Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра програмного забезпечення

Бакалаврська кваліфікаційна робота

на тему: Розробка вебзастосунку для пошуку та рекомендації туристичних місць міста

Виконала: студентка IV курсу, групи 3ПІ-21б
спеціальності
121 – Інженерія програмного забезпечення
(шифр і назва напряму підготовки, спеціальності)
Орчакова Юлія Володимирівна
(прізвище та ініціали)
Керівник: д-р. філос., асист. каф. ПЗ Карась О.В.
(прізвище та ініціали)
Рецензент: к.т.н., доц. каф. ОТ Черняк О. І.
(прізвище та ініціали)

Допущено до захисту

Зав. кафедри ПЗ О.Н. Романюк

«09» 06 2025 р.

ВНТУ – 2025

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення
Рівень вищої освіти перший (бакалаврський)
Галузь знань 12 «Інформаційні технології»
Спеціальність 121 «Інженерія програмного забезпечення»
Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
О. Н. Романюк
«21» березня 2025 р.

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Орчаковій Юлії Володимирівні

1. Тема роботи – розробка вебзастосунку для пошуку та рекомендації туристичних місць міста.

Керівник роботи: Карась Олександр Володимирович, д-р. філос., асист. каф. ПЗ, затверджені наказом вищого навчального закладу від «20» березня 2025 р. № 97.

2. Строк подання студентом роботи 2 червня 2025

3. Вихідні дані до роботи: середовище розробки Visual Studio Code, мова програмування JavaScript, фреймворк React, Google Maps API, бібліотеки react-toastify та react-modal, візуальне оформлення CSS, метод генерації рекомендацій – контентний.

4. Зміст текстової частини: вступ; аналіз стану технологій пошуку та рекомендації туристичних місць; порівняльний аналіз аналогів; аналіз методів розв'язання задачі; постановка задач дослідження; аналіз вхідних даних; розробка моделі вебзастосунку; розробка алгоритмів вебзастосунку; варіантний аналіз і обґрунтування вибору засобів для реалізації; розробка модуля реєстрації та авторизації користувача; розробка модуля взаємодії з картою та побудови маршрутів; розробка інтерфейсу вебзастосунку; тестування вебзастосунку; розробка інструкції користувача.

5. Перелік ілюстративної частини: титульний аркуш, актуальність розробки, мета, об'єкт та предмет дослідження, задачі дослідження, наукова новизна та практична цінність отриманих результатів, порівняльний аналіз аналогів, загальний алгоритм роботи вебзастосунку, алгоритм генерації персоналізованих рекомендацій, робота модуля реєстрації, робота модуля побудови маршрутів, демонстрація інтерфейсу вебзастосунку «головна сторінка», демонстрація

інтерфейсу вебзастосунку «блок персоналізованих рекомендацій», демонстрація інтерфейсу вебзастосунку «робота пошуку», демонстрація інтерфейсу вебзастосунку «побудова маршруту», демонстрація інтерфейсу вебзастосунку «збереження місць», демонстрація інтерфейсу вебзастосунку «сторінка місця та адаптивний дизайн», апробація та публікація матеріалів бакалаврської кваліфікаційної роботи.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
1-4	Карась О.В., д-р. філос., асист. каф. ПЗ	24.03.25	01.06.25

7. Дата видачі завдання 24 березня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз стану технологій пошуку та рекомендації туристичних місць та постановка задач дослідження	25.03.25 – 07.04.25	Викон.
2	Розробка моделі та алгоритмів вебзастосунку	07.04.25 – 20.04.25	Викон.
3	Розробка модулів вебзастосунку	20.04.25 – 03.05.25	Викон.
4	Тестування програмного продукту	03.05.25 – 16.05.25	Викон.
5	Оформлення матеріалів до захисту БКР	16.05.25– 30.05.25	Викон.

Студентка _____ **Ю. В. Орчакова**
(підпис) (ініціали та прізвище)

Керівник бакалаврської кваліфікаційної роботи _____ **О. В. Карась**
(підпис) (ініціали та прізвище)

АНОТАЦІЯ

УДК 004.4

Орчакова Ю.В. Розробка вебзастосунку для пошуку та рекомендації туристичних місць міста : бакалаврська кваліфікаційна робота зі спеціальності 121 – інженерія програмного забезпечення, освітня програма – інженерія програмного забезпечення. Вінниця: ВНТУ, 2025. 50 с.

Бакалаврська кваліфікаційна робота містить 50 сторінок формату А4, на яких наведено 27 рисунків та 1 таблиця, а список використаних джерел налічує 21 найменуваня.

У бакалаврській кваліфікаційній роботі було розроблено вебзастосунок для пошуку та рекомендації туристичних місць міста. Проведено аналіз стану технологій пошуку та рекомендації туристичних місць. Обгрунтовано актуальність власної розробки на основі проведеного порівняльного аналізу аналогів.

Розроблено модулі реєстрації та авторизації, взаємодії з картами для побудови маршрутів.

Розроблено алгоритми пошуку та збереження місць, додавання відгуків, а також алгоритм персоналізованих рекомендацій.

Проведене тестування підтвердило достовірність теоритичних досліджень методів пошуку та рекомендації туристичних місць. Розроблений програмний продукт можна використати в туристичних інформаційних системах, путівниках по містах та мобільних застосунках для туристів.

Ключові слова: персоналізовані рекомендації, маршрут, туристичні місця, карта, пошук місць, фільтрація.

ABSTRACT

UDC 004.4

Orchakova, Y.V. Development of a web application for searching and recommending tourist attractions in the city: bachelor's thesis in the field of 121 – software engineering, educational program – software engineering. Vinnytsia: VNTU, 2025. 50 p.

The bachelor's thesis contains 50 pages in A4 format, with 27 figures and 1 table, and the list of references includes 21 items.

In the bachelor's thesis, a web application for searching and recommending tourist places in the city was developed. The author analyzed the state of the art of search and recommendation of tourist places. The relevance of our own development is substantiated on the basis of a comparative analysis of analogues.

The modules of registration and authorization, interaction with maps for route building are developed.

Algorithms for searching and saving places, adding reviews, and an algorithm for personalized recommendations have been developed.

The testing confirmed the validity of theoretical studies of methods for searching and recommending tourist destinations. The developed software product can be used in tourist information systems, city guides, and mobile applications for tourists.

Keywords: personalized recommendations, route, tourist places, map, place search, filtering.

ЗМІСТ

ВСТУП

Обґрунтування вибору теми дослідження.

Сучасні інформаційні технології відіграють важливу роль у розвитку туристичної сфери, пропонуючи користувачам зручні системи для пошуку та вибору цікавих місць і подій. Такі системи допомагають туристам швидко знаходити місця, які відповідають їхнім інтересам, бюджету та географічному розташуванню.

Більшість традиційних платформ для пошуку туристичних локацій обмежуються недостатньо точними рекомендаціями. У зв’язку з цим сучасні технології, які можуть інтегруватися з картографічними сервісами та використовувати машинне навчання [1] для надання інформації на основі вподобань користувачів, відкривають нові шляхи для покращення користувацького досвіду та підвищення точності таких рекомендацій.

Персоналізовані системи рекомендацій [2] використовують дані про взаємодію користувача для визначення його вподобань. Таким чином, користувачі отримують кращі варіанти для відвідування. А якщо система інтегрована з мапами, вона не лише вказує місцезнаходження об’єктів, але й полегшує шлях до них.

Існуючі туристичні онлайн-системи мають ряд недоліків, таких як відсутність гнучкої фільтрації, незначна інтеграція з картографічними сервісами або її відсутність взагалі, а також нижча точність рекомендацій.

Тому актуальною є розробка вебзастосунку, що поєднуватиме сучасні технології з алгоритмами персоналізації.

Зв’язок роботи з науковими програмами, планами, темами. Робота виконувалася згідно плану виконання наукових досліджень на кафедрі програмного забезпечення.

Мета та завдання дослідження. Метою дослідження є покращення якості процесу пошуку актуальних туристичних подій та місць, підвищення комфорту

планування дозвілля за рахунок розробки програмних модулів, які забезпечують пошук, персоналізовані рекомендації, побудову маршрутів, зручну взаємодію з користувачем.

Відповідно до поставленої мети в бакалаврській кваліфікаційній роботі потрібно вирішити такі **задачі**:

– розробити алгоритм персоналізованих рекомендацій, що забезпечить точне визначення інтересів користувачів та формування відповідних пропозицій;

– розробити модуль пошуку та фільтрації, який дозволить користувачам швидко знаходити цікаві місця за категоріями;

– реалізувати інтеграцію з картографічними сервісами, що забезпечить відображення місць на карті та побудову маршрутів;

– розробити адаптивний інтерфейс користувача, який буде зручним для користування і на комп’ютерах, і на мобільних пристроях.

Об’єктом дослідження є процес розробки вебзастосунку пошуку і рекомендацій туристичних місць.

Предметом дослідження є методи та засоби розробки вебзастосунку пошуку і рекомендацій туристичних місць.

Методи дослідження. У процесі дослідження застосовувалися: теорія алгоритмів для розробки функцій пошуку за назвою, динамічної фільтрації місць за категоріями, а також для створення персоналізованих рекомендацій, принципи роботи з геоінформаційними технологіями та картографічними сервісами, техніки розробки адаптивного інтерфейсу.

Наукова новизна отриманих результатів:

1. Удосконалено метод реалізації персоналізованих рекомендацій для туристичних вебсистем, який, на відміну від існуючих, що часто потребують значних серверних ресурсів та складних алгоритмів, використовує аналіз даних про взаємодію користувача, що зберігаються на стороні клієнта, особливість якого полягає у визначенні пріоритетних для користувача категорій на основі цих даних та формуванні списку релевантних, але ще не відомих йому

пропозицій, що дозволило спростити архітектуру та реалізувати функціонал персоналізації, підвищуючи залученість користувача.

2. Удосконалено підхід до інтеграції функціоналу планування маршрутів безпосередньо у клієнтський вебзастосунок, який відрізняється від простого відображення статичної карти тим, що в ньому використано комбінацію компонентів для розрахунку та візуалізації маршруту з можливістю введення користувачем початкової точки як текстової адреси, так і через геолокацію, що дозволило забезпечити інтерактивне відображення маршруту на тій самій карті, де показано цільовий об’єкт, та спростити процес планування для користувача без необхідності переходу до сторонніх картографічних сервісів.

Практичне значення отриманих результатів полягає у можливості використання вебзастосунку туристами для швидкого знаходження цікавих місць відповідно до їхніх вподобань. Розроблений програмний продукт може бути використаний для вдосконалення існуючих платформ або як основа для створення нових туристичних сервісів.

Особистий внесок здобувача. Усі наукові результати отримано здобувачем самостійно. У працях, опублікованих у співавторстві, студенту належать: [3] – проаналізовано алгоритми для створення рекомендаційних систем; [4] – описано принципи роботи найпоширеніших алгоритмів маршрутизації.

Апробація матеріалів бакалаврської кваліфікаційної роботи. Основні положення БКР доповідалися та обговорювалися на Міжнародних і Всеукраїнських конференціях: LIV Всеукраїнська науково-технічна конференція факультету інформаційних технологій та комп’ютерної інженерії (м. Вінниця, 2025), Міжнародна науково-практична інтернет-конференція «Молодь в науці: дослідження, проблеми, перспективи (МН-2025)» (м. Вінниця, 2025).

Публікації. За темою бакалаврської роботи надруковано 2 праці у матеріалах всеукраїнських і міжнародних конференціях.

Структура та обсяг БКР. Бакалаврська кваліфікаційна робота складається зі вступу, чотирьох розділів, висновків, списку використаних джерел і додатків, у які винесено технічне завдання, протокол перевірки на плагіат, лістинг програмного забезпечення й ілюстративний матеріал до захисту роботи.

1 АНАЛІЗ СТАНУ ТЕХНОЛОГІЙ ПОШУКУ ТА РЕКОМЕНДАЦІЇ ТУРИСТИЧНИХ МІСЦЬ ТА ПОСТАНОВКА ЗАДАЧ ДОСЛІДЖЕННЯ

1.1 Аналіз стану технологій пошуку та рекомендації туристичних місць

Сучасний етап розвитку інформаційних технологій характеризується активним використанням інтелектуальних вебсистем, що пропонують користувачеві персоналізований досвід. Особливо це стосується туристичної галузі, де легкість доступу до інформації про визначні пам’ятки, події та послуги має безпосередній вплив на рішення мандрівників.

Сьогодні більшість туристичних сайтів, таких як Google Maps, TripAdvisor, Booking.com, використовують рекомендаційні алгоритми, щоб забезпечити кращий досвід для користувачів. Провідними підходами до побудови таких рекомендацій є колаборативна фільтрація на основі поведінки інших користувачів та контентна фільтрація на основі інформації про об’єкти, а також комбіновані алгоритми, які поєднують ці два підходи [5].

Завдяки прогресу в машинному навчанні вебзастосунки тепер можуть обробляти величезні обсяги даних, включаючи історію переглядів, рейтинги, уподобання користувачів, моделі поведінки користувачів і контекстну інформацію. Це дозволяє надавати більш точні та релевантні рекомендації, що підвищує залученість та інтерес користувачів до системи.

Разом з розробкою алгоритмів рекомендацій все більшого значення набуває адаптивний дизайн. Все частіше користувачі взаємодіють з системами за допомогою мобільних пристроїв, і вебзастосунки повинні бути оптимізовані для різних розмірів екранів.

Великі туристичні сайти не завжди пропонують персоналізовані рекомендації. Саме тому розробка вебзастосунку, який враховує інтереси користувача, його історію переглядів та фільтрацію за типом місця є актуальною.

Отже, дослідження сучасного стану застосування вебтехнологій у туристичній галузі обґрунтовує актуальність розробки вебзастосунку для пошуку та рекомендації туристичних місць та подій. Це дозволить підвищити якість користувацького досвіду, релевантність контенту, а також надасть більш точні та цікаві варіанти для користувачів. Розробка системи, що поєднує сучасні моделі рекомендацій та адаптивний інтерфейс забезпечить створення ефективного інструменту для туристичних потреб.

1.2 Порівняльний аналіз аналогів

На даний момент існує значна кількість вебплатформ, які пропонують користувачам інформацію про туристичні напрямки, події, готелі та інші послуги. Серед найвідоміших – MoeMisto.ua, Booking.com та TripAdvisor. Порівняння цих вебсистем дозволяє виявити сильні та слабкі сторони, а також можливості для розробки власного вебзастосунку.

MoeMisto.ua – український інтернет-портал, який спеціалізується на розміщенні інформації про події в містах України [6]. Портал пропонує події за категоріями такими, як концерти, виставки, театри та можливість додавання до обраного. Сайт має зручний адаптивний інтерфейс та інтеграцію з картографічними сервісами. Він не надає персоналізованих рекомендацій, рейтингової системи чи аналітики інтересів користувачів. Робота з платформою MoeMisto.ua наведена на рисунку 1.1.

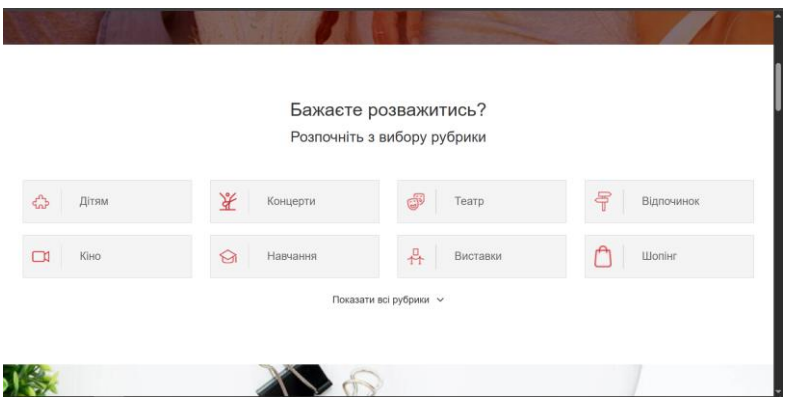

Рисунок 1.1 – Приклад роботи вебсистеми MoeMisto.ua

Booking.com – один з найпоширеніших світових сайтів для бронювання готелів [7]. Хоча основною функцією сайту є бронювання житла, на ньому також є розділ «Чим зайнятися», в якому перераховані визначні пам’ятки та заходи. Особливостями сайту є потужна система фільтрації, рекомендації на основі попередніх бронювань, персоналізація та велика кількість відгуків. Однак, сайт орієнтований насамперед на туристів, а не на місцеві події. Робота з платформою Booking.com наведена на рисунку 1.2.

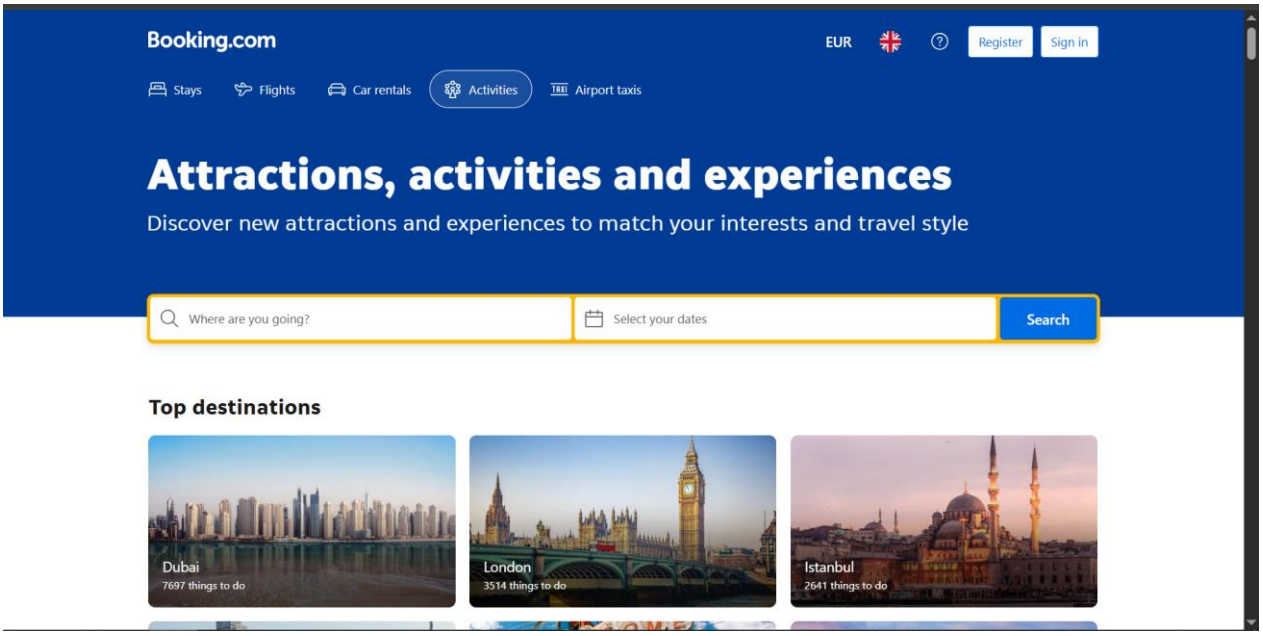

Рисунок 1.2 – Приклад роботи вебсистеми Booking.com

TripAdvisor – це глобальний вебсайт з великою базою даних про визначні пам’ятки, готелі, ресторани та розваги [8]. Він може похвалитися високорозвиненою системою рейтингів, відгуків та рекомендацій. Алгоритми сайту враховують як популярність локації, так і відповідність інтересам користувача. Недоліком є інформаційна перевантаженість і неможливість швидко знайти події в режимі реального часу. Робота з платформою TripAdvisor наведена на рисунку 1.3.

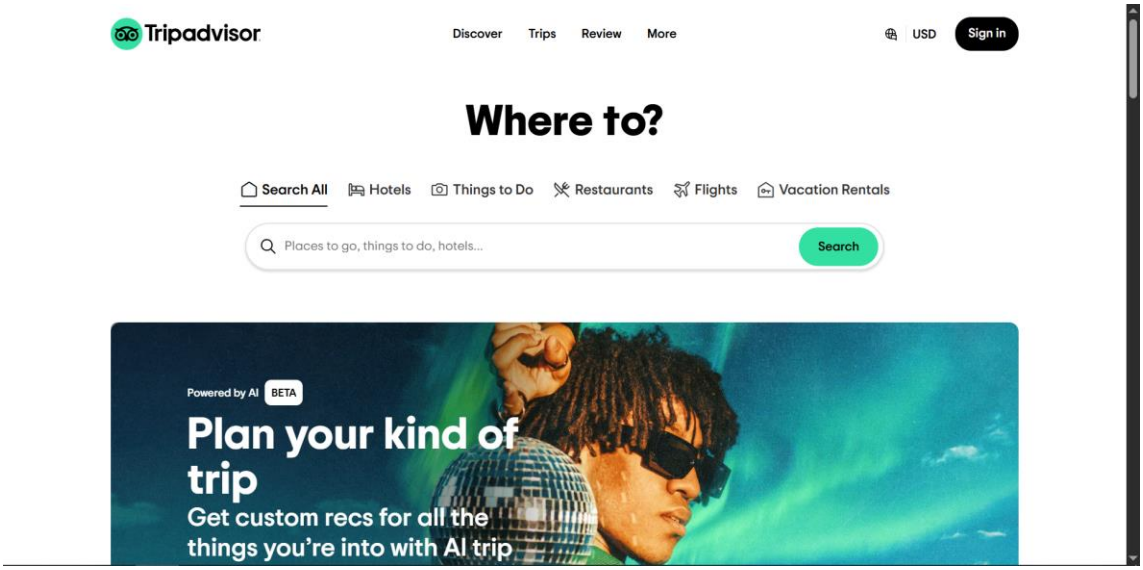

Рисунок 1.3 – Приклад роботи вебсистеми TripAdvisor

Аналіз аналогів дозволив виявити основні функціональні переваги, які необхідно врахувати при розробці власного вебзастосунку. Серед них персональні рекомендації, адаптивний інтерфейс, підтримка українського контенту, інтеграція з картами, можливість збереження вибраного, класифікація місць та подій за категоріями. Це дозволяє сформувати уявлення про майбутню платформу, яка буде зручною для користувачів.

Результати порівняння аналогів наведено у таблиці 1.1.

Таблиця 1.1. – Порівняльний аналіз аналогів

Критерій	MoeMisto.ua	Booking.com	TripAdvisor	Власна розробка
Персоналізовані рекомендації	-	+	+	+
Відгуки користувачів	-	+	+	+
Адаптивність інтерфейсу	+	+	+	+
Підтримка українських місцевих подій	+	-	-	+

Продовження таблиці 1.1

Критерій	MoeMisto.ua	Booking.com	TripAdvisor	Власна розробка
Можливість збереження вибраного	-	+	+	+
Фільтр за категорією	+	+	+	+
Можливість побудови маршрутів до локації	-	-	-	+
Загальна оцінка	40%	70%	70%	100%

Порівняльний аналіз сучасних вебсайтів MoeMisto.ua, Booking.com та TripAdvisor дозволив оцінити їхні можливості відповідно до потреб користувача, який шукає туристичні пам’ятки та події в Україні. Аналіз показав, що кожен з цих сервісів має певні переваги, але вони не реалізують всіх функціональних можливостей, необхідних для зручного та персоналізованого пошуку туристичних місць та подій.

MoeMisto.ua інформує про українські місцеві події, пропонує фільтрацію за категоріями та має адаптивний дизайн для різних пристроїв. Однак йому бракує функціональності, яка б покращила користувацький досвід. На платформі відсутні персоналізовані рекомендації, які б дозволили користувачам знаходити нові цікаві місця на основі їхніх минулих вподобань. Також немає можливості прочитати відгуки інших або написати свій, що ускладнює оцінку якості локації чи події. Також відсутність функції збереження улюблених локацій чи подій та відсутність можливості побудови маршрутів змушує користувачів використовувати інші інструменти або нотатки для планування.

Міжнародні сайти, такі як Booking.com та TripAdvisor, мають ширший функціонал. Вони надають персоналізовані пропозиції, систему відгуків та оцінок, збереження обраного та фільтрацію на основі декількох критеріїв. Їхні вебсторінки також адаптивні. Але їх міжнародна спрямованість означає, що

місцеві українські події або менш відомі місця не так добре представлені, як на спеціалізованих українських сайтах. Ще одним недоліком є відсутність можливості побудувати маршрут до обраної локації безпосередньо на сайті. Користувачеві пропонується лише побачити локацію на карті, а для перегляду маршруту потрібно скористатись окремою картографічною програмою, що є менш зручним.

Таким чином, дослідження аналогів показало відсутність єдиного ресурсу, який би поєднував український місцевий матеріал із сучасними та зручними для користувача функціями, такими як персоналізація, збереження обраного, система відгуків, вбудований механізм прокладання маршрутів.

Для того, щоб створити максимально зручний і корисний інструмент для українських користувачів, які шукають місця та події в Україні, було прийнято рішення про розробку власного вебзастосунку. Це дозволить включити всі важливі функціональні можливості, які були визначені під час аналізу існуючих рішень. А саме надання персоналізованих рекомендацій з урахуванням інтересів користувача, система відгуків та рейтингів для об’єктивної оцінки, адаптивний та мінімалістичний інтерфейс, підтримка українського контенту, інтеграція карт та можливість прокладання маршрутів безпосередньо на сайті, додавання обраних місць та подій до збережених для подальшого планування дозвілля. Реалізація цих функцій в одній системі дозволить створити платформу, яка буде зручною для користувачів, що планують своє дозвілля в Україні.

1.3 Аналіз методів розв’язання задачі

Розробка вебзастосунку для пошуку та рекомендації туристичних місць та подій передбачає використання сучасних технологій, здатних забезпечити швидкий доступ до інформації, зручність пошуку, а також особисті рекомендації. Для ефективного розв’язання поставлених задач існують різні техніки та підходи, які мають свої сильні сторони та обмеження.

Для того, щоб розробленим вебзастосунком було зручно користуватися на будь-яких розмірах та типах екранів, його інтерфейс повинен вміти адаптуватися, тобто автоматично підлаштовуватися до розміру екрану, на якому його переглядають. Без такої адаптації сайт, що добре виглядає на комп'ютері, може бути абсолютно незручним або навіть нечитабельним на телефоні.

Для вирішення цієї задачі, тобто щоб вебзастосунок можна було комфортно використовувати з різних пристроїв, було застосовано метод адаптивного макетування. Метод дозволяє динамічно налаштовувати зовнішній вигляд сайту залежно від роздільної здатності екрану. Він реалізований за допомогою сучасних інструментів вебстилізації CSS, а саме технологій CSS Grid та Flexbox, у поєднанні з медіа-запитами [9].

Flexbox допомагає розташовувати елементи в один ряд або стовпчик і керувати проміжками між ними. CSS Grid дозволяє створювати складніші двовимірні сітки, що складаються з рядків та стовпців, і точно розміщувати елементи в цих сітках.

Адаптивність активується за допомогою медіа-запитів. Це спеціальні правила в CSS, які дозволяють застосовувати різні стилі залежно від певних характеристик пристрою. Наприклад, за допомогою медіа-запиту можна встановити правило, що якщо ширина екрану менша за 800 пікселів, то колонки контенту мають розташовуватися одна під одною, а не поруч, а розмір шрифту тексту має стати більшим. Таким чином, сайт динамічно змінює свій зовнішній вигляд, щоб залишатися зручним на будь-якому екрані.

Перевагами такого підходу є його гнучкість, адже елементи можуть плавно змінювати своє положення та розміри, та простота реалізації. З недоліків є те, що розробка адаптивного дизайну потребує додаткового часу на тестування. Необхідно перевірити, як сайт виглядає та працює на різних розмірах екранів, щоб переконатися у відсутності помилок верстки та забезпечити якісний досвід для всіх користувачів.

Для наочного відображення розташування подій та визначних місць на карті у вебзастосунку використовується сервіс Google Maps API [10]. Це набір інструментів від Google, який дозволяє розробникам вбудовувати Google Карти та їхні функції безпосередньо на власний вебсайт. У контексті даної розробки це дозволяє показувати інтерактивну карту прямо на сторінці конкретного місця. Під інтерактивністю мається на увазі, що користувачі можуть робити те саме, що й на звичайних Google Картах. Тобто перетягувати карту мишкою, щоб побачити навколишню територію, наближати чи віддаляти масштаб за допомогою кнопок або коліщатка миші.

На цій карті за допомогою маркерів, маленьких візуальних позначок, позначаються точні місця розташування туристичних об’єктів чи подій. Кожний маркер ставиться у відповідності до географічних координат, тобто широти та довготи, які для кожного місця зберігаються у підготовленому списку даних, що виконує роль бази даних.

Однією з головних переваг використання Google Maps API є висока інтерактивність та довіра користувачів. Більшість людей добре знайомі з тим, як виглядають і працюють Google Карти, тому їм легко орієнтуватися на вбудованій карті сайту, і вони зазвичай довіряють її точності. А також завдяки сучасним інструментам та допоміжним бібліотекам впровадження базових функцій, таких як показ карти та маркерів, є відносно простим завданням для розробника у порівнянні зі створенням подібного функціоналу власноруч від початку.

Однак, існують певні недоліки та ризики. Робота карти на сайті повністю залежить від того, чи працюють сервери Google Maps. Якщо у Google виникнуть технічні проблеми або збої, карта на сайті може перестати відображатися або працювати коректно. Також варто зауважити, що Google, як власник сервісу, може в майбутньому змінити умови використання, цінову політику або навіть технічні деталі роботи API, що може потребувати внесення відповідних змін у код розробки для підтримки її працездатності.

Для того, щоб зробити взаємодію з сайтом більш цікавою та заохотити користувачів повертатися, у системі реалізовано алгоритми персоналізованих рекомендацій. Кожному зареєстрованому користувачеві показуватиметься не просто загальний список місць, а добірка, яка відповідатиме саме його індивідуальним смакам та інтересам. Це допомагає легше знаходити нові цікаві локації чи події.

Найбільш популярними підходами є контентний підхід та колаборативна фільтрація. Контентний підхід аналізує вподобання користувача на основі взаємодії з категоріями, збереженнями, переглядами. Колаборативна фільтрація вивчає поведінку схожих користувачів.

