на сторінці можна перейти до форми бронювання коворкінгу.

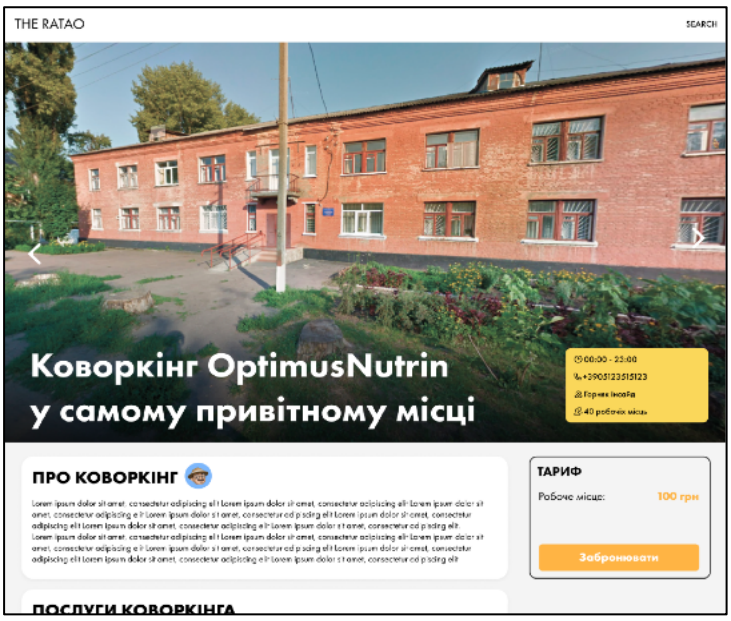


Рисунок 3.20 – Перша частина сторінки коворкінгу (рисунок виконаний самостійно)

На рисунку 3.21 зображена друга частина сторінки коворкінгу, на сторінці можна передивитись відгуки, та залишити свій відгук.

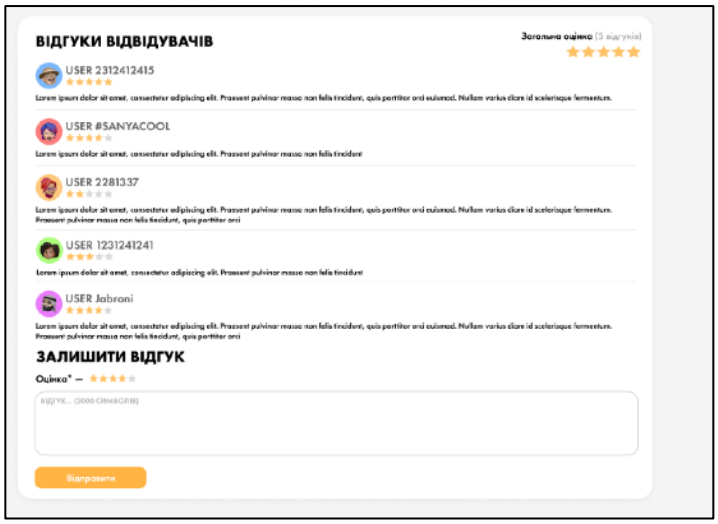


Рисунок 3.21 – Друга частина сторінки коворкінгу (рисунок виконаний самостійно)

На рисунку 3.22 зображена анкета бронювання, через неї клінт може вписати своє ім’я, електрону пошту, телефон, обрати дату та час бронювання.

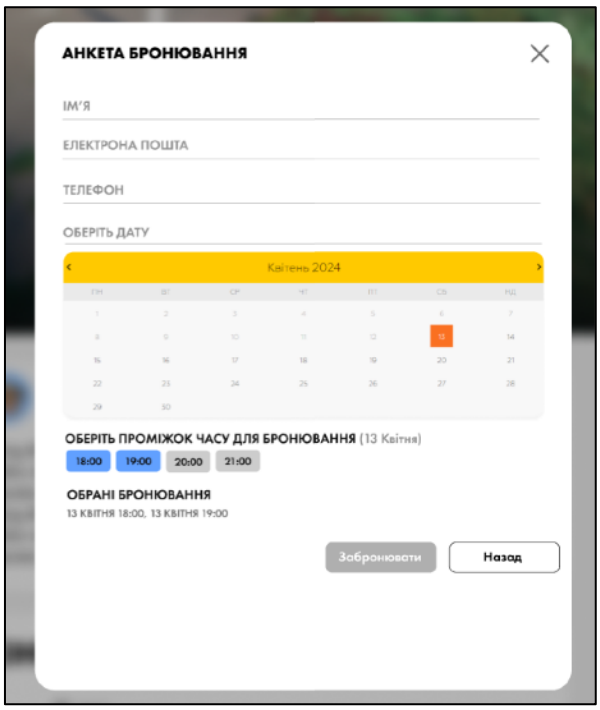


Рисунок 3.22 – Анкета бронювання коворкінгу (рисунок виконаний самостійно)

Після відправки анкети користувач може передивитись повідомлення, яке повідомляє про успішність або помилку при бронюванні, після чого місце і час який обрав користувач буде зарезервованим за цим користувачем.

4 ОПИС ПРИЙНЯТИХ ПРОГРАМНИХ РІШЕНЬ

Розглянуті програмні рішення були обрані з урахуванням вимог до проєкту, а також з метою забезпечення ефективності, безпеки та зручності користування системою як для користувачів, так і для адміністраторів.

Серверна частина програми реалізована з використанням середовища виконання JavaScript – Node.js. Мова програмування TypeScript використовувалась для строгої типізації, що покращує надійність та підвищує швидкість розробки. TypeScript дозволяє виявляти та виправляти помилки ще на етапі написання коду, що робить розробку більш прозорою та ефективною [4]. Node.js дозволяє створювати масштабовані серверні додатки з використанням неблокуючого вводу/виводу, що робить його ідеальним вибором для реалізації веб-сервера з можливістю обробки багатьох одночасних запитів [5]. MongoDB була обрана як база даних сервера через свою гнучкість та простоту використання. Як нереляційна база даних, MongoDB дозволяє зберігати дані у вигляді документів JSON, що полегшує роботу з ними в середовищі JavaScript [6].

Для реалізації клієнтської частини програми було обрано бібліотеку React.js для веб-застосунку та фреймворк React Native для мобільного додатку. Одним із важливих факторів у виборі React була його популярність та широке співтовариство розробників, що сприяє доступності багатої документації та ресурсів для розвитку проєкту. React надає можливість швидкого розгортання та розробки інтерфейсу користувача за допомогою компонентного підходу [7]. Це дозволяє розділити інтерфейс на невеликі незалежні компоненти, що полегшує розробку, розширення та підтримку програми в майбутньому. Для пришвидшення розробки мобільної версії застосунку було вирішено використати React Native. Цей фреймворк дозволяє створювати мобільні додатки, використовуючи звичні засоби розробки на React, що забезпечує однаковий дизайн і функціональність для обох платформ Android та iOS [8].

Для забезпечення ізоляції програми від середовища на серверній частині проекту було створено файл «.env», що містить наступні змінні:

– NODE_ENV: визначає поточне середовище виконання додатка;

– PORT: вказує порт, на якому буде запущений сервер;

– FRONTEND_APP_URL: URL-адреса веб-додатку для клієнтської частини;

– MOBILE_APP_URL: URL-адреса мобільного додатку для клієнтської частини;

– ADMIN_APP_URL: URL-адреса адміністративної частини додатка;

– BACKEND_APP_URL: URL-адреса серверної частини додатка;

– JWT_ACCESS_SECRET: секретний ключ для підпису та перевірки JWT токенів доступу;

– JWT_ACCESS_SECRET_LIFETIME: тривалість життя JWT токенів доступу;

– UPLOADS_DESTINATION: шлях для завантаження файлів;

– MONGO_PATH: шлях до MongoDB, включаючи ім’я користувача та пароль;

– MONGO_USER: ім’я користувача MongoDB;

– MONGO_PASSWORD: пароль користувача MongoDB;

– SECRET_KEY: секретний ключ для шифрування даних або генерації підписів;

– MAILER_SMPT_HOST: хост SMTP-сервера для відправлення електронної пошти;

– MAILER_SMPT_PORT: порт SMTP-сервера;

– MAILER_SMPT_SECURE: вказує, чи використовується SSL для забезпечення безпеки з’єднання з SMTP-сервером;

– MAILER_SMPT_USER: користувацьке ім’я або адреса електронної пошти SMTP.

– MAILER_SMPT_PASSWORD: пароль для автентифікації на SMTP-сервері.

Цей конфігураційний файл дозволяє налаштувати різні параметри додатка залежно від середовища виконання та вимог проєкту.

Однією з ключових складових ініціалізації серверної частини проєкту за допомогою конструктору є налаштування Express-додатку та визначення порту, на якому він буде працювати. Express володіє простим та зрозумілим API, що сприяє легкості розробки веб-додатків [9]. Він не нав'язує жорстких обмежень або структур і є швидким та легким у порівнянні з іншими фреймворками:

```
constructor(controllers: Controller[], port: number) {
    this.express = express();
    this.port = port;
    this.initialiseFileStorage();
    this.initialiseDatabaseConnection();
    this.initialiseMiddleware();
    this.initialiseControllers(controllers);
    this.initialiseErrorHandling();
}
```

Функція «initialiseMiddleware» встановлює проміжне програмне забезпечення для Express, включаючи захист Helmet, CORS, обробку JSON та URL-кодованих даних, компресію та роботу з куками:

```
private initialiseMiddleware(): void {
    this.express.use(helmet({crossOriginResourcePolicy: false,}));
          this.express.use(cors({credentials: true, origin: [`${process.env.FRONTEND_APP_URL}`, `${process.env.MOBILE_APP_URL}`, `${process.env.ADMIN_APP_URL}`, `${process.env.BACKEND_APP_URL}`,],}));
    this.express.use(morgan('dev'));
    this.express.use(express.json());
    this.express.use(express.urlencoded({ extended: false }));
    this.express.use(compression());
    this.express.use(cookie());
this.express.use(`${process.env.UPLOADS_DESTINATION}`.substring(1), express.static(`${process.env.UPLOADS_DESTINATION}`));
}
```

Функція «initialiseFileStorage» відповідає за ініціалізацію файлового сховища. Вона перевіряє наявність каталогу для завантажень та, у випадку відсутності, створює його за допомогою «fs.mkdirSync»:

```
private initialiseFileStorage(): void {
    if (!fs.existsSync(`${process.env.UPLOADS_DESTINATION}`)) {
        fs.mkdirSync(`${process.env.UPLOADS_DESTINATION}`);}}
```

Функція «initialiseControllers» встановлює контролери для маршрутів, додаючи їх у Express-додаток. Кожен контролер зв'язується з підшляхом «/api»:

```
private initialiseControllers(controllers: Controller[]): void {
    controllers.forEach((controller: Controller) => {
        this.express.use('/api', controller.router);
    });
```

```
}
```

Функція «initialiseErrorHandling» встановлює обробник помилок для програми Express , використовуючи власний обробник помилок «ErrorMiddleware»:

```
private initialiseErrorHandling(): void {
    this.express.use(ErrorMiddleware);
}
```

Функція «initialiseDatabaseConnection» встановлює підключення до бази даних MongoDB, використовуючи значення змінних середовища, таких як «MONGO_USER», «MONGO_PASSWORD» та «MONGO_PATH», та метод «mongoose.connect»:

```
private initialiseDatabaseConnection(): void {
    const { MONGO_USER, MONGO_PASSWORD, MONGO_PATH } = process.env;
    mongoose.connect(
`mongodb+srv://${MONGO_USER}:${MONGO_PASSWORD}${MONGO_PATH}`);
}
```

Таким чином, дане рішення створює гнучкий та масштабований Expressдодаток з підключенням до бази даних MongoDB, налаштуванням необхідного проміжного програмного забезпечення та контролерів для обробки маршрутів API.

Наступне рішення реалізоване для завантаження файлів на сервер за допомогою бібліотеки Multer, оскільки вона добре інтегрується з Express, зручно оброблює файли, дозволяє легко налаштувати спосіб їх збереження, має валідацію та підтримує завантаження багатьох типів файлів [10]. Функція «fileFilter» проводить фільтрацію завантажуваних файлів, перевіряючи MIMEтип файла та дозволяє завантаження лише файлів типу JPEG та PNG. Константи «memoryStorage» та «upload» використовуються для тимчасового зберігання файлів у пам’яті:

```
    const fileFilter = (req: Request, file: Express.Multer.File, cb:
FileFilterCallback) => {
        if (file.mimetype === 'image/jpeg' || file.mimetype === 'image/
png') {
            cb(null, true);
        } else {
            cb(new Error('Only JPEG and PNG files are allowed'));
        }
    };
    const memoryStorage = multer.memoryStorage();
    const upload = multer({storage: memoryStorage, fileFilter:
fileFilter});
```

Для асинхронного збереження файлів використовується бібліотека fs та path, а для генерації назви файлу було використано бібліотеку crypto:

```
    public async saveFile(file: Express.Multer.File): Promise<string | Error> {
        try {
            const fileExtension = extname(file.originalname);
                                    const fileUniqueSuffix = crypto.randomBytes(16).toString('hex');
                        const fileNewOriginalname = fileUniqueSuffix + fileExtension;
                await fs.promises.writeFile(`${this.uploadsDestination}/${fileNewOriginalname}`, file.buffer, 'binary');
            return `${this.uploadsDestination}/${fileNewOriginalname}`;
        } catch (error: any) {
            throw new Error(error.message);
        }
    }
    public async removeFile(path: string): Promise<void | Error> {
        try {
            await fs.promises.access(path, fs.constants.F_OK);
            await fs.promises.unlink(path);
        } catch (error: any) {}
    }
```

Таким чином, за допомогою Express та Multer вдається завантажувати файли, зберігаючи їх у пам’яті та оброблюючи за допомогою функцій для збереження та видалення файлів на сервері. Приклад збережених файлів зображено на рисунку 4.1.

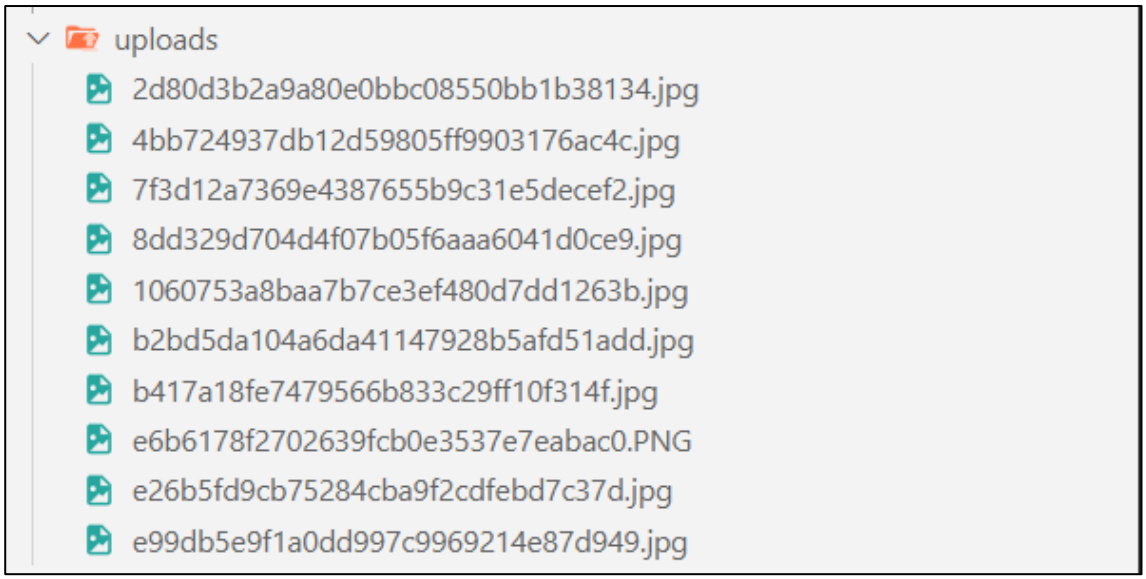


Рисунок 4.1 – Збережені файли на сервері (рисунок виконаний самостійно)

Для виведення зображення виконується запит до URL серверу та шляху до збереженого зображення, наприклад: «http://localhost:5000/uploads/file_name.jpg».

Для перевірки автентифікації користувача було створено middleware, який витягує з запиту токен доступу:

```
const accessToken = bearer.split('Bearer ')[1].trim();
```

Далі, шляхом декодування токена, отримується ідентифікатор користувача та проводиться пошук його в базі даних:

```
const payloadVerifyAccessToken: Token | jwt.JsonWebTokenError = await token.verifyAccessToken(accessToken);
if (payloadVerifyAccessToken instanceof jwt.JsonWebTokenError) {
    return next(new HttpException(401, 'Unauthorized'));
}
const user = await UserModel.findById(payloadVerifyAccessToken.id).select(['-password']);
if (!user) {
    return next(new HttpException(401, 'Unauthorized'));
}
req.user = user;
```

Такий підхід дозволяє контролювати та валідувати токени, забезпечуючи безпеку додатку. У цьому випадку, запити, які може виконувати лише автентифікований адміністратор, будуть доступні тільки для нього.

За допомогою бібліотеки Joi на сервері налаштовані не лише звичайні запити, що валідують вхідні параметри, а й middleware, який валідує дані вхідного запиту. Функція «validationMiddleware» отримує схему Joi у якості аргументу та повертає middleware для Express. У середині нього об’єднуються різноманітні частини запиту (тіло, параметри та файли) у єдиний об’єкт «data». У випадку наявності JSON-строк вони перетворюються на об’єкт, а далі проводиться валідація. Якщо вона пройшла успішно, то middleware передає дані запиту через «req.properties». У випадку, коли валідація не пройшла, викидається відповідна помилка у блоці «catch»:

```
const data = {...req.body,...req.query,...req.params,files: req.files,file: req.file,};
for (const key in data) {
    if (data.hasOwnProperty(key)) {
        try {
            data[key] = JSON.parse(data[key]);
        } catch (error) {}
    }
}
const value = await schema.validateAsync(data, validationOptions);
```

```
req.properties = value;
next();
```

Цей middleware забезпечує валідацію даних запиту на основі заданої схеми та надсилає повідомлення про помилки, якщо дані не відповідають цій схемі.

