Міністерство освіти і науки України
Луцький національний технічний університет
Факультет комп'ютерних та інформаційних технологій
Кафедра інженерії програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА
ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «МАГІСТР»

РОЗРОБКА ТА ДОСЛІДЖЕННЯ ВЕБПЛАТФОРМИ ДЛЯ ПІДТРИМКИ СОЦІАЛЬНИХ ІНІЦІАТИВ ІЗ ГЕОЛОКАЦІЙНИМИ ФУНКЦІЯМИ

DEVELOPMENT AND RESEARCH OF A WEB PLATFORM TO SUPPORT SOCIAL INITIATIVES WITH GEOLOCATION FUNCTIONS

спеціальність 121 «Інженерія програмного забезпечення»
освітня програма «Інженерія програмного забезпечення»

Виконав: здобувач вищої освіти
групи ІПЗм-21
Гузоватий С. В.
Керівник:
Андрущак І. Є.

Кваліфікаційну роботу
допущено до захисту
«___» _____ 20__ р.
Гарант освітньої програми:
к.т.н., доцент Суринович О. М.
_____

Луцьк – 2025 року

ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет *комп'ютерних та інформаційних технологій*
Кафедра *інженерії програмного забезпечення*
Ступінь вищої освіти *магістр*
Галузь знань: *12 «Інформаційні технології»*
Спеціальність: *121 «Інженерія програмного забезпечення»*
Освітня програма: *«Інженерія програмного забезпечення»*

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____
«___» _____ 202__ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧА
ДРУГОГО (МАГІСТЕРСЬКОГО) РІВНЯ ВИЩОЇ ОСВІТИ

Гузоватому Сергію Володимировичу
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи: Розробка та дослідження вебплатформи для підтримки соціальних ініціатив із геолокаційними функціями
Керівник роботи: д.т.н., професор Андрущак І. Є.
затверджені наказом закладу вищої освіти від «29» березня 2025 р. № 190/01-02

2. Строк подання здобувачем вищої освіти кваліфікаційної роботи: «04» грудня 2025 р.

3. Вихідні дані до роботи: *технічне та програмне забезпечення ЕОМ*

4. Зміст розрахунково-пояснювальної записки (перелік питань, що потрібно розробити): *Аналіз сучасного стану проблеми, існуючих методів і засобів її розв'язання, аналіз і вибір засобів проектування, опис функціонального наповнення об'єкта проектування, розробка й обґрунтування системного наповнення, оцінка ергономічних та надійністних параметрів проектованої системи, функціонально-структурна схема роботи об'єкта проектування.*

5. Перелік графічного матеріалу: 28 рисунків, 4 таблиці, 14 лістингів коду

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада Консультанта	Підпис	
		завдання видав	завдання прийняв
Аналіз проблеми за темою роботи та постановка завдань дослідження	*Андрущак І. Є.*		
Теоретичне дослідження та практична реалізація	*Андрущак І. Є.*		
Експериментальне дослідження системи	*Андрущак І. Є.*		
Нормоконтроль	*Повстяна Ю. С.*		
Гарант ОП	*Андрущак І. Є.*		
Показник запозичень тексту	___%		
Академічна доброчесність	*Андрущак І. Є.*		

7. Дата видачі завдання *« 02 » квітня 2025 р.*

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи магістра	Строк виконання етапів роботи	Примітка
1	*Провести огляд літературних джерел по темі кваліфікаційної роботи*	*05.09.2025 р.*	
2	*Провести аналіз загальної проблеми і вибір напрямків дослідження*	*24.09.2025 р.*	
3	*Розробити функціональну схему роботи програмного продукту*	*01.10.2025 р.*	
4	*Описати засоби розробки об'єкта проектування*	*19.10.2025 р.*	
5	*Практична реалізація об'єкта проектування*	*26.10.2025 р.*	
6	*Розробити методику для проведення експерименту*	*05.11.2025 р.*	
7	*Провести аналіз результатів експерименту*	*15.11.2025 р.*	
8	*Здача чистового варіанту магістерської роботи на кафедру*	*04.12.2025 р.*	

Здобувач вищої освіти _____ (підпис) Гузоватий С. В. (прізвище, ініціали)

Керівник кваліфікаційної роботи _____ (підпис) Андрущак І. Є. (прізвище, ініціали)

АНОТАЦІЯ

Гузоватий С. В. Розробка та дослідження вебплатформи для підтримки соціальних ініціатив із геолокаційними функціями. Рукопис.