У вебзастосунку, що розробляється, використовується контентний підхід до рекомендацій. Цей метод працює, аналізуючи те, що саме конкретний користувач робив на сайті раніше. Алгоритм аналізує, які місця користувач додав у збережені та яким місцям він поставив високу оцінку, наприклад, 4 чи 5 зірок. Дана інформація про дії користувача зберігається у локальному сховищі браузера.

Далі система визначає, до яких категорій, наприклад, «Музеї», «Парки», «Заклади», «Події» належать ті місця, які сподобалися користувачеві. Категорії, з місцями з яких користувач взаємодіяв найактивніше, тобто найчастіше зберігав або високо оцінював, система вважає його улюбленими або пріоритетними.

На основі визначених пріоритетних категорій система потім шукає інші місця, які також належать до цих категорій, але які користувач ще не зберігав і не оцінював. Тобто, вона намагається знайти щось схоже на те, що користувачеві вже сподобалось, але ще невідоме йому. Зі знайдених місць кандидатів можуть бути відібрані ті, що мають найвищий загальний рейтинг серед інших користувачів. Кілька таких найкращих пропозицій відображаються користувачеві у спеціальному блоці «Для Вас» на головній сторінці.

Варто зазначити, що це відносно простий варіант контентного підходу. Він добре працює, щоб пропонувати схожі місця в межах тих категорій, якими

користувач вже цікавився. Однак, він менше підходить для того, щоб рекомендувати щось зовсім нове з абсолютно інших категорій, які користувач ще не досліджував. Ефективність контентних рекомендацій також залежить від того, наскільки активно користувач взаємодіє з сайтом, тобто чим більше місць він збереже та оцінить, тим точнішими будуть рекомендації. Такий підхід потребує збору та обробки даних про взаємодію кожного окремого користувача, щоб сформувати для нього персональну добірку.

Аналіз методів свідчить про те, що найкращим способом реалізації поставлених задач є використання комбінації сучасних вебтехнологій таких, як адаптивний дизайн, інтерактивна карта, фільтри для персоналізації. Такий підхід забезпечить розробку інтуїтивно зрозумілого, корисного і функціонального вебзастосунку.

1.4 Постановка задач дослідження

Проаналізувавши функціональні можливості, переваги та недоліки існуючих туристичних вебплатформ, таких як MoeMisto, Booking.com та TripAdvisor, було визначено основні задачі, які необхідно реалізувати під час розробки вебзастосунку для пошуку та рекомендації туристичних місць і подій міста:

– розробити адаптивний інтерфейс, який забезпечить коректне відображення контенту на різних пристроях, а також забезпечить зручну навігацію по сайту;

– реалізувати фільтрацію туристичних місць та подій за категоріями;

– розробити сторінку місця або події, яка міститиме фотографії, назву, опис, адресу, ціновий діапазон, відгуки, кнопку переходу на платформу купівлі квитків для подій;

– розробити алгоритм збереження обраних місць/подій зареєстрованими користувачами для подальшого перегляду;

– розробити алгоритм персоналізованих рекомендацій на основі вподобань користувача, збережених місць;

– розробити алгоритм додавання відгуків та залишання оцінок зареєстрованими користувачами;

– розробити модуль інтеграції з Google Maps API для візуалізації місць, подій на мапі та побудови маршрутів;

– розробити модуль реєстрації та авторизації користувача;

– провести тестування вебсистеми згідно поставлених задач.

1.5 Висновки

У першому розділі було проведено аналіз існуючих туристичних вебплатформ, зокрема MoeMisto, Booking.com та TripAdvisor. У процесі аналізу було розглянуто функціональні можливості, переваги й недоліки кожної з платформ, інтерфейс та можливість персоналізації. На основі порівняльного аналізу визначено, що існуючі сервіси мають широкий функціонал, однак не реалізують всіх можливостей для планування відпочинку.

Було також проаналізовано методи до реалізації програмного продукту, а саме використання сучасних вебтехнологій, API інтеграцій, системи фільтрації та рекомендацій.

На основі аналізу було сформовано перелік задач для реалізації власної вебсистеми, серед яких – створення адаптивного інтерфейсу, реалізація пошуку й фільтрації, інтеграція мапи, впровадження персоналізованих рекомендацій та реалізація системи відгуків.

Таким чином, було підтверджено актуальність власної розробки, яка об’єднає функціональні можливості існуючих аналогів, усуне їх недоліки та забезпечить користувачам інтуїтивно зрозумілий, зручний та ефективний інструмент для планування відпочинку.

2 РОЗРОБКА МОДЕЛІ ТА АЛГОРИТМІВ ВЕБЗАСТОСУНКУ

2.1 Аналіз вхідних даних

Для розробки вебзастосунку для пошуку та рекомендації туристичних місць і подій потрібно визначити вхідні дані, які будуть оброблятися в системі.

Основу контенту вебсистеми складають дані про самі туристичні об’єкти. Це можуть бути як постійно існуючі місця, такі як музеї, парки, ресторани, так і тимчасові події, наприклад, фестивалі, виставки, концерти. У вебзастосунку, що розробляється, вся ця інформація зберігається у спеціальному підготовленому списку, який виконує роль простої бази даних. Для кожного такого об’єкта потрібно мати повний набір характеристик, щоб користувач отримав вичерпну інформацію. Сюди входять назва об’єкта, його детальний опис, адреса розташування, приналежність до певної категорії, орієнтовний ціновий діапазон, декілька фотографій, щоб можна було створити карусель, а також точні географічні координати, широта та довгота, які необхідні, щоб правильно показати об’єкт маркером на інтерактивній карті. Якщо об’єкт є подією, то крім загальної інформації, обов’язково вказуються дата та час її проведення, а також посилання на зовнішній сайт, де можна придбати квитки.

Дуже важливою частиною вхідних даних є також дії та інформація від самих користувачів. Коли незареєстрований користувач вирішує створити акаунт, він надає системі свої дані для реєстрації, а саме ім’я користувача, адресу електронної пошти та пароль. Для подальшого входу в систему користувач вводить своє ім’я та пароль. Коли зареєстрований користувач залишає відгук про місце, його текст стає вхідними даними, до яких система автоматично додає інформацію про автора, а саме його ім’я користувача та дату написання. Під час залишання оцінки місця, обираючи кількість зірочок, користувач надає системі числове значення від 1 до 5. Кожне натискання кнопки «Зберегти» біля певного місця теж важливий вхідний сигнал для системи, який фіксується і прив’язується до профілю конкретного користувача. Навіть такі прості дії, як вибір категорії

для перегляду або введення тексту в рядок пошуку, є вхідними даними, які система використовує для фільтрації та відображення відповідної інформації.

Для функції побудови маршрутів на карті система також потребує вхідних даних від користувача. Це початкова точка маршруту, якою може бути або адреса, яку користувач вводить вручну, або його поточні географічні координати, які браузер може визначити за дозволом користувача. Також це обраний користувачем спосіб пересування, наприклад, автомобілем, пішки, велосипедом чи громадським транспортом.

Окремо варто відзначити дані, що є вхідними для алгоритму персоналізованих рекомендацій. Хоча користувач безпосередньо не вводить свої інтереси, система використовує як вхідні дані результати його попередніх дій. Аналізується список збережених користувачем місць та ті місця, яким він поставив високі оцінки. Саме ця історія взаємодії користувача з контентом сайту слугує основою для того, щоб алгоритм міг запропонувати йому щось потенційно цікаве у блоці «Для Вас».

Крім даних про місця та дій користувачів, для коректної роботи певних модулів потрібні і конфігураційні вхідні дані. Це API-ключ для сервісу Google Maps, який має бути безпечно збережений і наданий системі для того, щоб вона могла взаємодіяти з картами.

Технічно розробка вебзастосунку відбуватиметься з використанням бібліотеки React.js [11], яка підтримує створення інтерактивного інтерфейсу користувача. Інтерфейс буде оформлятись за допомогою CSS [12]. Для представлення об’єктів на карті використовуватиметься Google Maps API, що дозволяє динамічно відображати місця на карті.

Отже, системі потрібно працювати з різноманітними типами вхідних даних, отриманих від користувачів або отриманих із бази даних. Ними можуть бути назви туристичних пам’яток, опис подій, фотографії, адреси, координати на карті, ціни, рейтинги та дані, залишені користувачами під час взаємодії з сайтом, тобто збережені події, відгуки. Це і статична інформація про місця та події, і

динамічні дані, що надходять від дій користувачів, і географічні координати, і конфігураційні параметри. Усі ці дані необхідно збирати, зберігати, обробляти, тобто фільтрувати за категоріями чи пошуком, аналізувати для рекомендацій, передавати в зовнішні сервіси, наприклад, Google Maps та належним чином комбінувати між собою для створення зручної вебсистеми.

Відвідувач повинен мати можливість швидко знайти саме те, що йому цікаво – концерт, виставку, визначну пам’ятку чи місце відпочинку. Тому система повинна обробляти пошукові запити користувача, точно фільтрувати результати за обраними категоріями та надавати корисні підказки чи персоналізовані рекомендації. Це забезпечить легкість і ефективність взаємодії з сайтом.

2.2 Розробка моделі вебзастосунку

Для детального опису взаємодії користувача з розробленим вебзастосунком для пошуку та рекомендації туристичних місць міста показано модель роботи системи у вигляді UML діаграми (див. рисунок 2.1).

Згідно з розробленою моделлю, початкова взаємодія користувача із системою відбувається на головній сторінці. Незареєстрованому користувачеві надається можливість перегляду туристичних місць за категоріями, а також використання пошуку конкретних локацій за назвою. При введенні запиту в поле пошуку система динамічно відображає випадаючий список з пропозиціями. Після вибору категорії або завершення пошуку система відображає перелік відповідних туристичних місць у вигляді карток, що містять зображення, назву, середній рейтинг та елементи керування «На мапі», «Зберегти». Користувач може перейти до детального перегляду місця, натиснувши на його картку.

Вебзастосунок надає можливість реєстрації нових користувачів та авторизації існуючих через відповідні модальні вікна. Процес реєстрації передбачає введення імені користувача, адреси електронної пошти та пароля. Система здійснює перевірку унікальності імені користувача та email. У разі

успішної валідації дані нового користувача зберігаються у локальному сховищі браузера. Процес входу передбачає введення імені користувача та пароля. Система перевіряє їх відповідність збереженим даним. Після успішного входу система оновлює інтерфейс, відображаючи ім’я користувача та надаючи доступ до перегляду збережених місць та виходу з системи.

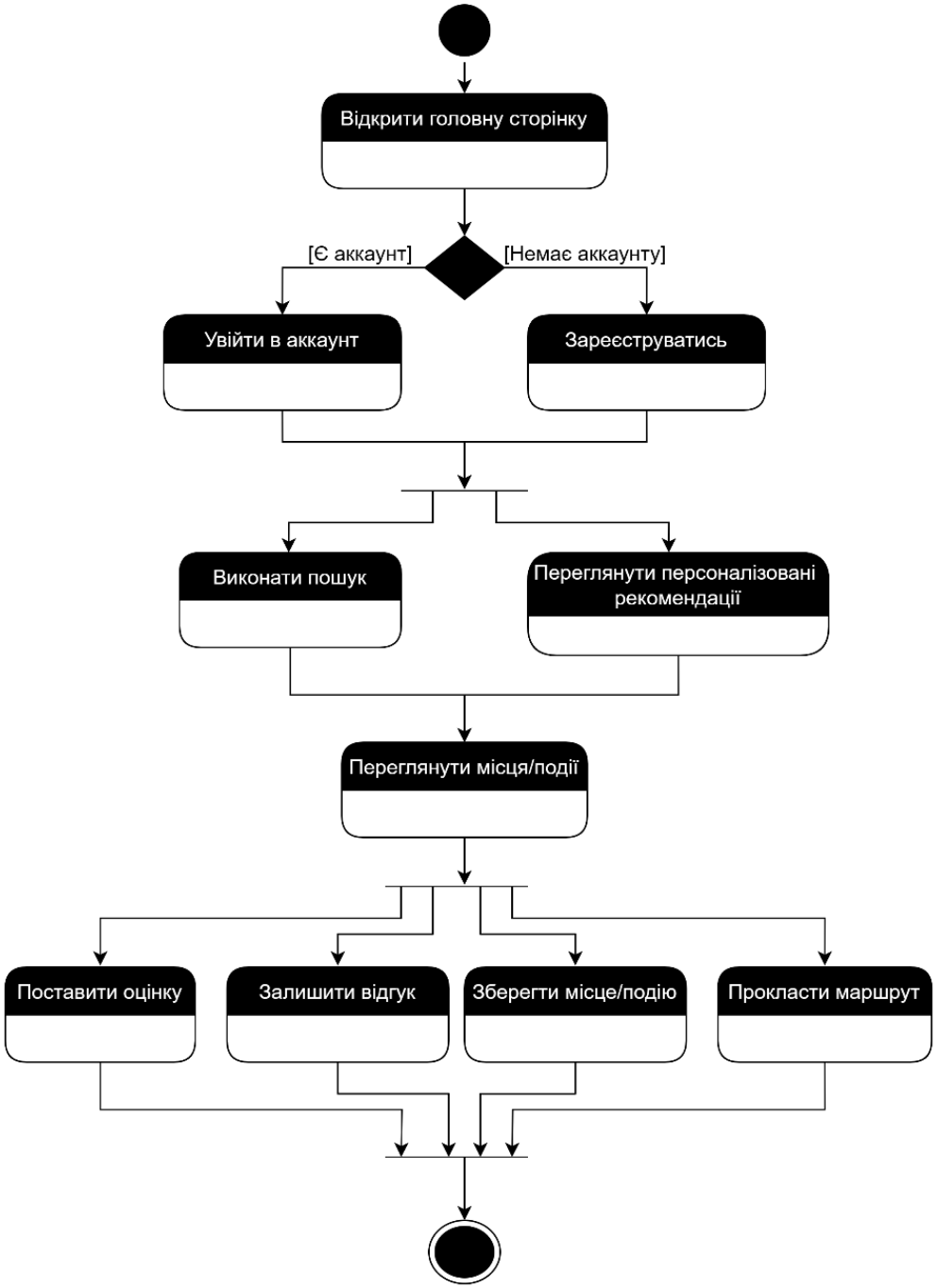

Рисунок 2.1 – Модель роботи вебсистеми у вигляді UML діаграми

Для зареєстрованого користувача на головній сторінці додатково відображається персоналізований блок рекомендацій. Алгоритм формування рекомендацій аналізує збережені користувачем місця та місця з високими оцінками, визначає пріоритетні категорії та пропонує інші популярні місця з цих категорій, які користувач ще не зберіг чи не оцінив.

На сторінці детального перегляду місця користувач отримує розширену інформацію: карусель з кількох зображень, опис, адресу, ціновий діапазон, мапу з розташуванням локації. Зареєстрований користувач має можливість зберегти місце, поставити оцінку та додати текстовий відгук. Система також розраховує та відображає середній рейтинг місця на основі оцінок користувачів. Для місць категорії «Події» доступна кнопка для переходу на зовнішній ресурс для придбання квитків.

Також реалізовано функціонал прокладання маршруту. Користувач може ввести початкову адресу або використати поточну геолокацію, вибрати спосіб пересування, після чого система, використовуючи DirectionsService та DirectionsRenderer Google Maps API, розраховує та візуалізує маршрут безпосередньо на карті сторінки.

2.3 Розробка алгоритмів вебзастосунку

Для забезпечення функціонування вебзастосунку для пошуку та рекомендації туристичних місць міста було розроблено алгоритми, що забезпечують роботу основних модулів. Загальний алгоритм роботи системи представлено у вигляді діаграми (див. рисунок 2.2).

Згідно з даним алгоритмом, робота програмного застосунку починається з ініціалізації інтерфейсу користувача та завантаження початкового набору даних про туристичні місця з модуля placesData. Система також перевіряє наявність активної сесії користувача у локальному сховищі браузера. Головний модуль відображення динамічно формує початковий вигляд сторінки. Для неавторизованого користувача відображаються кнопки категорій та поле

пошуку. Для авторизованого користувача додатково завантажується та відображається блок персоналізованих рекомендацій «Для Вас», згенерований на основі збережених даних користувача.

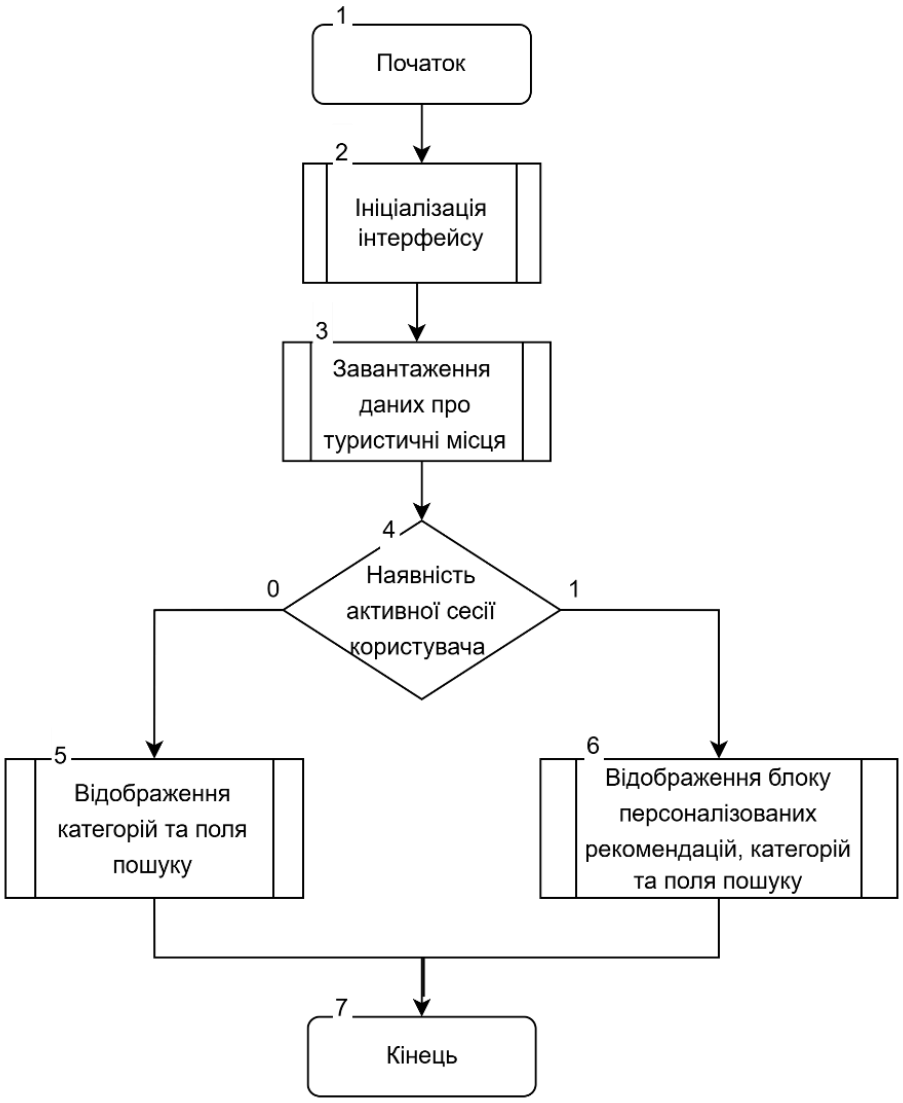

Рисунок 2.2 – Алгоритм роботи вебсистеми

Алгоритм пошуку та фільтрації активується при взаємодії користувача з полем пошуку або кнопками категорій. При введенні тексту користувачем у поле пошуку, система реагує в реальному часі. З кожним новим символом вона бере введений текст, перетворює його на нижній регістр для пошуку без урахування великих чи малих літер, і порівнює його з назвами всіх туристичних місць також

у нижньому регістрі за принципом входження, тобто перевіряє, чи містить назва місця введений текст. Ті місця, чиї назви містять введений фрагмент тексту, потрапляють у тимчасовий відфільтрований список. Невелика частина цього списку одразу відображається у випадаючому списку під полем пошуку, щоб користувач міг швидко вибрати потрібне місце, не дописуючи назву до кінця. Одночасно, якщо в цей момент жодна категорія не вибрана, основний список місць на сторінці також оновлюється в реальному часі, показуючи всі місця, що відповідають поточному пошуковому запиту, у вигляді карток.

Якщо ж користувач натискає на кнопку категорії, наприклад, «Музеї», система встановлює цю категорію як активний фільтр. Після цього основний список місць оновлюється, відображаючи лише ті об’єкти, які належать до вибраної категорії. Випадаючий список результатів пошуку при цьому ховається, оскільки пошуковий запит стає порожнім.

Коли користувач підтверджує свій пошуковий запит натисканням клавіші Enter у полі пошуку чи вибором конкретного місця зі списку підказок, що одразу перенаправляє на сторінку місця, чи просто кліком мишки поза межами поля пошуку та випадаючого списку, то випадаючий список автоматично зникає. Це важливо для зручності, оскільки залишає видимим основний, вже відфільтрований за останнім пошуковим запитом, список місць у вигляді карток. Це дозволяє користувачеві спокійно переглянути результати пошуку без елементів інтерфейсу, що можуть перекривати контент.

За взаємодію зареєстрованого користувача з функціями збереження місць, додавання відгуків та виставлення оцінок відповідає алгоритм управління даними користувача. При спробі зберегти місце система перевіряє статус авторизації, зчитує поточний список збережених місць користувача, додає або видаляє ідентифікатор обраного місця та записує оновлений список назад. Так само при додаванні відгуку або оцінки, система перевіряє авторизацію, зчитує відповідні дані для конкретного місця, додає відгук або оцінку користувача та

зберігає оновлені записи. Після оновлення оцінки користувача система також ініціює перерахунок та оновлення середнього рейтингу для даного місця.

Для кращого розуміння роботи алгоритму генерації персоналізованих рекомендацій було розроблено схему (див. рисунок 2.3).

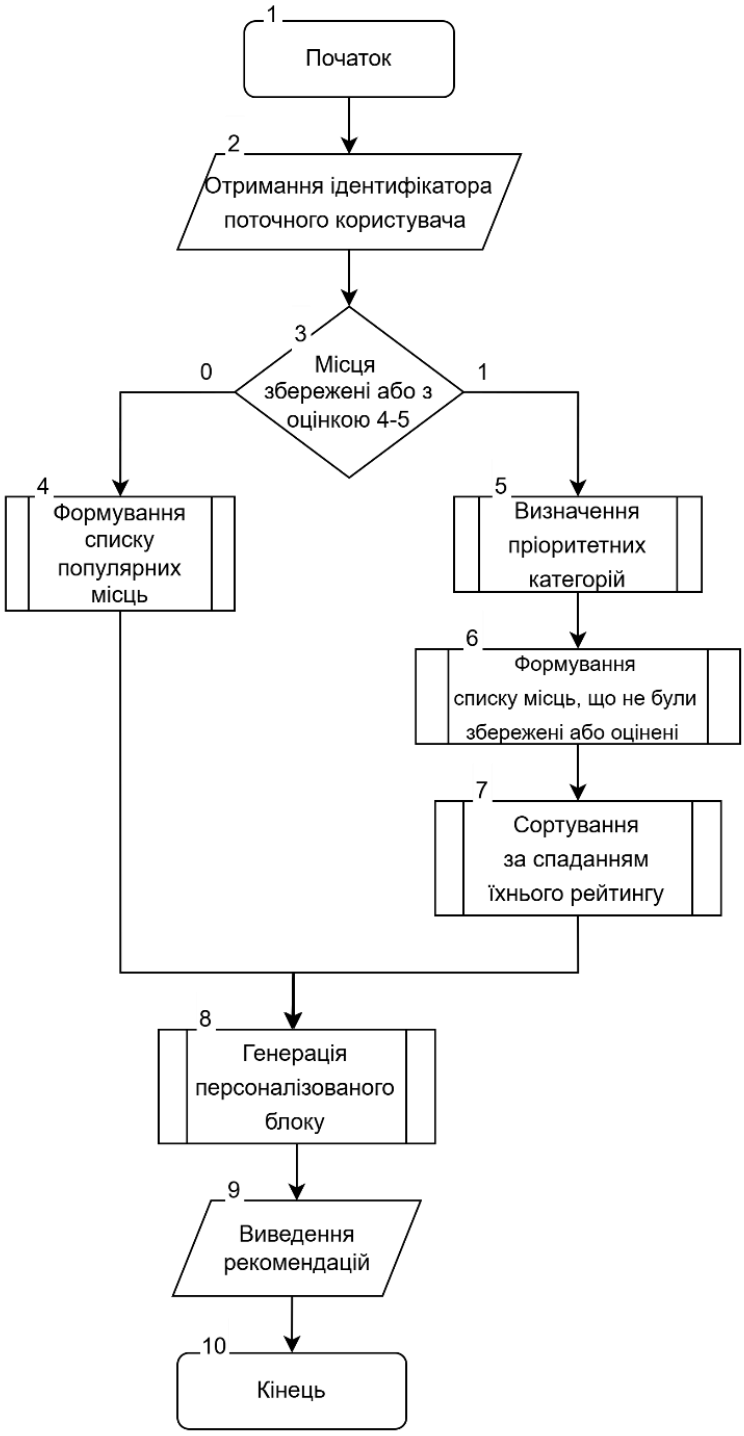

Рисунок 2.3 – Алгоритм генерації персоналізованих рекомендацій

Згідно з цим алгоритмом, першим кроком є отримання ідентифікатора поточного користувача та доступ до даних у місцевому сховищі. Система вибирає місця, збережені користувачем, а також ті, яким він поставив високу оцінку 4 або 5. На основі категорій цих місць для даного користувача визначаються ті категорії, які зустрічаються найчастіше. Далі система формує список місць, що належать до визначених раніше категорій, але ще не були збережені або оцінені користувачем. Отриманий список сортується за спаданням їхнього середнього рейтингу. Фінальний набір рекомендацій вибирається з початку відсортованого списку, тобто в рекомендації потрапляють перших 4 місця. Для нових користувачів або тих, хто ще не має достатньо даних для персоналізації, алгоритм передбачає відображення загально популярних місць, відсортованих за їхнім середнім рейтингом.

Результат роботи алгоритму у вигляді списку рекомендованих місць передається модулю відображення для генерації відповідного блоку «Для Вас» на головній сторінці.

2.4 Висновки

У другому розділі було проаналізовано вхідні дані, що забезпечують роботу вебзастосунку для пошуку та рекомендації туристичних місць і подій. Розглянуто різні типи інформації, з якими взаємодіє система, а саме статичні дані про місця та події такі, як назва, опис, адреса, координати, зображення, а також динамічні дані, що надходять від користувачів у процесі використання вебзастосунку, тобто відгуки, оцінки, збереження, пошукові запити, обрана категорія, маршрут.

Було розроблено і представлено модель взаємодії користувача із системою у вигляді UML-діаграми, що відображає логіку та сценарії використання вебзастосунку від перегляду місць до використання карти, фільтрації, авторизації, оцінювання та коментування.

Також було описано алгоритми, які забезпечують функціональність програмних модулів системи. Було розроблено загальний алгоритм роботи вебзастосунку, алгоритм пошуку та фільтрації об’єктів за категоріями та назвами, алгоритм збереження локацій та оновлення даних, а також алгоритм генерації персоналізованих рекомендацій, що враховує попередню активність користувача для формування пропозицій у блоці «Для Вас».

Це дало розуміння, які саме дані потрібні для роботи вебзастосунку, як вони надходять і використовуються, а також як користувачі взаємодіють із системою. Створені діаграми та алгоритми показали, як працюють окремі частини вебзастосунку. Отже, отримані результати стали важливими для подальшого процесу розробки функціонального та корисного вебзастосунку для туристів.

3. РОЗРОБКА МОДУЛІВ ВЕБЗАСТОСУНКУ

3.1 Варіантний аналіз і обґрунтування вибору засобів для реалізації

Під час розробки вебзастосунку для пошуку та рекомендації туристичних місць міста важливим етапом є вибір технологій, які забезпечать необхідну функціональність, інтерактивність та зручність для користувача.

Для розробки користувацького інтерфейсу було обрано бібліотеку React. React є інструментом, який допомагає створювати інтерактивні вебсторінки, що швидко реагують на дії користувача. Вона добре підходить для створення односторінкових застосунків SPA [13]. Це означає, що весь сайт працює ніби як одна велика сторінка. Тобто під час переходу між розділами браузер не перезавантажує сайт повністю, а лише динамічно оновлює потрібні частини. Це робить роботу з сайтом швидшою.

Однією з переваг React є висока продуктивність, яка досягається завдяки використанню віртуального DOM, Document Object Model [14]. Замість того, щоб повільно змінювати саму вебсторінку в браузері під час кожної зміни даних, наприклад, при оновленні списку місць, React спочатку робить ці зміни у своїй внутрішній, швидкій копії сторінки в пам’яті, що і є віртуальним DOM. Потім він порівнює цю копію з тим, якою вона була раніше, визначає, що саме змінилося, і після цього вносить тільки ці необхідні зміни у реальну сторінку, яку бачить користувач. Це дозволяє оновлювати інтерфейс значно швидше та ефективніше.

У React є підхід, який дозволяє розділяти весь інтерфейс на невеликі, незалежні частини – компоненти. Такий підхід надає можливість створити компонент один раз, наприклад, кнопку з певним дизайном, картку для відображення інформації про місце, блок для рейтингу, і потім використовувати його в багатьох різних місцях сайту. Кожен такий компонент має свою невелику логіку та зовнішній вигляд. Також такий компонентний підхід спрощує подальшу роботу з розробкою. Для прикладу, якщо потрібно змінити вигляд усіх

карток місць, це робиться лише в одному місці, а саме у файлі компонента картки.

Для того, щоб швидко розпочати розробку, було використано інструмент Create React App, CRA [15]. Це є готовий набір для React-проєктів, який надає вже налаштовану структуру папок та конфігурацію, дозволяючи одразу зосередитись на написанні самого коду вебзастосунку та його модулів.

Переходи між різними розділами вебзастосунку, наприклад, з головної сторінки на сторінку конкретного місця чи події, реалізовано без повного перезавантаження сторінки з сервера. Для цього використовується бібліотека React Router DOM. Вона працює на стороні браузера, відстежує зміни в адресному рядку і, відповідно до поточної адреси, показує користувачеві потрібні React-компоненти.