Для забезпечення безпеки користувачів системи було прийняте рішення про хешування паролів за допомогою бібліотеки bcrypt. Під час збереження паролю, отримане значення через тіло запиту передається в функцію моделі бази даних, яка відпрацьовує при збереженні нової сутності. З поля паролю витягується значення та хешується з додатковим додаванням «солі».

```
UserSchema.pre<User>('findOneAndUpdate', async function (this) {
    const update: any = { ...this.getUpdate() };
    if (update.password) {
        update.password = await bcrypt.hash(update.password, 10);
        this.setUpdate(update);
    }});
```

Такий підхід забезпечує надійність зберігання конфіденційної інформації користувачів, навіть у випадку отримання зловмисниками доступу до бази даних.

Для масштабованості клієнтської частини та мобільного програмного застосунку усі елементи, що повторюються у коді декілька разів, були винесені у папку компонентів та імпортовані до основних сторінок проєкту через один файл (див. рис. 4.2):

```
export { default as RatingStars } from "./rating-stars";
export { default as RaviewElement } from "./review-element";
export { default as MultiRange } from "./multi-range";
import { RatingStars, RaviewElement } from "../../components";
```

Рисунок 4.2 – Архітектура клієнтської частини та мобільного програмного застосунку (рисунок виконаний самостійно)

Такий підхід дозволяє уникнути надмірності коду та налаштувати стилі для усіх компонентів разом.

Для підключення до бекенду використовується проксі, а для виконання запитів використовується «fetch» з використанням кастомного хука «useHttp», що отримує на вхід URL запиту, метод, тіло запиту та заголовки, а повертає результат запиту, можливі помилки та булевий статус, що показує виконаний запит чи ні:

```
async (url, method = "GET", body = null, headers = {}) => {
      setLoading(true);
      try {
        if (body) {
          body = JSON.stringify(body);
          headers["Content-Type"] = "application/json";
        }
        const response = await fetch(url, { method,body,headers });
        const data = await response.json();
        if (!response.ok) {
          throw new Error(data.message || "Something went wrong");
        }
        setLoading(false);
        return data;
      } catch (e) {
        setLoading(false);
        setError(e.message);
        throw e;
```

```
        }
      },
      []
);
const clearError = useCallback(() => setError(null), []);
return { loading, request, error, clearError };
```

Цей код дозволяє легко викликати запити без надмірного забруднення коду, а також забезпечує додаткову валідацію помилок.

5 ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

5.1 Підхід до тестування ПЗ

Тестування має охоплювати всі функції системи, включаючи перегляд інформації про коворкінги, бронювання робочих місць, залишення відгуків та оцінок, пошук та фільтрацію коворкінгів, а також управління коворкінгами.

Враховуючи невеликий розмір проекту і комплексність функціональності було обрано ручне тестування, воно дозволить забезпечити більшу увагу до деталей та виявлення найбільш критичних проблем. Такий підхід також сприятиме швидшому виявленню та виправленню помилок, оскільки ручне тестування дозволяє тестувальнику активно взаємодіяти з системою і швидко реагувати на виявлені проблеми.

Крім того, в процесі ручного тестування можна здійснювати творчий підхід до пошуку неочевидних проблем або вразливостей, що може бути особливо корисним у випадку складних функцій або взаємодії різних компонентів системи.

5.2 Тестування ПЗ

Ручне тестування цієї програмної системи вимагає структурованого підходу для покриття всіх функціональних і нефункціональних вимог, зазначених у специфікації (див. додаток Е). Тестування програмного забезпечення складалося з наступних кроків:

а) планування:

1) складено тест-план (див. додаток Ж);
2) розроблено баг-репорт (див. додаток И);

б) перевірка функціональних вимог:

1) перегляд інформації про коворкінги:

– перевірено відображення інформації про коворкінги, включаючи фотографії, опис, послуги, адресу та інші деталі;

2) бронювання робочих місць:

– здійснено бронювання робочих місць з різними параметрами і перевірено, що було отримано талон, що підтверджує бронювання, на електронну пошту;

3) залишення відгуків та оцінок:

– додано відгук і оцінку для коворкінгів і перевірено, що вони відображаються правильно;

4) пошук та фільтрація:

– перевірено роботу функціоналу пошуку та фільтрації за різними параметрами;

в) перевірка нефункціональних вимог:

1) безпека:

– перевірено, що пароль зберігається в БД в вигляді хешу;

– автентифікація відбувається за допомогою токену, який має обмежений термін дії;

2) зручність використання:

– перевірено, що програмне забезпечення пристосовується до різних розмірів екранів (для веб-застосунків) та різних розширень екранів (для мобільних додатків);

– перевірено, що у разі помилки при заповненні полів для вводу користувач отримує повідомлення про це;

3) тестування на різних платформах:

– перевірено роботу веб-версії системи на різних веббраузерах, таких як Google Chrome, Mozilla Firefox, Edge та Safari;

– перевірено мобільний додаток на різних мобільних пристроях з операційними системами Android та iOS, щоб переконатися, що він працює належним чином на різних пристроях.

Цей підхід до тестування програмної системи є ефективним з кількох причин.

По-перше, структурований підхід, що базується на ретельному плануванні та розробці тест-плану та тест-кейсів, дозволяє забезпечити покриття всіх функціональних та нефункціональних вимог, вказаних у специфікації.

По-друге, проведення перевірки функціональних вимог, таких як перегляд інформації, бронювання робочих місць, залишення відгуків та оцінок, пошук та фільтрація, допомагає переконатися у коректному функціонуванні системи.

По-третє, перевірка нефункціональних вимог, таких як безпека та зручність використання, є не менш важливою, і цей підхід дозволяє систематично перевірити їх виконання.

Нарешті, тестування на різних платформах, включаючи різні веб-браузери та мобільні пристрої з різними операційними системами, підвищує рівень впевненості у тому, що програмне забезпечення працює належним чином у різних умовах.

ВИСНОВКИ

Під час виконання кваліфікаційної роботи було створено комплексну програмну систему, що включає наступні компоненти:

– клієнтська частина;

– мобільний програмний застосунок;

– серверна частина.

Серверна частина програмної системи розроблена на платформі Node.js з використанням мови програмування TypeScript, що забезпечує надійність і масштабованість системи. Для створення клієнтської частини застосовано бібліотеку React та мову програмування JavaScript, що дозволяє створювати інтерактивні та зручні у використанні інтерфейси. Мобільний додаток реалізовано з використанням мови програмування JavaScript та фреймворку React Native, що забезпечує високу продуктивність та кросплатформеність. Управління базами даних здійснюється за допомогою документо-орієнтованої системи MongoDB, яка дозволяє ефективно зберігати та обробляти дані.

Розроблена програмна система надає можливість не лише бронювати робочі місця для окремих користувачів, але й для цілих груп, що шукають зручний спосіб організації спільного простору. Крім того, система дозволяє адміністраторам коворкінгів автоматизувати процеси бронювання та керування простором, що значно підвищує ефективність їхньої роботи. Завдяки мобільному додатку користувачі можуть швидко та легко знаходити доступні робочі місця, що особливо важливо в сучасному ритмі життя, де мобільність і швидкий доступ до інформації є ключовими факторами успіху.

Мета кваліфікаційної роботи була успішно досягнута – створена програмна система для бронювання коворкінгів, яка включає серверну частину, клієнтську частину та мобільний додаток. Адміністратори коворкінгів користуватимуться спеціалізованою частиною клієнтського інтерфейсу, тоді як відвідувачі матимуть доступ до загальнодоступної клієнтської частини та

мобільного додатку. Обробка даних, отриманих з клієнтської частини та мобільного додатку, здійснюватиметься на серверній частині системи.

Отже, мета кваліфікаційної роботи бакалавра – розробка програмної системи для бронювання коворкінгів – виконана у повному обсязі.

Основні задачі розробки програмної системи виконані.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Європейська Бізнес Асоціація. EBA.COM.UA (дата звернення: 25.04.2024).

2. Михайлова Р. Коворкінг як модерне явище (на прикладі України) / Р. Михайлова, І. В. Антоненко, М. Різніченко // Актуальні проблеми сучасного дизайну : збірник матеріалів IV Міжнародної науково-практичної конференції, м. Київ, 27 квітня 2022 року. – В 2-х т. – Т. 2. – Київ : КНУТД, 2022. – С. 212-215.

3. Перетворюємо офіс на коворкінг: як це працює в Німеччині. USENKOV.PRO. URL: https://usenkov.pro/page6894986.html (дата звернення: 26.04.2024).

4. Documentation - TypeScript for the New Programmer. TypeScript: JavaScript With Syntax For Types. URL: https://www.typescriptlang.org/docs/handbook/typescript-from-scratch.html (дата звернення: 26.04.2024).

5. Index | Node.js v22.0.0 Documentation. Node.js – Run JavaScript Everywhere. URL: https://nodejs.org/docs/latest/api/ (дата звернення: 26.04.2024).

6. Team M. D. MongoDB Documentation. *MongoDB: The Developer Data Platform | MongoDB*. URL: https://www.mongodb.com/docs/ (дата звернення: 26.04.2024).

7. Getting Started – React. *React – A JavaScript library for building user interfaces*. URL: https://legacy.reactjs.org/docs/getting-started.html (дата звернення: 26.04.2024).

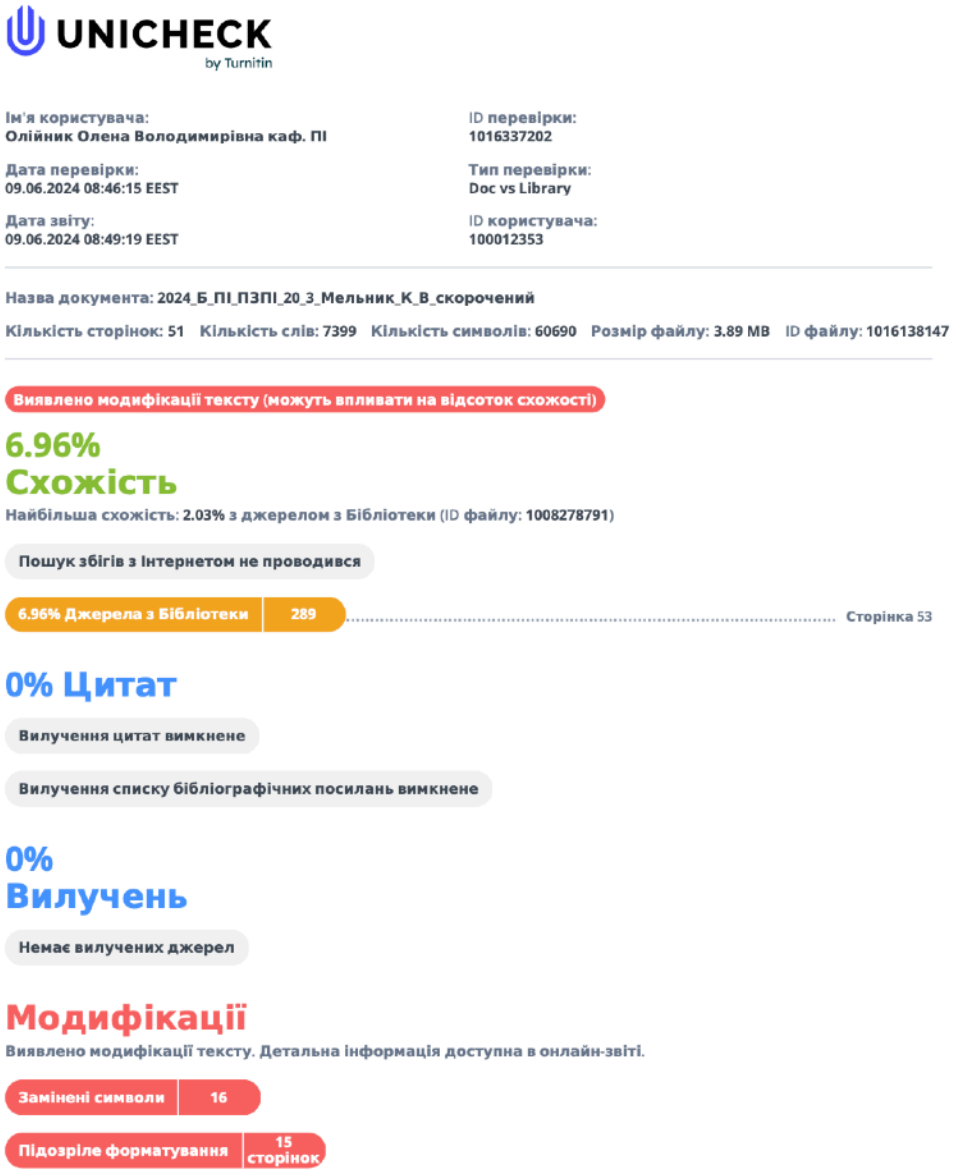
8. Introduction · React Native. *React Native · Learn once, write anywhere*. URL: https://reactnative.dev/docs/getting-started (дата звернення: 26.04.2024).

9. Express - Node.js web application framework. *Express - Node.js web application framework*. URL: https://expressjs.com/ (дата звернення: 26.04.2024).

10. Express multer middleware. *Express - Node.js web application framework*. URL: https://expressjs.com/en/resources/middleware/multer.html (дата звернення: 26.04.2024).

ДОДАТОК А

Звіт результатів перевірки на унікальність тексту в базі ХНУРЕ



UNICHECK
by Turnitin

Ім'я користувача:
Олійник Олена Володимирівна каф. ПІ

ID перевірки:
1016337202

Дата перевірки:
09.06.2024 08:46:15 EEST

Тип перевірки:
Doc vs Library

Дата звіту:
09.06.2024 08:49:19 EEST

ID користувача:
100012353

Назва документа: 2024_Б_ПІ_ПЗПІ_20_3_Мельник_К_В_скорочений

Кількість сторінок: 51 Кількість слів: 7399 Кількість символів: 60690 Розмір файлу: 3.89 MB ID файлу: 1016138147

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

6.96%
Схожість

Найбільша схожість: 2.03% з джерелом з Бібліотеки (ID файлу: 1008278791)

Пошук збігів з Інтернетом не проводився

6.96% Джерела з Бібліотеки 289 ..... Сторінка 53

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи 16

Підозріле форматування 15 сторінок

ДОДАТОК Б

Слайди презентації

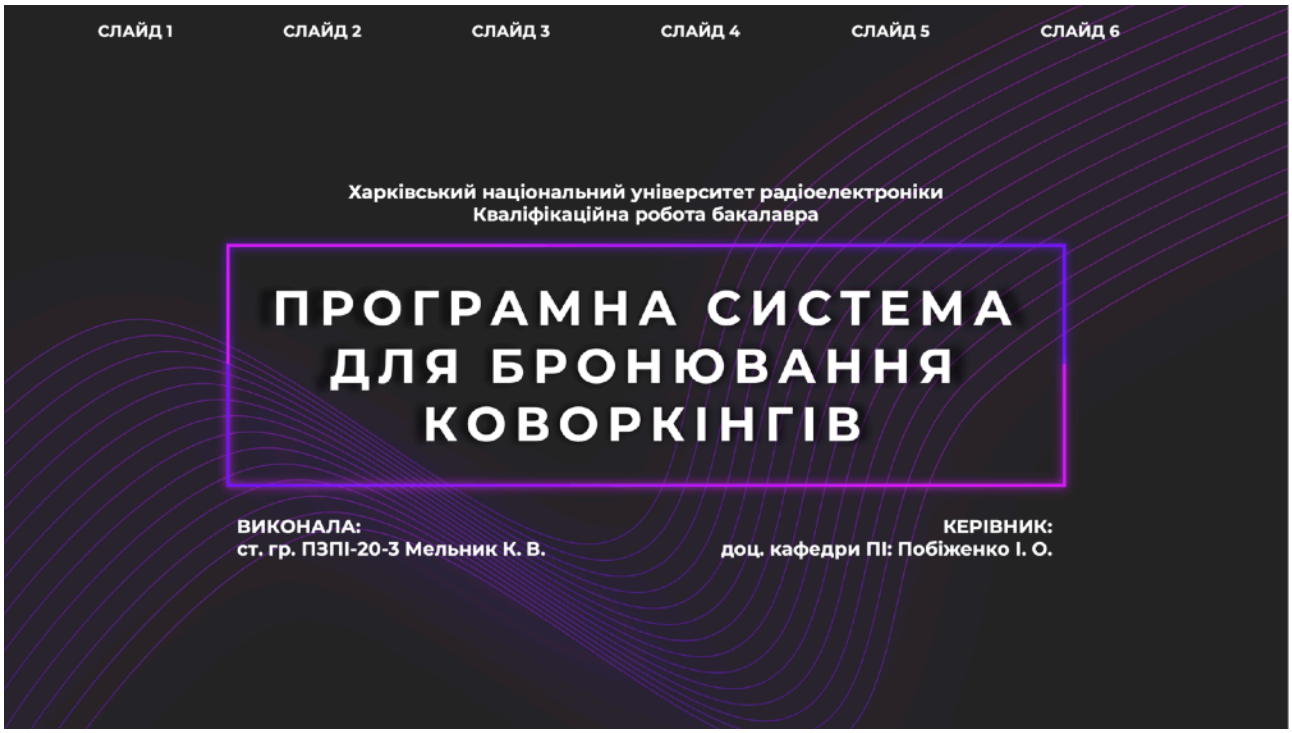


Рисунок Б.1 – Перший слайд презентації (рисунок виконаний самостійно)