Кваліфікаційна робота магістра ОП «Інженерія програмного забезпечення». Спеціальності 121 «Інженерія програмного забезпечення». Луцький національний технічний університет. Луцьк, 2025.

Кваліфікаційна робота магістра складається зі вступу, трьох розділів, висновків, списку використаних джерел (згідно структури кваліфікаційної роботи, затвердженої кафедрою).

У першому розділі здійснено аналіз сучасного стану проблеми координації соціальних ініціатив, розглянуто існуючі аналоги, методи та засоби розробки вебплатформи. У другому розділі спроєктовано архітектуру системи та базу даних, реалізовано програмні компоненти серверної та клієнтської частин із використанням геолокаційних сервісів. У третьому розділі проведено експериментальне дослідження ефективності алгоритмів геопошуку та швидкодії інтерфейсу. У висновках узагальнено інформацію, відображену у попередніх частинах.

Ключові слова: вебплатформа, соціальні ініціативи, геолокація, Laravel, React, Leaflet, API, кластеризація.

ABSTRACT

Huzovatyi S. V. Development and Research of a Web Platform to Support Social Initiatives With Geolocation Functions. Manuscript.

Master's qualification work of OP "Software Engineering". Specialties 121 "Software engineering". Lutsk National Technical University. Lutsk, 2025.

The master's thesis consists of an introduction, three chapters, conclusions, and a list of references (according to the structure of the qualification work approved by the department).

The first chapter analyses the current state of the problem of coordinating social initiatives and considers existing analogues, methods, and means of developing a web platform. The second chapter designs the system architecture and database, implements the software components of the server and client parts using geolocation services. The third chapter conducts an experimental study of the effectiveness of geolocation algorithms and interface performance. The conclusions summarize the information presented in the previous sections.

Keywords: web platform, social initiatives, geolocation, Laravel, React, Leaflet, API, clustering.

ЗМІСТ

ВСТУП

Актуальність кваліфікаційної роботи магістра пов'язана з активізацією волонтерського руху та соціальних ініціатив в Україні. В умовах постійних кризових викликів самоорганізація населення трансформувалась з інструменту вирішення локальних проблем у важливий елемент національної безпеки. Стрімка цифровізація суспільних процесів зумовила перехід до рішень, які базуються на вебплатформах чи мобільних додатках для зручності та залученню більшої кількості людей. Проте існуючі на ринку платформи мають суттєві обмеження, адже зосереджені переважно в одній сфері (фінанси, залучення людей). Критичним недоліком є відсутність належної інтеграції геолокаційних функцій, таких як інтерактивна карта, що унеможливлює ефективний просторовий пошук ініціатив.

Мета дослідження – створення програмного засобу на основі сучасних веб-технологій для підвищення ефективності координації та організації соціальних ініціатив.

Об'єкт дослідження – процес інформаційної підтримки та координації соціальних ініціатив за допомогою сучасних веб-технологій.

Предмет дослідження – методи, моделі та програмні засоби розробки вебплатформи для візуалізації та управління ініціативами з використанням геолокаційних сервісів.

Завданням дослідження є:

– провести аналіз особливостей волонтерської діяльності, включно з українським контекстом, та існуючих цифрових рішень для виявлення недоліків поточних платформ і формування вимог для нового ресурсу;

– здійснити порівняльний аналіз архітектурних підходів та сучасних веб-технологій для подальшого вибору оптимального стеку розробки;

– спроєктувати структуру бази даних та схему розмежування прав користувачів для забезпечення надійності та безпеки вебплатформи;

– розробити програмну реалізацію клієнтської та серверної частин

вебплатформи відповідно до сформованих функціональних вимог;

– інтегрувати геолокаційний функціонал з використанням інтерактивної карти для візуалізації подій, кластеризації маркерів та розрахунку відстаней;

– реалізувати адміністративний модуль для управління контентом, модерації подій;

– провести експериментальне дослідження розробленої вебплатформи для оцінки її продуктивності, швидкодії та ефективності алгоритмів обробки даних, зокрема геопросторового пошуку та кластеризації.