Для взаємодії з картографічними даними, візуалізації місцезнаходжень та прокладання маршрутів було інтегровано Google Maps JavaScript API. Цей сервіс надає інструменти для роботи з геоданими, включаючи відображення карт, маркерів, розрахунок та візуалізацію маршрутів. Для спрощення інтеграції API з React-компонентами використано бібліотеку @react-google-maps/api [16], яка надає готові React-компоненти GoogleMap, Marker, DirectionsService, DirectionsRenderer.

Центральним компонентом є <GoogleMap>, який безпосередньо відповідає за відображення самої карти на вебсторінці, дозволяючи налаштувати її початковий центр, масштаб та вигляд. Для позначення конкретних точок на цій карті, наприклад, розташування туристичних об’єктів, використовується компонент <Marker>, що розміщує візуальну позначку у заданих географічних координатах.

Щоб розрахувати маршрут між двома точками, застосовується компонент <DirectionsService>. Він приймає інформацію про початкову та кінцеву точки, а також бажаний спосіб пересування, наприклад, автомобіль чи пішки. Після чого звертається до сервісів Google для обчислення маршруту. Отриманий результат,

тобто сам маршрут, потім візуалізується на карті за допомогою компонента <DirectionsRenderer>, який малює лінію шляху та стандартні маркери початку і кінця маршруту A та B.

Оскільки використання Google Maps API потребує секретного API-ключа, його безпечне зберігання є дуже важливим, щоб запобігти несанкціонованому доступу, можливому зловживанню та пов’язаним з цим витратам. З цієї причини ключ не записується безпосередньо у вихідний код програми, який може стати доступним іншим. Безпечне зберігання API-ключа забезпечується за допомогою механізму змінних середовища та файлу .env, що унеможливлює його потрапляння у систему контролю версій [17]. Іншими словами ключ зберігається у спеціальному файлі .env у кореневій папці проєкту. Цей файл додається до .gitignore, тобто до списку ігнорування системи контролю версій, тому він не потрапляє у репозиторій коду. У самому коді програми до ключа відбувається звернення безпечно через об’єкт process.env. При публікації сайту на хостингу цей ключ так само не є частиною коду, а налаштовується як безпечна змінна середовища безпосередньо на сервері хостингової платформи.

Для покращення користувацького досвіду при відображенні сповіщень та реалізації модальних вікон було використано спеціальні бібліотеки react-toastify [18] та react-modal [19] відповідно. Вони надають готові, стилізовані та доступні компоненти, замінюючи стандартні функції браузера alert, prompt.

Бібліотека react-toastify дозволяє показувати неблокуючі спливаючі сповіщення. Вони з’являються на короткий час на екрані, інформуючи користувача про успішні дії, помилки чи попередження. Також варто зазначити, що вони не зупиняють взаємодію користувача з сайтом, на відміну від стандартного alert, який блокує всю сторінку.

Для реалізації модальних вікон, наприклад, для входу та реєстрації застосовано react-modal. Ця бібліотека значно спрощує створення діалогових вікон, які тимчасово з’являються поверх основного вмісту сторінки. Вона керує такими важливими деталями, як правильне фокусування для введення даних

всередині вікна, затемнення фону сторінки, що є доволі зручнішим за використання стандартного prompt чи ручне створення таких вікон.

Візуальне оформлення всього інтерфейсу, тобто те, як виглядають кнопки, текст, картки місць, їхні кольори, шрифти, відступи та загальне розташування на сторінці, реалізовано за допомогою CSS. Використання CSS надає повну гнучкість у налаштуванні зовнішнього вигляду кожного елемента.

Мовою програмування, на якій написана логіка роботи вебзастосунку, а саме обробка натискань кнопок, керування тим, що бачить користувач, збереження даних, взаємодія з Google Maps API, є JavaScript [20]. Для того, щоб зручно описувати, як саме має виглядати користувацький інтерфейс всередині React-компонентів, використовується спеціальне синтаксичне розширення JSX [21]. JSX дозволяє писати код, який схожий на HTML, тобто з тегами , <button>, <img> та іншими, безпосередньо в JavaScript-файлах. Це робить код компонентів більш наочним і зрозумілим, оскільки структура інтерфейсу візуально представлена там, де й логіка її роботи. Перед виконанням у браузері цей JSX-код автоматично перетворюється на звичайні JavaScript-команди.

Отже, обрані технології дозволяють створити інтерактивну, функціональну вебсистему для пошуку та рекомендації туристичних місць з використанням сучасних підходів до веброзробки. Комбінація цих інструментів забезпечує реалізацію всіх основних функцій, включаючи відображення даних, автентифікацію, взаємодію з картою, персоналізацію та збереження стану.

3.2 Розробка модуля реєстрації та авторизації користувача

Модуль автентифікації та керування даними користувача є важливим для реалізації функцій збереження місць, рекомендації, залишання оцінок та відгуків.

Для керування станом автентифікації користувача створено контекст AuthContext. Він надає дочірнім компонентам інформацію про поточний статус входу isLoggedIn та дані поточного користувача currentUser, а також функції для

виконання реєстрації, авторизації та розавторизації. Доступ до цих даних та функцій здійснюється за допомогою useAuth(). Компонент AuthProvider відповідає за ініціалізацію стану автентифікації при завантаженні сторінки, перевіряючи наявність даних про користувача у локальному сховищі.

Робота модуля реєстрації у вигляді UML діаграми наведена на рисунку 3.1.

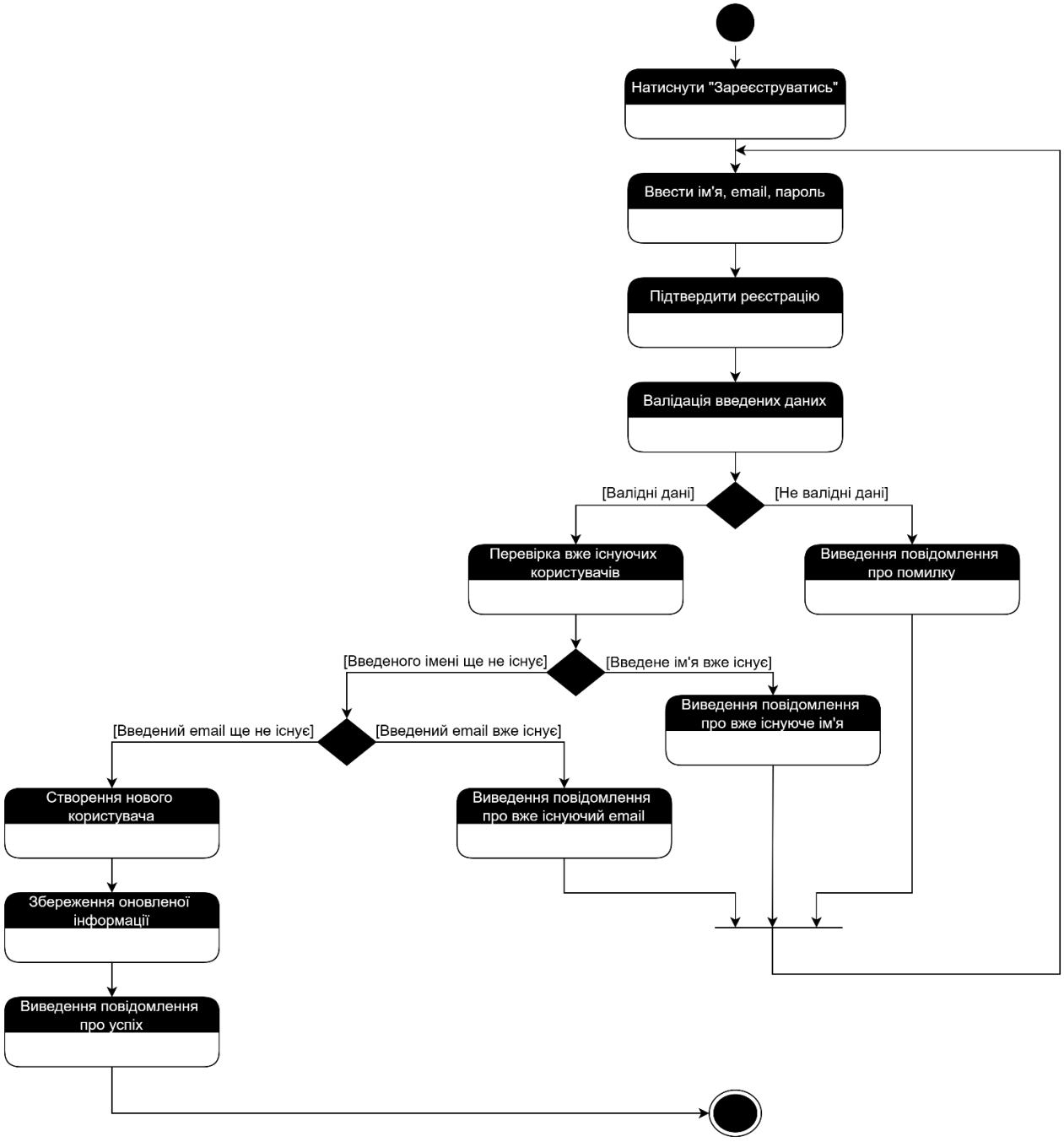

Рисунок 3.1 – Робота модуля реєстрації у вигляді UML діаграми

Алгоритм реєстрації ініціюється через модальне вікно RegistrationModal. Він передбачає отримання від користувача імені, адреси електронної пошти та пароля. Функція register в AuthContext починається з валідації вхідних даних на наявність та коректність формату email. Потім відбувається зчитування масиву існуючих користувачів за ключем STORAGE_KEYS.USERS. Після чого перевіряється унікальність введеного імені користувача та email серед існуючих записів. У разі виявлення дубліката процес переривається з відповідним сповіщенням. В іншому разі створюється та додається до масиву новий об’єкт користувача, що містить ім'я, email та пароль. При успішній реєстрації відбувається інформування про це користувача.

Алгоритм входу, що ініціюється через LoginModal, отримує ім’я користувача та пароль. Після чого зчитує масив існуючих користувачів та шукає користувача за введеним іменем. Якщо користувача знайдено, порівнюється введений пароль зі збереженим. У разі успішної перевірки користувач інформується про успішний вхід, а в іншому разі – про помилку.

3.3 Розробка модуля взаємодії з картою та побудови маршрутів

Модуль взаємодії з картою є важливим для візуалізації географічного розташування туристичних місць та прокладання маршрутів. Його реалізація базується на Google Maps JavaScript API та бібліотеці @react-google-maps/api.

Основним компонентом для роботи з картою є LoadScript, який відповідає за асинхронне завантаження скриптів Google Maps API. Йому передається API-ключ, отриманий із змінних середовища через process.env.REACT_APP_Maps_API_KEY, та список необхідних бібліотек. Для уникнення помилок перед рендерингом LoadScript перевіряється наявність ключа.

Відображення самої карти здійснюється компонентом GoogleMap. Він приймає стилі контейнера mapContainerStyle, початкові координати центру, що відповідають координатам обраного місця та рівень масштабування. Всередині

GoogleMap розміщується компонент Marker, що візуалізує маркер у зазначеній позиції.

Робота модуля побудови маршрутів у вигляді UML діаграми наведена на рисунку 3.2.

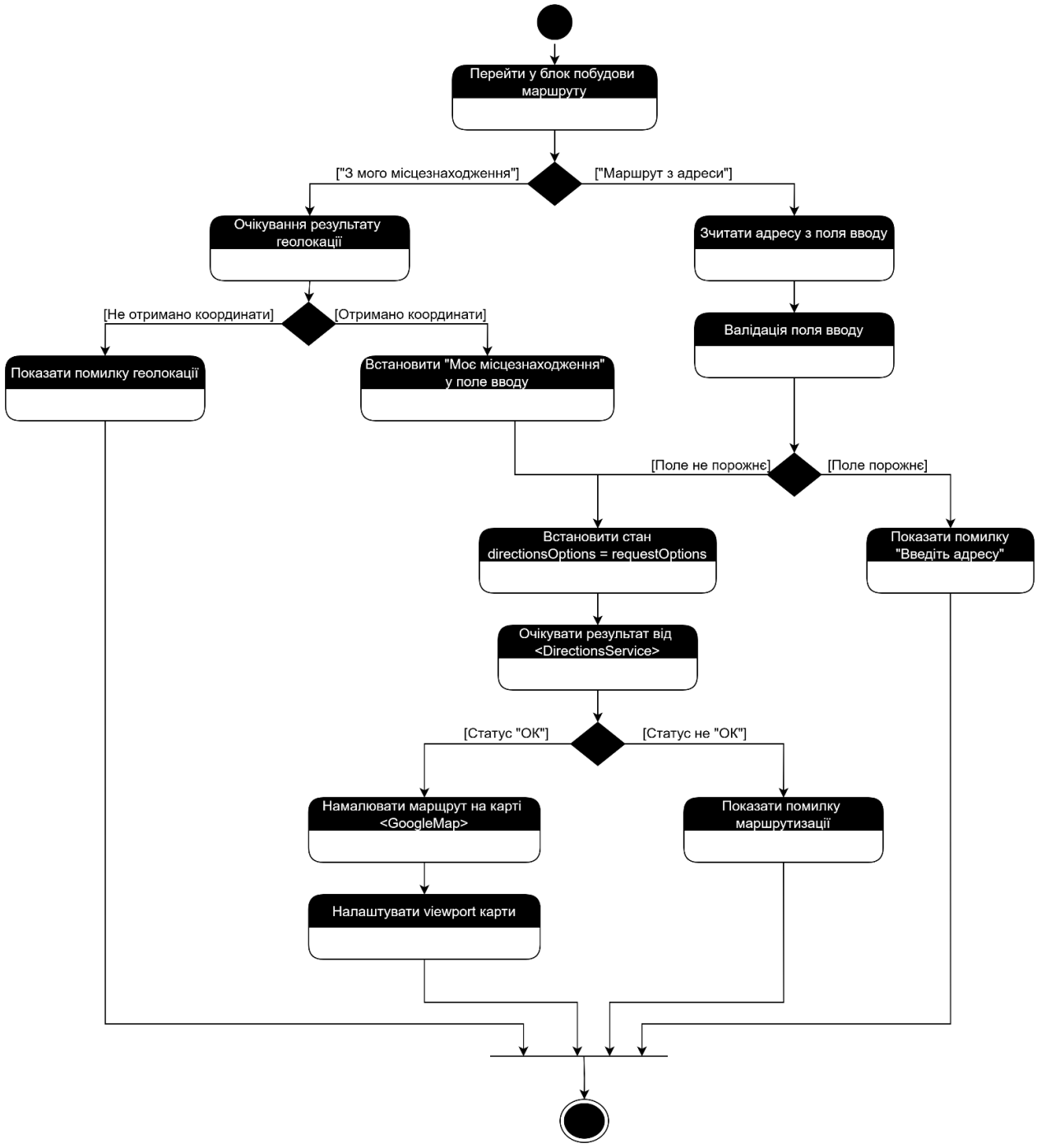

Рисунок 3.2 – Робота модуля побудови маршрутів у вигляді UML діаграми

Функціонал прокладання маршруту починається з ініціалізації запиту. Користувач вводить початкову адресу у поле вводу або натискає кнопку «З мого місцезнаходження». У другому випадку система використовує navigator.geolocation.getCurrentPosition() для отримання поточних координат користувача, обробляючи запит на дозвіл та можливі помилки. Після цього формуються опції запиту. Створюється об’єкт requestOptions, що містить початкову точку, кінцеву точку, обраний спосіб пересування. Цей об'єкт зберігається у стані компонента.

Наступним кроком відбувається виклик сервісу. Зміна стану requestOptions призводить до активації компонента <DirectionsService>. Цей компонент приймає requestOptions та функцію зворотного виклику. Він асинхронно надсилає запит до Google Maps Directions API. Після чого обробляється відповідь. Функція зворотного виклику отримує результат відповіді та статус від API. Якщо статус 'OK', отриманий об’єкт response, що містить деталі маршруту, зберігається у стані компонента directionsResponse. Попередні помилки скидаються. Якщо статус вказує на помилку, стан directionsResponse очищується, а у стан routeError записується відповідне повідомлення для користувача.

Зміна стану directionsResponse призводить до активації компонента <DirectionsRenderer>. Цей компонент приймає об’єкт response та автоматично малює лінію маршруту та маркери початку A і кінця B на екземплярі карти <GoogleMap>. Опції рендерера дозволяють налаштувати вигляд лінії та маркерів. За замовчуванням, рендерер також налаштовує видиму область карти так, щоб вмістити весь маршрут.

3.4 Розробка інтерфейсу вебзастосунку

При розробці інтерфейсу вебзастосунку для пошуку та рекомендації туристичних місць особливу увагу було приділено його зрозумілості, зручності та візуальній привабливості для користувача. Основною кольоровою гамою інтерфейсу було обрано поєднання бірюзового, білого та відтінків сірого.

Бірюзовий колір, що використовується в хедері та для акцентних елементів, асоціюється з подорожами та спокоєм, а білий та світло-сірий фон забезпечують читабельність. Контрастні кольори зелений, жовтий та помаранчевий застосовано для кнопок дій для покращення їх помітності.

При першому відкритті вебзастосунку користувача зустрічає головна сторінка (див. рисунок 3.3), що містить хедер з назвою сайту та кнопками автентифікації, поле пошуку, кнопки категорій.

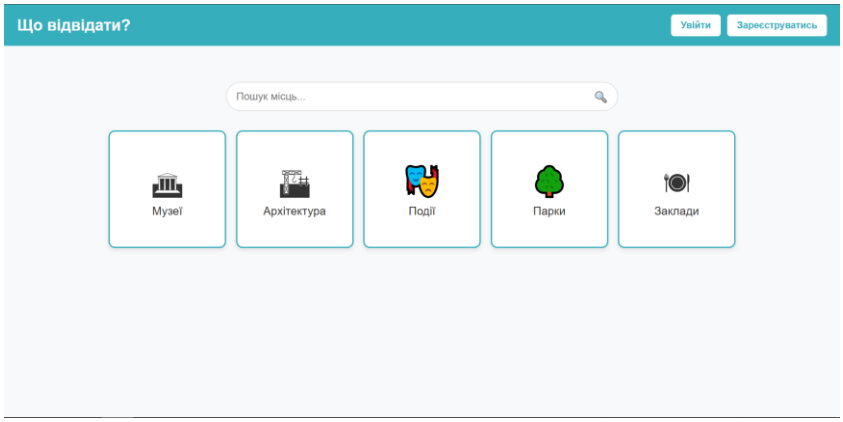

Рисунок 3.3 – Головна сторінка

Для авторизованого користувача вигляд хедера змінюється, відображуючи привітання, ім’я користувача як посилання на збережені місця та кнопку виходу. Також на головній сторінці з’являється блок персоналізованих рекомендацій «Для Вас» (див. рисунок 3.4).

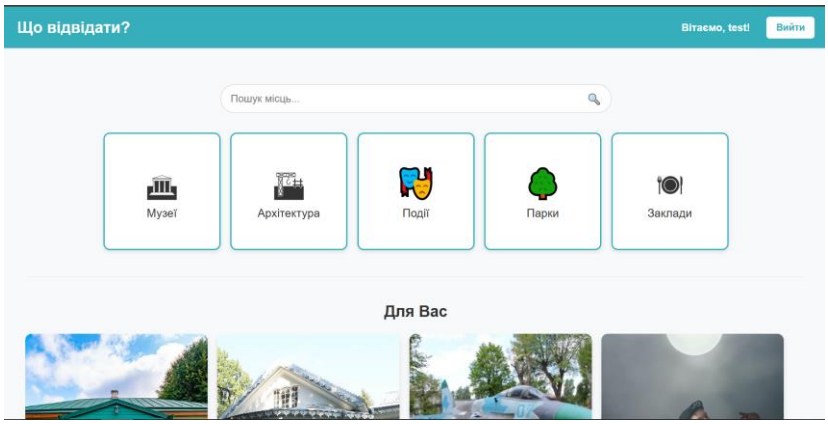

Рисунок 3.4 – Головна сторінка в авторизованого користувача

Інтерфейс побудовано за компонентним принципом з використанням React. Завжди видимий у верхній частині хедер забезпечує навігацію на головну сторінку та доступ до функцій автентифікації. Поле пошуку дозволяє вводити текст і динамічно відображає випадаючий список підказок, який зникає при підтвердженні пошуку. Кнопки категорій реалізовані як інтерактивні елементи з іконками та текстом. Картка місця відображається у списках та рекомендаціях, містить зображення, назву, рейтинг та кнопки дій.

Сторінка місця представляє детальну інформацію, включаючи карусель зображень, опис, адресу, рейтинг, секцію для прокладання маршрутів, інтерактивну карту та секцію відгуків з формою додавання. Також дизайн є адаптивним за допомогою медіа-запитів (див. рисунок 3.5).

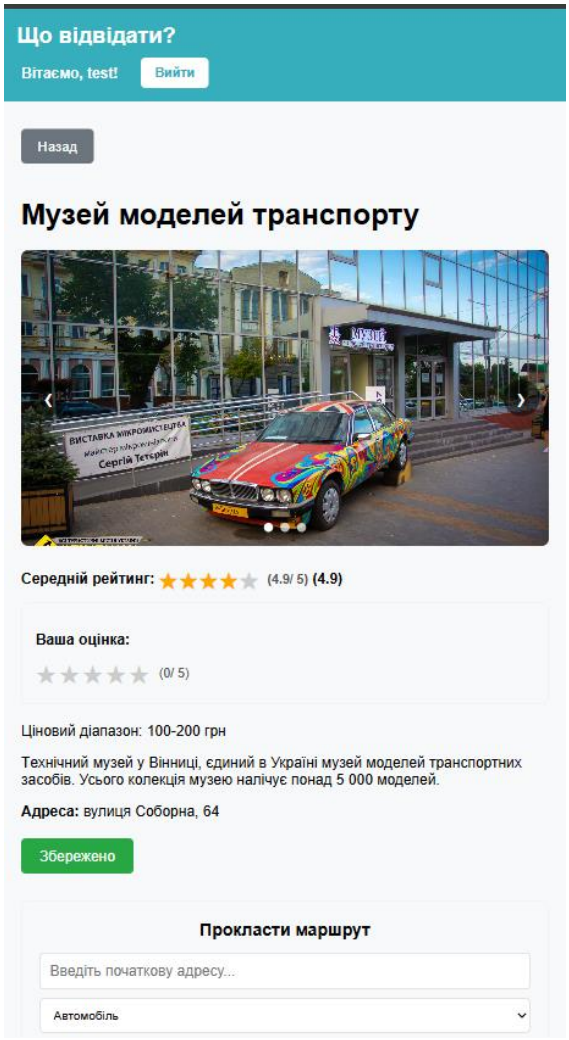

Рисунок 3.5 – Сторінка місця та адаптивний дизайн

Модальні вікна використовуються для процесів входу, реєстрації та відображення інформації. Для інформування про помилки чи попередження застосовуються спливаючі сповіщення (див. рисунок 3.6).

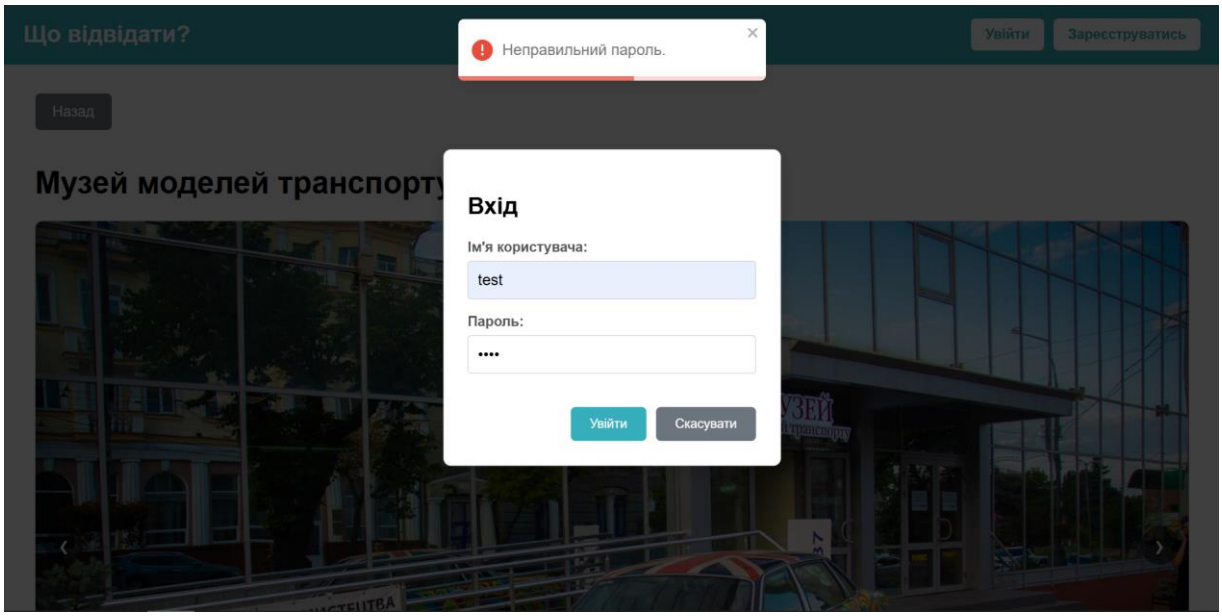

Рисунок 3.6 – Модальне вікно входу та спливаюче сповіщення

Структура сторінок використовує центральний контейнер для обмеження ширини контенту. Для розташування елементів використовуються CSS Flexbox та CSS Grid. Розробка графічного інтерфейсу була проведена з використанням бібліотеки React, синтаксису JSX, стилізації за допомогою CSS та допоміжних бібліотек React Router DOM, @react-google-maps/api, react-modal, react-toastify.

3.5 Висновки

У третьому розділі було обґрунтовано вибір технологій та інструментів, що використовувались для розробки модулів системи. Було обрано бібліотеку React, що дозволяє створити швидкий та інтерактивний інтерфейс завдяки компонентному підходу та використанню віртуального DOM. Обґрунтовано використання мови JavaScript з синтаксисом JSX, бібліотеки React Router DOM для зручної навігації між розділами сайту без перезавантаження сторінки, та

стандартного CSS для гнучкого візуального оформлення. Було аргументовано вибір Google Maps JavaScript API та бібліотеки @react-google-maps/api для реалізації картографічних функцій, а також зазначено важливість безпечного зберігання API-ключа за допомогою змінних середовища та файлу .env. Обґрунтовано використання допоміжних бібліотек react-toastify та react-modal для створення більш зручних сповіщень і модальних вікон замість стандартних браузерних функцій.

Було описано розробку модуля реєстрації та авторизації користувача. Описано алгоритми для реєстрації нового користувача з валідацією та перевіркою унікальності та для процесу входу існуючого користувача з перевіркою пароля, що було продемонстровано діаграмою діяльності.

Також було описано розробку модуля взаємодії з картою та побудови маршрутів. Описано використання компонентів бібліотеки @react-google-maps/api, а саме LoadScript для завантаження API, GoogleMap для відображення карти, Marker для позначення місць. Алгоритм побудови маршруту також було продемонстровано відповідною діаграмою діяльності.

Було описано розробку графічного інтерфейсу вебзастосунку. Розглянуто рішення щодо візуального стилю, кольорової гами та загальної структури сторінок. Описано компоненти інтерфейсу, такі як хедер, поле пошуку, кнопки категорій, картки місць, сторінки місць з каруселлю зображень, картою та секцією відгуків, а також модальні вікна та спливаючі сповіщення.

4 ТЕСТУВАННЯ ПРОГРАМНОГО ПРОДУКТУ

4.1 Тестування вебзастосунку

Тестування програмного забезпечення це дуже важливий етап розробки, спрямований на виявлення помилок, перевірку відповідності функціональним вимогам та забезпечення стабільної роботи програмного продукту. Основна мета тестування полягає у перевірці коректності роботи функцій та взаємодії між модулями системи.

Під час тестування перевіряється відповідність функціональним вимогам, тобто чи виконує програма всі ті завдання, для яких її створювали. Наприклад, для вебзастосунку пошуку та рекомендації туристичних місць, це означає перевірити чи дійсно пошук знаходить потрібні місця, чи правильно працює фільтрація за категоріями, чи може користувач успішно зареєструватися та увійти, чи зберігаються обрані місця.

Також тестування допомагає забезпечити стабільність програмного продукту, тобто переконатися, що він не «зависає» і поводиться передбачувано навіть при не зовсім стандартних діях користувача. Окрім перевірки коректності роботи окремих функцій, важливою частиною є перевірка взаємодії між різними модулями системи. Наприклад, чи правильно оновлюється статус кнопки «Зберегти» після того, як користувач увійшов у систему, чи коректно передаються координати з даних про місце на карту. Загалом, ретельне тестування дозволяє не тільки виправити помилки до того, як їх побачать кінцеві користувачі, але й підвищити загальну якість продукту, зробити його більш надійним та зручним, що в кінцевому підсумку призведе до кращого досвіду для всіх, хто буде користуватися розробленим вебзастосунком.

Першим кроком є функціональне тестування, під час якого перевірялася працездатність вебзастосунку. Було перевірено коректність навігації між різними сторінками сайту, адже саме від якості навігації залежить, наскільки

легко користувач зможе знайти потрібну інформацію та взаємодіяти з усіма можливостями вебзастосунку.

Було перевірено, чи всі переміщення між сторінками працюють належним чином, а саме чи призводить натискання на кнопки категорій до миттєвого відображення списку місць саме цієї категорії з відповідним заголовком, чи кожна картка місця у списку на головній сторінці, в результатах пошуку чи в рекомендаціях є діючим посиланням, що без помилок відкриває сторінку з детальною інформацією про відповідну локацію, чи посилання у хедері точно перенаправляють на очікувані сторінки. Окрему увагу було приділено роботі кнопки «Назад», яка реалізована через історію браузера. Також перевірялася відсутність конфліктів при використанні як власних навігаційних елементів сайту, так і стандартних кнопок «вперед/назад» самого браузера.