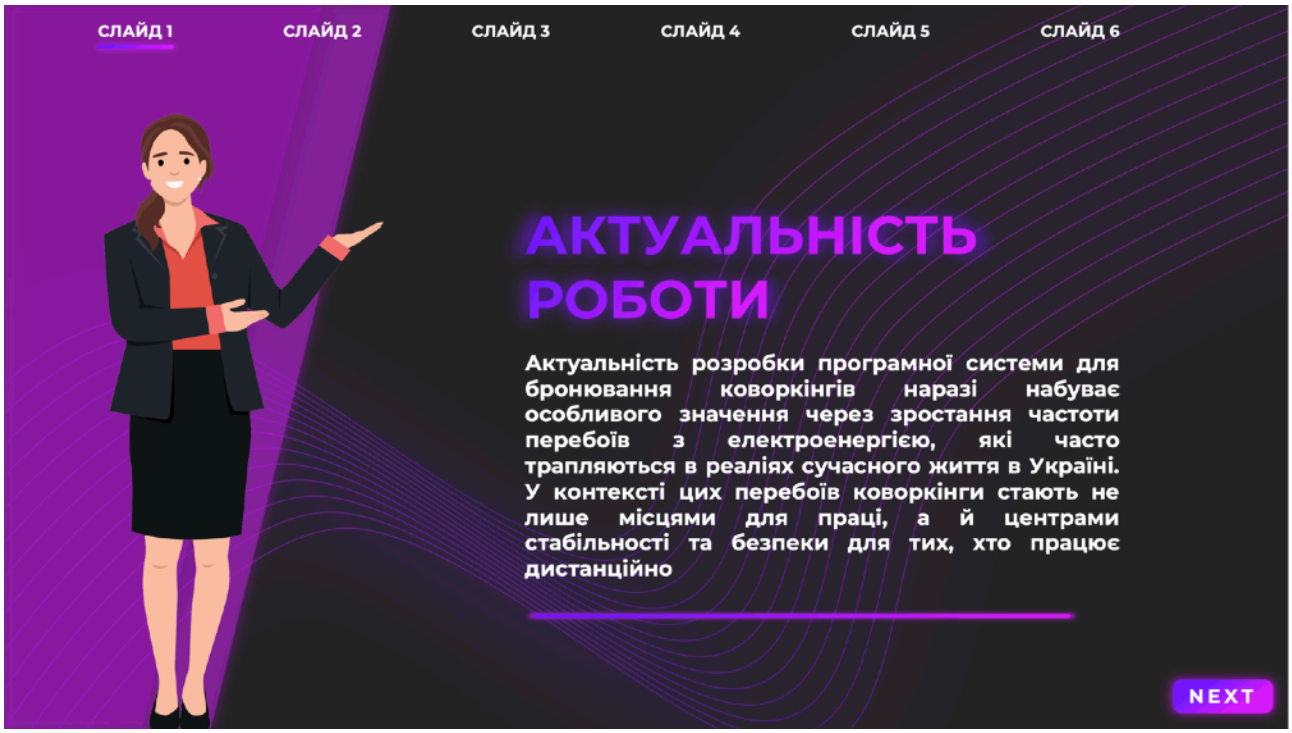


Рисунок Б.2 – Другий слайд презентації (рисунок виконаний самостійно)

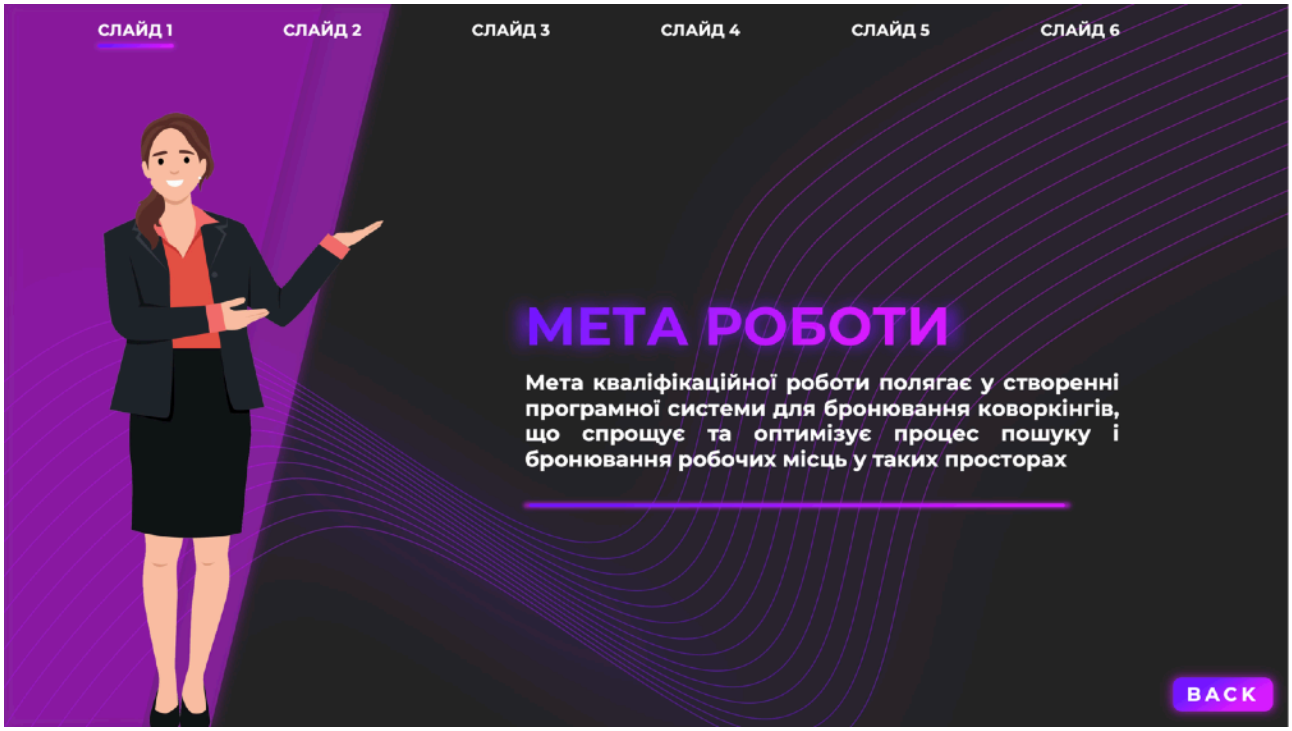


Рисунок Б.3 – Третій слайд презентації (рисунок виконаний самостійно)

Рисунок Б.4 – Четвертий слайд презентації (рисунок виконаний самостійно)

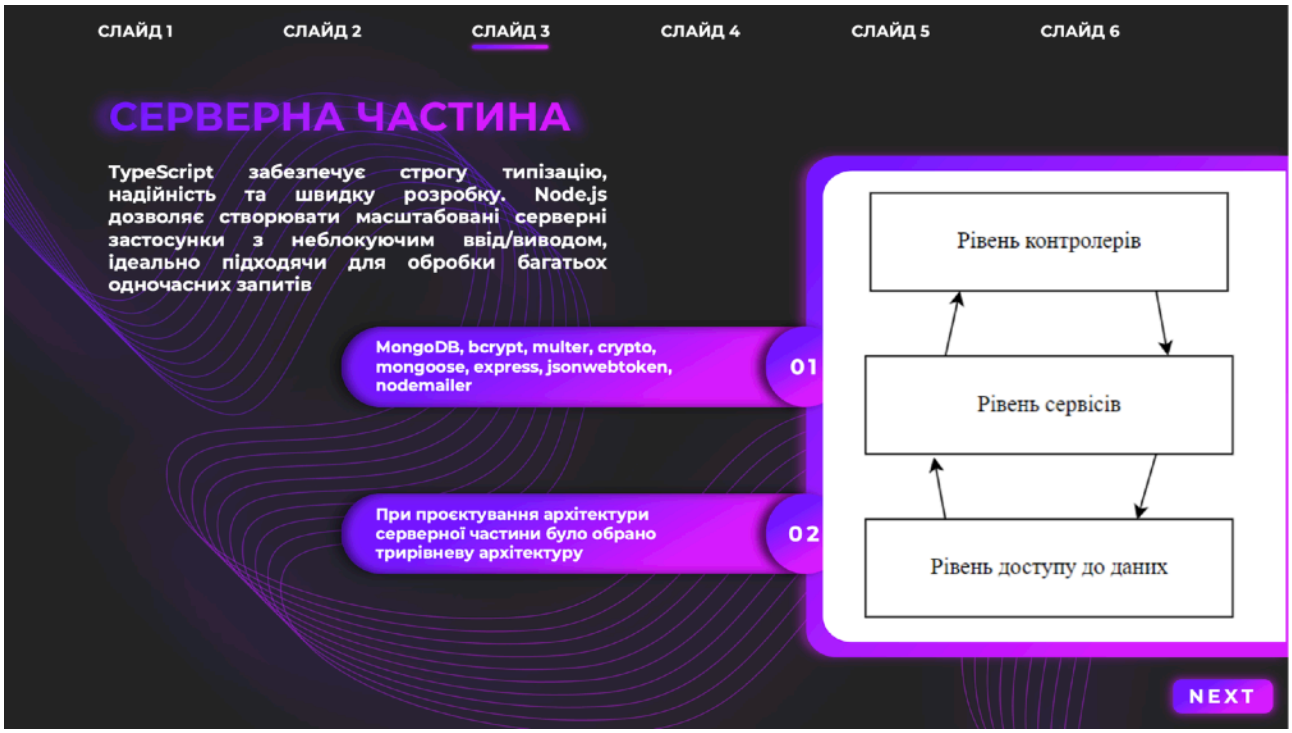


Рисунок Б.5 – П'ятий слайд презентації (рисунок виконаний самостійно)

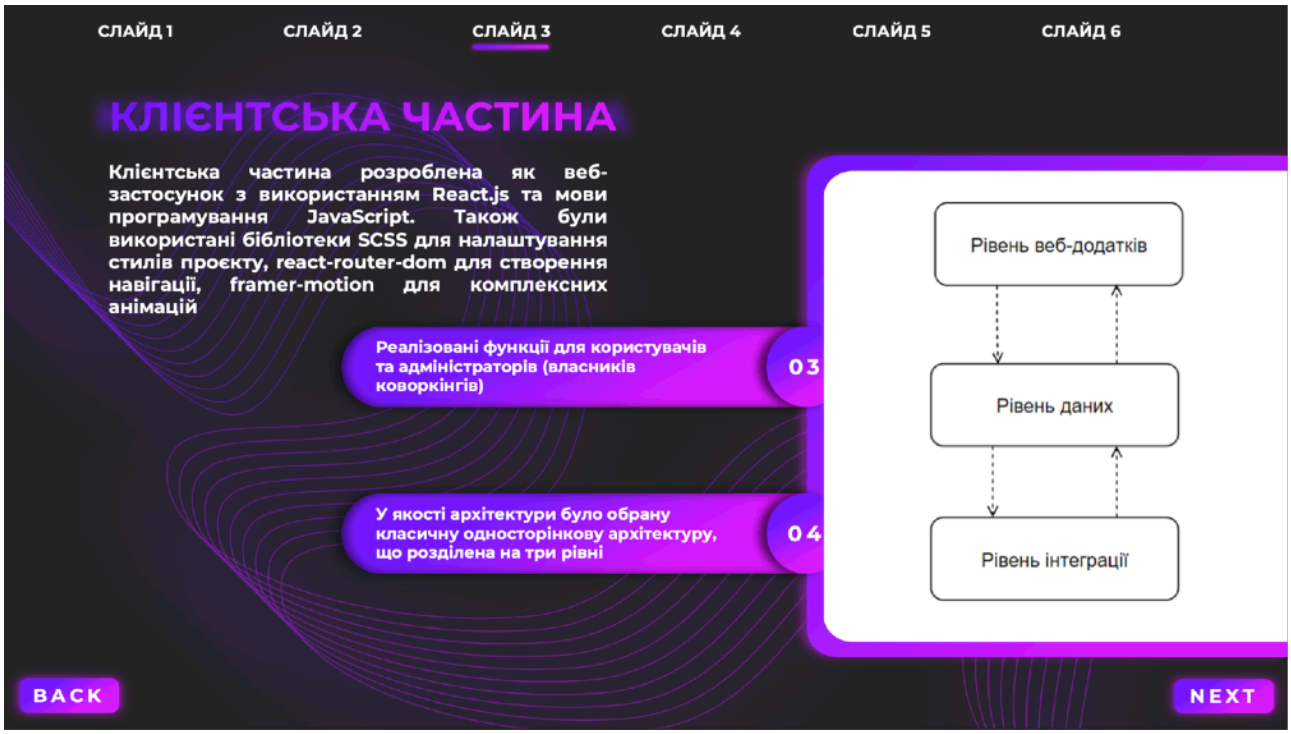


Рисунок Б.6 – Шостий слайд презентації (рисунок виконаний самостійно)

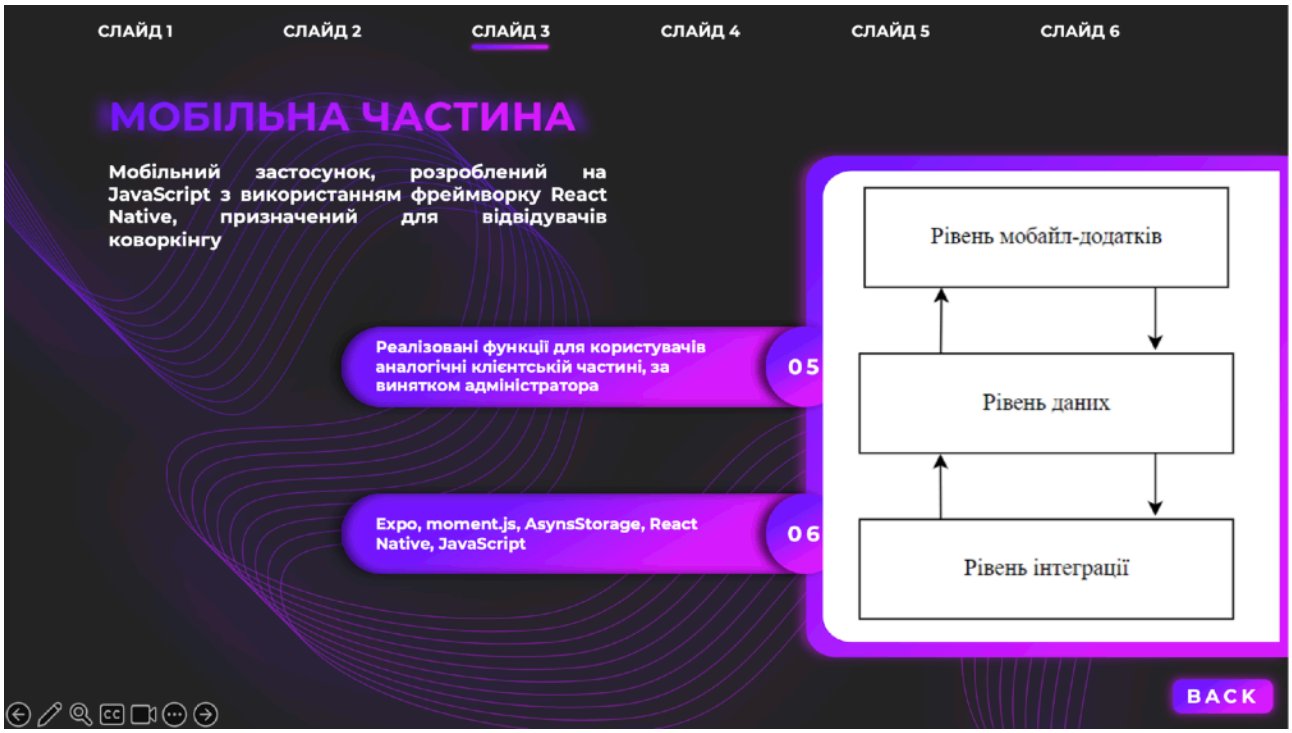


Рисунок Б.7 – Сьомий слайд презентації (рисунок виконаний самостійно)