Наукова новизна одержаних результатів кваліфікаційної роботи магістра полягає в архітектурному підході до реалізації високонавантажених інтерактивних інтерфейсів у веб-середовищі, який базується на поєднані методів оптимізації структур даних при передачі мережею та динамічного групування об'єктів на стороні клієнта, що забезпечує стабільність роботи системи при візуалізації значної кількості елементів.

Практичне значення та застосування отриманих результатів полягає у створенні повнофункціонального програмного продукту – вебплатформи, яка готова до впровадження у діяльність громадських організацій або волонтерських центрів для координації соціальних ініціатив на локальному рівні.

Апробація результатів кваліфікаційної роботи були сформовані в статті та опубліковані в науковому журналі «Комп'ютерно-інтегровані технології: освіта, наука, виробництво». Луцьк: Луцький НТУ, 2025. Вип. № 59, 61 [1, 2].

РОЗДІЛ 1
АНАЛІЗ ПРОБЛЕМАТИКИ ЗА ТЕМОЮ РОБОТИ ТА ПОСТАНОВКА ЗАВДАНЬ ДОСЛІДЖЕННЯ

1.1 Огляд і аналіз предметної області проблеми, результатів існуючих теоретичних та експериментальних досліджень.

Соціальні ініціативи – процеси, дії, проєкти запущені людьми, волонтерами для того, щоб вирішувати проблеми суспільства або ж покращувати рівень життя жителям своїх громад чи країн. Це можуть бути різні збори коштів, речей для людей, які цього потребують, толоки, культурні та освітні заходи, екологічні акції тощо [3].

Волонтерство та взаємодопомога є основою сталого та здорового громадянського суспільства. Волонтерська діяльність сприяє соціальному розвитку, розвиває навички учасників та закріпляє здатність до співпраці [4]. Важлива роль таких ініціатив настає в кризові періоди. Наприклад, під час пандемій чи стихійних лих, самоорганізація людей є ключовою для підтримки населення, коли державні інституції не встигають реагувати. Саме ініціативи в такі часи виявляються найбільш гнучкими та ефективними, підтримуючи загальну стійкість суспільства.

В українському контексті, особливо після 2014 року та з початком повномасштабного вторгнення в 2022 році, соціальні ініціативи та волонтерство зазнало великих змін. Це уже не просто прояв громадянської позиції та вирішення проблем суспільства, а питання національного спротиву та виживання. Тому волонтерський рух став невід'ємною частиною оборонних зусиль країни [5].

В 2022 році, 56,9 % дорослого населення заходу та центру України були залучені до волонтерської діяльності, є люди які допомагали тільки фізичною працею або ж тільки фінансово. Діяльність українських волонтерів охоплює надзвичайно широкий спектр завдань: від закупівлі зброї, транспорту, проведення зборів коштів, надання гуманітарної допомоги до реалізації

культурних проєктів, спрямованих на зміцнення національної ідентичності та солідарності [5].

Така інтеграція волонтерства в структуру національної безпеки та соціального захисту також змінює вимоги до інструментів, що підтримують соціальні ініціативи. Цифрові платформи такого роду діяльності в Україні перестають бути просто зручними сервісами для громадської активності, тому що від їх ефективності та надійності залежить здатність суспільства оперативно реагувати на екзистенційні загрози. Можна вважати, що це елемент критичної соціальної інфраструктури.

Цифрові інструменти, такі як вебплатформи, мобільні додатки, значно розширюють можливості волонтерської діяльності, тому що роблять громадську активність значно простішою та доступнішою для більшого кола людей, дозволяють поширювати інформацію з більшою швидкістю та масштабом [6].

Цифрові інструменти мають очевидні переваги, але і несуть ризики, які особливо загострюються в умовах війни. Певні групи населення, які не мають доступу до інтернету або необхідних технічних навичок, не зможуть користуватися вебплатформами або мобільними додатками. Також питання кібербезпеки, витік конфіденційної інформації про волонтерів, їхні проєкти, потреби та місця дислокації може становити пряму загрозу життю, в умовах війни. Крім того, соціальні мережі та відкриті платформи можуть використовуватись для поширення дезінформації з метою дискредитації волонтерського руху [6].

Це означає, що для вебплатформи з підтримки соціальних ініціатив, що розробляється для українського контексту, безпека, верифікація користувачів та механізми протидії дезінформації не є додатковими функціями, а вимогами до проєкту.