Тестувалася робота пошуку, оскільки він є основним інструментом для користувачів, що шукають щось конкретне серед великої кількості місць та подій. Перевірка включала введення різноманітних пошукових запитів, від одного символу до повних назв, включаючи запити різним регістром літер, щоб оцінити релевантність та повноту результатів. Під час введення тексту в реальному часі відстежувалася швидкість реакції системи та перевірялося, чи динамічно та без помітних затримок з’являється випадаючий список з підказками (див. рисунок 4.1).

Оцінювалася зручність використання цього списку, тобто чи легко вибрати потрібний елемент, чи є можливість прокрутки, якщо підказок багато. Перевірялося, чи коректно працює перехід на сторінку місця при кліку на підказку. Також контролювалася правильність фільтрації основного списку місць на сторінці відповідно до введеного тексту, чи дійсно відображаються лише ті місця, що відповідають запиту, чи відбувається це оновлення плавно, без візуальних збоїв.

Було підтверджено, що пошук працює незалежно від регістру літер. Також тестувалася поведінка системи при натисканні клавіші Enter у полі пошуку.

Випадаючий список мав зникати, залишаючи основні результати для перегляду. Перевірялося, як система обробляє запити, для яких не знайдено жодного результату, тобто чи виводиться відповідне повідомлення.

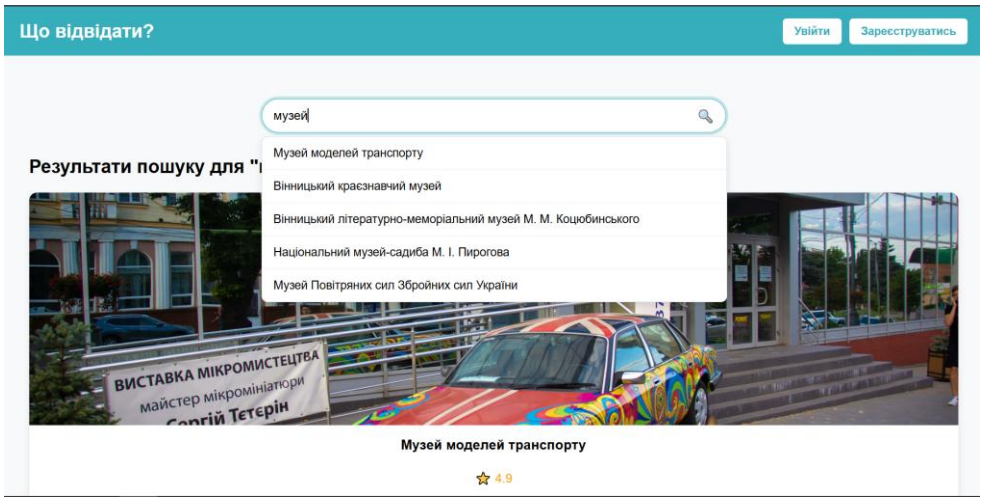

Рисунок 4.1 – Робота пошуку

Модуль автентифікації перевірявся на успішну реєстрацію через модальне вікно, коректну обробку помилок при спробі використання існуючих даних (див. рисунок 4.2), успішний вхід (див. рисунок 4.3), обробку неправильних даних для входу (див. рисунок 4.4), роботу функції виходу та збереження стану сесії після оновлення сторінки.

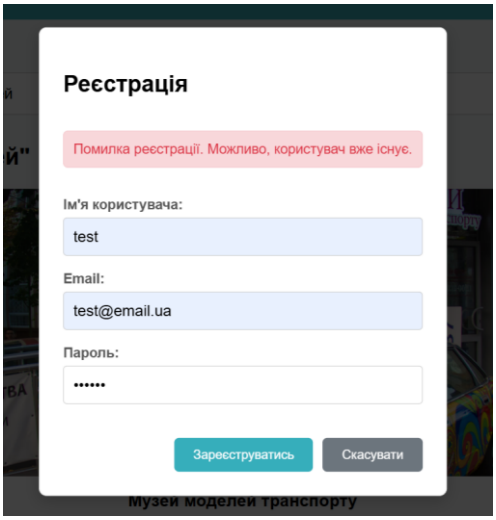

Рисунок 4.2 – Реакція системи на введення вже існуючих даних

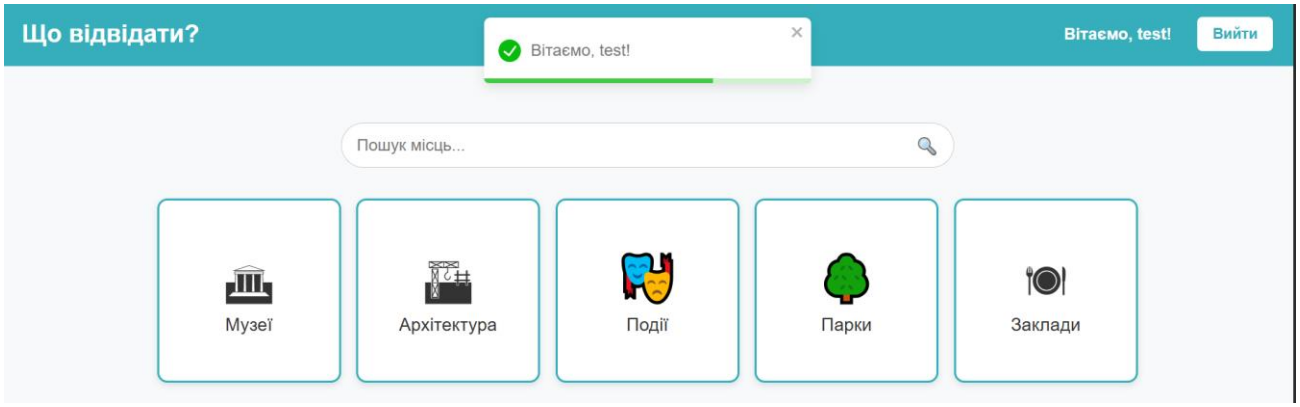

Рисунок 4.3 – Успішний вхід

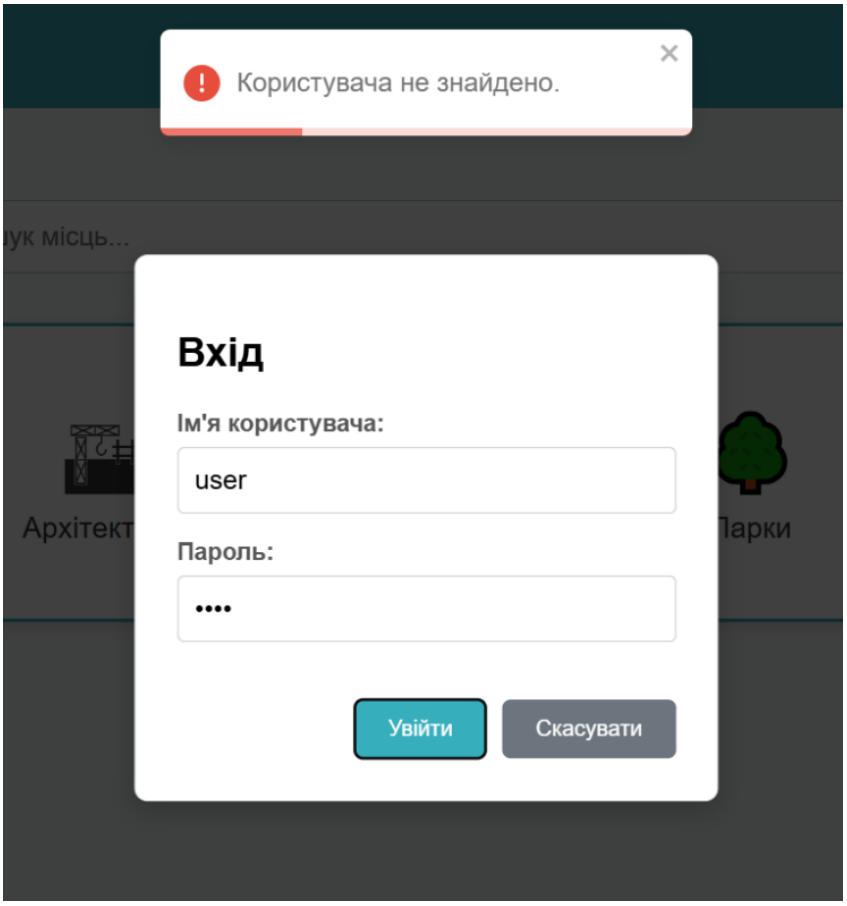

Рисунок 4.4 – Реакція системи на неіснуючі вхідні дані

Перевірялася взаємодія користувача з місцями, а саме додавання та видалення зі списку збережених, що відображалося на зміні стану кнопки (див. рисунок 4.5) та оновленні сторінки збережених місць (див. рисунок 4.6).

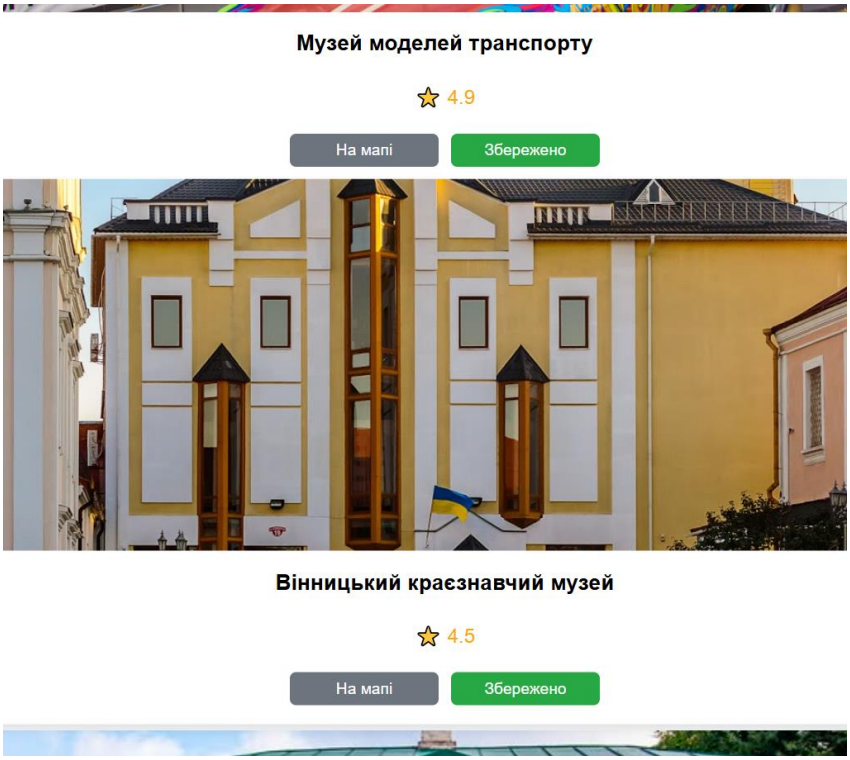

Рисунок 4.5 – Зміна стану кнопки

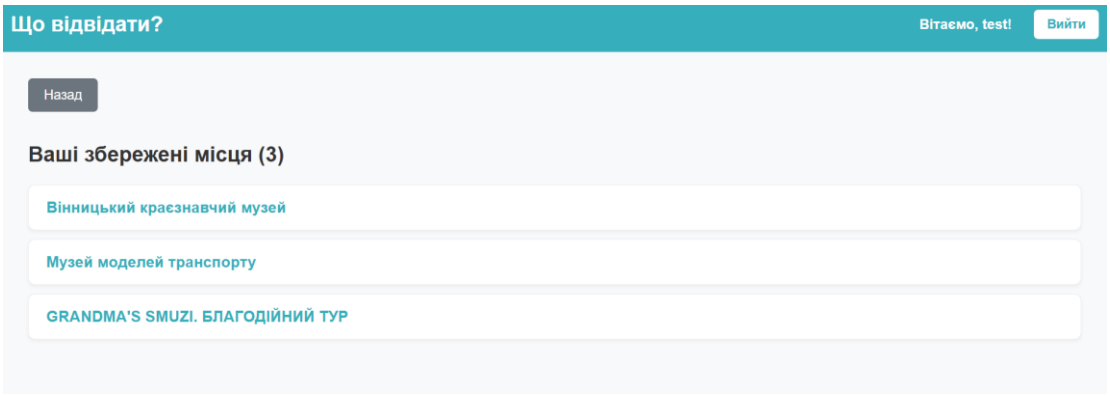

Рисунок 4.6 – Оновлення сторінки збережених місць

Також тестувалося додавання відгуків та виставлення оцінок. Було підтверджено блокування цих дій для неавторизованих користувачів. Функціонування блоку рекомендацій перевірялося для авторизованих користувачів. Перевірялася коректність відкриття та закриття всіх модальних вікон.

Робота карти та прокладання маршрутів тестувалася на коректне відображення маркера, успішну побудову маршруту від введеної адреси та

геолокації (див. рисунок 4.7), а також на обробку помилок при неможливості побудови маршруту (див. рисунок 4.8).

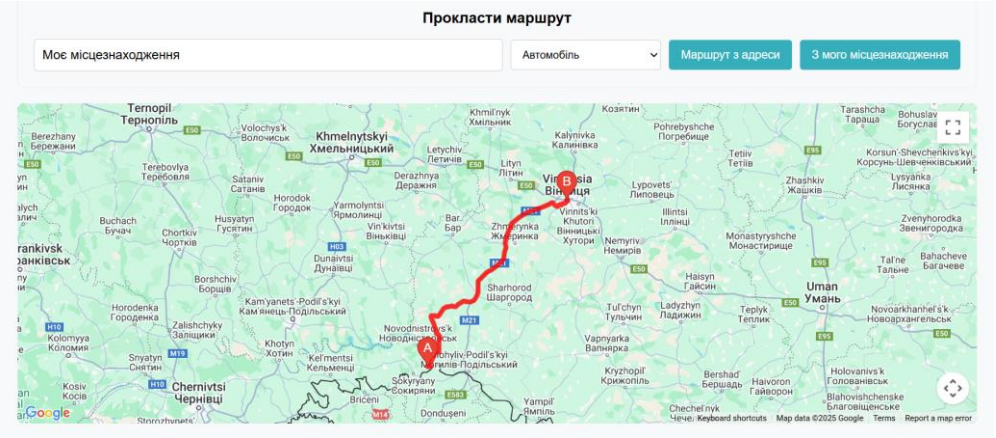

Рисунок 4.7 – Успішна побудова маршруту

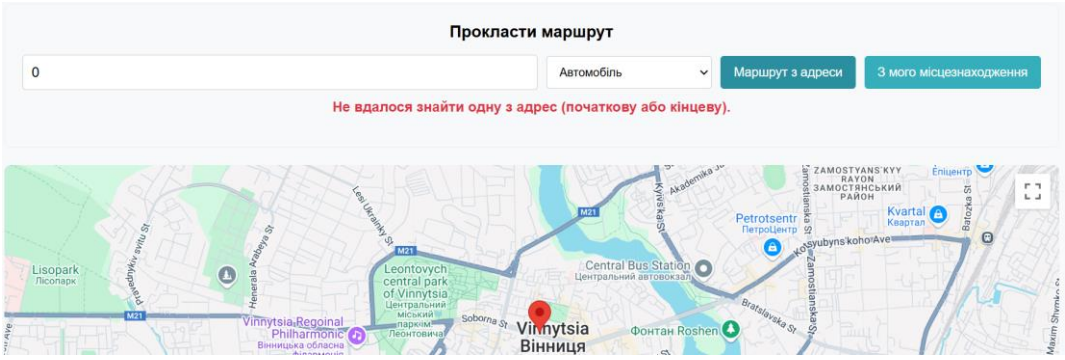

Рисунок 4.8 – Реакція на помилку

Також проводилося тестування користувацького інтерфейсу для перевірки візуального відображення елементів та їх читабельності. Адаптивність інтерфейсу тестувалася на різних розмірах екрану за допомогою інструментів розробника.

Було проведено тестування обробки помилок, а саме реакції системи на спроби виконати дії, що вимагають авторизації, без входу в систему, та на помилки, що повертаються Google Maps API.

Під час тестування було виявлено та виправлено незначні недоліки у стилях та покращено обробку деяких помилкових сценаріїв. У результаті

тестування встановлено, що функціонал вебзастосунку працює відповідно до встановлених задач.

4.2 Розробка інструкції користувача

Для коректної роботи вебзастосунку рекомендовано використовувати останню версію веббраузера Google Chrome, Mozilla Firefox, Microsoft Edge або Safari на пристрої зі стабільним підключенням до Інтернету. JavaScript у браузері має бути увімкнено. Для використання функції «З мого місцезнаходження» при побудові маршруту необхідно надати відповідний дозвіл у браузері.

Щоб розпочати роботу, потрібно відкрити вебзастосунок у браузері. Користувач побачить головну сторінку з основними елементами керування. Для пошуку місць можна натиснути на одну з кнопок категорій у верхній частині сторінки, щоб побачити список місць цієї категорії (див. рисунок 4.9), або ввести назву чи ключове слово у поле пошуку. Під час введення може з’явитися список підказок. Для перегляду результатів потрібно натиснути Enter або вибрати підказку (див. рисунок 4.10).

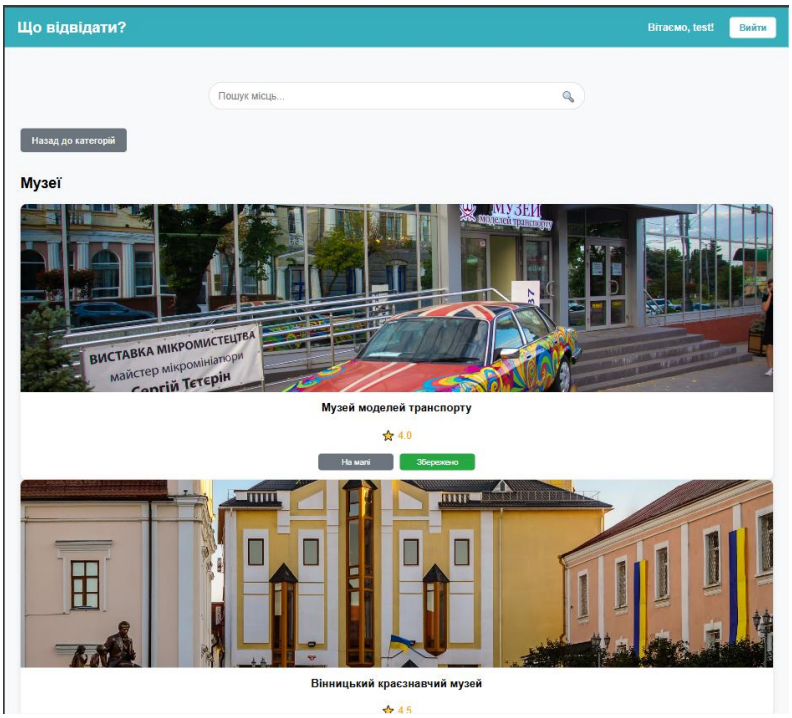

Рисунок 4.9 – Відображення списку місць після вибору певної категорії

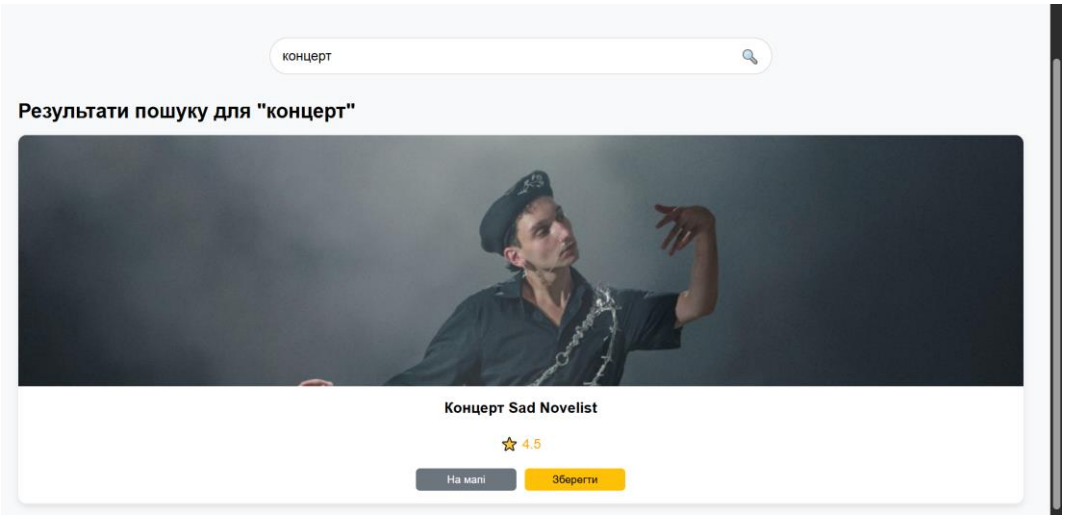

Рисунок 4.10 – Відображення результатів пошуку

Щоб перейти на сторінку з детальною інформацією, у списку місць необхідно натиснути на картку потрібного об’єкта. Там користувач зможе переглянути фотографії, прочитати опис, побачити розташування на карті, переглянути відгуки та рейтинг.

Для доступу до всіх функцій необхідно зареєструватися або увійти. Натиснувши відповідну кнопку «Зареєструватись» або «Увійти», потрібно заповнити поля у модальному вікні, що з’явиться. Після успішного входу на головній сторінці з’явиться блок персональних рекомендацій «Для Вас» (див. рисунок 4.11).

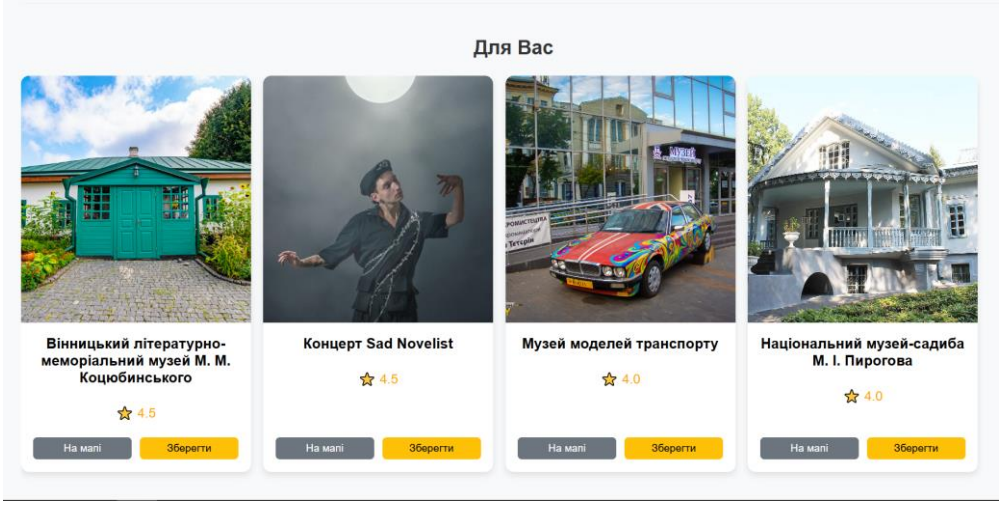

Рисунок 4.11 – Відображення блоку персональних рекомендацій

Для збереження локації потрібно натискати кнопку «Зберегти» на картках або сторінках місць. Переглянути збережені місця можна, натиснувши на своє ім’я користувача у хедері.

На сторінці місця можна поставити власну оцінку, натиснувши на бажаний номер зірочки (див. рисунок 4.12). Також можна залишити відгук, ввівши текст у відповідне поле та натиснувши «Надіслати» (див. рисунок 4.13), попередньо ознайомившись з правилами коментування через кнопку «Правила коментування» (див. рисунок 4.14).

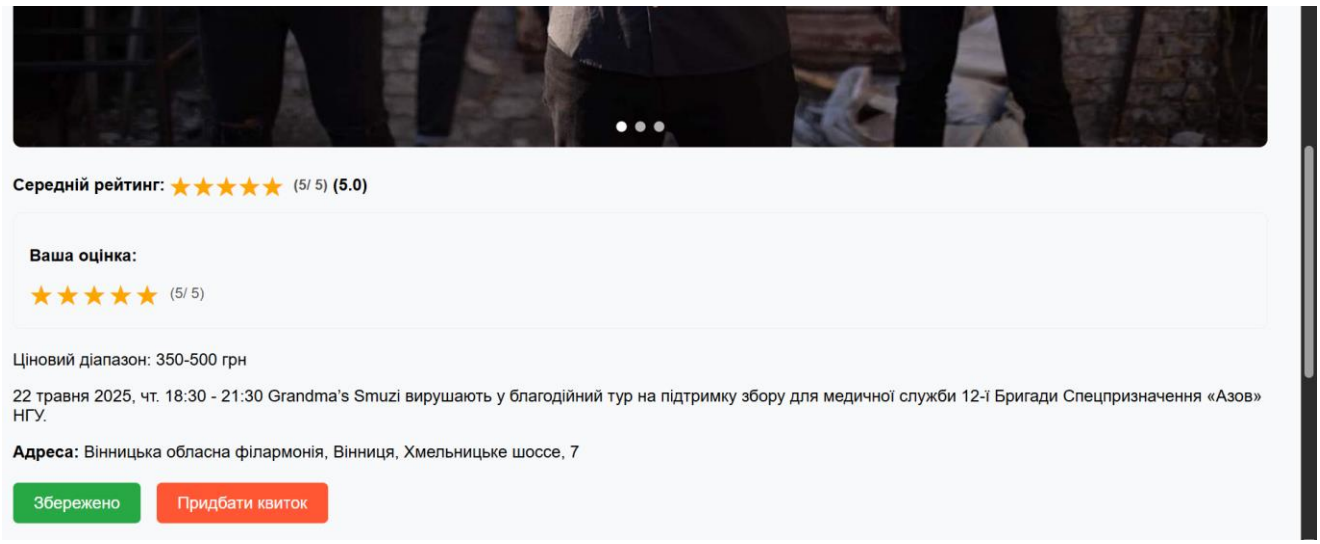

Рисунок 4.12 – Виставлення оцінки користувачем

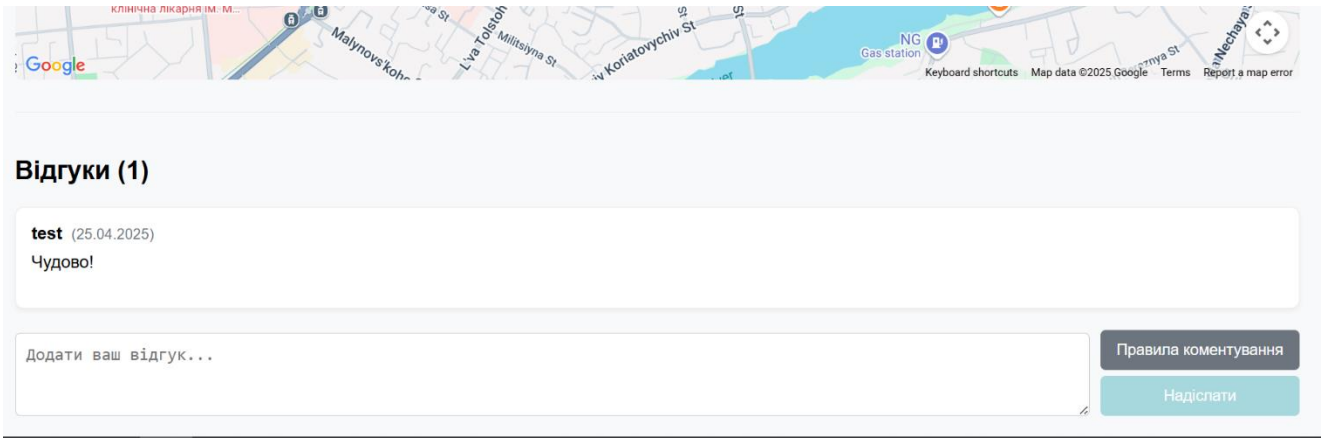

Рисунок 4.13– Форма додавання відгуків

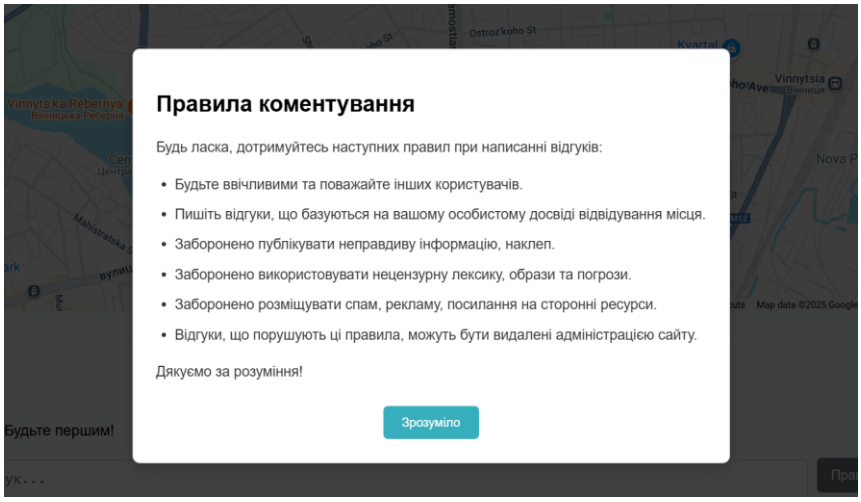

Рисунок 4.14 – Вікно для ознайомлення з правилами коментування

Для побудови маршруту на сторінці місця необхідно перейти у секцію «Прокласти маршрут». Після чого користувачу потрібно ввести адресу або натиснути «З мого місцезнаходження», обрати спосіб пересування та натиснути кнопку для розрахунку. Тоді маршрут з’явиться на карті у вигляді червоної лінії та маркерів початкової точки та кінцевої. На рисунку 4.15 показано побудований маршрут з введеної адреси та з обраним способом пересування громадський транспорт. Система будує маршрут до найближчої зупинки, показує тип громадського транспорту та його номер і прокладає маршрут цим транспортом до кінцевої точки.