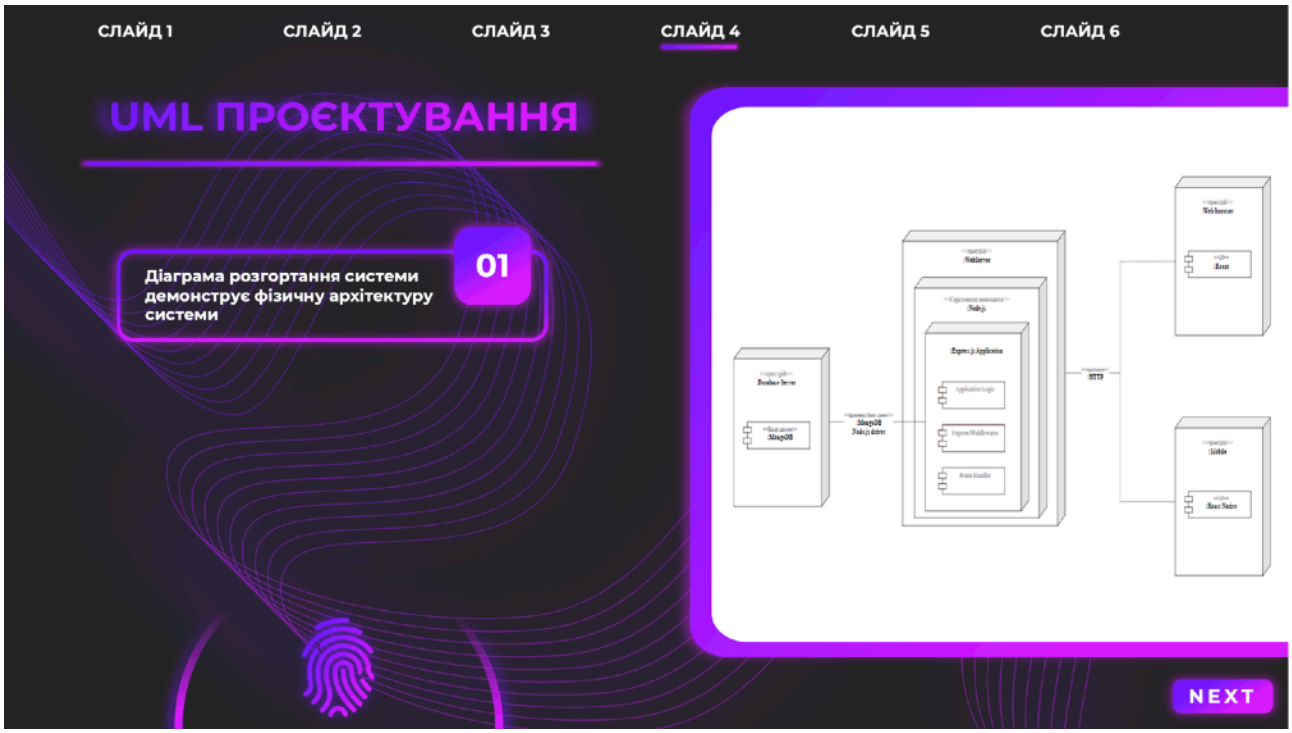


Рисунок Б.8 – Восьмий слайд презентації (рисунок виконаний самостійно)

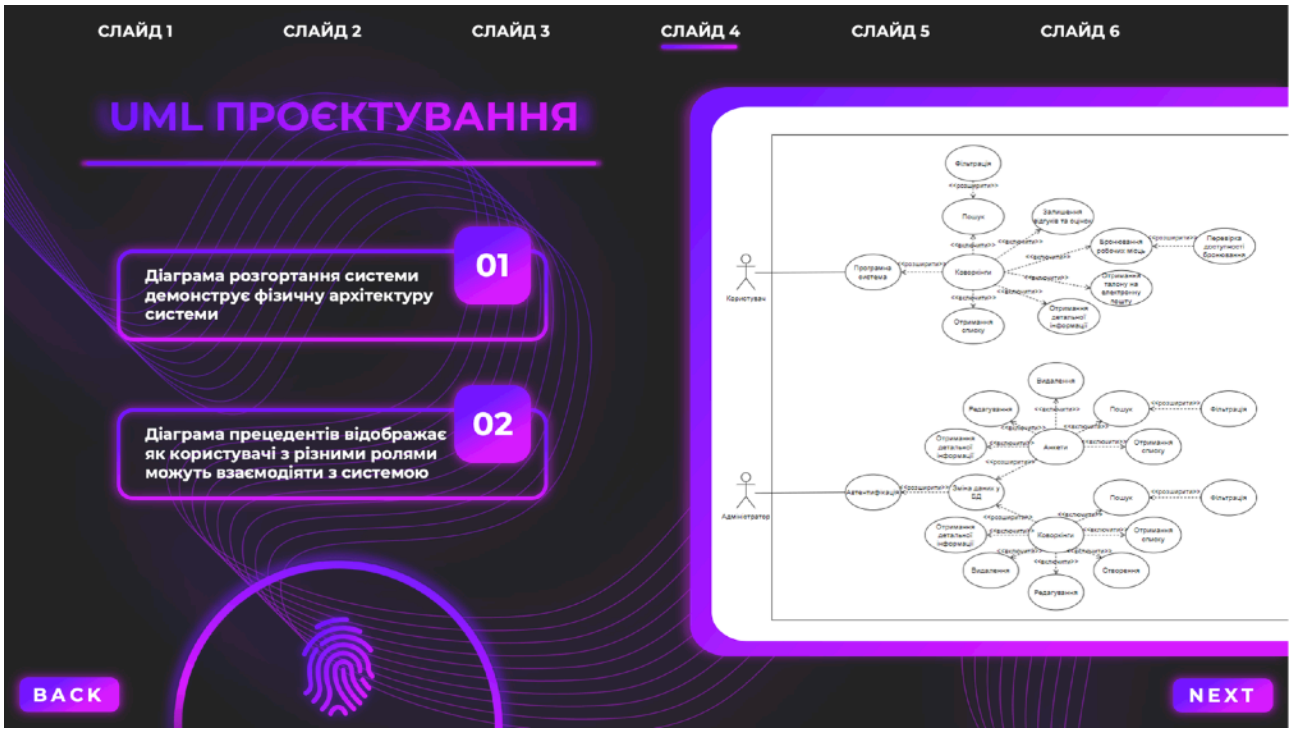


Рисунок Б.9 – Дев'ятий слайд презентації (рисунок виконаний самостійно)

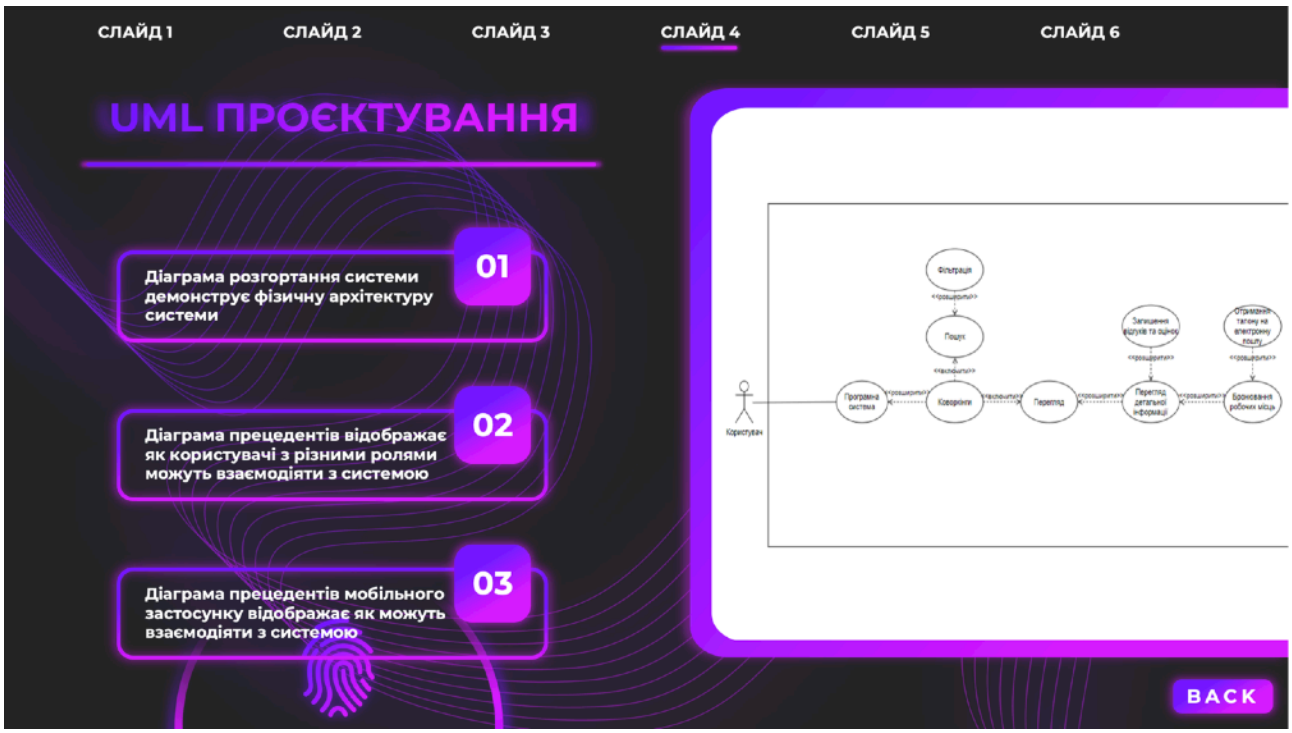


Рисунок Б.10 – Десятий слайд презентації (рисунок виконаний самостійно)

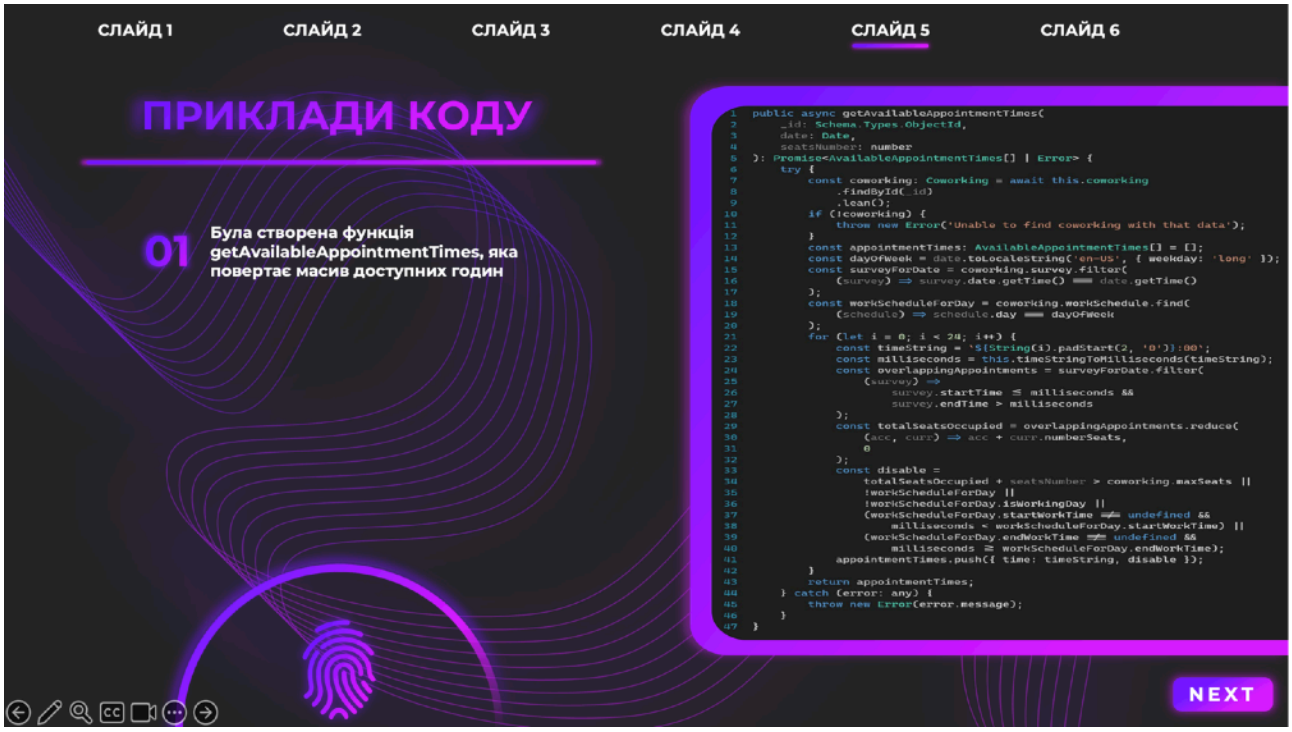


Рисунок Б.11 – Одинадцятий слайд презентації (рисунок виконаний самостійно)

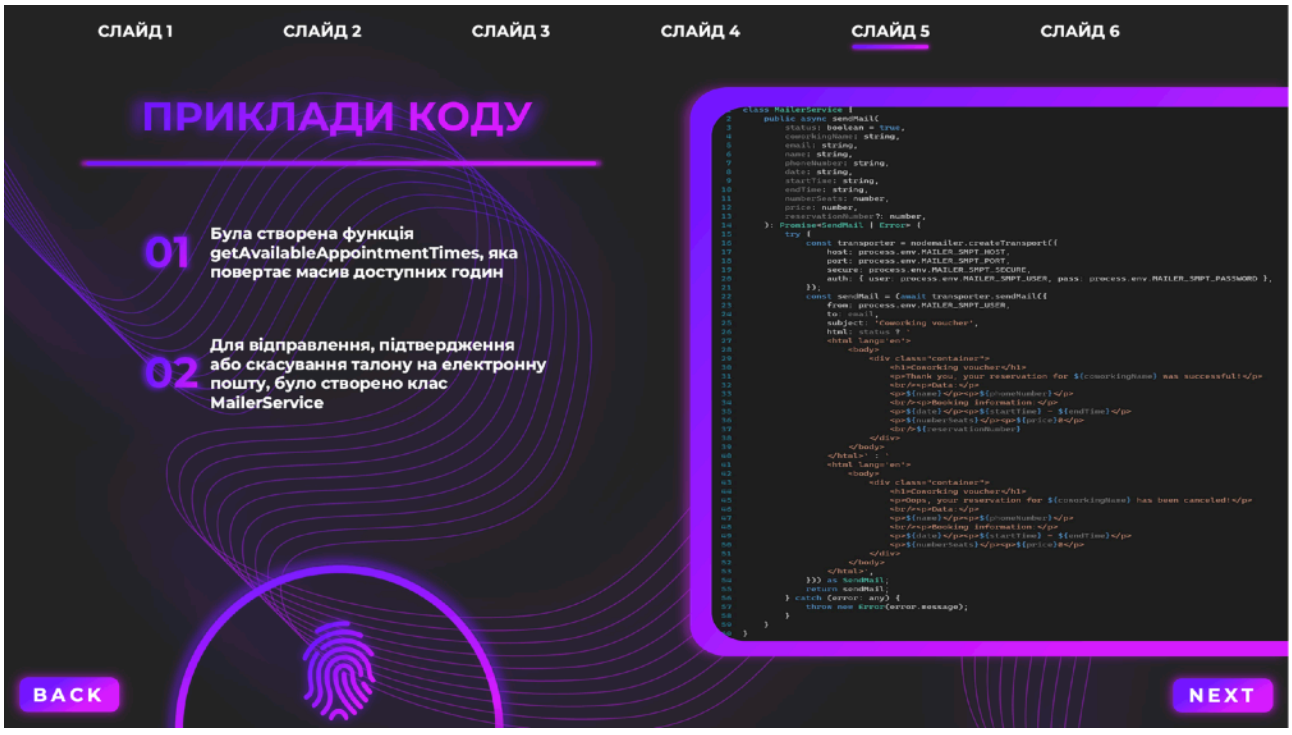


Рисунок Б.12 – Дванадцятий слайд презентації (рисунок виконаний самостійно)

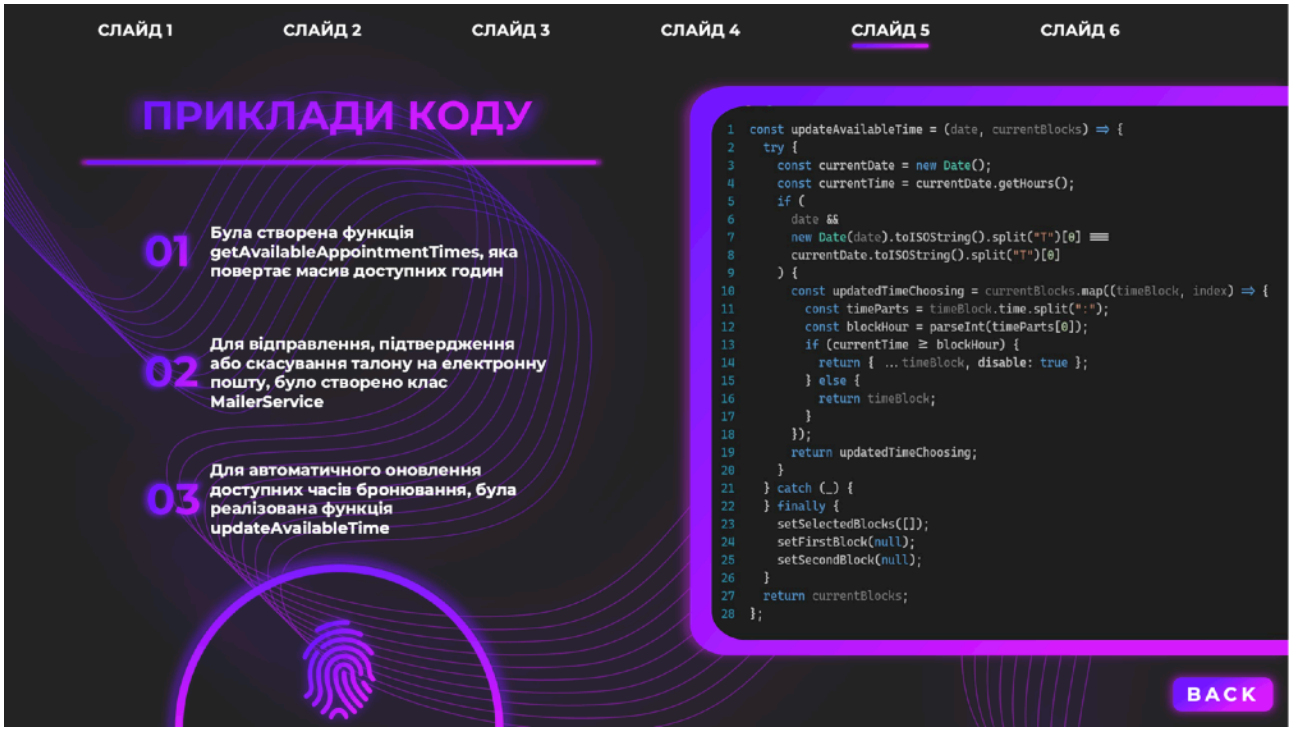


Рисунок Б.13 – Тринадцятий слайд презентації (рисунок виконаний самостійно)