Для обґрунтування необхідності розробки нової вебплатформи, необхідно провести аналіз існуючих проєктів. Для підтримки соціальних ініціатив існує низка платформ, які можна класифікувати за їх основним призначенням на три категорії: краудфандингові платформи (Crowdfunding Platforms), платформи

координації волонтерів, платформи для локальних спільнот.

Перша категорія – краудфандингові платформи, які орієнтовані на збір коштів для реалізації проєктів або підтримки певних ініціатив. Прикладом цієї категорії є Kickstarter, в українському просторі є платформа «Спільнокошт».

Kickstarter є найпопулярнішою платформою у цій категорії, заснованою у 2009 році. Ідеально підходить для фінансування проєктів з кінцевим продуктом, такі як фільми, ігри, музика і так далі, тому що кошти збираються лише у випадку досягнення заявленої мети до визначеного кінцевого терміну, як це показано на рисунку 1.1 [7, 8]. Така структура є неспроможною для підтримки нематеріальних соціальних ініціатив. Геолокаційні функції на платформі мінімальні і зводяться до фільтрації проєктів за країною походження автора.

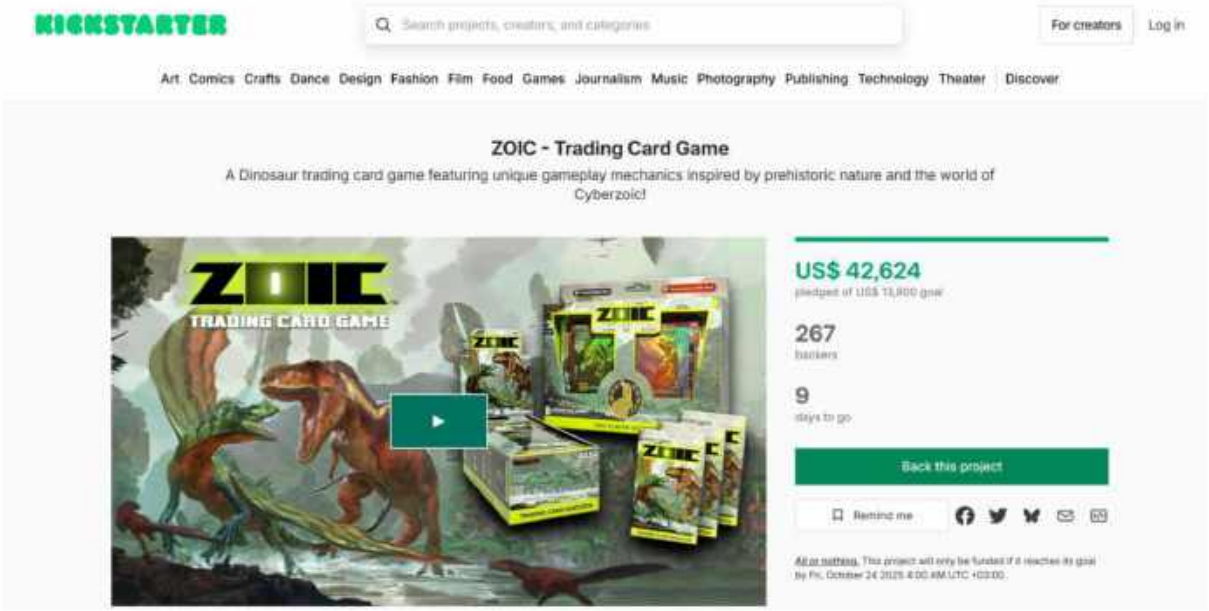

Рисунок 1.1 – Приклад сторінки проєкту на Kickstarter [8]

«Спільнокошт» (на платформі biggggidea) є найбільшою українською платформою у цій категорії, що функціонує з 2012 року [9]. Фокусується на підтримці добре продуманих громадських та освітніх ініціатив, через що має високий рівень довіри в українському суспільстві. Основний функціонал також зосереджений на зборі коштів, що продемонстровано на рисунку 1.2. Геолокаційні функції відсутні.