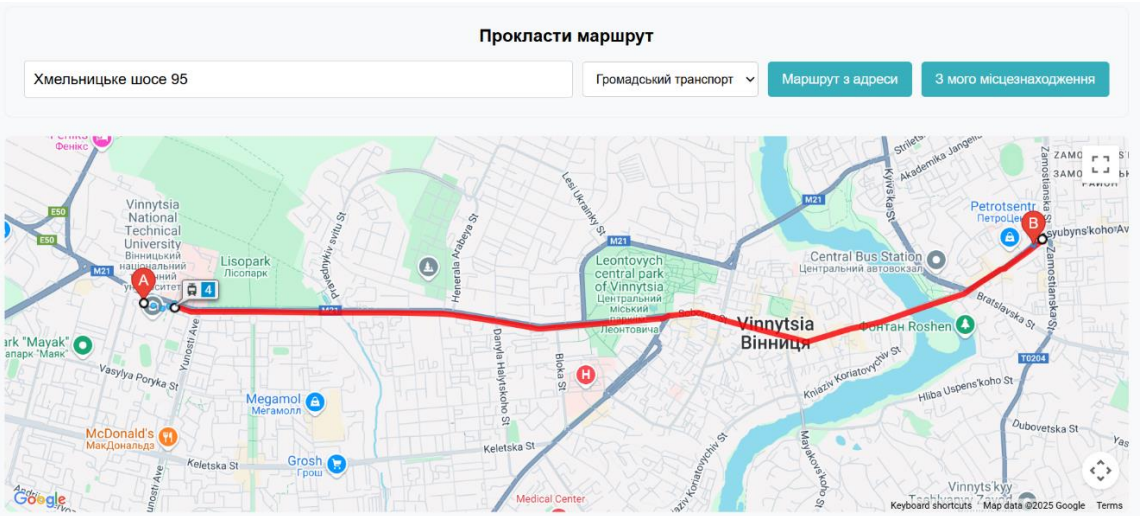

Рисунок 4.15 – Побудований маршрут з введеної адреси

Щоб завершити сесію у вебзастосунку, користувачу потрібно натиснути кнопку «Вийти» у хедері.

4.3 Висновки

У четвертому розділі було проведено тестування програмних модулів вебзастосунку для пошуку та рекомендації туристичних місць. Було проведено функціональне тестування, яке підтвердило коректну роботу всіх запланованих можливостей, а саме користувачі можуть успішно переходити між сторінками, здійснювати пошук з підказками та отримувати відфільтровані результати. Було перевірено процеси реєстрації нових користувачів та входу існуючих через модальні вікна, а також перевірено, що система правильно реагує на спроби ввести некоректні дані чи зареєструвати вже існуючого користувача.

Тестування підтвердило, що зареєстровані користувачі можуть взаємодіяти з місцями, а саме успішно додавати їх до списку збережених та видаляти звідти, що коректно відображається як на самій кнопці, так і на окремій сторінці збережених місць. Також було перевірено можливість залишати відгуки та виставляти оцінки, і підтверджено, що ці функції недоступні для неавторизованих користувачів. Було перевірено, що для авторизованих користувачів дійсно з’являється блок персональних рекомендацій. Окремо тестувалася коректність взаємодії з картографічним модулем. Перевірено відображення карти, маркерів та успішну побудову маршрутів як від введеної адреси, так і від поточної геолокації користувача.

Було проведено тестування реакції системи на можливі помилкові ситуації, наприклад, коли не вдається знайти маршрут або коли користувач намагається виконати дію, що потребує входу в акаунт, не будучи авторизованим. Тестування показало, що система адекватно обробляє такі сценарії, надаючи користувачеві зрозумілі повідомлення чи сповіщення. Додатково було проведено базове тестування користувацького інтерфейсу на предмет візуальної коректності, читабельності та адаптивності на різних розмірах екранів за допомогою

інструментів розробника. Результати тестування підтвердили, що розроблений вебзастосунок функціонує стабільно та відповідно до поставлених задач, а виявлені під час перевірок незначні недоліки у стилях чи обробці помилок було оперативно виправлено.

Також було розроблено інструкцію користувача для взаємодії з вебзастосунком. Інструкція містить опис системних вимог та надає пояснення щодо використання всіх основних функцій вебзастосунку.

ВИСНОВКИ

У бакалаврській кваліфікаційній роботі було розроблено вебзастосунок для пошуку та рекомендації туристичних місць міста. Проведено аналіз стану технологій пошуку та рекомендації туристичних місць. Обґрунтовано актуальність власної розробки на основі проведеного порівняльного аналізу аналогів та поставлено задачі дослідження.

Під час аналізу технологій розробки було обґрунтовано вибір мови програмування JavaScript та бібліотеки React.

У роботі реалізовано фільтрацію туристичних місць та подій за категоріями. Розроблено модулі реєстрації та авторизації, а також модуль інтеграції з Google Maps API для візуалізації місць, подій на мапі та побудови маршрутів.

Розроблено алгоритми пошуку, збереження обраних місць/подій зареєстрованими користувачами, додавання відгуків та залишання оцінок, персоналізованих рекомендацій на основі збережених місць користувача.

Було розроблено схеми загального алгоритму роботи вебзастосунку та алгоритму генерації персоналізованих рекомендацій. Також було розроблено модель роботи вебзастосунку з використанням UML діаграм.

Було проведено комплексне тестування роботи всіх модулів. Тестування системи показало повну працездатність даного програмного продукту. Також було розроблено інструкцію користувача та реалізовано адаптивний інтерфейс для комфортного використання з різних пристроїв.

Завдяки розробленим модулям пошуку, персоналізованих рекомендацій, адаптивному інтерфейсу та функціоналу побудови маршрутів, досягнуто поставленої мети – підвищення зручності та ефективності планування дозвілля для користувача.

Ключові слова: персоналізовані рекомендації, маршрут, туристичні місця, карта, пошук місць, фільтрація.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Aurelien Geron. Hands-On Learningn with Scikit-Learn, Keras, and TensorFlow. USA: O'Reilly Media, 2022. – 864 p.

2. Рекомендаційні системи [Електронний ресурс] – Режим доступу: https://skalar.ua/ua/expertise/recommender-systems (дата звернення: 08.03.2025)

3. Орчакова Ю.В. Методи рекомендацій туристичних місць: контентний підхід, колаборативна фільтрація та гібридні системи. / Ю.В. Орчакова, О.В. Карась / Матеріали LIV Всеукраїнської науково-технічної конференції підрозділів Вінницького національного технічного університету (НТКП ВНТУ–2025) : збірник доповідей [Електронний ресурс] – Режим доступу: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2025/paper/view/24278/20114

4. Орчакова Ю.В. Оптимізація маршруту для туристів за допомогою геоінформаційних технологій. / Ю.В. Орчакова, О.В. Карась / Матеріали Міжнародної науково-практичної інтернет-конференції Молодь в науці: дослідження, проблеми, перспективи (МН-2025) : збірник доповідей [Електронний ресурс] – Режим доступу: https://conferences.vntu.edu.ua/index.php/mn/mn2025/paper/viewFile/24945/20584

5. What is a hybrid recommender system? [Електронний ресурс] – Режим доступу: https://zilliz.com/ai-faq/what-is-a-hybrid-recommender-system (дата звернення: 15.03.2024)

6. MoeMisto.ua [Електронний ресурс] – Режим доступу: https://moemisto.ua/ (дата звернення: 20.03.2024)

7. Booking.com [Електронний ресурс] – Режим доступу: https://surl.li/ewcscc (дата звернення: 20.03.2024)

8. Tripadvisor Official Site [Електронний ресурс] – Режим доступу: https://shorturl.at/dJcEO (дата звернення: 20.03.2024)

9. Jamie Steane. Aurelien Geron. The Principles and Processes of Interactive Design. London: Bloomsbury Publishing, 2023. – 248 p.

10. Google Maps Platform [Електронний ресурс] – Режим доступу: https://developers.google.com/maps (дата звернення: 03.04.2024)

11. Zac Gordon. React Explained: Your Step-by-Step Guide to React. Chicago: Independently published, 2020. – 366 p.

12. Martine Dowden, Michael Gearon.Tiny CSS Projects. New York: Manning 2023. – 392 p.

13. Односторінкові (spa) і багатосторінкові (pwa) веб-додатки [Електронний ресурс] – Режим доступу: https://embo.com.ua/uk/blog/odnostorinkovi-spa-i-bagatostorinkovi-pwa/ (дата звернення: 03.04.2024)

14. Document Object Model (DOM) - Web APIs - MDN Web Docs [Електронний ресурс] – Режим доступу: https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model (дата звернення: 06.04.2024)

15. Getting Started [Електронний ресурс] – Режим доступу: https://create-react-app.dev/docs/getting-started/ (дата звернення: 06.04.2024)

16. react-google-maps/api [Електронний ресурс] – Режим доступу: https://www.npmjs.com/package/@react-google-maps/api (дата звернення: 10.04.2024)

17. Використання змінних середовища (environment variables) [Електронний ресурс] – Режим доступу: https://pupenasan.github.io/NodeREDGuidUKR/base/envvar.html (дата звернення: 10.04.2024)

18. react-toastify [Електронний ресурс] – Режим доступу: https://www.npmjs.com/package/react-toastify (дата звернення: 13.04.2024)

19. react-modal [Електронний ресурс] – Режим доступу: https://www.npmjs.com/package/react-modal (дата звернення: 13.04.2024)

20. Laurence Lars Svekis, Maaike van Putten, Codestars By Rob Percival. JavaScript from Beginner to Professional: Learn JavaScript quickly by building fun, interactive, and dynamic web apps, games, and pages. Birmingham: Packt Publishing Ltd, 2021. – 546 p.

21. Introducing JSX [Електронний ресурс] – Режим доступу: https://legacy.reactjs.org/docs/introducing-jsx.html (дата звернення: 13.04.2024)

ДОДАТКИ

57

Додаток А – Технічне завдання

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

ЗАТВЕРДЖУЮ
д.т.н., проф. О. Н. Романюк
«24» 03 2025 р.

Технічне завдання
на бакалаврську кваліфікаційну роботу «Розробка вебзастосунку для пошуку та рекомендації туристичних місць міста» за спеціальністю
121 Інженерія програмного забезпечення

Керівник бакалаврської кваліфікаційної роботи:
к-р. філос., асист. каф. ПЗ О. В. Карась
«24» 03 2025 р.

Виконала:
студентка гр.3ПІ-21б Ю. В. Орчакова
«24» 03 2025 р.

Вінниця – 2025 року

1. Найменування та галузь застосування

Бакалаврська кваліфікаційна робота: «Розробка вебзастосунку для пошуку та рекомендації туристичних місць міста». Галузь застосування – туризм.

2. Підстава для розробки

Підставою для виконання бакалаврської кваліфікаційної роботи є індивідуальне завдання на БКР та наказ №97 від «20» березня 2025 р. ректора ВНТУ про закріплення тем БКР.

3. Мета та призначення розробки

Метою дослідження є покращення якості процесу пошуку актуальних туристичних подій та місць, підвищення комфорту планування дозвілля за рахунок розробки програмних модулів, які забезпечують пошук, персоналізовані рекомендації, побудову маршрутів, зручну взаємодію з користувачем.

4. Вихідні дані для проведення НДР

Перелік основних літературних джерел, на основі яких буде виконуватись БКР.

1. Рекомендаційні системи [Електронний ресурс] – Режим доступу: https://skalar.ua/ua/expertise/recommender-systems

2. Zac Gordon. React Explained: Your Step-by-Step Guide to React. Chicago: Independently published, 2020. – 366 p.

3. Laurence Lars Svekis, Maaike van Putten, Codestars By Rob Percival. JavaScript from Beginner to Professional: Learn JavaScript quickly by building fun, interactive, and dynamic web apps, games, and pages. Birmingham: Packt Publishing Ltd, 2021. – 546 p.

5. Технічні вимоги

Вихідні дані до роботи: остання версія одного із веббраузерів Google Chrome, Mozilla Firefox, Microsoft Edge або Safari на пристрої зі стабільним підключенням до Інтернету. JavaScript у браузері має бути увімкнено. Для використання функції «З мого місцезнаходження» при побудові маршруту необхідно надати відповідний дозвіл у браузері.

6. Конструктивні вимоги

Графічна та текстова документація повинна відповідати діючим стандартам України.

7. Перелік технічної документації, що пред'являється по закінченню робіт:

1. Пояснювальна записка до БКР.
2. Технічне завдання.
3. Лістинги програми.

8. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

9. Стадії та етапи розробки:

№ з/п	Назва етапів бакалаврської кваліфікаційної роботи	Строк виконання етапів роботи
1	Аналіз стану технологій пошуку та рекомендації туристичних місць та постановка задач дослідження	25.03.25 – 07.04.25
2	Розробка моделі та алгоритмів вебзастосунку	07.04.25 – 20.04.25
3	Розробка модулів вебзастосунку	20.04.25 – 03.05.25
4	Тестування програмного продукту	03.05.25 – 16.05.25
5	Оформлення матеріалів до захисту БКР	16.05.25– 30.05.25

10. Порядок контролю та прийняття

Виконання етапів бакалаврської кваліфікаційної роботи контролюється керівником згідно з графіком виконання роботи. Прийняття бакалаврської кваліфікаційної роботи здійснюється ДЕК, затвердженою зав. кафедрою згідно з графіком.

61

Додаток Б – Протокол перевірки на плагіат

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: Розробка вебзастосунку для пошуку та рекомендації туристичних місць міста

Тип роботи: бакалаврська кваліфікаційна робота
(бакалаврська кваліфікаційна робота / магістерська кваліфікаційна робота)

Підрозділ кафедра програмного забезпечення, ФІТКІ, ЗПІ-21б
(кафедра, факультет, навчальна група)

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism 5/5 %

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

■ Запозичення, виявлені у роботі, є законними і не містять ознак плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту

☐ У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.

☐ У роботі виявлено ознаки плагіату та/або текстових маніпуляцій як спроб укриття плагіату, фабрикації, фальсифікації, що суперечить вимогам законодавства та нормам академічної доброчесності. Робота до захисту не приймається.

Експертна комісія:

_____ _____
(прізвище, ініціали, посада) (підпис)

_____ _____
(прізвище, ініціали, посада) (підпис)

Особа, відповідальна за перевірку _____ Черноволик Г. О.

З висновком експертної комісії ознайомлений(-на)

Керівник _____ д-р. філос., асист. каф. ПЗ Карась О. В.
(підпис) (прізвище, ініціали, посада)

Здобувач _____ Орчакова Ю. В.
(підпис) (прізвище, ініціали)