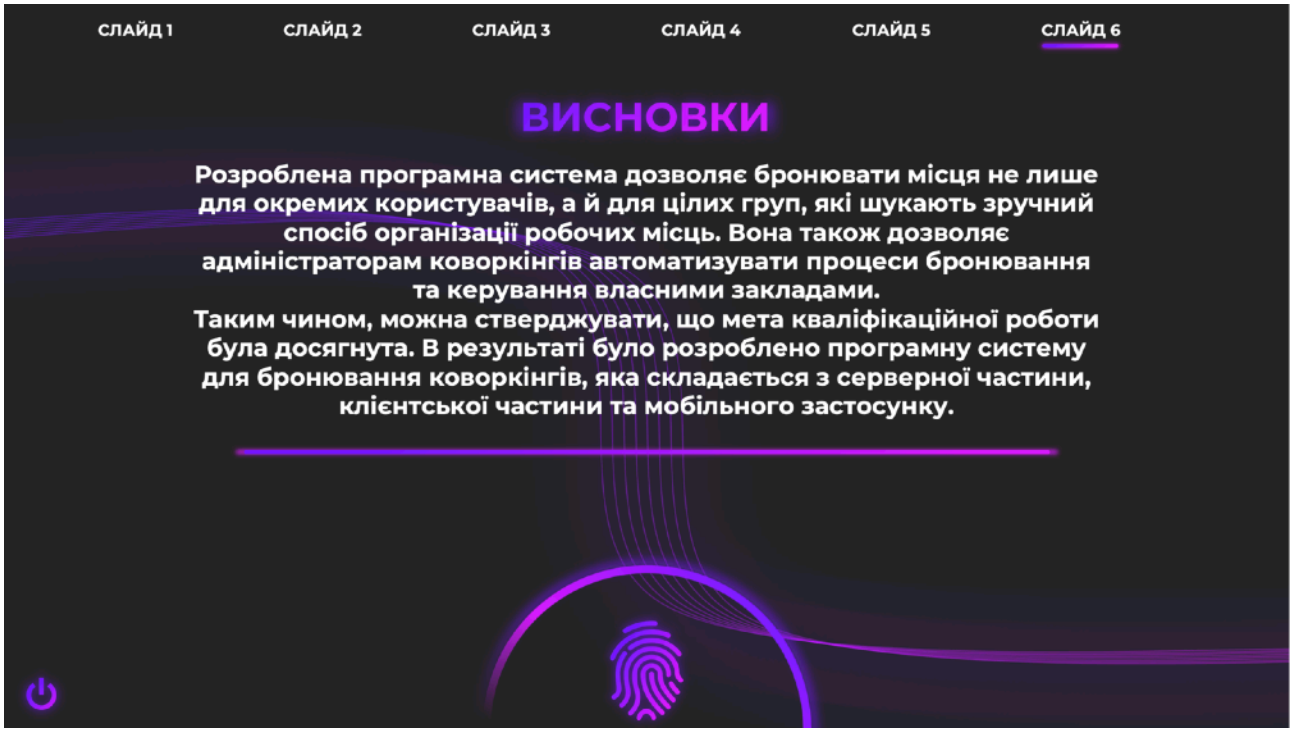


Рисунок Б.14 – Чотирнадцятий слайд презентації (рисунок виконаний самостійно)

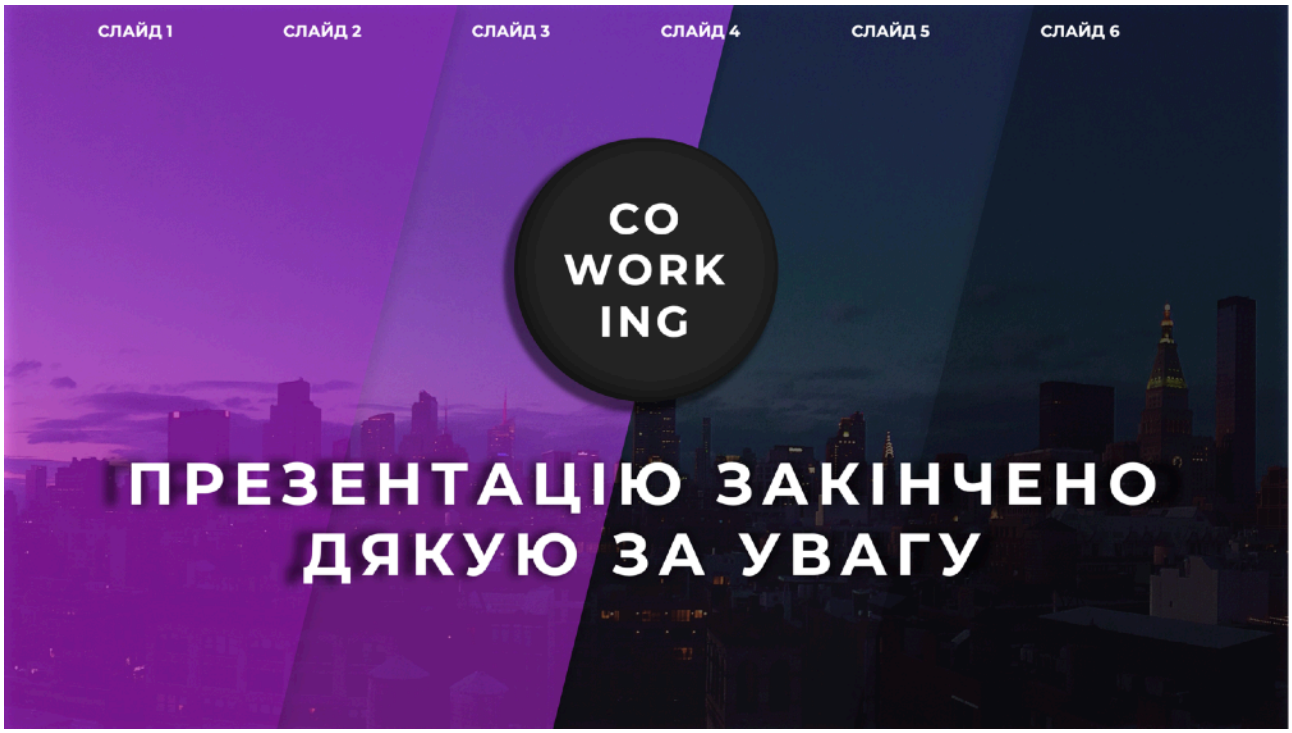


Рисунок Б.15 – П'ятнадцятий слайд презентації (рисунок виконаний самостійно)

ДОДАТОК В

Функція на серверній частині проєкту для перевірки доступності бронювання робочого місця

```
public async getAvailableAppointmentTimes(
_id: Schema.Types.ObjectId,
date: Date,
seatsNumber: number
): Promise<AvailableAppointmentTimes[] | Error> {
try {
const coworking: Coworking = await this.coworking
.findById(_id)
.lean();
if (!coworking) {
throw new Error('Unable to find coworking with that
data');}
const appointmentTimes: AvailableAppointmentTimes[] = [];
const dayOfWeek = date.toLocaleString('en-US', { weekday:
'long' });
const surveyForDate = coworking.survey.filter(
(survey) => survey.date.getTime() === date.getTime() );
const workScheduleForDay = coworking.workSchedule.find(
(schedule) => schedule.day === dayOfWeek );
for (let i = 0; i < 24; i++) {
const timeString = `${String(i).padStart(2, '0')}:00`;
const milliseconds =
this.timeStringToMilliseconds(timeString);
const overlappingAppointments = surveyForDate.filter(
(survey) =>
survey.startTime <= milliseconds &&
survey.endTime > milliseconds );
const totalSeatsOccupied =
overlappingAppointments.reduce(
```

```
30   (acc, curr) => acc + curr.numberSeats, 0 );
31   const disable =
32   totalSeatsOccupied + seatsNumber > coworking.maxSeats ||
33   !workScheduleForDay ||
34   !workScheduleForDay.isWorkingDay ||
35   (workScheduleForDay.startWorkTime !== undefined &&
36   milliseconds < workScheduleForDay.startWorkTime) ||
37   (workScheduleForDay.endWorkTime !== undefined &&
38   milliseconds >= workScheduleForDay.endWorkTime);
39   appointmentTimes.push({ time: timeString, disable });
40   }
41   return appointmentTimes;
42   } catch (error: any) {
43   throw new Error(error.message);
44   }
45   }
```

ДОДАТОК Г

Клас на серверній частині проєкту для відправлення талону на електронну пошту

```
1    class MailerService {
2    public async sendMail(
3    status: boolean = true,
4    coworkingName: string,
5    email: string,
6    name: string,
7    phoneNumber: string,
8    date: string,
9    startTime: string,
10   endTime: string,
11   numberSeats: number,
12   price: number,
13   reservationNumber?: number,
14   ): Promise<SendMail | Error> {
15   try {
16   const transporter = nodemailer.createTransport({
17   host: process.env.MAILER_SMPT_HOST,
18   port: process.env.MAILER_SMPT_PORT,
19   secure: process.env.MAILER_SMPT_SECURE,
20   auth: {
21   user: process.env.MAILER_SMPT_USER,
22   pass: process.env.MAILER_SMPT_PASSWORD,
23   },
24   });
25   const sendMail = (await transporter.sendMail({
26   from: process.env.MAILER_SMPT_USER,
27   to: email,
28   subject: 'Coworking voucher',
29   html: status ? `
30   <html lang='en'>
```

```
31   <body>
32   <div class="container">
33   <h1>Coworking voucher</h1>
34   <p>Thank you, your reservation for ${coworkingName} was
35   successful!</p>
36   <br/>
37   <p>Data:</p>
38   <p>${name}</p>
39   <p>${phoneNumber}</p>
40   <br/>
41   <p>Booking information:</p>
42   <p>${date}</p>
43   <p>${startTime} - ${endTime}</p>
44   <p>${numberSeats}</p>
45   <p>${price}₴</p>
46   <br/>
47   ${reservationNumber}
48   </div>
49   </body>
50   </html>
51   `
52   : `
53   <html lang='en'>
54   <body>
55   <div class="container">
56   <h1>Coworking voucher</h1>
57   <p>Oops, your reservation for ${coworkingName} has been
58   canceled!</p>
59   <br/>
60   <p>Data:</p>
61   <p>${name}</p>
62   <p>${phoneNumber}</p>
63   <br/>
64   <p>Booking information:</p>
65   <p>${date}</p>
```

```
66   <p>${startTime} - ${endTime}</p>
67   <p>${numberSeats}</p>
68   <p>${price}₴</p>
69   </div>
70   </body>
71   </html>
72   `,
73   })) as SendMail;
74   return sendMail;
75   } catch (error: any) {
76   throw new Error(error.message);
77   }
78   }
79   }
80   export default MailerService;
```

ДОДАТОК Д

Функція на клієнтській частині проєкту та мобільному програмному застосунку для оновлення доступного часу бронювання

```
1    const updateAvailableTime = (date, currentBlocks) => {
2    try {
3    const currentDate = new Date();
4    const currentTime = currentDate.getHours();
5    if (
6    date &&
7    new Date(date).toISOString().split("T")[0] ===
8    currentDate.toISOString().split(«T")[0]  ) {
9    const updatedTimeChoosing = currentBlocks.map((timeBlock,
10   index) => {
11   const timeParts = timeBlock.time.split(":");
12   const blockHour = parseInt(timeParts[0]);
13   if (currentTime >= blockHour) {
14   return { ...timeBlock, disable: true };
15   } else {
16   return timeBlock;
17   }
18   });
19   return updatedTimeChoosing;
20   }
21   } catch (_) {
22   } finally {
23   setSelectedBlocks([]);
24   setFirstBlock(null);
25   setSecondBlock(null);
26   }
27   return currentBlocks;
28   };
29   useEffect(() => {
```

```
30   const updateAvailableTimeHandler = () => {
31   const updatedTimeChoosing = updateAvailableTime(
32   survey.date,
33   availableTimeChoosing );
34   if ( JSON.stringify(updatedTimeChoosing) !==
35   JSON.stringify(availableTimeChoosing)
36   ) {
37   setAvailableTimeChoosing(updatedTimeChoosing);
38   setAvailableTime(updatedTimeChoosing);
39   }
40   const currentDate = new Date();
41   const secondsUntilNextHour =
42   3600 - (currentDate.getMinutes() * 60 +
43   currentDate.getSeconds());
44   const timeUntilNextHour = secondsUntilNextHour * 1000;
45   setTimeout(updateAvailableTimeHandler,
46   timeUntilNextHour);
47   };
48   if (survey.date && survey.date !== "" &&
49   survey.numberSeats > 0) {
50   updateAvailableTimeHandler();
51   }
52   }, [survey.date, survey.numberSeats,
53   availableTimeChoosing]);
```

ДОДАТОК Е

Специфікація програмної системи для бронювання коворкінгів

ЗМІСТ

1 ВСТУП

1.1 Огляд продукту

Програмна система для бронювання коворкінгів призначена для забезпечення зручного та ефективного інструменту для користувачів у пошуку, перегляді та бронюванні робочих місць у коворкінгах. Система спрощує процес вибору коворкінгу шляхом швидкого доступу до необхідної інформації та можливості бронювання в реальному часі.

1.2 Мета

Метою цього SRS документу вимог є представлення вичерпного опису вимог до програмного забезпечення для розробки системи бронювання робочих місць у коворкінгах. Цей документ створений для інженера-програміста та інших учасників проекту з метою належного розуміння потреб та функціональних вимог системи.

1.3 Межі

Програмна системи ідентифікується під назвою «Система бронювання робочих місць у коворкінгах».

Програмна система для бронювання коворкінгів повинна надавати інструмент для ефективного управління коворкінгами та їх бронюваннями з боку адміністраторів, а також у спрощенні процесу перегляду та бронювання для відвідувачів.

Конкретні цілі системи включають:

а) для користувачів:

– переглядати інформацію про коворкінги;

– бронювати робочі місця у коворкінгах;

– залишати відгуки та оцінки;

– здійснювати пошук та фільтрацію коворкінгів.

а) для адміністраторів:

– управління інформацією про коворкінги;

– обробка запитів на бронювання робочих місць від користувачів;

– отримання зворотного зв’язку від користувачів через відгуки.

Програмна система повинна складатися з серверної частини, клієнтської частини та мобільного додатку.

Серверна частина повинна обробляти дані, які були відправлені з клієнтського веб-застосунку та мобільного додатку.

Мобільний додаток призначений для відвідувачів коворкінгу. Вони повинні шукати й переглядати коворкінги, бронювати в них місця та залишати відгуки.

Клієнтський веб-застосунок призначений для адміністраторів коворкінгів і відвідувачів. Він повинен мати окремі видимі частини для відвідувачів і невидимі для адміністраторів для правильного функціонування системи в цілому.

1.4 Посилання

Перелік документів, на які є посилання в інших частинах SRS документу або в окремому, визначеному документі, відсутній. SRS документ не містить інших посилань.

1.5 Означення та абревіатури

SRS – software requirements specification (специфікація вимог до програмного забезпечення)

2 ЗАГАЛЬНИЙ ОПИС

2.1 Перспективи продукту

Програмна система для бронювання коворкінгів – це сучасний інструмент, розроблений з урахуванням потреб сучасних підприємців, фрілансерів та команд, які шукають ефективний простір для роботи.

Основні переваги та перспективи продукту:

– система дозволяє користувачам легко і швидко знаходити та бронювати вільні робочі місця або конференц-зали в коворкінгах у будь-якому місті.

– користувачі можуть вибирати місце розташування, тип робочого місця (відкрите просторове або приватне кабінетне), а також часовий інтервал бронювання.

– завдяки системі коворкінги можуть легко розширювати свою клієнтську базу, а користувачі отримують доступ до все більшого числа якісних робочих просторів.

Загалом, програмна система для бронювання коворкінгів відкриває нові можливості для організації робочого простору, сприяючи ефективності та комфорту користувачів.

2.2 Функції продукту

Програмна система для бронювання коворкінгів має ряд функцій, які полегшують процес бронювання та управління коворкінгом для його власників і користувачів. Ось декілька ключових функцій, які можна реалізувати:

– онлайн-календар для перегляду доступності робочих місць;

– можливість вибору кількості місць;

– пошук та фільрація коворкінгів;

– додавання, видалення та редагування коворкінгів;

– можливість оцінити та залишити коментар на сторінці коворкінгу;

– мобільний додаток для бронювання та управління коворкінгом з будь-якого місця.

2.3 Характеристики користувачів

Користувачі системи для бронювання коворкінгів:

– адміністратор, має можливість додавати товари та видаляти коворкінги, редагувати інформацію про них;

– користувач, має можливість бронювати місця в коворкінгах, залишати коментарі, використовувати пошукову систему коворкінгів.

2.4 Загальні обмеження

Ось переклад обмежень для програмної системи для бронювання коворкінгів:

Використання бібліотеки React для реалізації фронтенду. Обмеження обумовлене тим, що ця технологія надає великий набір ресурсів для розробки веб-додатків. Веб-додатки, написані з використанням цієї бібліотеки, відзначаються високою продуктивністю, оскільки React взаємодіє зі своїм легковажним еквівалентом - віртуальним DOM, замість повільних та незручних взаємодій безпосередньо з реальним DOM. Реальний DOM оновлюється лише після взаємодії з віртуальним DOM. Також можлива повторна використання компонентів, що скорочує час розробки.

Використання середовища Node.js для реалізації бекенду. Обмеження обумовлене тим, що серверна частина, написана за допомогою Node.js, має

високу продуктивність. Використання JavaScript є зручним як на серверній, так і на клієнтській стороні. Також наявні асинхронні бібліотеки, що є дуже корисним, оскільки сервери Node.js не чекають відповіді від API, а переходять до наступного запиту.