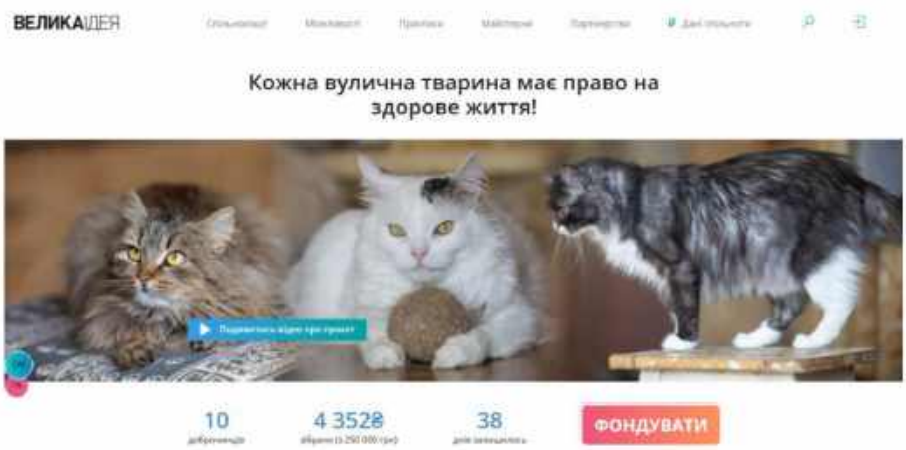

Рисунок 1.2 – Приклад ініціативи на платформі «Спільнокошт» [10]

Друга категорія – платформи координації волонтерів зосереджені на пошук людей, які готові надати свій час та навички, з організаціями та ініціативами, що потребують допомоги. Прикладом цієї категорії є VolunteerMatch, в українському просторі – «Волонтерська Платформа».

VolunteerMatch (на платформі Idealist) функціонує як велика дошка оголошень для волонтерських можливостей, заснована у 1998 році [11]. Неприбуткові організації розміщують свої запити, а потенційні волонтери можуть шукати їх за допомогою фільтрів за місцем розташування, сферою інтересів та необхідними навичками, як це зображено на рисунку 1.3. Геолокаційні функції реалізовані через пошук за містом та радіусом від нього.

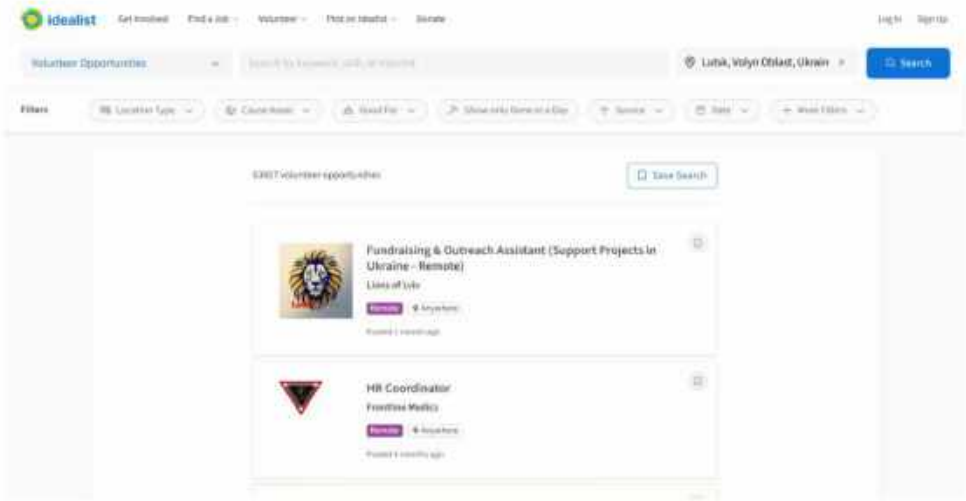

Рисунок 1.3 – Інтерфейс пошуку на VolunteerMatch [12]

«Волонтерська Платформа» є частиною діяльності Української Волонтерської Служби та створена спільно з Дитячим фондом ООН за підтримки Міністерства молоді та спорту [13]. Платформа об'єднує сотні тисяч користувачів та близько тисячі організацій, виконуючи роль дошки оголошень для волонтерів, як це продемонстровано на рисунку 1.4. Геолокаційні функції реалізовані тільки через пошук за містом.