Додаток В – Лістинг програми

```
//App.js

import React, { useState, useEffect, useMemo, useRef } from "react";
import { BrowserRouter as Router, Routes, Route, useNavigate } from "react-router-dom";
import { motion } from "framer-motion";
import "./styles.css";
import PlacePage from "./PlacePage";
import { AuthProvider, useAuth } from './AuthContext';
import Layout from './Layout';
import placesData from "./placesData";
import { getItem, setItem, STORAGE_KEYS } from './localStorageUtils';
import SavedPlacesPage from './SavedPlacesPage';
import { ToastContainer, toast } from 'react-toastify';
import 'react-toastify/dist/ReactToastify.css';
import Modal from 'react-modal';
Modal.setAppElement('#root');

const categories = [
  { name: "Музеї", icon: "🏛️" },
  { name: "Архітектура", icon: "🏗️" },
  { name: "Події", icon: "🎭" },
  { name: "Парки", icon: "🌳" },
  { name: "Заклади", icon: "🍽️" }
];

const places = placesData;

function Home() {
  const [selectedCategory, setSelectedCategory] = useState(null);
  const [searchQuery, setSearchQuery] = useState("");
  const [showDropdown, setShowDropdown] = useState(false);
  const [savedPlacesForUser, setSavedPlacesForUser] = useState([]);
  const [recommendations, setRecommendations] = useState([]);
  const navigate = useNavigate();
  const { isLoggedIn, currentUser } = useAuth();
  const searchContainerRef = useRef(null);

  useEffect(() => {
    if (isLoggedIn && currentUser) {
      const allSavedPlaces = getItem(STORAGE_KEYS.SAVED_PLACES, {});
      setSavedPlacesForUser(allSavedPlaces[currentUser.username] || []);
    } else {
      setSavedPlacesForUser([]);
    }
  }, [isLoggedIn, currentUser]);

  useEffect(() => {
```

```
    if (!isLoggedIn || !currentUser) {
      setRecommendations([]);
      return;
    }

    const savedPlaces = savedPlacesForUser;
    const allRatings = getItem(STORAGE_KEYS.RATINGS, {});
    const userRatingsData = allRatings[currentUser.username] || {};
    const highlyRatedPlaceNames = Object.entries(userRatingsData)
      .filter(([, rating]) => rating >= 4)
      .map(([name]) => name);
    const positiveInteractionPlaces = [...new Set([...savedPlaces, ...highlyRatedPlaceNames])];

    if (positiveInteractionPlaces.length === 0) {
      const popularPlaces = [...places]
        .sort((a, b) => (getItem(STORAGE_KEYS.AVERAGE_RATINGS, {})[b.name] || b.rating ||
0) - (getItem(STORAGE_KEYS.AVERAGE_RATINGS, {})[a.name] || a.rating || 0))
        .slice(0, 4);
      setRecommendations(popularPlaces);
      return;
    }
    const categoryCounts = positiveInteractionPlaces.reduce((counts, placeName) => {
      const place = places.find(p => p.name === placeName);
      if (place) {
        counts[place.category] = (counts[place.category] || 0) + 1;
      }
      return counts;
    }, {});
    const sortedCategories = Object.entries(categoryCounts)
      .sort(([, countA], [, countB]) => countB - countA)
      .map(([category]) => category);
    const preferredCategories = sortedCategories.slice(0, 3);
    const candidatePlaces = places.filter(place => preferredCategories.includes(place.category));
    const filteredCandidates = candidatePlaces.filter(place =>
!positiveInteractionPlaces.includes(place.name));
    const avgRatings = getItem(STORAGE_KEYS.AVERAGE_RATINGS, {});
    const sortedCandidates = filteredCandidates.sort((a, b) => {
      const ratingA = avgRatings[a.name] || a.rating || 0;
      const ratingB = avgRatings[b.name] || b.rating || 0;
      return ratingB - ratingA;
    });
    setRecommendations(sortedCandidates.slice(0, 5));

  }, [isLoggedIn, currentUser, savedPlacesForUser]);

  const filteredPlaces = useMemo(() => places.filter(
    place =>
      (!selectedCategory || place.category === selectedCategory) &&
      place.name.toLowerCase().includes(searchQuery.toLowerCase())
  ), [selectedCategory, searchQuery]);
```

```
const searchResults = useMemo(() => searchQuery.length > 0 ? places.filter(place =>
  place.name.toLowerCase().includes(searchQuery.toLowerCase())
) : [], [searchQuery]);

const handleSaveToggle = (e, placeName) => {
  e.stopPropagation();
  if (!isLoggedIn || !currentUser) {
    return;
  }

  const allSavedPlaces = getItem(STORAGE_KEYS.SAVED_PLACES, {});
  const userPlaces = allSavedPlaces[currentUser.username] || [];
  let updatedUserPlaces;
  let isNowSaved;

  if (userPlaces.includes(placeName)) {
    updatedUserPlaces = userPlaces.filter(name => name !== placeName);
    isNowSaved = false;
  } else {
    updatedUserPlaces = [...userPlaces, placeName];
    isNowSaved = true;
  }

  allSavedPlaces[currentUser.username] = updatedUserPlaces;
  setItem(STORAGE_KEYS.SAVED_PLACES, allSavedPlaces);
  setSavedPlacesForUser(updatedUserPlaces);
};

const handleMapClick = (e, placeName) => {
  e.stopPropagation();
  alert(`Показати "${placeName}" на мапі! (потрібна реальна логіка)`);
};

useEffect(() => {
  const handleClickOutside = (event) => {
    if (searchContainerRef.current && !searchContainerRef.current.contains(event.target)) {
      setShowDropdown(false);
    }
  };

  if (showDropdown) {
    document.addEventListener('mousedown', handleClickOutside);
  } else {
    document.removeEventListener('mousedown', handleClickOutside);
  }
  return () => {
    document.removeEventListener('mousedown', handleClickOutside);
  };
}, [showDropdown]);
const handleSearchKeyDown = (event) => {
  if (event.key === 'Enter') {
```

```
    setShowDropdown(false);
    event.target.blur();
  }
};

return (
  <div className="container">
    <div className="search-container" ref={searchContainerRef}>
      <input
        type="text"
        className="search-input"
        placeholder="Пошук місць..."
        value={searchQuery}
        onChange={(e) => {
          setSearchQuery(e.target.value);
          setShowDropdown(e.target.value.length > 0);
          setSelectedCategory(null);
        }}
        onFocus={() => {
          if (searchQuery.length > 0) {
            setShowDropdown(true);
          }
        }}
        onKeyDown={handleSearchKeyDown}
      />
      {showDropdown && searchResults.length > 0 && (
        <ul className="dropdown">
          {searchResults.map((place, index) => (
            <li key={index} onClick={() => {
              navigate(`/place/${encodeURIComponent(place.name)}`);
              setShowDropdown(false);
            }}>
              {place.name}
            </li>
          ))}
        </ul>
      )}
    </div>

    {!selectedCategory && searchQuery === "" ? (
      <>
      <div className="categories-grid">
        {categories.map((cat, index) => (
          <motion.button
            key={index}
            whileHover={{ scale: 0.9 }}
            className="category-btn"
            onClick={() => setSelectedCategory(cat.name)}
          >
            <span className="icon">{cat.icon}</span>
            {cat.name}
```

```
          </motion.button>
        ))}
        </div>

        {isLoggedIn && recommendations.length > 0 && (
          <div className="recommendations-section">
            <h2>Для Вас</h2>
            <div className="places-grid recommendations-grid">
              {recommendations.map((place, index) => {
                const isSaved = savedPlacesForUser.includes(place.name);
                const saveButtonTitle = !isLoggedIn ? "Увійдіть, щоб зберегти" : (isSaved ?
"Видалити зі збережених" : "Зберегти місце");
                const avgRatings = getItem(STORAGE_KEYS.AVERAGE_RATINGS, {});
                const displayRating = avgRatings[place.name] || place.rating;

                return (
                  <div key={`rec-${index}-${place.name}`} className="place-card" onClick={() =>
navigate(`/place/${encodeURIComponent(place.name)}`)}>
                    <img src={place.images[0]} alt={place.name} className="place-image" />
                    <div className="place-card-content">
                      <h3>{place.name}</h3>
                      {displayRating && <p className="rating">☆ {displayRating.toFixed(1)}</p>}
                      <div className="card-buttons-inline">
                        <button className="map-btn" onClick={(e) => handleMapClick(e,
place.name)}>На мапі</button>
                        <button
                          className={`save-btn ${isSaved ? 'saved' : ''}`}
                          onClick={(e) => handleSaveToggle(e, place.name)}
                          disabled={!isLoggedIn}
                          title={saveButtonTitle}
                        >
                          {isSaved ? 'Збережено' : 'Зберегти'}
                        </button>
                      </div>
                    </div>
                  </div>
                );
              })}
            </div>
          </div>
        )}
      </>
    ) : (
      <div className="places-container">
        {selectedCategory && <button className="back-btn" onClick={() =>
setSelectedCategory(null)}>Назад до категорій</button>}
        <h2>{selectedCategory || `Результати пошуку для "${searchQuery}"`}</h2>

        {filteredPlaces.length === 0 ? (
          <p>Нічого не знайдено</p>
```

```
        ) : (
          <div className="places-grid">
            {filteredPlaces.map((place, index) => {
              const isSaved = savedPlacesForUser.includes(place.name);
              const saveButtonTitle = !isLoggedIn ? "Увійдіть, щоб зберегти" : (isSaved ? "Видалити
зі збережених" : "Зберегти місце");
              const avgRatings = getItem(STORAGE_KEYS.AVERAGE_RATINGS, {});
              const displayRating = avgRatings[place.name] || place.rating;

              return (
                <div key={`filtered-${index}-${place.name}`} className="place-card" onClick={() =>
navigate(`/place/${encodeURIComponent(place.name)}`)}>
                  <img src={place.images[0]} alt={place.name} className="place-image" />
                  <div className="place-card-content">
                  <h3>{place.name}</h3>
                    {displayRating && <p className="rating">☆ {displayRating.toFixed(1)}</p>}
                    <div className="card-buttons-inline">
                    <button className="map-btn" onClick={(e) => handleMapClick(e, place.name)}>На
мапі</button>
                    <button
                      className={`save-btn ${isSaved ? 'saved' : ''}`}
                      onClick={(e) => handleSaveToggle(e, place.name)}
                      disabled={!isLoggedIn}
                      title={saveButtonTitle}
                    >
                      {isSaved ? 'Збережено' : 'Зберегти'}
                    </button>
                  </div>
                  </div>
                </div>
              );
            })}
          </div>
        )}
      </div>
    )}
  </div>
 );
}

export default function App() {
 useEffect(() => {
  const existingRatings = getItem(STORAGE_KEYS.AVERAGE_RATINGS);
  if (!existingRatings) {
    const initialAvgRatings = places.reduce((acc, place) => {
      if (place.rating) {
        acc[place.name] = place.rating;
      }
      return acc;
    }, {});
```

```
      setItem(STORAGE_KEYS.AVERAGE_RATINGS, initialAvgRatings);
    }
  }, []);
  return (
    <AuthProvider>
      <Router>
        <Layout>
          <Routes>
            <Route path="/" element={<Home />} />
            <Route path="/place/:name" element={<PlacePage places={places} />} />
            <Route path="/saved" element={<SavedPlacesPage />} />
          </Routes>
        </Layout>
        <ToastContainer
          position="top-center"
          autoClose={4000}
          hideProgressBar={false}
          newestOnTop={false}
          closeOnClick
          rtl={false}
          pauseOnFocusLoss
          draggable
          pauseOnHover
          theme="light"
        />
      </Router>
    </AuthProvider>
  );
}

//Header.js

import React, { useState } from 'react';
import { Link } from 'react-router-dom';
import { useAuth } from './AuthContext';
import RegistrationModal from './RegistrationModal';
import LoginModal from './LoginModal';
import './styles.css';

function Header() {
    const { isLoggedIn, currentUser, login, logout, register } = useAuth();
    const [isRegisterModalOpen, setIsRegisterModalOpen] = useState(false);
    const [isLoginModalOpen, setIsLoginModalOpen] = useState(false);

    const handleRegisterSubmit = async ({ username, email, password }) => {
        const success = await register(username, password, email);
        return success;
    };
    const handleLoginSubmit = async ({ username, password }) => {
        const success = await login(username, password);
        return success;
```

```
  };

  return (
    <>
      <header className="header">
        <h1><Link to="/" className="header-title-link">Що відвідати?</Link></h1>
        <div className="auth-buttons">
          {isLoggedIn && currentUser ? (
            <>
              <Link to="/saved" className="username-link">
                <span className="username-display">Вітаємо,
{currentUser.username}!</span>
              </Link>
              <button className="register-btn" onClick={logout}>Вийти</button>
            </>
          ) : (
            <>
              <button className="login-btn" onClick={() => setIsLoginModalOpen(true)}>
                Увійти
              </button>
              <button className="register-btn" onClick={() =>
setIsRegisterModalOpen(true)}>
                Зареєструватись
              </button>
            </>
          )}
        </div>
      </header>

      <RegistrationModal
        isOpen={isRegisterModalOpen}
        onRequestClose={() => setIsRegisterModalOpen(false)}
        onRegisterSubmit={handleRegisterSubmit}
      />
      <LoginModal
        isOpen={isLoginModalOpen}
        onRequestClose={() => setIsLoginModalOpen(false)}
        onLoginSubmit={handleLoginSubmit}
      />
    </>
  );
}

export default Header;

//Layout.js

import React from 'react';
import Header from './Header';

function Layout({ children }) {
```

```
    return (
        <div>
            <Header />
            <main>{children}</main>
        </div>
    );
}

export default Layout;
//PlacePage.js

import React, { useState, useEffect, useCallback } from "react";
import { useParams, useNavigate } from "react-router-dom";
import { GoogleMap, LoadScript, Marker, DirectionsService, DirectionsRenderer } from "@react-
google-maps/api";
import StarRating from './StarRating';
import { useAuth } from './AuthContext';
import { getItem, setItem, STORAGE_KEYS } from './localStorageUtils';
import { toast } from 'react-toastify';
import InfoModal from './InfoModal';
import "./styles.css";

const mapContainerStyle = {
    width: "100%",
    height: "400px",
    borderRadius: "8px",
    marginTop: '20px'
};

const googleMapsApiKey=process.env.REACT_APP_Maps_API_KEY;

const calculateAverageRating = (placeName) => {
    const allRatings = getItem(STORAGE_KEYS.RATINGS, {});
    const placeRatings = allRatings[placeName];

    if (!placeRatings || Object.keys(placeRatings).length === 0) {
        return 0;
    }

    const ratingsArray = Object.values(placeRatings);
    const sum = ratingsArray.reduce((acc, rating) => acc + rating, 0);
    const average = sum / ratingsArray.length;
    return parseFloat(average.toFixed(1));
};

const libraries = ['places', 'directions'];

export default function PlacePage({ places }) {
    const { name: encodedName } = useParams();
    const placeName = decodeURIComponent(encodedName);
    const navigate = useNavigate();
```

```
  const { isLoggedIn, currentUser } = useAuth();

  const [place, setPlace] = useState(null);
  const [currentImageIndex, setCurrentImageIndex] = useState(0);
  const [reviews, setReviews] = useState([]);
  const [newReviewText, setNewReviewText] = useState("");
  const [userRating, setUserRating] = useState(0);
  const [averageRating, setAverageRating] = useState(0);
  const [isPlaceSaved, setIsPlaceSaved] = useState(false);

  const [originInput, setOriginInput] = useState('');
  const [directionsOptions, setDirectionsOptions] = useState(null);
  const [directionsResponse, setDirectionsResponse] = useState(null);
  const [isCalculatingRoute, setIsCalculatingRoute] = useState(false);
  const [routeError, setRouteError] = useState(null);
  const [travelMode, setTravelMode] = useState('DRIVING');

  const [isRulesModalOpen, setIsRulesModalOpen] = useState(false);

  const scrollToTop = (e) => {
    e.preventDefault();
    window.scrollTo({ top: 0, behavior: 'smooth' });
  };

  useEffect(() => {
    const foundPlace = places.find(p => p.name === placeName);
    if (foundPlace) {
      setPlace(foundPlace);

      const allReviews = getItem(STORAGE_KEYS.REVIEWS, {});
      setReviews(allReviews[placeName] || []);

      if (currentUser) {
        const allRatings = getItem(STORAGE_KEYS.RATINGS, {});
        const placeRatings = allRatings[placeName] || {};
        setUserRating(placeRatings[currentUser.username] || 0);
        const allSavedPlaces = getItem(STORAGE_KEYS.SAVED_PLACES, {});
        const userSaved = allSavedPlaces[currentUser.username] || [];
        setIsPlaceSaved(userSaved.includes(placeName));
      } else {
        setUserRating(0);
        setIsPlaceSaved(false);
      }

      const avgRatingsStorage = getItem(STORAGE_KEYS.AVERAGE_RATINGS, {});
      setAverageRating(avgRatingsStorage[placeName] || calculateAverageRating(placeName) ||
foundPlace.rating || 0);

      setCurrentImageIndex(0);
      setNewReviewText("");
```

```
            setDirectionsResponse(null);
            setDirectionsOptions(null);
            setOriginInput('');
            setRouteError(null);

        } else {
            navigate('/');
        }
    }, [placeName, places, navigate, currentUser]);

    const updateAverageRating = useCallback((placeNameForUpdate) => {
        const newAverage = calculateAverageRating(placeNameForUpdate);
        setAverageRating(newAverage);

        const allAverageRatings = getItem(STORAGE_KEYS.AVERAGE_RATINGS, {});
        allAverageRatings[placeNameForUpdate] = newAverage;
        setItem(STORAGE_KEYS.AVERAGE_RATINGS, allAverageRatings);
    }, []);
    const handleSaveTogglePage = () => {
        if (!isLoggedIn || !currentUser) {
            return;
        }
        const allSavedPlaces = getItem(STORAGE_KEYS.SAVED_PLACES, {});
        const userPlaces = allSavedPlaces[currentUser.username] || [];
        let updatedUserPlaces;
        let isNowSaved;

        if (userPlaces.includes(placeName)) {
            updatedUserPlaces = userPlaces.filter(name => name !== placeName);
            isNowSaved = false;
        } else {
            updatedUserPlaces = [...userPlaces, placeName];
            isNowSaved = true;
        }
        allSavedPlaces[currentUser.username] = updatedUserPlaces;
        setItem(STORAGE_KEYS.SAVED_PLACES, allSavedPlaces);
        setIsPlaceSaved(isNowSaved);
    };

    const handleNextImage = () => setCurrentImageIndex((prevIndex) => (prevIndex + 1) %
place.images.length);
    const handlePrevImage = () => setCurrentImageIndex((prevIndex) => (prevIndex - 1 +
place.images.length) % place.images.length);

    const handleAddReview = () => {
        if (!isLoggedIn || !currentUser) {
            toast.warn("Будь ласка, увійдіть або зареєструйтесь, щоб залишити відгук.");
            return;
        }
        if (newReviewText.trim() === "") {
            toast.warn("Відгук не може бути порожнім.");
```

```
        return;
    }

    const newReview = {
        author: currentUser.username,
        text: newReviewText,
        date: new Date().toLocaleDateString()
    };

    const allReviews = getItem(STORAGE_KEYS.REVIEWS, {});
    const placeReviews = allReviews[placeName] || [];
    placeReviews.push(newReview);
    allReviews[placeName] = placeReviews;
    setItem(STORAGE_KEYS.REVIEWS, allReviews);

    setReviews(placeReviews);
    setNewReviewText("");
};

const handleRatePlace = (rating) => {
    if (!isLoggedIn || !currentUser) {
        toast.warn("Будь ласка, увійдіть або зареєструйтесь, щоб поставити оцінку.");
        return;
    }

    const allRatings = getItem(STORAGE_KEYS.RATINGS, {});
    if (!allRatings[placeName]) {
        allRatings[placeName] = {};
    }
    allRatings[placeName][currentUser.username] = rating;
    setItem(STORAGE_KEYS.RATINGS, allRatings);
    setUserRating(rating);
    updateAverageRating(placeName);
};

const calculateRoute = (origin) => {
    if (!window.google || !window.google.maps || !place?.location) {
        setRouteError("Карта ще завантажується або сталася помилка. Спробуйте пізніше.");
        setIsCalculatingRoute(false);
        return;
    }
    if (!window.google.maps.TravelMode[travelMode]) {
        console.error("Неправильний TravelMode:", travelMode);
        setRouteError("Вибрано недійсний спосіб пересування.");
        setIsCalculatingRoute(false);
        return;
    }

    console.log(`Calculating route from:`, origin, `to:`, place.location, `via:`, travelMode);

    setDirectionsResponse(null);
```

```
        setRouteError(null);
        setIsCalculatingRoute(true);

        setDirectionsOptions({
            origin: origin,
            destination: place.location,
            travelMode: window.google.maps.TravelMode[travelMode],
        });
    };

    const handleGetDirectionsFromInput = () => {
        if (!originInput.trim()) {
            setRouteError('Будь ласка, введіть початкову адресу.');
            return;
        }
        calculateRoute(originInput);
    };

    const handleGetDirectionsFromLocation = () => {
        if (!navigator.geolocation) {
            setRouteError('Геолокація не підтримується вашим браузером.');
            return;
        }

        setIsCalculatingRoute(true);
        setDirectionsResponse(null);
        setRouteError(null);

        navigator.geolocation.getCurrentPosition(
            (position) => {
                const userLocation = {
                    lat: position.coords.latitude,
                    lng: position.coords.longitude,
                };
                console.log("User location found:", userLocation);
                setOriginInput("Моє місцезнаходження");
                calculateRoute(userLocation);
            },
            (err) => {
                console.error("Geolocation error:", err);
                let message = `Помилка геолокації: ${err.message}`;
                if (err.code === 1) message = "Ви не надали дозвіл на визначення
місцезнаходження.";
                setRouteError(message);
                setIsCalculatingRoute(false);
            },
            { enableHighAccuracy: true, timeout: 10000, maximumAge: 0 }
        );
    };

    const directionsCallback = useCallback((response, status) => {
```

```
        console.log("Directions Callback - Status:", status, "Response:", response);
        if (status === 'OK' && response) {
            setDirectionsResponse(response);
            setRouteError(null);
            console.log("Route successfully calculated.");
        } else {
            setDirectionsResponse(null);
            let errorMsg = `Не вдалося прокласти маршрут. Статус: ${status}`;
            if (status === 'ZERO_RESULTS') errorMsg = "Маршрут не знайдено. Спробуйте іншу адресу або спосіб пересування.";
            if (status === 'NOT_FOUND') errorMsg = "Не вдалося знайти одну з адрес (початкову або кінцеву).";
            setRouteError(errorMsg);
            console.warn("Directions request failed with status:", status);
        }
        setIsCalculatingRoute(false);
    }, []);

    if (!place) return <div className="container"><p>Завантаження...</p></div>;

    const saveButtonTitle = !isLoggedIn ? "Увійдіть, щоб зберегти" : (isPlaceSaved ? "Видалити зі збережених" : "Зберегти місце");
    const submitReviewTitle = !isLoggedIn ? "Увійдіть, щоб залишити відгук" : (newReviewText.trim() === "" ? "Введіть текст відгуку" : "Надіслати відгук");
    const ratingTitle = !isLoggedIn ? "Увійдіть, щоб оцінити" : "Оцініть місце";

    return (
        <>
        <div className="place-page container">
            <button className="back-btn" onClick={() => navigate(-1)}>Назад</button>

            <h1>{place.name}</h1>

            {place.images && place.images.length > 0 && (
                <div className="image-carousel">
                    {place.images.length > 1 && (<button onClick={handlePrevImage} className="carousel-btn prev-btn">‹</button>)}
                    <img src={place.images[currentImageIndex]} alt={`${place.name} - зображення ${currentImageIndex + 1}`} className="place-image-large" />
                    {place.images.length > 1 && (<button onClick={handleNextImage} className="carousel-btn next-btn">›</button>)}
                    {place.images.length > 1 && (
                        <div className="carousel-dots">
                            {place.images.map((_, index) => (<span key={index} className={`dot ${index === currentImageIndex ? 'active' : ''}`} onClick={() => setCurrentImageIndex(index)}></span>))}
                        </div>
                    )}
                </div>
            )}

            <div className="average-rating">
```

```
            Середній рейтинг:
            <StarRating
              key={`avg-${averageRating}`}
              initialRating={averageRating}
              readonly={true}
              showRatingValue={true}
            />
            ({averageRating > 0 ? averageRating.toFixed(1) : 'Немає'})
            </div>

          <div className="user-rating-section" title={ratingTitle}>
            <p><strong>{isLoggedIn ? "Ваша оцінка:" : "Оцініть місце:"}</strong></p>
            <StarRating
              key={`user-${userRating}-${placeName}`}
              initialRating={userRating}
              onRate={handleRatePlace}
              readonly={!isLoggedIn}
            />
            {!isLoggedIn && (
              <p className="auth-prompt">
                (Будь ласка, <a href="#login" onClick={scrollToTop}>увійдіть</a>, щоб
оцінити)
              </p>)}
          </div>

          <p className="price-range">Ціновий діапазон: {place.priceRange || "Не вказано"}</p>
          <p>{place.description}</p>
          <p><strong>Адреса:</strong> {place.address}</p>

          <div className="place-page-actions">
            <button
              className={`save-btn ${isPlaceSaved ? 'saved' : ''}`}
              onClick={handleSaveTogglePage}
              disabled={!isLoggedIn}
              title={saveButtonTitle}
            >
              {isPlaceSaved ? 'Збережено' : 'Зберегти'}
            </button>
            {place.category === 'Події' && place.ticketLink && (
              <a href={place.ticketLink} target="_blank" rel="noopener noreferrer"
className="ticket-btn">
                Придбати квиток
              </a>
            )}
          </div>

          <div className="directions-section">
            <h3>Прокласти маршрут</h3>
            <div className="directions-controls">
              <input
                type="text"
```

```
                    className="origin-input"
                    placeholder="Введіть початкову адресу..."
                    value={originInput}
                    onChange={(e) => setOriginInput(e.target.value)}
                    disabled={isCalculatingRoute}
                  />
                  <select
                    className="travel-mode-select"
                    value={travelMode}
                    onChange={(e) => setTravelMode(e.target.value)}
                    disabled={isCalculatingRoute}
                  >
                    <option value="DRIVING">Автомобіль</option>
                    <option value="WALKING">Пішки</option>
                    <option value="BICYCLING">Велосипед</option>
                    <option value="TRANSIT">Громадський транспорт</option>
                  </select>
                  <button
                    onClick={handleGetDirectionsFromInput}
                    disabled={isCalculatingRoute || !originInput.trim()}
                    title={!originInput.trim() ? "Введіть адресу" : ""}
                  >
                    {isCalculatingRoute ? 'Шукаємо...' : 'Маршрут з адреси'}
                  </button>
                  <button onClick={handleGetDirectionsFromLocation}
disabled={isCalculatingRoute}>
                    {isCalculatingRoute ? 'Шукаємо...' : 'З мого місцезнаходження'}
                  </button>
                </div>
                {routeError && <p className="error-message">{routeError}</p>}
              </div>

              {googleMapsApiKey ? (
                <LoadScript googleMapsApiKey={googleMapsApiKey}
                  libraries={libraries}
                  onError={(err) => {
                    console.error("Помилка завантаження Google Maps скрипту:", err);
                    setRouteError("Не вдалося завантажити карту.")
                  }}>
                  <GoogleMap mapContainerStyle={mapContainerStyle}
                    center={place.location} zoom={15}
                    options={{ streetViewControl: false, mapTypeControl: false }}>
                    {!directionsResponse && <Marker position={place.location} title={place.name}
/>}
                    {directionsOptions && (
                      <DirectionsService
                        options={directionsOptions}
                        callback={directionsCallback}
                        onLoad={() => console.log('DirectionsService loaded.')}
                        onUnmount={() => console.log('DirectionsService unmounted.')}
                      />
```

```
                )}
                {directionsResponse && (
                  <DirectionsRenderer
                    directions={directionsResponse}
                    options={{
                      preserveViewport: true,
                      suppressMarkers: false,
                      polylineOptions: {
                        strokeColor: '#FF0000',
                        strokeOpacity: 0.8,
                        strokeWeight: 6
                      }
                    }}
                  />
                )}
              </GoogleMap>
            </LoadScript>
          ) : (
            <div className="error-message" style={{ padding: '20px', border: '1px solid red' }}>
              Помилка конфігурації: Не вдалося завантажити ключ Google Maps API.
Маршрути та карта недоступні.
            </div>
          )}

          <div className="reviews-section">
            <h2>Відгуки ({reviews.length})</h2>
            <ul className="reviews">
              {reviews.length > 0 ? (
                reviews.map((review, index) => (
                  <li key={`${index}-${review.date}-${review.author}`} className="review-
item">
                    <strong className="review-author">{review.author}</strong>
                    <span className="review-date"> ({review.date})</span>
                    <p className="review-text">{review.text}</p>
                  </li>
                ))
              ) : (
                <p>Ще немає відгуків. Будьте першим!</p>
              )}
            </ul>

            <div className="review-form">
              <textarea
                placeholder={isLoggedIn ? "Додати ваш відгук..." : "Увійдіть, щоб додати
відгук"}
                className="review-input"
                value={newReviewText}
                onChange={(e) => setNewReviewText(e.target.value)}
                disabled={!isLoggedIn}
                rows="3"
                title={!isLoggedIn ? "Увійдіть, щоб додати відгук" : ""}
```

```
                ></textarea>
                <div className="review-actions">
                    <button className="comment-rules-btn"
                      onClick={() => setIsRulesModalOpen(true)}
                      >
                      Правила коментування
                    </button>
                  <button
                    className="submit-review-btn"
                    onClick={handleAddReview}
                    disabled={!isLoggedIn || newReviewText.trim() === ""}
                    title={submitReviewTitle}
                  >
                    Надіслати
                  </button>
                </div>
              </div>
              {!isLoggedIn && (
                <p className="auth-prompt">
                  Будь ласка, <a href="#login" onClick={scrollToTop}>увійдіть</a>
                   або <a href="#register" onClick={scrollToTop}>зареєструйтесь</a>,
                  щоб залишити відгук.
                </p>
              )}
            </div>
          </div>
            <InfoModal
              isOpen={isRulesModalOpen}
              onRequestClose={() => setIsRulesModalOpen(false)}
              title="Правила коментування"
            >
              <p>Будь ласка, дотримуйтесь наступних правил при написанні відгуків:</p>
              <ul>
                <li>Будьте ввічливими та поважайте інших користувачів.</li>
                <li>Пишіть відгуки, що базуються на вашому особистому досвіді відвідування
місця.</li>
                <li>Заборонено публікувати неправдиву інформацію, наклеп.</li>
                <li>Заборонено використовувати нецензурну лексику, образи та погрози.</li>
                <li>Заборонено розміщувати спам, рекламу, посилання на сторонні ресурси.</li>
                <li>Відгуки, що порушують ці правила, можуть бути видалені адміністрацією
сайту.</li>
              </ul>
              <p>Дякуємо за розуміння!</p>
            </InfoModal>
          </>
    );
}

//placesData.js
```

```
const places = [
  {
    name: "Музей моделей транспорту",
    category: "Музеї",
    images: ["/images/transport1.jpg", "/images/transport2.jpg", "/images/transport3.jpg"],
    priceRange: "100-200 грн",
    description: "Технічний музей у Вінниці, єдиний в Україні музей моделей транспортних засобів. Усього колекція музею налічує понад 5 000 моделей.",
    address: "вулиця Соборна, 64",
    location: { lat: 49.233333, lng: 28.466944 },
    rating: 4.9
  },
  {
    name: "Вінницький краєзнавчий музей",
    category: "Музеї",
    images: ["/images/krai1.jpeg", "/images/krai2.jpg", "/images/krai3.jpg"],
    priceRange: "100-200 грн",
    description: "Обласний краєзнавчий музей у місті Вінниці, найбільше зібрання матеріалів і документів з історії, етнографії і культури Східного Поділля. Розташований у центрі міста, в будинку за адресою: вулиця Соборна, № 19, який є частиною історичного комплексу «Вінницькі мури».",
    address: "вулиця Соборна, 19",
    location: { lat: 49.232666, lng: 28.476159 },
    rating: 4.5
  },
  {
    name: "Вінницький літературно-меморіальний музей М. М. Коцюбинського",
    category: "Музеї",
    images: ["/images/kocub1.jpg", "/images/kocub2.jpeg", "/images/kocub3.jpeg"],
    priceRange: "100-200 грн",
    description: "Музей-садиба письменника-класика М. М. Коцюбинського у Вінниці, де він народився.",
    address: "вулиця І. Бевза, 15",
    location: { lat: 49.237174, lng: 28.487909 },
    rating: 4.5
  },
  {
    name: "Національний музей-садиба М. І. Пирогова",
    category: "Музеї",
    images: ["/images/pyrogov1.jpg", "/images/pyrogov2.jpg", "/images/pyrogov3.jpg"],
    priceRange: "100-200 грн",
    description: "Музей у Вінниці, присвячений життю та діяльності Миколи Івановича Пирогова, видатного українського науковця, хірурга та педагога. Режим роботи: 10:00 — 18:30, понеділок — вихідний",
    address: "вулиця Пирогова, 155",
    location: { lat: 49.216031, lng: 28.408375 },
    rating: 4.5
  },
  {
    name: "Музей Повітряних сил Збройних сил України",
    category: "Музеї",
```

```
        images: ["/images/airforce1.jpg", "/images/airforce2.png", "/images/airforce3.jpg"],
        priceRange: "100-200 грн",
        description: "Музей авіаційної техніки і засобів ППО просто неба у Вінниці.
Розташований на території штабу командування Повітряних Сил Збройних Сил України.",
        address: "вулиця Стрілецька, 105",
        location: { lat: 49.251044, lng: 28.500725 },
        rating: 4.5
    },
    {
        name: "Концерт Sad Noelist",
        category: "Події",
        images: ["/images/sadNovelist1.jpg", "/images/coffee_fest_people.jpg",
"/images/coffee_cup.jpg"],
        priceRange: "350-500 грн",
        description: "26 квітня 2025, сб. 21:30 Цієї весни Sad Novelist вирушає у перший тур із
лайв-бендом!",
        address: "Docker Pub, Вінниця, пр-т Коцюбинського, 39",
        location: { lat: 49.23818443785771, lng: 28.492074696450278 },
        ticketLink: "https://concert.ua/uk/booking/sad-novelist-vinnytsia/",
        rating: 4.5
    },
    {
        name: "GRANDMA'S SMUZI. БЛАГОДІЙНИЙ ТУР",
        category: "Події",
        images: ["/images/grandma's1.jpeg", "/images/coffee_fest_people.jpg",
"/images/coffee_cup.jpg"],
        priceRange: "350-500 грн",
        description: "22 травня 2025, чт. 18:30 - 21:30 Grandma’s Smuzi вирушають у благодійний
тур на підтримку збору для медичної служби 12-ї Бригади Спецпризначення «Азов» НГУ.",
        address: "Вінницька обласна філармонія, Вінниця, Хмельницьке шоссе, 7",
        location: { lat: 49.232529929161586, lng: 28.449081809942076 },
        ticketLink: "https://concert.ua/uk/booking/grandmas-smuzi-vinnytsia25/",
        rating: 4.5
    },
];

export default places;

//AuthContext.js

import React, { createContext, useState, useContext, useEffect, useCallback } from 'react';
import { getItem, setItem, removeItem, STORAGE_KEYS } from './localStorageUtils';
import { toast } from 'react-toastify';

const AuthContext = createContext(null);

export const AuthProvider = ({ children }) => {
    const [currentUser, setCurrentUser] = useState(null);
    const [isLoggedIn, setIsLoggedIn] = useState(false);

    useEffect(() => {
```

```
    const storedUser = getItem(STORAGE_KEYS.CURRENT_USER);
    if (storedUser && storedUser.username) {
        setCurrentUser(storedUser);
        setIsLoggedIn(true);
    } else {
        setIsLoggedIn(false);
        setCurrentUser(null);
    }
}, []);

const register = useCallback((username, password, email) => {
    if (!username || !password || !email) {
        toast.warn("Ім'я користувача, email та пароль не можуть бути порожніми.");
        return false;
    }
    if (!/\S+@\S+\.\S+/.test(email)) {
        toast.warn('Будь ласка, введіть дійсну адресу email.');
        return false;
    }
    const users = getItem(STORAGE_KEYS.USERS, []);
    const existingUser = users.find(user => user.username === username);
    const existingEmail = users.find(user => user.email === email);

    if (existingUser) {
        toast.error('Користувач з таким іменем вже існує!');
        return false;
    }
    if (existingEmail) {
        toast.error('Користувач з таким email вже існує!');
        return false;
    }

    const newUser = { username, password, email };
    users.push(newUser);
    setItem(STORAGE_KEYS.USERS, users);
    toast.success('Реєстрація успішна! Тепер ви можете увійти.');
    return true;
}, []);

const login = useCallback((username, password) => {
    const users = getItem(STORAGE_KEYS.USERS, []);
    const foundUser = users.find(user => user.username === username);

    if (!foundUser) {
        toast.error('Користувача не знайдено.');
        return false;
    }

    if (foundUser.password !== password) {
        toast.error('Неправильний пароль.');
        return false;
```

```
        }

        const userData = { username: foundUser.username };
        setItem(STORAGE_KEYS.CURRENT_USER, userData);
        setCurrentUser(userData);
        setIsLoggedIn(true);
        toast.success(`Вітаємо, ${username}!`);
        return true;
    }, []);

    const logout = useCallback(() => {
        removeItem(STORAGE_KEYS.CURRENT_USER);
        setCurrentUser(null);
        setIsLoggedIn(false);
        toast.info('Ви вийшли з системи.');
    }, []);

    return (
        <AuthContext.Provider value={{ isLoggedIn, currentUser, login, logout, register }}>
            {children}
        </AuthContext.Provider>
    );
};

export const useAuth = () => {
    return useContext(AuthContext);
};

//localStorageUtils.js

/**
 *
 * @param {string} key
 * @param {*} defaultValue
 * @returns {*}
 */
export const getItem = (key, defaultValue = null) => {
    try {
        const item = localStorage.getItem(key);
        return item ? JSON.parse(item) : defaultValue;
    } catch (error) {
        console.error(`Помилка читання localStorage ключа “${key}”:`, error);
        return defaultValue;
    }
};

/**
 *
 * @param {string} key
 * @param {*} value
 */
```

```
export const setItem = (key, value) => {
   try {
      localStorage.setItem(key, JSON.stringify(value));
   } catch (error) {
      console.error(`Помилка запису localStorage ключа “${key}”:`, error);
   }
};

/**
 *
 * @param {string} key
 */
export const removeItem = (key) => {
   try {
      localStorage.removeItem(key);
   } catch (error) {
      console.error(`Помилка видалення localStorage ключа “${key}”:`, error);
   }
};

export const STORAGE_KEYS = {
   USERS: 'app_users',
   CURRENT_USER: 'app_currentUser',
   SAVED_PLACES: 'app_savedPlaces',
   REVIEWS: 'app_reviews',
   RATINGS: 'app_ratings',
   AVERAGE_RATINGS: 'app_averageRatings'
};

//RegistrationModal.js

import React, { useState } from 'react';
import Modal from 'react-modal';
import './styles.css';

const customStyles = {
   content: {
      top: '50%',
      left: '50%',
      right: 'auto',
      bottom: 'auto',
      marginRight: '-50%',
      transform: 'translate(-50%, -50%)',
      minWidth: '300px',
      maxWidth: '500px',
      padding: '25px',
      borderRadius: '8px',
      boxShadow: '0 5px 15px rgba(0,0,0,0.2)'
   },
   overlay: {
      backgroundColor: 'rgba(0, 0, 0, 0.75)',
```

```
    zIndex: 1000
  },
};

function RegistrationModal({ isOpen, onRequestClose, onRegisterSubmit }) {
  const [username, setUsername] = useState('');
  const [email, setEmail] = useState('');
  const [password, setPassword] = useState('');
  const [error, setError] = useState('');

  const handleSubmit = async (event) => {
    event.preventDefault();
    setError('');

    if (!username || !email || !password) {
      setError('Будь ласка, заповніть усі поля.');
      return;
    }
    if (!/\S+@\S+\.\S+/.test(email)) {
      setError('Будь ласка, введіть дійсну адресу email.');
      return;
    }
    const success = await onRegisterSubmit({ username, email, password });

    if (success) {
      setUsername('');
      setEmail('');
      setPassword('');
      onRequestClose();
    } else {
      setError('Помилка реєстрації. Можливо, користувач вже існує.');
    }
  };

  const handleClose = () => {
    setError('');
    setUsername('');
    setEmail('');
    setPassword('');
    onRequestClose();
  }

  return (
    <Modal
      isOpen={isOpen}
      onRequestClose={handleClose}
      style={customStyles}
      contentLabel="Форма Реєстрації"
    >
      <h2>Реєстрація</h2>
      <form onSubmit={handleSubmit} className="modal-form">
```

```
            {error && <p className="error-message">{error}</p>}
            <div className="form-group">
              <label htmlFor="reg-username">Ім'я користувача:</label>
              <input
                type="text"
                id="reg-username"
                value={username}
                onChange={(e) => setUsername(e.target.value)}
                required
              />
            </div>
            <div className="form-group">
              <label htmlFor="reg-email">Email:</label>
              <input
                type="email"
                id="reg-email"
                value={email}
                onChange={(e) => setEmail(e.target.value)}
                required
              />
            </div>
            <div className="form-group">
              <label htmlFor="reg-password">Пароль:</label>
              <input
                type="password"
                id="reg-password"
                value={password}
                onChange={(e) => setPassword(e.target.value)}
                required
              />
            </div>
            <div className="modal-actions">
              <button type="submit" className="submit-btn">Зареєструватись</button>
              <button type="button" onClick={handleClose} className="cancel-
btn">Скасувати</button>
            </div>
          </form>
        </Modal>
    );
}

export default RegistrationModal;

//LoginModal.js

import React, { useState } from 'react';
import Modal from 'react-modal';
import './styles.css';

const customStyles = {
  content: {
```

```
      top: '50%',
      left: '50%',
      right: 'auto',
      bottom: 'auto',
      marginRight: '-50%',
      transform: 'translate(-50%, -50%)',
      minWidth: '300px',
      maxWidth: '500px',
      padding: '25px',
      borderRadius: '8px',
      boxShadow: '0 5px 15px rgba(0,0,0,0.2)'
    },
    overlay: {
      backgroundColor: 'rgba(0, 0, 0, 0.75)',
      zIndex: 1000
    },
};

function LoginModal({ isOpen, onRequestClose, onLoginSubmit }) {
    const [username, setUsername] = useState('');
    const [password, setPassword] = useState('');
    const [error, setError] = useState('');

    const handleSubmit = async (event) => {
      event.preventDefault();
      setError('');

      if (!username || !password) {
        setError('Будь ласка, заповніть ім\'я користувача та пароль.');
        return;
      }

      const success = await onLoginSubmit({ username, password });

      if (success) {
        setUsername('');
        setPassword('');
        onRequestClose();
      } else {
      }
    };

    const handleClose = () => {
      setError('');
      setUsername('');
      setPassword('');
      onRequestClose();
    }

    return (
      <Modal
```

```
        isOpen={isOpen}
        onRequestClose={handleClose}
        style={customStyles}
        contentLabel="Форма Входу"
      >
        <h2>Вхід</h2>
        <form onSubmit={handleSubmit} className="modal-form">
          {error && <p className="error-message">{error}</p>}
          <div className="form-group">
            <label htmlFor="login-username">Ім'я користувача:</label>
            <input
              type="text"
              id="login-username"
              value={username}
              onChange={(e) => setUsername(e.target.value)}
              required
            />
          </div>
          <div className="form-group">
            <label htmlFor="login-password">Пароль:</label>
            <input
              type="password"
              id="login-password"
              value={password}
              onChange={(e) => setPassword(e.target.value)}
              required
            />
          </div>
          <div className="modal-actions">
            <button type="submit" className="submit-btn">Увійти</button>
            <button type="button" onClick={handleClose} className="cancel-
btn">Скасувати</button>
          </div>
        </form>
      </Modal>
    );
}

export default LoginModal;

//InfoModal.js

import React from 'react';
import Modal from 'react-modal';
import './styles.css';

const customStyles = {
  content: {
    top: '50%',
    left: '50%',
    right: 'auto',
```

```
        bottom: 'auto',
        marginRight: '-50%',
        transform: 'translate(-50%, -50%)',
        minWidth: '300px',
        maxWidth: '600px',
        maxHeight: '80vh',
        overflowY: 'auto',
        padding: '25px',
        borderRadius: '8px',
        boxShadow: '0 5px 15px rgba(0,0,0,0.2)'
    },
    overlay: {
        backgroundColor: 'rgba(0, 0, 0, 0.75)',
        zIndex: 1000
    },
};

function InfoModal({ isOpen, onRequestClose, title, children }) {
    return (
        <Modal
            isOpen={isOpen}
            onRequestClose={onRequestClose}
            style={customStyles}
            contentLabel={title || "Інформація"}
        >
                <div className="modal-content info-modal-content">
                {title && <h2>{title}</h2>}
                <div className="info-modal-body">
                    {children}
                </div>
                <div className="modal-actions" style={{ justifyContent: 'center' }}>
                    <button onClick={onRequestClose} className="submit-btn">
                        Зрозуміло
                    </button>
                </div>
            </div>
        </Modal>
    );
}

export default InfoModal;

//SavedPlacesPage.js

import React, { useState, useEffect } from 'react';
import { Link, useNavigate } from 'react-router-dom';
import { useAuth } from './AuthContext';
import { getItem, STORAGE_KEYS } from './localStorageUtils';
import './styles.css';

function SavedPlacesPage() {
```

```
    const { currentUser, isLoggedIn } = useAuth();
    const [savedPlaces, setSavedPlaces] = useState([]);
    const navigate = useNavigate();

    useEffect(() => {
        if (!isLoggedIn || !currentUser) {
            navigate('/');
            return;
        }

        const allSavedPlaces = getItem(STORAGE_KEYS.SAVED_PLACES, {});
        setSavedPlaces(allSavedPlaces[currentUser.username] || []);

    }, [isLoggedIn, currentUser, navigate]);

    if (!isLoggedIn || !currentUser) {
        return <div className="container"><p>Перевірка авторизації...</p></div>;
    }

    return (
        <div className="container saved-places-page">
            <button className="back-btn" onClick={() => navigate(-1)}>Назад</button>
            <h2>Ваші збережені місця ({savedPlaces.length})</h2>
            {savedPlaces.length === 0 ? (
                <p>У вас ще немає збережених місць.</p>
            ) : (
                <ul className="saved-places-list">
                    {savedPlaces.map((placeName, index) => (
                        <li key={index} className="saved-place-item">
                            <Link to={`/place/${encodeURIComponent(placeName)}`}>
                                {placeName}
                            </Link>
                        </li>
                    ))}
                </ul>
            )}
        </div>
    );
}

export default SavedPlacesPage;

//StarRating.js


import React, { useState } from 'react';

function StarRating({ totalStars = 5, initialRating = 0, onRate, readonly = false, showRatingValue =
true }) {
    const [hoverRating, setHoverRating] = useState(0);
    const [currentRating, setCurrentRating] = useState(initialRating);
```

```
    const handleMouseOver = (rating) => {
        if (!readonly) {
            setHoverRating(rating);
        }
    };

    const handleMouseLeave = () => {
        if (!readonly) {
            setHoverRating(0);
        }
    };

    const handleClick = (rating) => {
        if (!readonly) {
            setCurrentRating(rating);
            if (onRate) {
                onRate(rating);
            }
        }
    };

    const displayRating = hoverRating || currentRating;

    return (
        <div className="star-rating-container">
            {[...Array(totalStars)].map((_, index) => {
                const starValue = index + 1;
                return (
                    <span
                        key={starValue}
                        className={`star ${starValue <= displayRating ? 'filled' : ''} ${readonly ? 'readonly'
: ''}`}
                        onMouseOver={() => handleMouseOver(starValue)}
                        onMouseLeave={handleMouseLeave}
                        onClick={() => handleClick(starValue)}
                    >
                        ★
                    </span>
                );
            })}
            {showRatingValue && <span className="rating-value">({displayRating > 0 ?
displayRating : initialRating}/ {totalStars})</span>}
        </div>
    );
}

export default StarRating;

//styles.css
```

```
body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
    background-color: #f8f9fa;
}

.container {
    max-width: 1200px;
    margin: 0 auto;
    padding: 20px;
}

.header {
    display: flex;
    flex-wrap: wrap;
    justify-content: space-between;
    align-items: center;
    background-color: #36aebb;
    color: white;
    padding: 15px 20px;
    margin-bottom: 10px;
}

.header h1 {
    margin: 0;
    font-size: 24px;
    flex: 1;
    white-space: nowrap;
    margin-right: 20px;
}

.header-title-link {
    color: white;
    text-decoration: none;
}

.header-title-link:hover {
    text-decoration: underline;
}

.auth-buttons {
    display: flex;
    align-items: center;
    gap: 10px;
}

.username-link {
    color: white;
    text-decoration: none;
```

```
    display: inline-block;
    padding: 5px;
    border-radius: 4px;
    transition: background-color 0.2s ease;
}

.username-link:hover {
    background-color: rgba(255, 255, 255, 0.2);
    text-decoration: none;
}

.username-display {
    margin-right: 10px;
    font-weight: bold;
    white-space: nowrap;
}

.login-btn,
.register-btn {
    background-color: white;
    color: #36aebb;
    border: none;
    padding: 8px 15px;
    cursor: pointer;
    font-weight: bold;
    border-radius: 5px;
    white-space: nowrap;
    font-size: 0.9rem;
}

.login-btn:hover,
.register-btn:hover {
    background-color: #e9ecef;
}

@media (max-width: 768px) {
    .header {
        flex-direction: column;
        align-items: flex-start;
        padding: 15px;
    }

    .header h1 {
        margin-bottom: 10px;
        margin-right: 0;
    }

    .auth-buttons {
        width: 100%;
        justify-content: flex-start;
    }
```

```
    .username-display {
        margin-bottom: 5px;
    }
}

@media (max-width: 480px) {
    .auth-buttons {
        flex-direction: column;
        align-items: stretch;
    }

    .username-display {
        text-align: center;
        margin-right: 0;
        margin-bottom: 10px;
    }

    .login-btn,
    .register-btn {
        width: 100%;
        text-align: center;
    }
}

.save-btn {
    background-color: #ffc107;
    color: black;
    border: none;
    padding: 8px 12px;
    cursor: pointer;
    border-radius: 5px;
    font-size: 0.85rem;
    transition: background-color 0.2s ease, color 0.2s ease;
}

.save-btn:hover:not(:disabled) {
    background-color: #e0a800;
}

.save-btn.saved {
    background-color: #28a745;
    color: white;
}

.save-btn.saved:hover:not(:disabled) {
    background-color: #218838;
}

.save-btn:disabled {
    background-color: #ccc;
```

```
    cursor: not-allowed;
    opacity: 0.7;
}

.place-card {
    background: white;
    border-radius: 10px;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
    text-align: center;
    overflow: hidden;
    display: flex;
    flex-direction: column;
    cursor: pointer;
    transition: transform 0.2s ease-in-out;
}

.place-card:hover {
    transform: translateY(-5px);
}

.place-image {
    width: 100%;
    height: 300px;
    object-fit: cover;
    display: block;
}

.place-card-content {
    padding: 15px;
    flex-grow: 1;
    display: flex;
    flex-direction: column;
    align-items: center;
}

.place-card-content h3 {
    margin-top: 0;
    margin-bottom: 5px;
    font-size: 1.1rem;
}

.place-card-content .rating {
    font-size: 1rem;
    color: #ffa500;
    margin-bottom: 10px;
}

.card-buttons-inline {
    display: flex;
    justify-content: center;
    gap: 10px;
```

```
    margin-top: auto;
    padding-top: 10px;
    width: 100%;
}

.card-buttons-inline .map-btn,
.card-buttons-inline .save-btn {
    padding: 6px 10px;
    font-size: 0.8rem;
    flex-grow: 1;
    max-width: 120px;
}

.place-page-actions {
    margin-top: 20px;
    margin-bottom: 20px;
    display: flex;
    gap: 15px;
    align-items: center;
}

.place-page-actions .save-btn,
.place-page-actions .ticket-btn {
    padding: 10px 20px;
    font-size: 1rem;
}

.map-btn {
    background-color: #6c757d;
    color: white;
    border: none;
    cursor: pointer;
    border-radius: 5px;
    transition: background-color 0.2s ease;
}

.map-btn:hover {
    background-color: #5a6268;
}

.ticket-btn {
    background-color: #ff5733;
    color: white;
    padding: 10px 15px;
    text-decoration: none;
    display: inline-block;
    border-radius: 5px;
    transition: background-color 0.2s ease;
    font-size: 1rem;
    border: none;
}
```

```
.ticket-btn:hover {
    background-color: #d84529;
}

.image-carousel {
    position: relative;
    width: 100%;
    margin-bottom: 20px;
    overflow: hidden;
    border-radius: 8px;
}

.place-image-large {
    display: block;
    width: 100%;
    aspect-ratio: 16 / 9;
    object-fit: cover;
    border-radius: 8px;
}

.carousel-btn {
    position: absolute;
    top: 50%;
    transform: translateY(-50%);
    background-color: rgba(0, 0, 0, 0.5);
    color: white;
    border: none;
    font-size: 2rem;
    padding: 5px 15px;
    cursor: pointer;
    border-radius: 50%;
    z-index: 10;
    line-height: 1;
    width: 40px;
    height: 40px;
    display: flex;
    align-items: center;
    justify-content: center;
}

.carousel-btn:hover {
    background-color: rgba(0, 0, 0, 0.8);
}

.prev-btn {
    left: 10px;
}

.next-btn {
    right: 10px;
```

```
}

.carousel-dots {
    position: absolute;
    bottom: 15px;
    left: 50%;
    transform: translateX(-50%);
    display: flex;
    gap: 8px;
    z-index: 10;
}

.dot {
    width: 10px;
    height: 10px;
    background-color: rgba(255, 255, 255, 0.7);
    border-radius: 50%;
    cursor: pointer;
    transition: background-color 0.3s ease;
}

.dot.active {
    background-color: white;
}

.star-rating-container {
    display: inline-flex;
    align-items: center;
    gap: 2px;
}

.star {
    font-size: 24px;
    color: #ccc;
    cursor: pointer;
    transition: color 0.2s ease-in-out;
}

.star.filled {
    color: #ffa500;
}

.star.readonly {
    cursor: default;
}

.rating-value {
    margin-left: 8px;
    font-size: 14px;
    color: #555;
}
```

```
.average-rating {
    margin-top: 15px;
    margin-bottom: 10px;
    font-weight: bold;
    display: flex;
    align-items: center;
    gap: 5px;
}

.user-rating-section {
    margin-top: 10px;
    margin-bottom: 20px;
    padding: 15px;
    background-color: #f8f9fa;
    border-radius: 8px;
    border: 1px solid #eee;
}

.user-rating-section p {
    margin-bottom: 8px;
}

.user-rating-section .star {
    font-size: 28px;
}

.auth-prompt {
    font-size: 0.9em;
    color: #6c757d;
    margin-top: 5px;
}

.auth-prompt a {
    color: #007bff;
    text-decoration: none;
}

.auth-prompt a:hover {
    text-decoration: underline;
}

.reviews-section {
    margin-top: 30px;
    padding-top: 20px;
    border-top: 1px solid #eee;
}

.reviews {
    list-style: none;
    padding: 0;
```

```
    margin-bottom: 20px;
}

.review-item {
    background-color: #fff;
    padding: 15px;
    margin-bottom: 10px;
    border-radius: 8px;
    border: 1px solid #eee;
    box-shadow: 0 2px 4px rgba(0, 0, 0, 0.05);
}

.review-author {
    font-weight: bold;
}

.review-date {
    font-size: 0.85em;
    color: #6c757d;
    margin-left: 5px;
}

.review-text {
    margin-top: 5px;
    line-height: 1.5;
}

.review-form {
    display: flex;
    flex-wrap: wrap;
    gap: 10px;
    align-items: flex-end;
    margin-top: 20px;
}

.review-input {
    flex-grow: 1;
    min-width: 200px;
    height: auto;
    padding: 10px;
    border: 1px solid #ccc;
    border-radius: 5px;
    font-size: 1rem;
    resize: vertical;
}

.review-input:disabled {
    background-color: #e9ecef;
    cursor: not-allowed;
}
```

```
.review-actions {
    display: flex;
    flex-direction: column;
    gap: 5px;
}

.comment-rules-btn,
.submit-review-btn {
    padding: 10px 15px;
    border-radius: 5px;
    cursor: pointer;
    border: none;
    font-size: 0.9rem;
    white-space: nowrap;
}

.comment-rules-btn {
    background-color: #6c757d;
    color: white;
}

.comment-rules-btn:hover {
    background-color: #5a6268;
}

.submit-review-btn {
    background-color: #36aebb;
    color: white;
    width: auto;
    margin-top: 0;
}

.submit-review-btn:hover:not(:disabled) {
    background-color: #2c8fa0;
}

.submit-review-btn:disabled {
    background-color: #a7d8de;
    cursor: not-allowed;
}

@media (max-width: 600px) {
    .review-form {
        flex-direction: column;
        align-items: stretch;
    }

    .review-actions {
        flex-direction: row;
        justify-content: flex-end;
    }
```

```
}

.saved-places-page h2 {
   margin-bottom: 20px;
   color: #333;
}

.saved-places-list {
   list-style: none;
   padding: 0;
}

.saved-place-item {
   background-color: #fff;
   padding: 15px 20px;
   margin-bottom: 10px;
   border-radius: 8px;
   border: 1px solid #eee;
   box-shadow: 0 2px 4px rgba(0, 0, 0, 0.05);
   transition: box-shadow 0.2s ease;
}

.saved-place-item:hover {
   box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
}

.saved-place-item a {
   text-decoration: none;
   color: #36aebb;
   font-weight: bold;
   font-size: 1.1rem;
}

.saved-place-item a:hover {
   text-decoration: underline;
}

.recommendations-section {
   margin-top: 40px;
   padding-top: 20px;
   border-top: 1px solid #eee;
}
.recommendations-section h2 {
   margin-bottom: 20px;
   text-align: center;
   color: #333;
}
.recommendations-grid {
   display: flex;
   flex-wrap: nowrap;
   overflow-x: auto;
```

```
    gap: 15px;
    padding-bottom: 15px;
    align-items: stretch;
}
.recommendations-grid::-webkit-scrollbar {
    height: 8px;
}
.recommendations-grid::-webkit-scrollbar-track {
    background: #f1f1f1;
    border-radius: 4px;
}
.recommendations-grid::-webkit-scrollbar-thumb {
    background: #ccc;
    border-radius: 4px;
}
.recommendations-grid::-webkit-scrollbar-thumb:hover {
    background: #aaa;
}
.recommendations-grid .place-card {
    flex-shrink: 0;
    width: 280px;
    height: auto;
}

.directions-section {
    margin-top: 30px;
    margin-bottom: 20px;
    padding: 20px;
    background-color: #f8f9fa;
    border-radius: 8px;
    border: 1px solid #eee;
}
.directions-section h3 {
    margin-top: 0;
    margin-bottom: 15px;
    text-align: center;
}
.directions-controls {
    display: flex;
    flex-wrap: wrap;
    gap: 10px;
    align-items: center;
    justify-content: center;
}
.origin-input {
    flex-grow: 1;
    padding: 10px;
    border: 1px solid #ccc;
    border-radius: 4px;
    min-width: 200px;
    font-size: 1rem;
```

```
}
.travel-mode-select {
    padding: 10px;
    border: 1px solid #ccc;
    border-radius: 4px;
    background-color: white;
    cursor: pointer;
    min-width: 100px;
}
.directions-controls button {
    padding: 10px 15px;
    background-color: #36aebb;
    color: white;
    border: none;
    border-radius: 4px;
    cursor: pointer;
    font-size: 0.9rem;
    transition: background-color 0.2s ease;
    white-space: nowrap;
    /* Запобігає переносу тексту */
}
.directions-controls button:hover:not(:disabled) {
    background-color: #2c8fa0;
}
.directions-controls button:disabled {
    background-color: #a7d8de;
    cursor: not-allowed;
}
.directions-section .error-message {
    color: #dc3545;
    font-weight: bold;
    text-align: center;
    margin-top: 10px;
    width: 100%;
}

@media (max-width: 768px) {
    .directions-controls {
        flex-direction: column;
        align-items: stretch;
    }

    .origin-input {
        min-width: unset;
    }
}

.back-btn {
    background-color: #6c757d;
    color: white;
    border: none;
```

```
    padding: 10px 18px;
    cursor: pointer;
    border-radius: 5px;
    margin-bottom: 15px;
    font-size: 0.95rem;
    transition: background-color 0.2s ease;
    align-items: center;
    gap: 5px;
}

.back-btn:hover {
    background-color: #5a6268;
}

.category-btn {
    background-color: white;
    border: 2px solid #36aebb;
    padding: 25px;
    font-size: 1.1rem;
    border-radius: 10px;
    cursor: pointer;
    text-align: center;
    display: flex;
    flex-direction: column;
    align-items: center;
    justify-content: center;
    width: 180px;
    height: 180px;
    transition: all 0.2s ease-in-out;
    color: #333;
    box-shadow: 0 3px 6px rgba(0, 0, 0, 0.1);
}
.categories-grid{
    display: flex;
    flex-direction: row;
    gap: 15px;
    justify-content: center;
}

.category-btn .icon {
    font-size: 45px;
    margin-bottom: 8px;
}

.category-btn:hover {
    transform: translateY(-4px);
    box-shadow: 0 6px 12px rgba(0, 0, 0, 0.15);
    background-color: #e1f7f9;
    border-color: #2c8fa0;
}
```

```
.search-container {
    position: relative;
    width: 100%;
    max-width: 600px;
    margin: 25px auto 30px;
}

.search-input {
    width: 100%;
    padding: 12px 15px;
    padding-right: 40px;
    font-size: 1rem;
    border: 1px solid #ccc;
    border-radius: 25px;
    outline: none;
    transition: border-color 0.2s ease, box-shadow 0.2s ease;
    box-sizing: border-box;
}

.search-input:focus {
    border-color: #36aebb;
    box-shadow: 0 0 0 3px rgba(54, 174, 187, 0.25);
}

.search-container::after {
    content: '🔍';
    position: absolute;
    right: 15px;
    top: 50%;
    transform: translateY(-50%);
    opacity: 0.6;
}

.dropdown {
    position: absolute;
    width: 100%;
    background: white;
    border: 1px solid #ddd;
    border-radius: 8px;
    margin-top: 5px;
    list-style: none;
    padding: 0;
    max-height: 250px;
    overflow-y: auto;
    z-index: 1000;
    box-shadow: 0 4px 10px rgba(0, 0, 0, 0.1);
}

.dropdown li {
    padding: 12px 15px;
```

```
    cursor: pointer;
    border-bottom: 1px solid #eee;
    font-size: 0.95rem;
}

.dropdown li:last-child {
    border-bottom: none;
}

.dropdown li:hover {
    background: #e1f7f9;
    color: #333;
}

.modal-form {
    display: flex;
    flex-direction: column;
    gap: 15px;
}

.modal-form h2 {
    text-align: center;
    margin-top: 0;
    margin-bottom: 20px;
    color: #333;
}

.form-group {
    display: flex;
    flex-direction: column;
}

.form-group label {
    margin-bottom: 5px;
    font-weight: bold;
    color: #555;
    font-size: 0.9rem;
}

.form-group input {
    padding: 10px;
    border: 1px solid #ccc;
    border-radius: 4px;
    font-size: 1rem;
}

.form-group input:focus {
    border-color: #36aebb;
    outline: none;
    box-shadow: 0 0 0 2px rgba(54, 174, 187, 0.2);
}
```

```
.modal-actions {
    margin-top: 20px;
    display: flex;
    justify-content: flex-end;
    gap: 10px;
}

.modal-actions .submit-btn {
    background-color: #36aebb;
    color: white;
    padding: 10px 20px;
    border: none;
    border-radius: 5px;
    cursor: pointer;
    transition: background-color 0.2s ease;
}

.modal-actions .submit-btn:hover {
    background-color: #2c8fa0;
}

.modal-actions .cancel-btn {
    background-color: #6c757d;
    color: white;
    padding: 10px 20px;
    border: none;
    border-radius: 5px;
    cursor: pointer;
    transition: background-color 0.2s ease;
}

.modal-actions .cancel-btn:hover {
    background-color: #5a6268;
}

.modal-form .error-message {
    color: #dc3545;
    background-color: #f8d7da;
    border: 1px solid #f5c6cb;
    padding: 10px;
    border-radius: 5px;
    margin-bottom: 15px;
    text-align: center;
    font-size: 0.9rem;
}

.info-modal-body {
    margin-top: 15px;
    margin-bottom: 25px;
    line-height: 1.6;
```

```
    font-size: 0.95rem;
    color: #333;
}

.info-modal-body ul {
    padding-left: 20px;
    list-style: disc;
}

.info-modal-body li {
    margin-bottom: 8px;
}
```