Використання платформи React Native для реалізації фронтенду. Обмеження обумовлене тим, що додатки, розроблені на React Native, виглядають однаково на обох платформах - iOS і Android. Код додатку також є однаковим. За допомогою React Native можливе повторне використання коду, що прискорює розробку додатків.

Використання СКБД MongoDB. Обмеження обумовлене тим, що у MongoDB є чітка структура для кожного об'єкта, вона легко масштабується, а дані зберігаються у вигляді JSON-документів. Також ця база даних базується на колекціях різних документів. Кількість полів, їх зміст і розмір цих документів може варіюватися. Тому різні сутності не повинні бути ідентичними за структурою. Вона підходить, якщо структура бази даних змінюється під час розробки.

2.5 Припущення й залежності

Щоб використовувати веб-програму на пристрої, потрібен браузер із доступом до Інтернету.

Для використання мобільного додатку потрібен пристрій Android/iOS з доступом до Інтернету.

3 КОНКРЕТНІ ВИМОГИ

3.1 Вимоги до зовнішніх інтерфейсів

3.1.1 Інтерфейс користувача

При заході користувача на сайт, він переходить на головну сторінку (див. рис. 3.1 та див. рис. 3.2), де може вибрати одну з двох опцій: переглянути список всіх коворкінгів або перейти на сторінку конкретного коворкінгу.

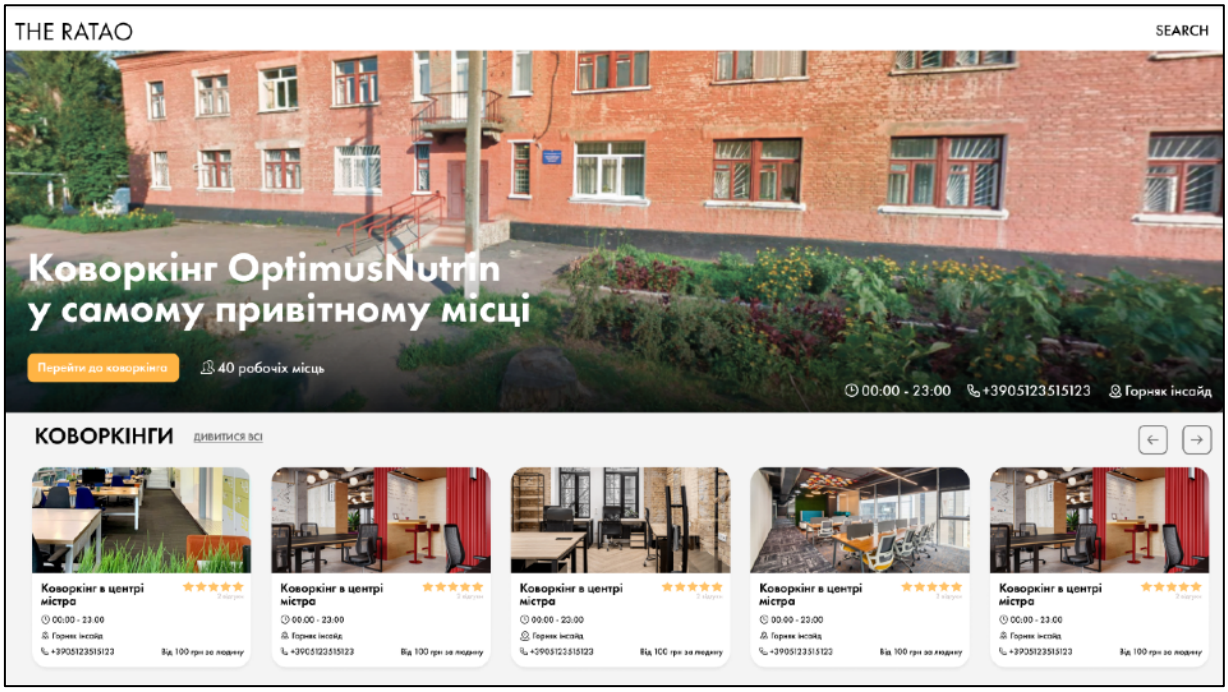


Рисунок 3.1 – Головна сторінка (рисунок виконаний самостійно)

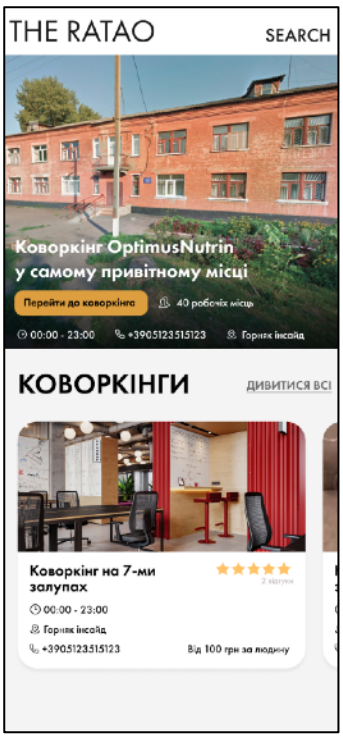


Рисунок 3.2 – Головна сторінка на смартфоні (рисунок виконаний самостійно)

Якщо користувач натискає кнопку «Дивитися всі» на головній сторінці, він потрапить на сторінку зі всіма коворкінгами (див. рис. 3.3 та див. рис. 3.4).

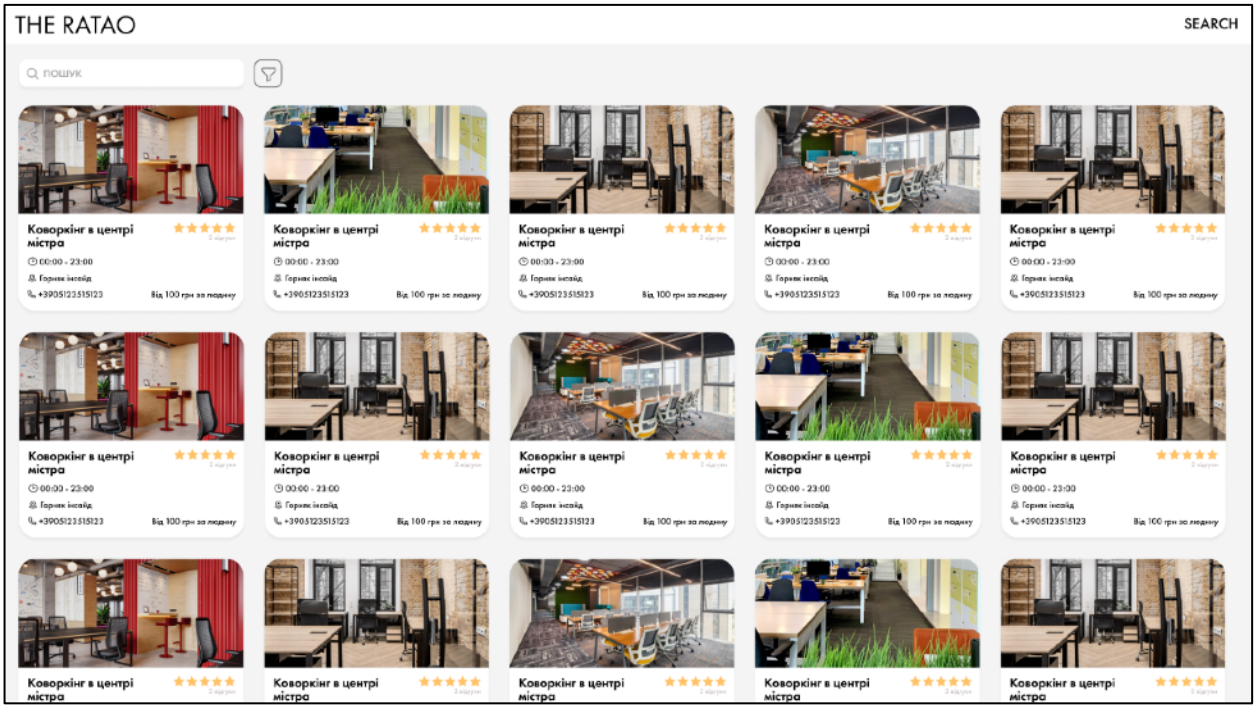


Рисунок 3.3 – Сторінка зі списком усіх коворкінгів (рисунок виконаний самостійно)

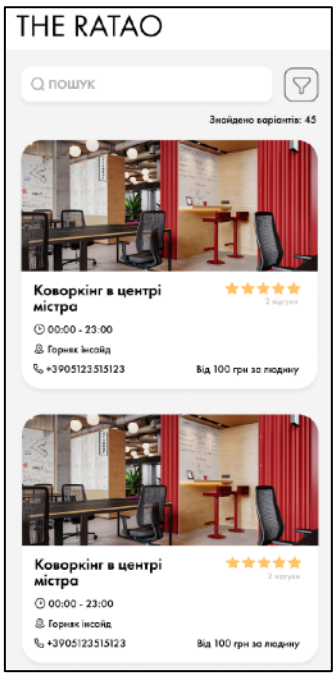


Рисунок 3.4 – Сторінка зі списком усіх коворкінгів на смартфоні (рисунок виконаний самостійно)

На сторінці зі всіма коворкінгами користувач може здійснити пошук та фільтрацію закладів за допомогою модального вінка (див. рис. 3.5), а також перейти на сторінку конкретного коворкінгу (див. рис. 3.6 та рис. 3.7)

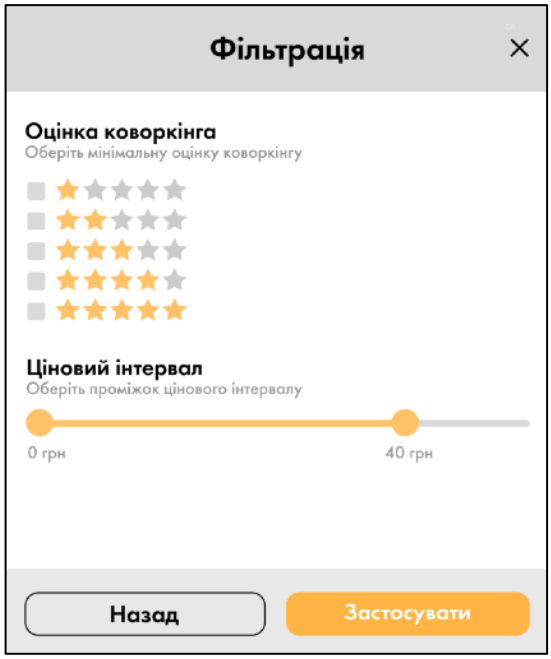


Рисунок 3.5 – Модальне вікно для фільтрації коворкінгів(рисунок виконаний самостійно)

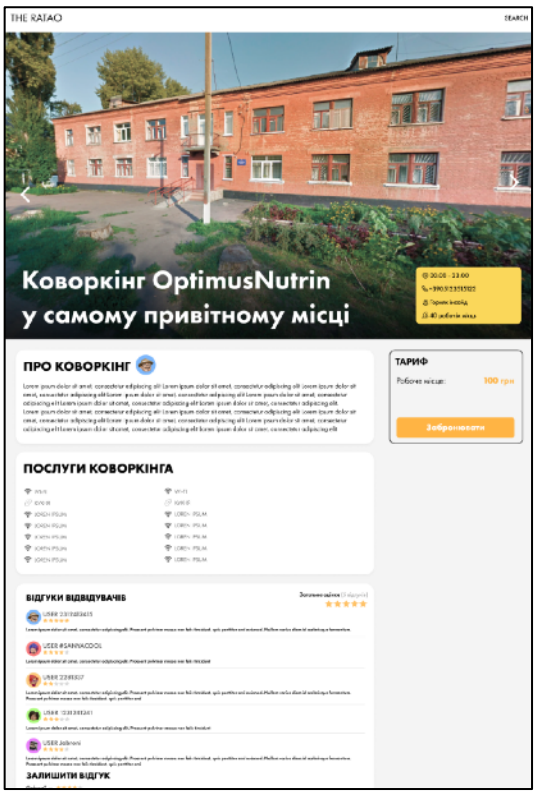


Рисунок 3.7 – Сторінка конкретного коворкінгу (рисунок виконаний самостійно)

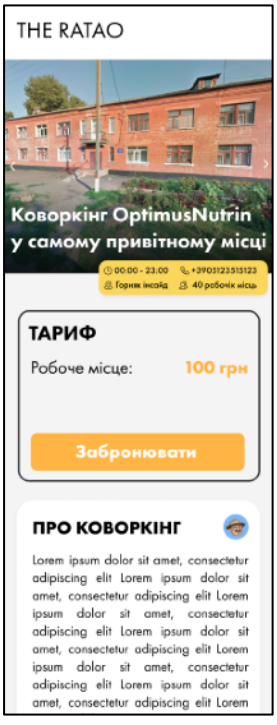


Рисунок 3.8 – Сторінка конкретного коворкінгу на смартфоні (рисунок виконаний самостійно)

Якщо користувач натискає кнопку «Забронювати» на сторінці конкретного коворкінгу, виводиться модальне вікно, в якому він може забронювати місця у обраному коворкінгу (див. рис. 3.9).

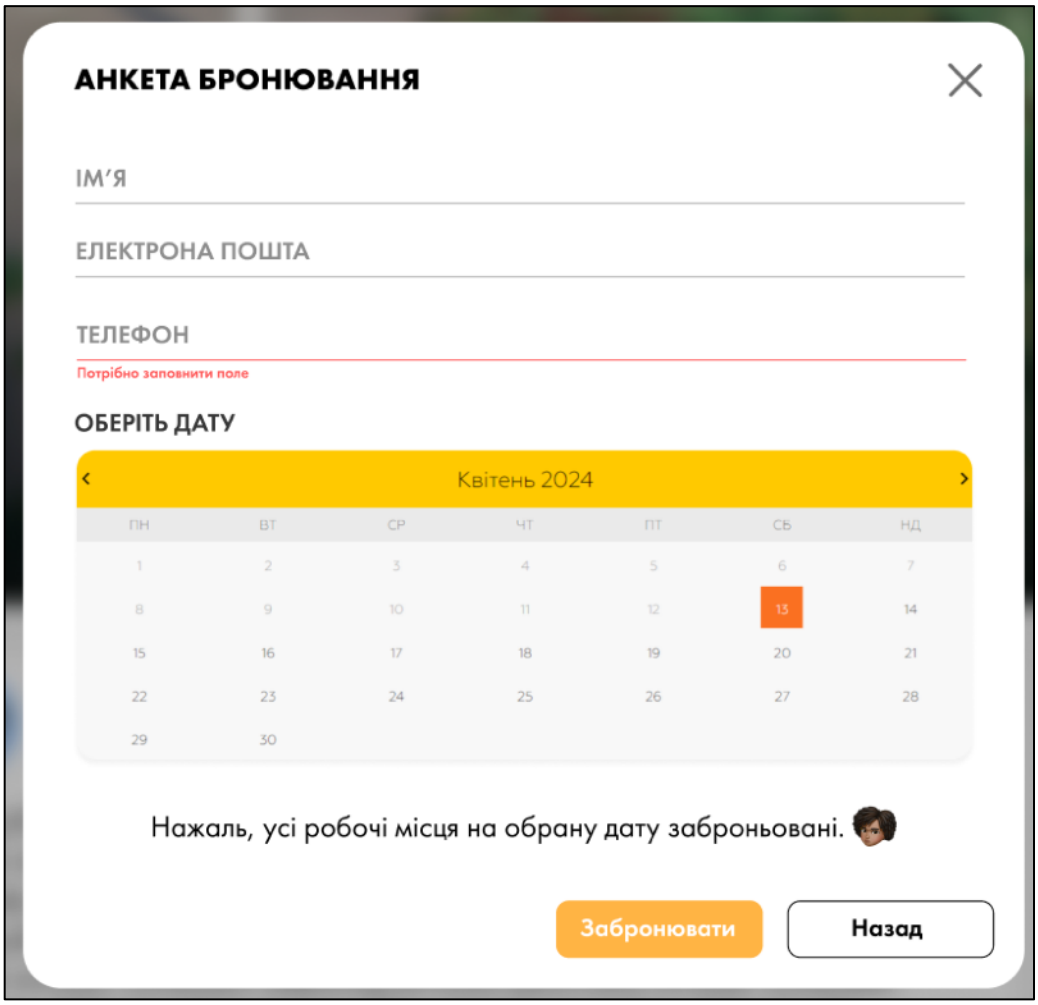


Рисунок 3.9 – Модальне вікно бронювання місць у коворкінгу (рисунок виконаний самостійно)

3.1.2 Апаратний інтерфейс

З набору апаратних інтерфейсів потрібні базові пристрої введення/ виведення.

3.1.3 Програмний інтерфейс

Користувач повинен мати браузер для використання веб-програми або смартфон/планшет/емулятор для використання мобільної програми.

3.1.4 Комунікаційний протокол

Додаток не потребує спеціальних інтерфейсів.

3.1.5 Обмеження пам’яті

Система повинна ефективно використовувати пам’ять пристроїв користувачів та серверів, забезпечуючи оптимальну продуктивність та швидкий доступ до інформації.