Додаток Г – Графічна частина

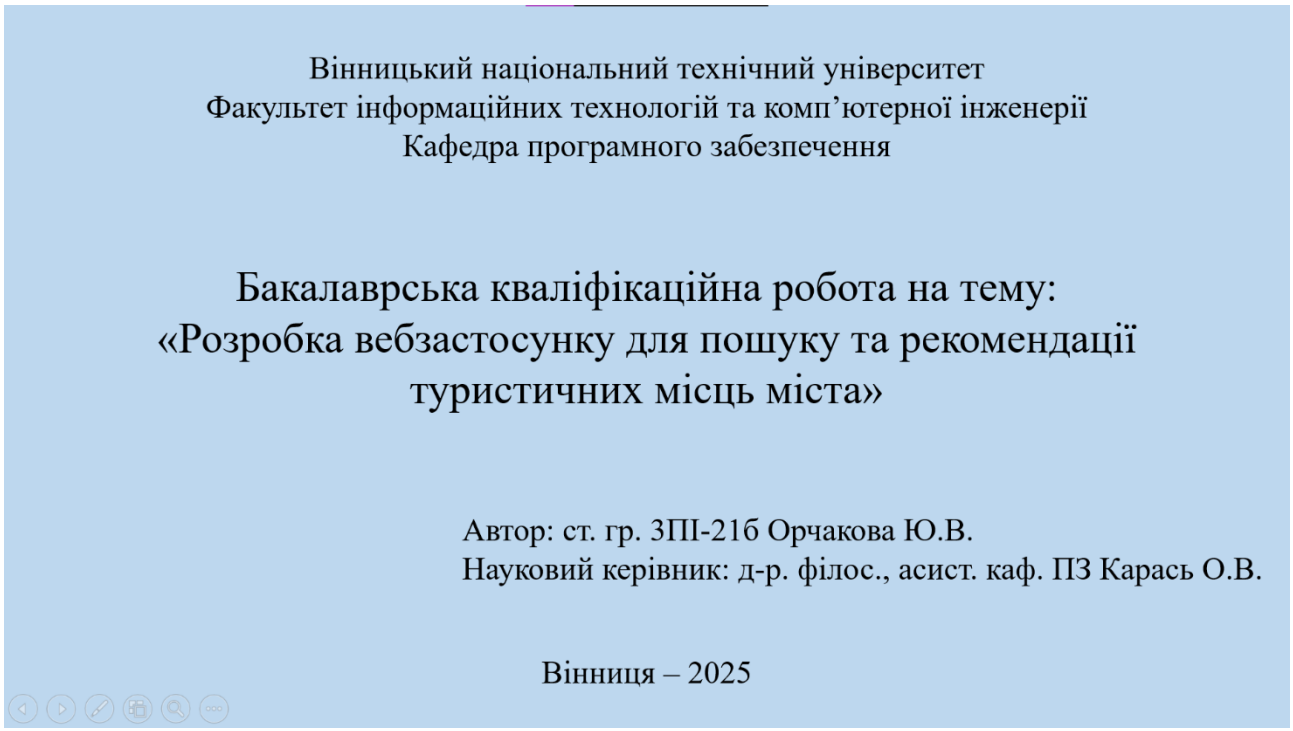

Рисунок Г.1 – Титульний аркуш

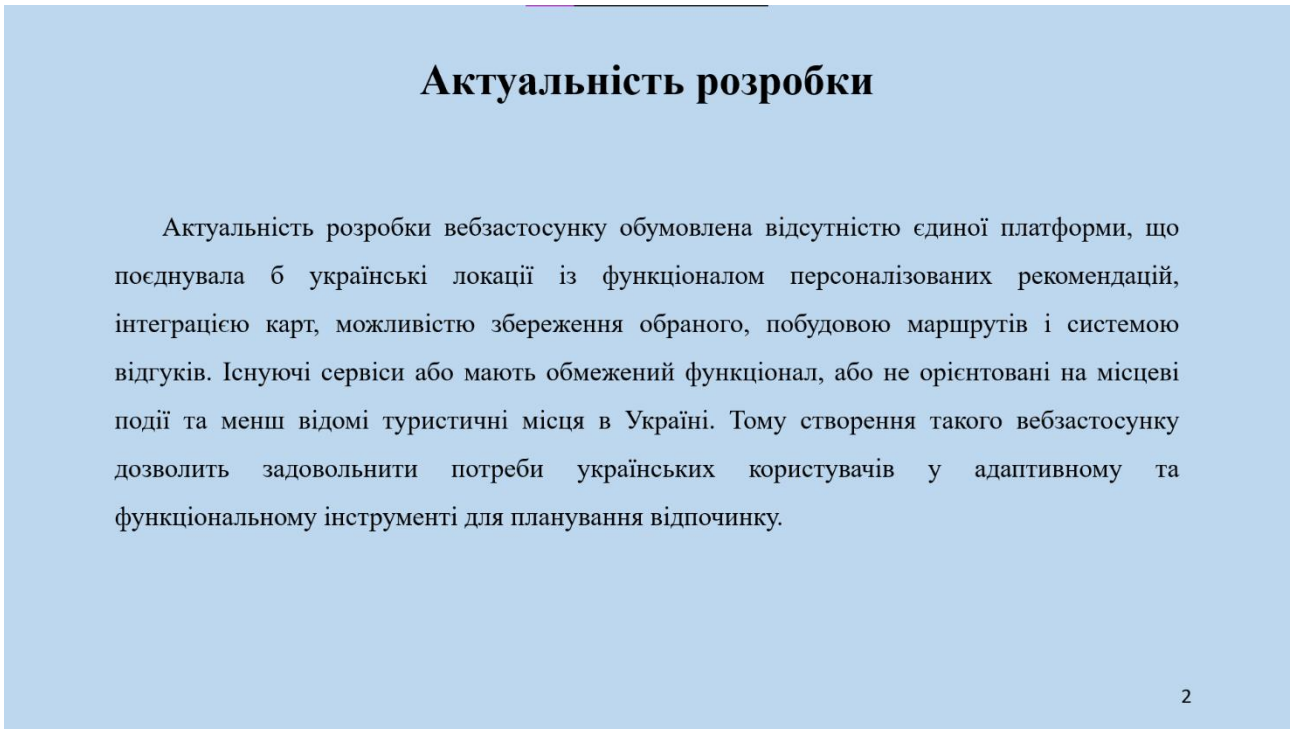

Рисунок Г.2 – Актуальність розробки

Мета, об’єкт та предмет дослідження

Метою дослідження є покращення якості процесу пошуку актуальних туристичних подій та місць, підвищення комфорту планування дозвілля за рахунок розробки програмних модулів, які забезпечують пошук, персоналізовані рекомендації, побудову маршрутів.

Об’єктом дослідження є процес розробки вебзастосунку пошуку і рекомендацій туристичних місць.

Предметом дослідження є методи та засоби розробки вебзастосунку пошуку і рекомендацій туристичних місць.

3

Рисунок Г.3 – Мета, об’єкт та предмет дослідження

Задачі дослідження

- розробити алгоритм персоналізованих рекомендацій, що забезпечить точне визначення інтересів користувачів та формування відповідних пропозицій;
- розробити модуль пошуку та фільтрації, який дозволить користувачам швидко знаходити цікаві місця за категоріями;
- реалізувати інтеграцію з картографічними сервісами, що забезпечить відображення місць на карті та побудову маршрутів;
- розробити адаптивний інтерфейс користувача, який буде зручним для користування і на комп’ютерах, і на мобільних пристроях.

4

Рисунок Г.4 – Задачі дослідження

Наукова новизна та практична цінність отриманих результатів

1. Удосконалено метод реалізації персоналізованих рекомендацій для туристичних вебсистем, який, на відміну від існуючих, використовує аналіз даних про взаємодію користувача, особливість якого полягає у визначенні пріоритетних для користувача категорій та формуванні списку ще не відомих йому пропозицій, що дозволило спростити архітектуру та реалізувати функціонал персоналізації, підвищуючи залученість користувача.

2. Удосконалено підхід до інтеграції функціоналу планування маршрутів, який відрізняється від простого відображення статичної карти тим, що в ньому використано комбінацію компонентів для розрахунку та візуалізації маршруту з можливістю введення користувачем початкової точки як текстової адреси, так і через геолокацію, що дозволило забезпечити відображення маршруту на тій самій карті, де показано цільовий об'єкт, та спростити процес планування для користувача без необхідності переходу до сторонніх картографічних сервісів.

Практичне значення полягає у можливості використання вебзастосунку туристами для швидкого знаходження цікавих місць відповідно до їхніх вподобань. Розроблений програмний продукт може бути використаний для вдосконалення існуючих платформ або як основа для створення нових туристичних сервісів.