3.1.6 Операції

Для користувачів:

– користувач може переглянути список усіх коворкінгів, які знаходяться у системі. Для цього він може відкрити додаток та переглянути нові коворкінги на головній сторінці додатку, а також натиснути кнопку «Дивитися всі», щоб перейти на список усіх доступних коворкінгів;

– користувач може здійснювати пошук коворкінгів за містом та їх назвою, а також фільтрацію за рейтингом коворкінгів, мінімальною та максимальною цінами. Для цього він на головній сторінці натискає кнопку «Дивитися всі». На цій сторінці є поле «Пошук», де користувач може ввести свій запит, а також кнопка «Фільтрація», що відкриває модальне вікно з фільтрацією. Користувач може обрати за допомогою чекбоксів декілька оцінок для фільтрації коворкінгів, а також за допомогою елементу «діапазон вибору» обрати мінімальну та максимальну ціни для фільтрації коворкінгів. Після цього для застосування обраних параметрів фільтрації він може натиснути кнопку «Застосувати». У разі потреби очистити ці поля, він може натиснути кнопку «Назад» або хрестик у куті екрану;

– користувач може переглядати детальну інформацію про коворкінги, для цього він може перейти до неї з головної сторінки чи зі сторінки зі списком усіх коворкінгів, натиснувши на певний елемент списку;

– користувач може залишити відгук та оцінку про коворкінг. Для цього він може перейти до сторінки конкретного коворкінгу з головної сторінки чи зі сторінки зі списком усіх коворкінгів, натиснувши на певний елемент списку, написати свій відгук, встановити оцінку від одного до п’яти та натиснути кнопку «Відправити»;

– користувач може забронювати коворкінг на певну дату та час. Для цього він може перейти до сторінки конкретного коворкінгу з головної сторінки чи зі сторінки зі списком усіх коворкінгів, натиснувши на певний елемент списку, далі на сторінці елементу натиснути кнопку «Забронювати». У відкритому модальному вікні користувач вводить ім’я, номер телефону, електрону пошту, кількість місць для бронювання, обирає дату та відповідний час, якщо вони доступні, далі він натискає кнопку «Забронювати». Через деякий час, коли адміністратор підтвердить бронювання, користувач отримає повідомлення на вказану в анкеті електрону пошту з талоном підтвердження чи скасування бронювання.

Для адміністраторів (власників коворкінгів):

– адміністратор може авторизуватися на адміністративній панелі, використовуючи електронну пошту та пароль, заповнивши їх у відповідні полі

– авторизований адміністратор системи може переглядати список усіх коворкінгів, обравши у навігаційній панелі «Коворкінги»;

– авторизований адміністратор може переглянути детальну інформацію про коворкінги, перейшовши до пункту «Коворкінги» у навігаційній панелі та обравши потрібний коворкінг зі списку;

– авторизований адміністратор може здійснювати пошук коворкінгів, перейшовши до пункту «Коворкінги» у навігаційній панелі та вказавши назву або місто коворкінгу до відповідного поля пошуку;

– авторизований адміністратор може створити нові коворкінги, перейшовши до пункту «Коворкінги» у навігаційній панелі та натиснувши кнопку «Створити». Він вводить наступні поля: назва, регіон, місто, адреса, номер телефону, електронна пошта, опис, ціна, максимальна кількість місць, сервіси, що пропонує коворкінг, робочий час та додає фотографії. Після введення всіх полів, він натискає кнопку «Зберегти»;

– авторизований адміністратор може редагувати існуючі коворкінги, перейшовши до пункту «Коворкінги» у навігаційній панелі, обравши потрібний коворкінг зі списку та натиснувши кнопку «Змінити». Він може змінити наступні поля: назва, регіон, місто, адреса, номер телефону, електронна пошта, опис, ціна, максимальна кількість місць, сервіси, що пропонує коворкінг, робочий час та фотографії. Після внесення всіх змін, він натискає кнопку «Зберегти»;

– авторизований адміністратор може видалити коворкінг, перейшовши до пункту «Коворкінги» у навігаційній панелі, обравши потрібний коворкінг зі списку та натиснувши кнопку «Видалити»;

– авторизований адміністратор системи може переглядати список усіх анкет, обравши у навігаційній панелі «Анкети»;

– авторизований адміністратор може переглянути детальну інформацію про анкети, перейшовши до пункту «Анкети» у навігаційній панелі та обравши потрібну анкету зі списку;

– авторизований адміністратор може здійснювати пошук та фільтрацію анкет, перейшовши до пункту «Анкети» у навігаційній панелі, вказавши назву коворкінгу у відповідне поле та обравши статус бронювання з випадаючого списку;

– авторизований адміністратор може підтверджувати бронювання, перейшовши до пункту «Анкети» у навігаційній панелі, обравши потрібну анкету зі списку та натиснувши кнопку «Змінити». Він може змінити її статус та натиснути кнопку «Зберегти»;

– авторизований адміністратор може скасувати бронювання, перейшовши до пункту «Анкети» у навігаційній панелі, обравши потрібну анкету зі списку та натиснувши кнопку «Видалити».

3.1.7 Функції продукту

а) перегляд інформації про коворкінги:

1) відображення детальної інформації про кожен коворкінг, включаючи назву, фотографії, опис, послуги, час роботи, номер телефону, адреса, кількість робочих місць, тариф та відгуки користувачів;

2) перегляд списку доступних коворкінгів;

б) бронювання робочих місць:

1) заповнення інформації для бронювання робочого місця в обраному коворкінгу, включаючи ім’я, номер телефону, електронну пошту, дату, час та кількість робочих місць;

2) визначення доступності місць на обрану дату та обраний час;

3) отримання талону на скасування або підтвердження бронювання через електронну пошту;

в) залишення відгуків та оцінок:

1) залишення відгуків та оцінок про коворкінги;

2) відображення середньої оцінки кожного коворкінгу на основі отриманих відгуків;

г) пошук та фільтрація коворкінгів:

1) пошук коворкінгів за містом або назвою;

2) фільтрація коворкінгів за ціною та рейтингом;

д) управління коворкінгами (для адміністраторів):

1) перегляд, додавання, редагування та видалення інформації про власні коворкінги;

2) пошук та фільтрація запитів на бронювання робочих місць за назвою коворкінгу та статусом;

3) обробка запитів на бронювання робочих місць від користувачів;

4) отримання зворотного зв’язку від користувачів через систему.

3.1.8 Припущення та залежності

Основні функції, які будуть включені в початковий випуск продукту, включають наступне:

– програма буде працювати на версіях Android 5.0+ та iOS 10.0+;

– додаток може бути використаний громадянами, що перебувають на території України;

– для використання системи необхідний доступ до комп’ютера або мобільного пристрою з підключенням до Інтернету;

– система повинна бути забезпечена відповідними заходами безпеки, щоб забезпечити захист конфіденційної інформації;

– використання системи має дозволяти шукати та переглядати дані про коворкінги, залишати відгуки та бронювати робочі місця на певну дату та час, а також отримувати талони на скасування чи підтвердження бронювання на вказану електронну пошту.

3.2 Властивості програмного продукту

а) керованість і зручність використання:

1) програмний продукт має інтуїтивний і легкий у використанні інтерфейс користувача, що дозволяє користувачам легко знаходити потрібну інформацію та взаємодіяти з системою без зайвих труднощів;

2) наявність інструментів пошуку та фільтрації дозволяє користувачам швидко знаходити необхідні коворкінги згідно їхніх вимог;

б) підтримка різних платформ:

1) програмний продукт доступний як для веб-браузерів, так і для мобільних платформ, забезпечуючи користувачам доступ до системи незалежно від їхнього пристрою;

в) надійність та безпека:

1) система забезпечує надійне зберігання та обробку персональної інформації користувачів, забезпечуючи конфіденційність і безпеку їх даних;

г) функціональність бронювання:

1) програмний продукт забезпечує користувачам можливість бронювати місця в коворкінгах на обрану дату та час;

2) система забезпечує можливість отримання талону підтвердження або скасування бронювання коворкінгу на вказану електронну пошту користувача. Талон містить інформацію про коворкінг, особу, що здійснила бронювання, а також інформацію про бронювання, включаючи ціну та унікальний номер анкети користувача;

д) адміністрування даних:

1) система надає можливість адміністраторам керувати власними коворкінгами з веб-інтерфейсу (додавання, редагування та видалення об’єктів);

2) Система забезпечує можливість адміністрування анкетами користувачів, включаючи їх підтвердження або скасування.

3.3 Атрибути програмного продукту

3.3.1 Надійність

Система повинна мати високу надійність для забезпечення безперебійної роботи та задоволення потреб користувачів. Для досягнення цієї мети враховуються наступні аспекти: доступність сервісу, стійкість до помилок, відновлення після збоїв, моніторинг та логування.

3.3.2 Доступність

Система повинна бути доступною для використання з будь-якого місця. Якщо система отримує помилку під час надсилання інформації на сервер, інформацію можна надіслати повторно для підтвердження, якщо помилку неможливо усунути, вона буде оброблена та відображена користувачеві.

3.3.3 Безпека

Для безпеки облікових записів кожного користувача буде використовуватися пароль, який буде відомий тільки йому. Для захисту паролів вони будуть хешуватися.

3.3.4 Супроводжуваність

Система повинна мати забезпечений ефективний механізм супроводження для забезпечення її безперебійної роботи та підтримки користувачів після впровадження. Для досягнення цієї мети враховуються наступні аспекти: технічна підтримка, моніторинг та аналіз продуктивності, оновлення та покращення.

3.3.5 Переносимість

Система працюватиме на всіх платформах, на яких вона буде використовуватися (веб і мобільна).

3.3.6 Продуктивність

Система повинна забезпечувати швидку та ефективну обробку запитів користувачів навіть при великому навантаженні. Оптимізація запитів до бази даних та моніторинг продуктивності є важливими аспектами для забезпечення оптимальної роботи системи та задоволення потреб користувачів.

3.4 Вимоги бази даних

Цей проєкт використовуватиме NoSQL MongoDB. Усі дані матимуть типи «string», «date», «number», «array», «null», «object» (залежно від типу вхідних даних і потреб).

3.5. Інші вимоги

Додаткових вимог немає.

ДОДАТОК Ж

Тест-план програмної системи для бронювання коворкінгів

ЗМІСТ

1 ВСТУП

1.1 Мета

Цей документ тест-плану для «Програмна система для бронювання коворкінгів» підтримує наступні цілі:

– визначення наявної інформації про проект та програмні компоненти, які слід протестувати

– перерахування рекомендованих вимог до тестування;

– рекомендації та опис стратегій тестування, які будуть використані;

– визначення необхідних ресурсів та надання оцінки зусиль, необхідних для тестування;

– перелік елементів, які повинні бути отримані в результаті тестового проєкту.

1.2 Передумови

Програмна система для бронювання коворкінгів забезпечує зручний та ефективний інструмент для пошуку, перегляду та бронювання робочих місць у різних коворкінгах. Основна мета системи полягає у спрощенні процесу вибору коворкінгу для користувачів, надаючи їм швидкий доступ до важливої інформації та можливість бронювання робочих місць. Для адміністраторів програмна система надає можливість керувати своїми коворкінгами та обробляти заявки на бронювання.

Архітектура системи передбачає взаємодію клієнтської та серверної складових через мобільний додаток та веб-інтерфейс. Клієнтська частина буде реалізована як веб-застосунок з використанням React.js, а також мобільний додаток

для операційних систем Android та iOS на базі React Native. Серверна частина системи буде побудована на Node.js та TypeScript, з використанням фреймворка Express.js . База даних буде реалізована за допомогою MongoDB.

Ця система є результатом попереднього аналізу та досліджень у сфері коворкінгів та їхнього бронювання. Вона відповідає сучасним тенденціям у сфері організації робочого простору та використанню інформаційних технологій для покращення бізнес-процесів. Проект розробляється з метою забезпечення ефективного управління коворкінгами та їхніми бронюваннями, а також сприяння комфортному користуванню системою як адміністраторами, так і відвідувачами коворкінгів.

Основними функціями системи є перегляд інформації про коворкінги, бронювання робочих місць, залишення відгуків та оцінок, пошук та фільтрація коворкінгів, а також управління коворкінгами зі сторони адміністраторів. Всі ці функції мають бути доступні як через мобільний додаток, так і через веб-інтерфейс, забезпечуючи зручність та доступність для різних категорій користувачів.

Цей опис передумов системи надає загальне уявлення про її призначення, архітектуру та основні функції, що допоможе зорієнтуватися в плануванні тестування та визначенні стратегій.

1.3 Межі

У цьому плані тестування передбачається розгляд тестування на рівнях Unit, Integration та System, а також типів тестування, таких як Function та Performance.

Етапи тестування включають:

– модульне тестування: перевірка окремих компонентів програмної системи на коректність їх роботи;

– інтеграційне тестування: перевірка взаємодії між компонентами системи та їх інтеграцію в єдину систему;

– системне тестування: тестування системи в цілому, щоб переконатися, що вона відповідає вимогам та очікуванням користувачів.

Типи тестування, які будуть розглядатися:

– функціональне тестування: перевірка функціональності системи відповідно до вимог;

– тестування інтерфейсу користувача: перевірка коректності та зручності використання інтерфейсу системи;

– тестування конфігурації: перевірка працездатності системи при різних конфігураціях середовища;

– тестування встановлення: перевірка коректності встановлення та розгортання системи.

Можливості та функції об'єкта тестування, які будуть тестуватися, включають усі функціональності, зазначені в специфікації проєкту, такі як перегляд інформації про коворкінги, бронювання робочих місць, залишення відгуків та оцінок, пошук та фільтрація коворкінгів, управління коворкінгами та доступність системи для користувачів через веб-браузери та мобільні додатки.

Припущення, зроблені під час розроблення цього плану тестування, включають усі аспекти, що відповідають опису функціоналу та технологічному стеку системи, а також правильне функціонування веб-інтерфейсу та мобільних додатків на платформах Android та iOS.

Ризики тестування можуть включати можливі проблеми зі сумісністю з різними веб-браузерами, та мобільними пристроями. Окрім цього потрібно враховувати різну роздільну здатність екранів.

Обмеження, які можуть вплинути на проєктування, розробку або реалізацію тестування, включають обмеження часу та ресурсів, доступність тестових середовищ та обмеження зв'язані з розгортанням на реальних пристроях для мобільних додатків.

1.4 Ідентифікація проєкту

Ідентифікація проєкту наведена в таблиці 1.1. Вона надає контекст для проведення функціонального тестування.

Таблиця 1.1 – Ідентифікація проекту (таблиця виконана самостійно)

Документ	Створено або доступно	Отримано або розглянуто	Автор або ресурс
SRS документ	Так	Так	Мельник К. В.
Прототип веб-застосунку	Так	Так	Мельник К. В.
Прототип мобільного застосунку	Так	Так	Мельник К. В.
Прототип серверної частини	Так	Так	Мельник К. В.

Інформація про те, які документи і прототипи були створені та переглянуті, допоможе тестувальникам краще зрозуміти функціональні вимоги та очікувані результати.

2 ВИМОГИ ДО ТЕСТУВАННЯ

У наведеному нижче переліку визначено ті елементи – функціональні вимоги та нефункціональні вимоги – які були визначені як цілі для тестування.

Функціональні вимоги:

– перевірка можливості перегляду детальної інформації про кожен коворкінг, включаючи назву, фотографії, опис, послуги, час роботи, номер телефону, адреса, кількість робочих місць, тариф та відгуки користувачів;

– перевірка можливості бронювання робочих місць у коворкінгах, заповнення інформації для бронювання, визначення доступності місць та отримання підтвердження або скасування бронювання;

– перевірка можливості залишення відгуків та оцінок про коворкінги та відображення середньої оцінки кожного коворкінгу на основі отриманих відгуків;

– перевірка функціоналу пошуку та фільтрації коворкінгів за містом або назвою, а також за ціною та рейтингом;

– перевірка можливості управління коворкінгами для адміністраторів, включаючи перегляд, додавання, редагування та видалення інформації про коворкінги та обробку запитів на бронювання робочих місць.

Нефункціональні вимоги:

а) перевірка безпеки:

– захист даних користувачів за допомогою механізму автентифікації;

– хешування паролю користувачів;

б) перевірка сумісності:

– сумісність з різними веб-браузерами та мобільними пристроями;

– перевірено, що програмне забезпечення пристосовується до різних розмірів екранів (для веб-застосунків) та різних розширень екранів (для мобільних додатків);

в) перевірка зручності використання:

– інтуїтивно зрозумілий та легкий у використанні інтерфейс користувача;

– у разі помилки при заповненні полів для вводу користувач отримує повідомлення про це.

Цей список представляє те, що буде протестовано.

3 СТРАТЕГІЯ ТЕСТУВАННЯ

3.1 Типи тестування

3.1.1 Функціональне тестування

Функціональне тестування об'єкта тестування повинно зосереджуватися на будь-яких вимогах до тесту, які можна простежити безпосередньо до варіантів використання або бізнес-функцій та бізнес-правил. Метою цих тестів є перевірка належного приймання, обробки та пошуку даних, а також належного виконання бізнес-правил. Цей тип тестування базується на методах «чорної скриньки», тобто перевірці програми та її внутрішніх процесів шляхом взаємодії з програмою через графічний інтерфейс користувача (GUI) та аналізу вихідних даних або результатів.

Техніка, мета випробування, критерії завершення та особливі міркування щодо функціонального тестування наведені в таблиці 3.1.

Таблиця 3.1 – Функціональне тестування (таблиця виконана самостійно)

Мета випробування:	Забезпечити належну функціональність об'єкта тестування, включаючи навігацію, введення, обробку та пошук даних.
Критерії завершення:	- всі заплановані тести були виконані. - всі виявлені дефекти були усунені.

Продовження таблиці 3.1

Особливі міркування:	– перевірити відповідність функціоналу програмної системи для бронювання коворкінгів зазначеним функціональним вимогам; – переконатися, що всі бізнес-правила, включаючи процес бронювання та керування коворкінгами, виконуються належним чином; – забезпечити, що користувачі отримують відповідні повідомлення про помилки та інформацію при взаємодії з системою; – виконати тестування кожного варіанту використання, використовуючи як достовірні, так і недостовірні дані, щоб переконатися, що очікувані результати з'являються при використанні достовірних даних, а також що відповідні повідомлення про помилки відображаються при використанні недостовірних даних.