5

Рисунок Г.5 – Наукова новизна та практична цінність отриманих результатів

Порівняльний аналіз аналогів

Критерій	MoeMisto.ua	Booking.com	TripAdvisor	Власна розробка
Персоналізовані рекомендації	-	+	+	+
Відгуки користувачів	-	+	+	+
Адаптивність інтерфейсу	+	+	+	+
Підтримка українських місцевих подій	+	-	-	+
Можливість збереження вибраного	-	+	+	+
Фільтр за категорією	+	+	+	+
Можливість побудови маршрутів до локації	-	-	-	+
Загальна оцінка	40%	70%	70%	100%

6

Рисунок Г.6 – Порівняльний аналіз аналогів

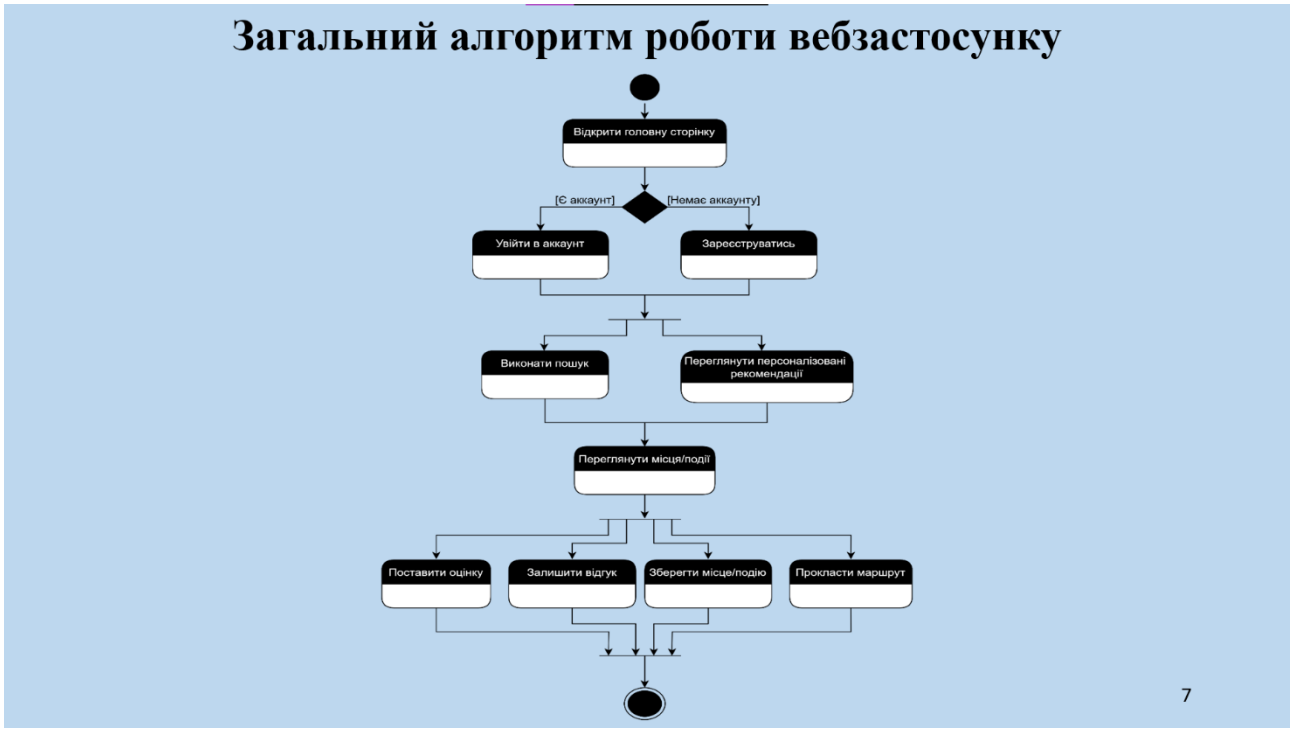

Рисунок Г.7 – Загальний алгоритм роботи вебзастосунку

Рисунок Г.8 – Алгоритм генерації персоналізованих рекомендацій

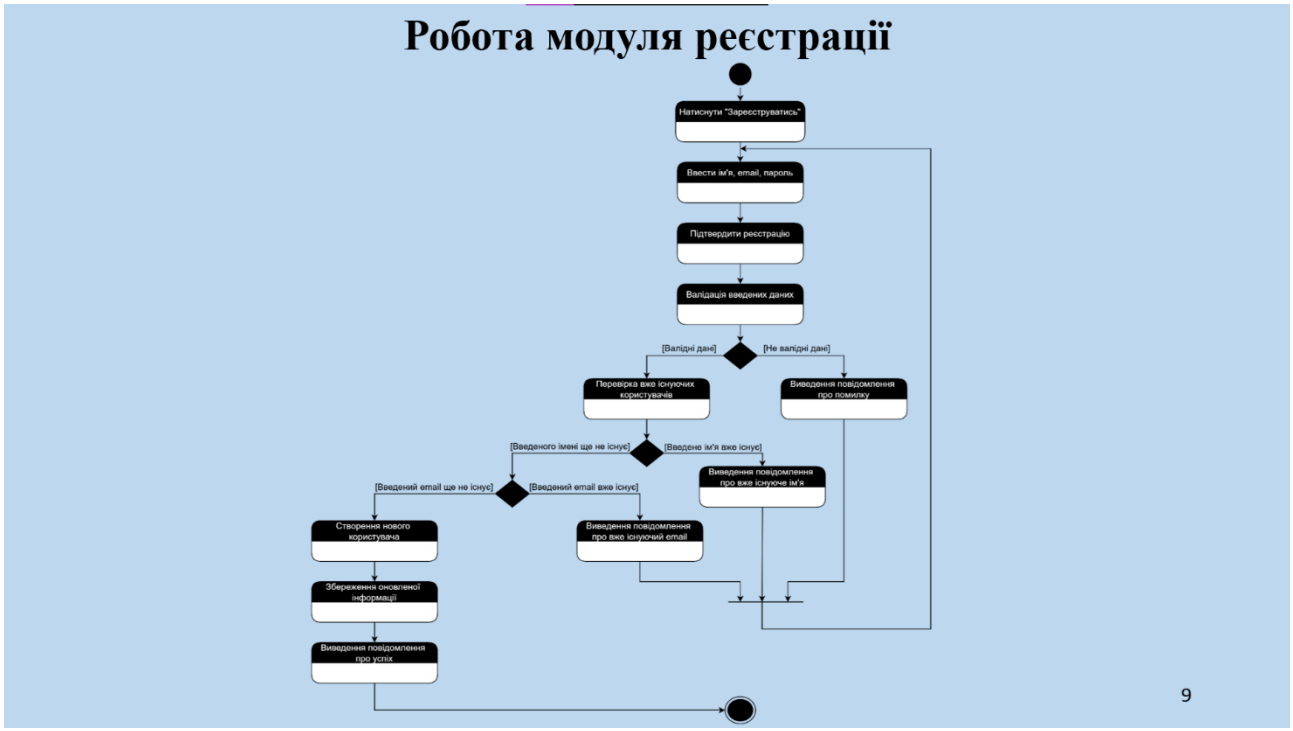

Рисунок Г.9 – Робота модуля реєстрації

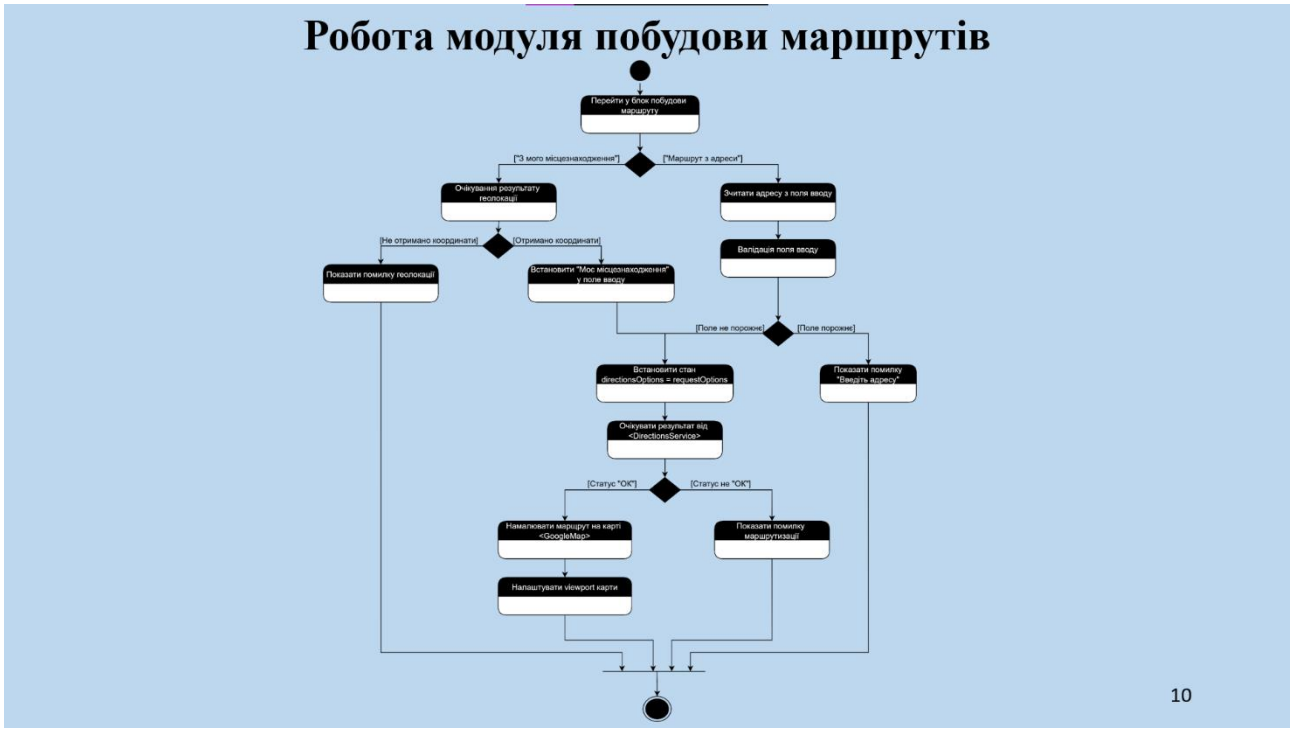

Рисунок Г.10 – Робота модуля побудови маршрутів

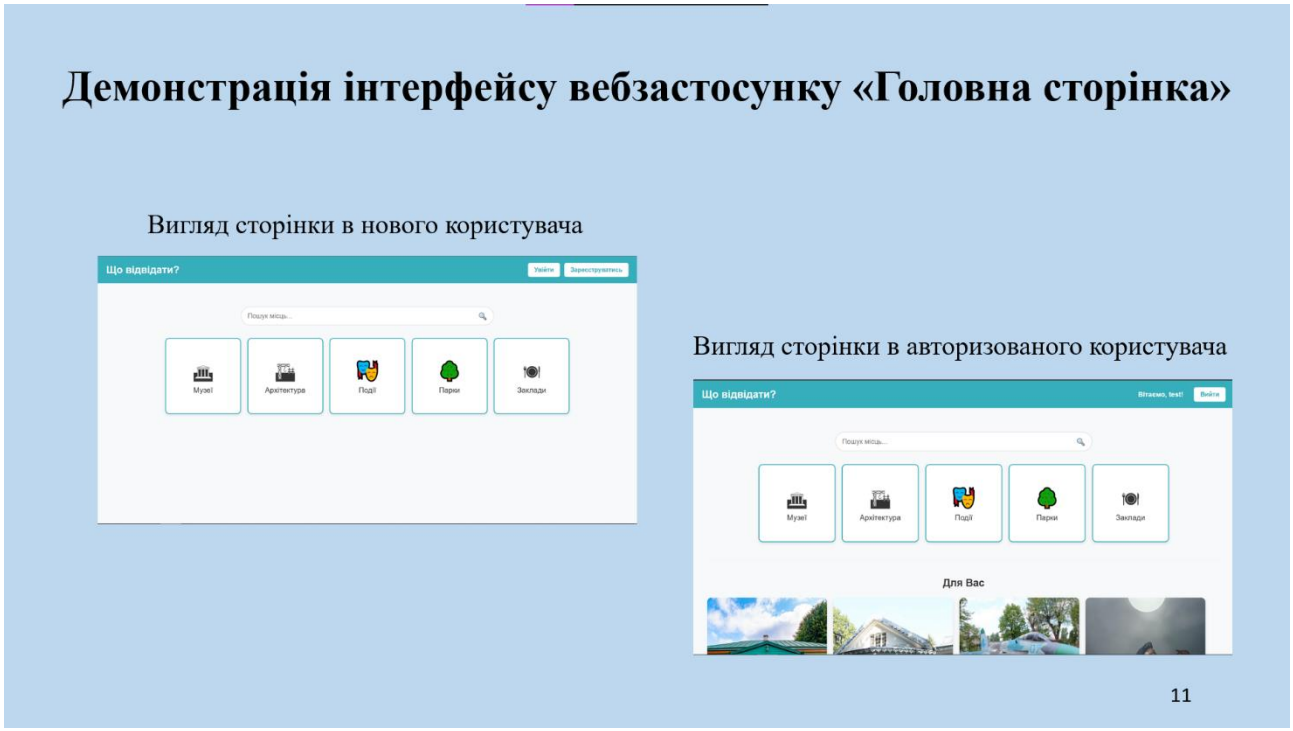

Рисунок Г.11 – Демонстрація інтерфейсу вебзастосунку «Головна сторінка»

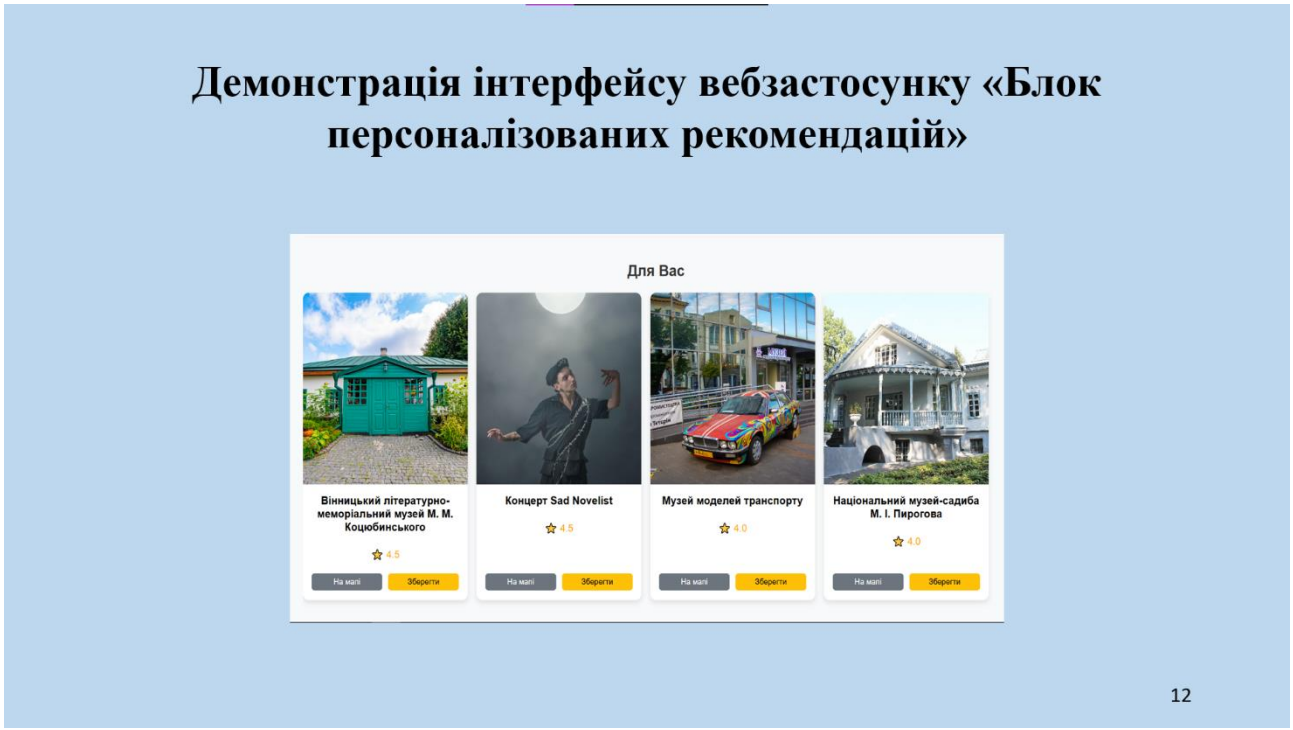

Рисунок Г.12 – Демонстрація інтерфейсу вебзастосунку «Блок персоналізованих рекомендацій»

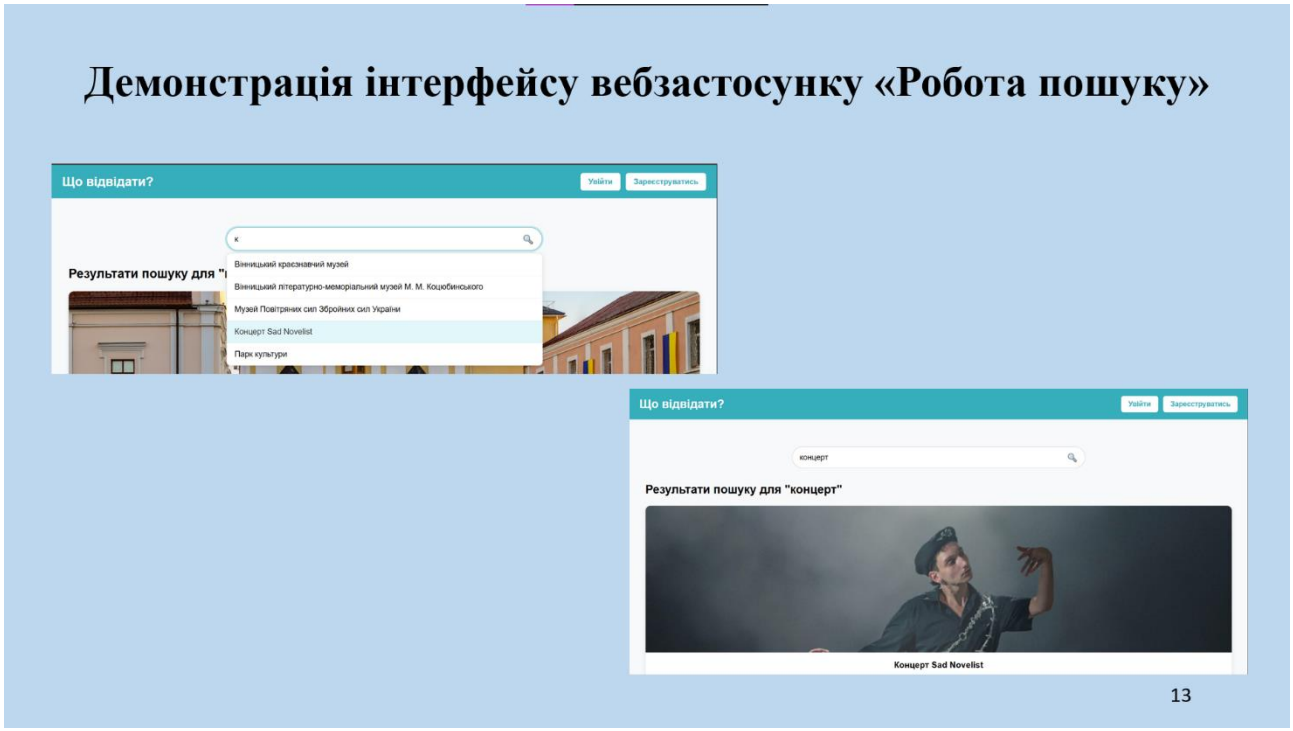

Рисунок Г.13 – Демонстрація інтерфейсу вебзастосунку «Робота пошуку»

Рисунок Г.14 – Демонстрація інтерфейсу вебзастосунку «Побудова маршруту»

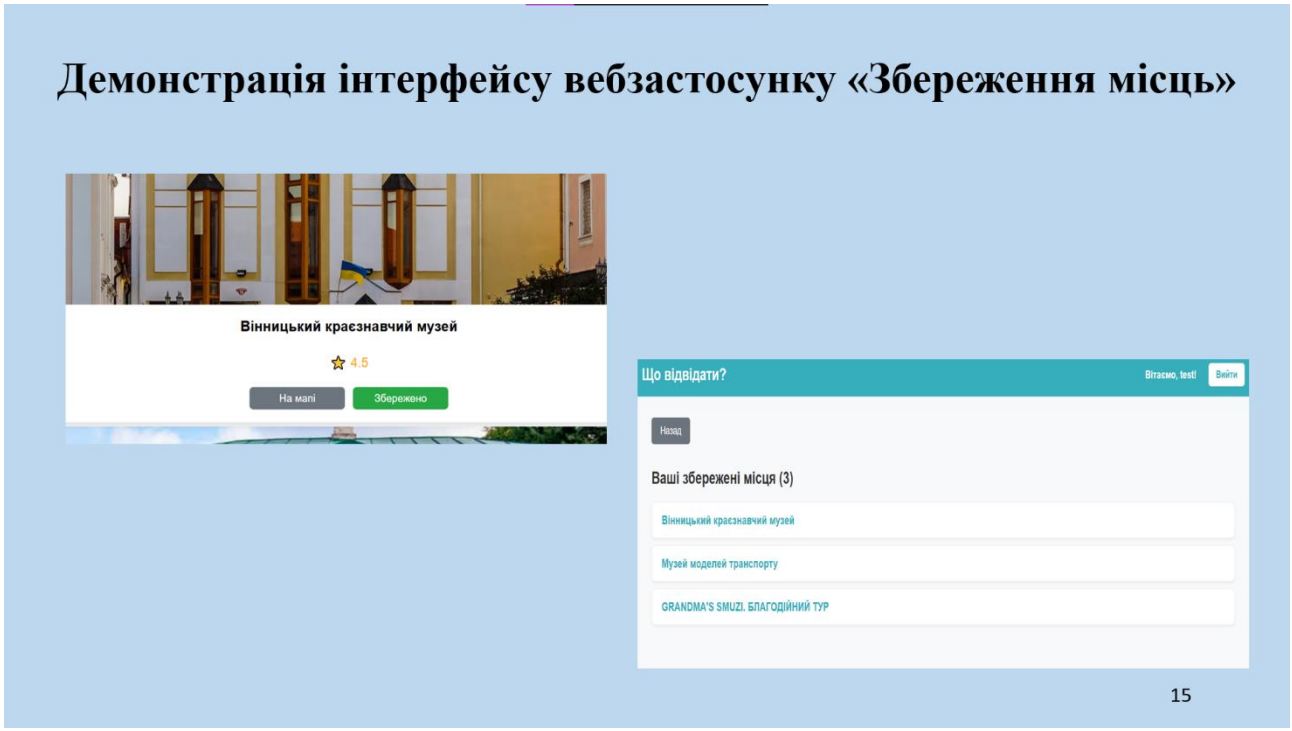

Рисунок Г.15 – Демонстрація інтерфейсу вебзастосунку «Збереження місць»

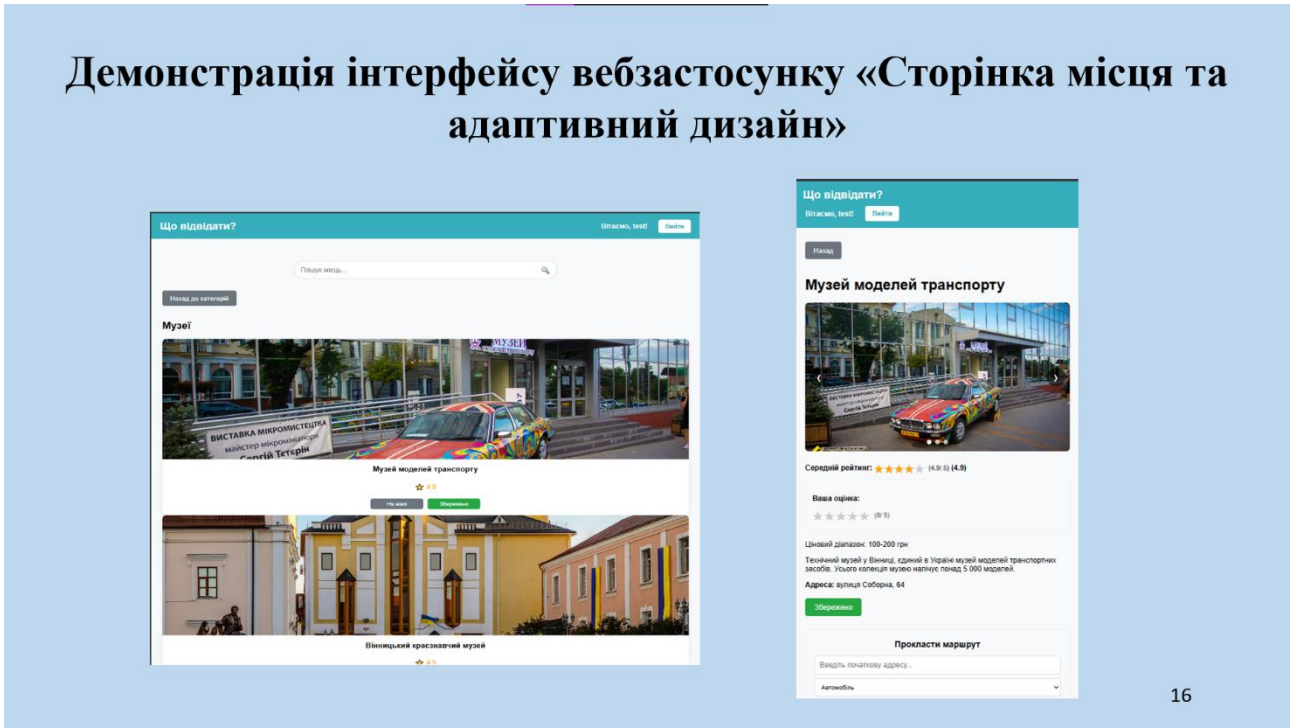

Рисунок Г.16 – Демонстрація інтерфейсу вебзастосунку «Сторінка місця та адаптивний дизайн»

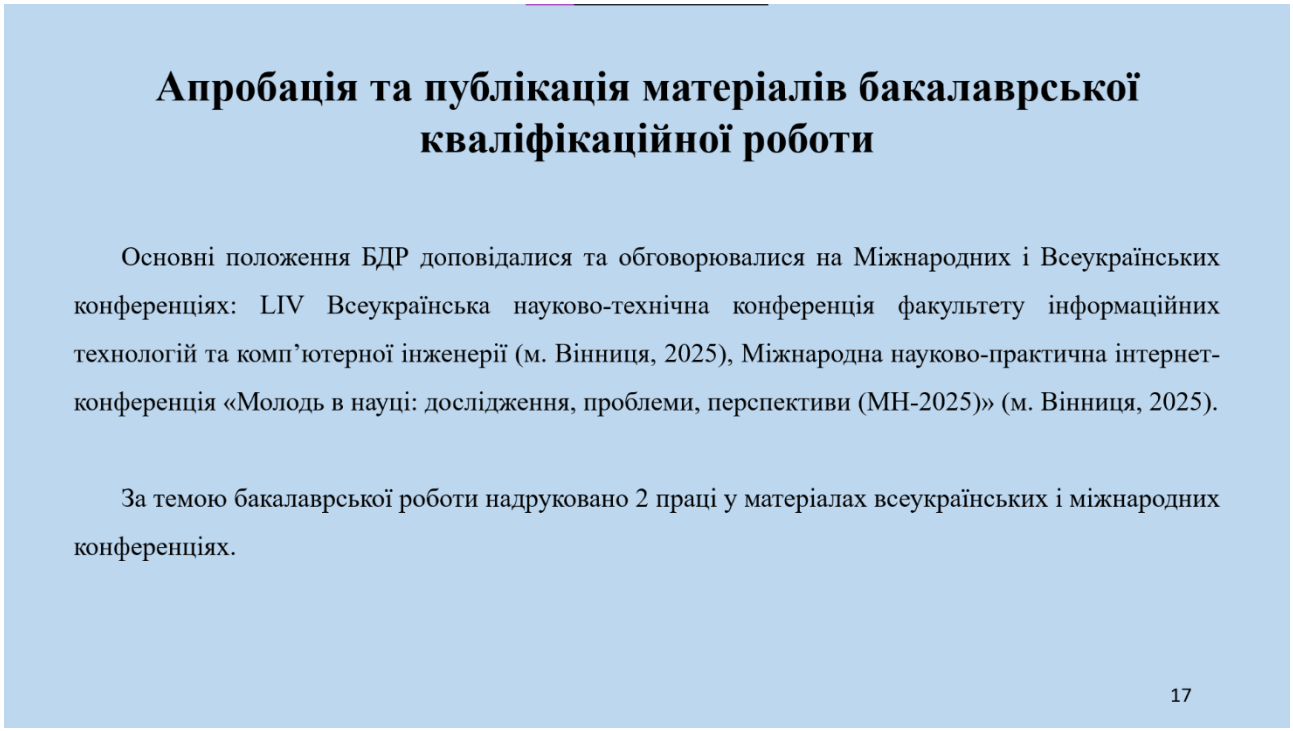

Рисунок Г.17 – Апробація та публікація матеріалів бакалаврської кваліфікаційної роботи

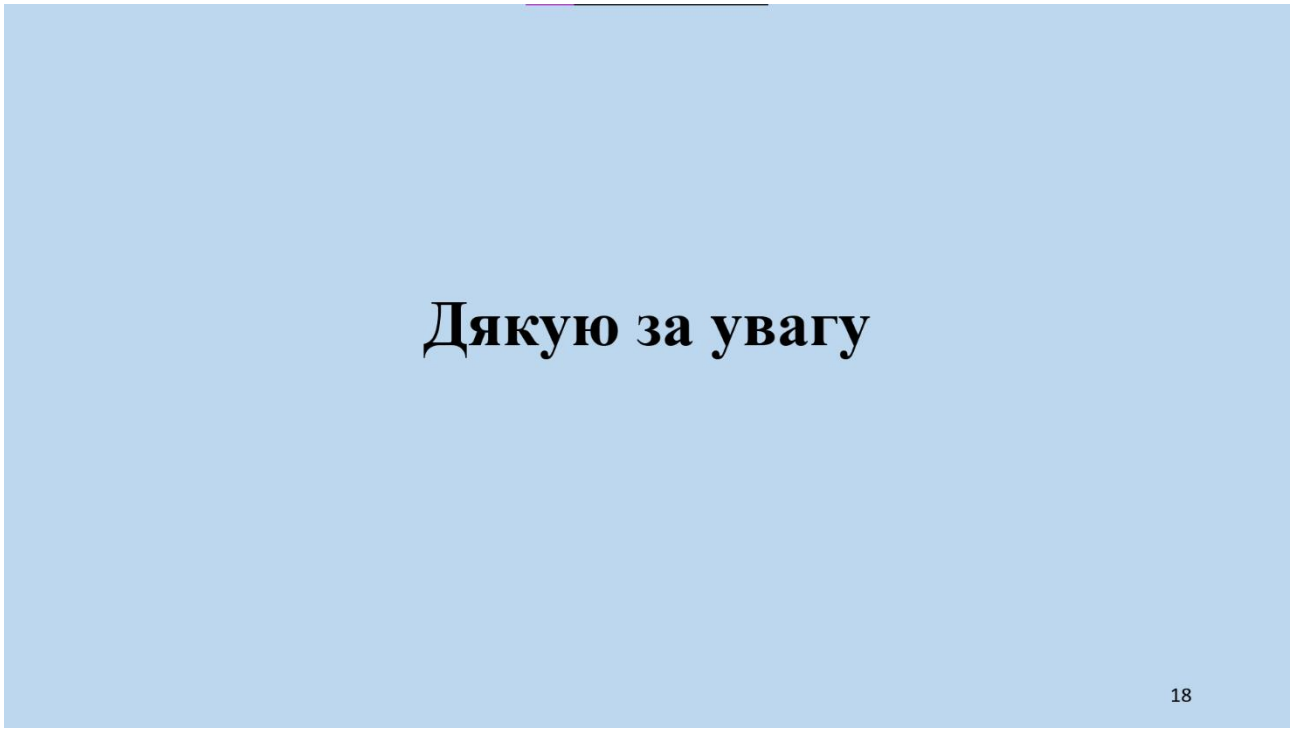

Рисунок Г.18 – Закінчення презентації