Кінець таблиці 3.1

Техніка:	Виконайте кожен варіант використання, потік варіантів використання або функцію, використовуючи достовірні та недостовірні дані, щоб перевірити наступне: – очікувані результати з'являються, коли використовуються достовірні дані; – відповідні повідомлення про помилки або попередження відображаються, коли використовуються невірні дані; – кожне бізнес-правило застосовується належним чином.

Ця таблиця допомагає структурувати процес тестування та забезпечує систематичний підхід до перевірки функціональних можливостей системи. Завдяки цій систематизації тестерам легше оцінювати відповідність продукту вимогам і здійснювати необхідні корективи для забезпечення його якості та надійності перед випуском на ринок.

3.1.2 Тестування інтерфейсу користувача

Тестування інтерфейсу користувача (UI) важливий етап в розробці програмної системи, який спрямований на перевірку коректності та зручності взаємодії користувача з програмним продуктом.

Техніка, мета випробування, критерії завершення та особливі міркування щодо тестування інтерфейсу користувача наведені в таблиці 3.2.

Таблиця 3.2 – Тестування інтерфейсу користувача (таблиця виконана самостійно)

Мета випробування:	Переконатися, що користувальницький інтерфейс системи відповідає вимогам, забезпечує зручну навігацію та ефективність взаємодії користувача з функціоналом системи. Крім того, це тестування покликане переконатися, що всі елементи і компоненти інтерфейсу працюють стабільно і відповідають встановленим стандартам якості, включаючи корпоративні та галузеві стандарти.
Техніка:	Техніка тестування UI зосереджена на перевірці коректності та зручності взаємодії користувача з інтерфейсом програмного забезпечення. Однією з ключових аспектів цієї техніки є перевірка навігації по об'єкту тестування. Вона включає в себе переходи між різними екранами, сторінками, а також взаємодію з окремими елементами інтерфейсу.

Кінець таблиці 3.2

Критерії завершення:	– правильність переходів: всі переходи між вікнами, сторінками та екранами мають бути здійснені вірно та без затримок; – зручність взаємодії: користувач повинен легко розуміти, як взаємодіяти з окремими елементами інтерфейсу та використовувати методи доступу (клавіші табуляції, рухи миші, клавіші акселератора); – відповідність вимогам: всі функції навігації повинні відповідати зазначеним в специфікації вимогам та бізнес-функціям.
Особливі міркування:	– безпека: деякі властивості можуть бути обмежені з метою захисту від несанкціонованого доступу або зловмисних атак; – конфіденційність: доступ до певних даних може бути обмежений з метою збереження конфіденційності інформації користувачів; – стабільність: обмеження можуть бути встановлені для забезпечення стабільності та надійності програмного продукту.

Ця таблиця сприяє організації процесу тестування, забезпечуючи систематичний підхід до перевірки функціональних можливостей системи. Ця систематизація допомагає тестерам легше оцінювати відповідність продукту вимогам і вносити необхідні зміни для забезпечення його якості та надійності перед випуском на ринок.

3.1.3 Тестування конфігурації

Тестування конфігурації перевіряє роботу мобільного застосунку, веб-додатку та серверної частини на різних конфігураціях апаратного та програмного забезпечення. Це включає в себе перевірку сумісності з різними версіями операційних систем, веб-браузерів, а також різними версіями та конфігураціями серверного середовища.

Техніка, мета випробування, критерії завершення та особливі міркування щодо тестування конфігурації наведені в таблиці 3.3.

Таблиця 3.3 – Тестування конфігурації (таблиця виконана самостійно)

Мета випробування:	Переконатися, що мобільний застосунок, веб-додаток та серверна частина працюють належним чином на різних апаратних і програмних конфігураціях.
Критерії завершення:	Для кожної комбінації тестового та не-тестового програмного забезпечення всі функції успішно виконуються без збоїв або некоректної поведінки.

Кінець таблиці 3.3

Техніка:	– запуск мобільного застосунку на різних версіях операційних систем (Android, iOS) з різними версіями пристроїв; – тестування веб-додатку в різних веб-браузерах (Chrome, Firefox, Safari, Edge) на різних операційних системах (Windows, macOS, Linux); – запуск серверної частини на різних конфігураціях серверів з різними версіями Node.js та операційних систем.
Особливі міркування:	– необхідно враховувати різноманітність апаратних і програмних конфігурацій, зокрема версії операційних систем, типи пристроїв, їх розширення та обладнання; – потрібно визначити набір ключових конфігурацій для тестування, які охоплюють основні характеристики цільових аудиторій.

Ця таблиця спрощує організацію процесу тестування, створюючи рамки для систематичного аналізу функціональних можливостей системи. Цей структурований підхід дозволяє тестерам ефективніше оцінювати відповідність продукту вимогам та вносити необхідні зміни, щоб гарантувати його якість і надійність перед випуском на ринок.

3.1.4 Тестування встановлення

Тестування встановлення має на меті переконатися, що мобільний додаток, розроблений на React Native, може бути успішно встановлений на мобільний пристрій під різними умовами, включаючи нормальні та ненормальні ситуації. Крім того, воно спрямоване на перевірку працездатності додатку після встановлення.

Техніка, мета випробування, критерії завершення та особливі міркування щодо тестування встановлення наведені в таблиці 3.4.

Таблиця 3.4 – Тестування встановлення (таблиця виконана самостійно)

Мета випробування:	Переконатися, що мобільний додаток правильно встановлюється на мобільний пристрій за наступних умов: – нова інсталяція: додаток встановлюється на мобільний пристрій, на якому раніше не було встановлено додаток «Програмна система для бронювання коворкінгів».
Критерії завершення:	Мобільний додаток «Програмна система для бронювання коворкінгів» успішно встановлюється та працює без помилок чи збоїв на мобільному пристрої під управлінням операційних систем iOS та Android.

Кінець таблиці 3.4

Техніка:	– використання ручних методів для перевірки стану мобільного пристрою (нового, на якому раніше не встановлювався додаток «Програмна система для бронювання коворкінгів» >; або пристрою, на якому вже встановлена попередня версія додатку «Програмна система для бронювання коворкінгів»); – запуск процесу встановлення мобільного додатку на пристрої під управлінням операційної системи iOS та Android; – виконання тестових сценаріїв для перевірки функціональності додатку після встановлення.
Особливі міркування:	Необхідно обрати транзакції додатку «Програмна система для бронювання коворкінгів», які найкраще підходять для перевірки впевненості у тому, що мобільний додаток React Native був успішно встановлений і відсутні будь-які основні проблеми з його компонентами під обидвома платформами.

Ця таблиця спрощує організацію процесу тестування, створюючи основу для систематичного аналізу функціональних можливостей системи. Такий структурований підхід дозволяє тестерам ефективніше оцінювати, наскільки продукт відповідає вимогам, та вносити необхідні зміни для гарантування його якості та надійності перед випуском на ринок.

3.2 Інструменти

Для забезпечення якісного тестування програмного забезпечення цього проекту будуть використані інструменти з таблиці 3.5.

Таблиця 3.5 – Інструменти (таблиця виконана самостійно)

	Інструмент	Постачальник / власна розробка	Версія
Управління тестуванням	Jira	Atlassian	9.13
ASQ Tool для функціонального тестування	Jest	OpenJS Foundation	29.6
Інструмент ASQ для тестування продуктивності	JMeter	Apache	5.4.3
Управління проектами	Jira	Atlassian	9.13
Інструменти для роботи з СУБД	Postman	Postman	10.21

Кожен з цих інструментів дозволяє протестувати кожний компонент програмного забезпечення.

4 РЕСУРСИ

4.1 Ролі

Кадрові припущення для проекту наведені в таблиці 4.1.

Таблиця 4.1 – Кадрові припущення для проекту (таблиця виконана самостійно)

Працівники		
Посада	Рекомендовані мінімальні ресурси (кількість штатних ролей, призначених)	Конкретні обов'язки або коментарі
Тестувальник	1	Виконати весь цикл тестування

Ця таблиця описує посаду, кількість та обов’язки працівників.

4.2 Система

У таблиці 4.2 наведено системні ресурси для проєкту тестування:

Таблиця 4.1 – Системні ресурси (таблиця виконана самостійно)

Системні ресурси	
Ресурс	Назва / Тип
База даних	MongoDB 7.0
Node.js	Node.js 16.13.0

Кінець таблиці 4.1

Системні ресурси	
Ресурс	Назва / Тип
ПК	Intel Core i5, 8 ГБ DDR4 RAM, NVIDIA GeForce, Windows 10, Full HD монітор
Смартфон	– Android: Samsung Galaxy S21 з Android 11 – iOS: iPhone 13 з iOS 15.

Тут вказані наступні ресурси: база даних, Node.js, ПК, смартфон.

5 ЕТАПИ ПРОЄКТУ

Для забезпечення якості програмної система для бронювання коворкінгів, тестування буде проведено на різних етапах розробки. Етапи проєкту можна побачити в таблиці 5.1.

Таблиця 5.1 – Етапи проєкту

Milestone Task	Effort	Start Date	End Date
Plan Test	3	2024-04-24	2024-04-27
Design Test	4	2024-04-28	2024-05-01
Implement Test	3	2024-05-02	2024-05-05
Execute Test	5	2024-05-05	2024-05-10
Evaluate Test	3	2024-05-10	2024-05-13

Кожен етап тестування включатиме специфічні тестові заходи, які відповідають вимогам та функціональності проекту, визначеним у попередніх розділах.

6 РЕЗУЛЬТАТИ

6.1 Тестова модель

Тест-план: це основний документ, який визначає стратегію тестування, область застосування, ресурси, ризики, графік та інші ключові аспекти тестового процесу.

Звіт про виконання тестів: цей звіт містить інформацію про кількість виконаних тестів, кількість успішно пройдених, кількість виявлених дефектів та їх статус.

6.2 Журнали випробувань

Для запису та звітування про результати тестування та статус тестування буде використовуватися система журналів випробувань, яка включатиме такі компоненти:

а) інструменти:

1) система журналів випробувань буде реалізована з використанням інтегрованих функцій у розробницькому середовищі Visual Studio Code;

б) методи:

1) журнали випробувань будуть вести кожен член команди, який бере участь у тестуванні.

6.3 Звіти про дефекти

Для відстеження дефектів та їх статусу буде використана система керування задачами, така як Jira. Кожен виявлений дефект буде документований у системі керування задачами, включаючи детальний опис проблеми, кроки для відтворення, пріоритет та відповідального за виправлення. Дефекти будуть регулярно оглядатися командою розробників та тестувальників для визначення статусу та пріоритету виправлення.

ДОДАТОК З

Завдання проєкту

Нижче наведені завдання, пов'язані з тестом:

а) спланувати тест:

1) визначити вимоги до тесту;

2) оцінити ризики;

3) розробити стратегію тестування;

4) визначити тестові ресурси;

5) створити розклад;

6) сформувати тест-план;

б) спроєктувати тест:

1) підготувати аналіз робочого навантаження;

2) визначити та описати тестов-кейси;

3) визначити та структурувати тестові процедури;

4) проаналізувати та оцінити тестове покриття;

в) реалізувати тест:

1) записати або програмувати тестові скрипти;

2) визначити специфічну для тестів функціональність в моделі проектування та реалізації;

3) створити зовнішні набори даних;

г) виконати тест:

1) виконати тестові процедури;

2) оцінити виконання тесту;

3) відновити тест після його зупинки;

4) перевірити результати;

5) дослідити неочікувані результати;

6) реєструвати дефекти;

д) оцінити тест:

1) оцінити покриття тест-кейсів;

2) оцінити покриття коду;

3) проаналізувати дефекти;

4) визначити, чи були досягнуті критерії завершення тесту та критерії успіху.

ДОДАТОК И

Тест-кейси програмної системи для бронювання коворкінгів

Таблиця К.1 – Тест-кейс №1 (таблиця виконана самостійно)

Інформація про тест-кейс			
Ідентифікатор тесту:	Тест-кейс №1		
Опис функції:	Обробка заявок на бронювання робочих місць в коворкінгу		
Власник тесту:	Мельник Катерина Вадимівна		
Дата створення:	22.04.2024		
Мета тесту:	Перевірити коректність обробки заявок на бронювання робочих місць в коворкінгу		
Передумова			
№	Опис випадку	Очікуваний результат	Висновок
1	Відкрити адміністративну частину веб-застосунку	Користувач має доступ до сайту, який відкритий	Пройдено
2	Має обліковий запис та хоче авторизуватися	Перенаправлено на сторінку профілю	Пройдено
Підтвердження заявки на бронювання робочих місць в коворкінгу			
№	Опис випадку	Очікуваний результат	Висновок
1	Відкрити сторінку поданих заявок	З’являється список поданих заявок	Пройдено
2	Натиснути на елемент списку	Відкривається детальна інформація заявки	Пройдено

Продовження таблиці К.1

Підтвердження заявки на бронювання робочих місць в коворкінгу			
№	Опис випадку	Очікуваний результат	Висновок
3	Натиснути кнопку типу «radio» з підписом «статус», щоб змінити положення на «включено»	Значення статусу змінено на «включено»	Пройдено
4	Натиснути кнопку «Зберегти»	Дані збережено. На вказану в заявці електронну пошту надіслано лист із підтвердженням	Пройдено
Відхилення заявки на бронювання робочих місць в коворкінгу			
№	Опис випадку	Очікуваний результат	Висновок
1	Відкрити сторінку поданих заявок	З’являється список поданих заявок	Пройдено
2	Натиснути на елемент списку	Відкривається детальна інформація заявки	Пройдено
3	Натиснути кнопку типу «radio» з підписом «статус», щоб змінити положення на «відключено»	Значення статусу змінено на «відключено»	Пройдено
4	Натиснути кнопку «Зберегти»	Дані збережено. На вказану в заявці електронну пошту надіслано лист із відхиленням	Пройдено

Кінець таблиці К.1

Результати тестування		
Тестувальник: Мельник К. В.	Дата прогону тесту: 22.04.2024	Результат тесту (P/F/B): ПРОЙДЕНО (P)

Таблиця К.2 – Тест-кейс №2 (таблиця виконана самостійно)

Інформація про тест-кейс			
Ідентифікатор тесту:	Тест-кейс №2		
Опис функції:	Подання заявки на бронювання робочих місць в коворкінгу		
Власник тесту:	Мельник Катерина Вадимівна		
Дата створення:	22.04.2024		
Мета тесту:	Перевірити коректність подання заявки на бронювання робочих місць в коворкінгу		
Передумова			
№	Опис випадку	Очікуваний результат	Висновок
1	Відкрити загальнодоступну частину веб-застосунку	Користувач має доступ до сайту, який відкритий	Пройдено
2	Відкрити головну сторінку	Перенаправлено на головну сторінку	Пройдено
3	Натиснути на один з відображених коворкінгів	Перенаправлено на сторінку коворкінгу	Пройдено
Подання заявки на бронювання робочих місць в коворкінгу			
№	Опис випадку	Очікуваний результат	Висновок
1	Натиснути на кнопку «Забронювати»	Відкривається вікно з параметрами заявки	Пройдено

Кінець таблиці К.2

Подання заявки на бронювання робочих місць в коворкінгу			
№	Опис випадку	Очікуваний результат	Висновок
2	Заповнити обов’язкове поле «ІМ’Я»	Було введено ім’я	Пройдено
3	Заповнити обов’язкове поле «ЕЛЕКТРОННА ПОШТА»	Була введена електронна пошта	Пройдено
4	Заповнити обов’язкове поле «ТЕЛЕФОН»	Був введений телефон	Пройдено
4	Заповнити обов’язкове поле «КІЛЬКІСТЬ МІСЦЬ»	Було введено кількість місць	Пройдено
5	Заповнити обов’язкове поле з датою	Була введена дата	Пройдено
6	Обрати час	Було обрано час	Пройдено
7	Натиснути кнопку «Забронювати»	Заявка була закрита та відправлена адміністратору. Отримано повідомлення про це.	Пройдено
Результати тестування			
Тестувальник: Мельник К. В.		Дата прогону тесту: 22.04.2024	Результат тесту (P/F/B): ПРОЙДЕНО (P)

Таблиця К.3 – Тест-кейс №3 (таблиця виконана самостійно)

Інформація про тест-кейс			
Ідентифікатор тесту:	Тест-кейс №3		
Опис функції:	Написання відгуку до коворкінгу з мобільного застосунку		
Власник тесту:	Мельник Катерина Вадимівна		
Дата створення:	22.04.2024		
Мета тесту:	Перевірити коректність написання відгуку до коворкінгу з мобільного застосунку		
Передумова			
№	Опис випадку	Очікуваний результат	Висновок
1	Відкрити загальнодоступну частину мобільного застосунку	Користувач має доступ до мобільного застосунку, який відкритий	Пройдено
2	Відкрити головний екран	Перенаправлено на головний екран	Пройдено
3	Натиснути на один з відображених коворкінгів	Перенаправлено на екран коворкінгу	Пройдено
Написання відгуку до коворкінгу з мобільного застосунку			
№	Опис випадку	Очікуваний результат	Висновок
1	Обрати оцінку, натиснувши на зірочку	Оцінка обрана	Пройдено
2	Заповнити обов’язкове поле «ВІДГУК»	Було введено відгук	Пройдено

Кінець таблиці К.3

Написання відгуку до коворкінгу з мобільного застосунку			
№	Опис випадку	Очікуваний результат	Висновок
3	Натиснути кнопку «Відправити»	Відгук було залишено. Отримано повідомлення про це Відгук з’явився в списку всіх відгуків.	Пройдено
Результати тестування			
Тестувальник: Мельник К. В.		Дата прогону тесту: 22.04.2024	Результат тесту (P/F/B): ПРОЙДЕНО (P)