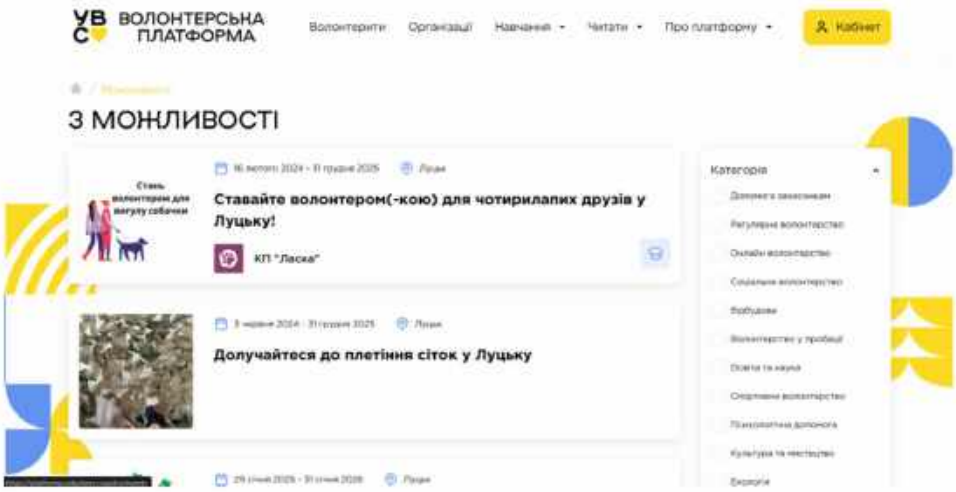

Рисунок 1.4 – Інтерфейс пошуку на «Волонтерська Платформа» [14]

Третя категорія – платформи для локальних спільнот фокусуються на взаємодії людей у межах певного географічного простору або спільноти за інтересами. Прикладом цієї категорії є Meetup та Nextdoor.

Meetup заснована у 2002 році, платформа дозволяє людям знаходити та створювати групи за спільними інтересами від вивчення мов до зустрічей в книжковому клубі [15]. Основна функція – організація реальних зустрічей та подій. Геолокація є ключовим елементом, оскільки дозволяє користувачам шукати події на інтерактивній мапі, але тільки в мобільному додатку, на сайті обмежились пошуком за містом та радіусом від нього.

Nextdoor – це гіперлокальна соціальна мережа, побудована на принципі верифікованих районів. Щоб приєднатися до спільноти, користувач повинен підтвердити свою реальну адресу проживання, що створює унікальне середовище високої довіри між сусідами. Платформа дозволяє обговорювати

місцеві новини, рекомендувати сервіси, продавати або віддавати речі, а також організовувати локальні заходи [16]. Геолокаційні функції є основою Nextdoor – інтерактивна карта сповіщень про безпеку, погоду чи надзвичайні ситуації, можливість додавати геотеги до постів та взаємодіяти з сусідніми районами, як це продемонстрована на рисунку 1.5.

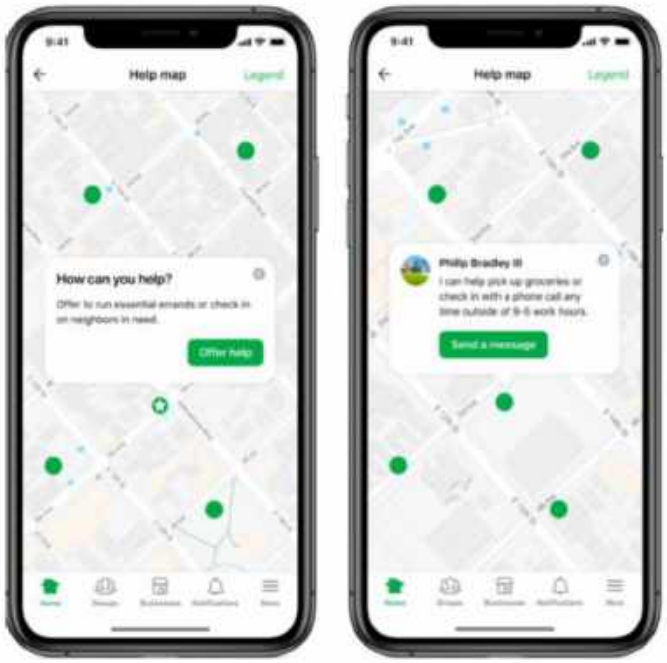

Рисунок 1.5 – Інтерфейс карти на платформі Nextdoor [17]

Проведений аналіз існуючих рішень для підтримки соціальних ініціатив демонструє, що кожна категорія платформ вирішує лише якусь частину завдань. На ринку відсутня платформа, яка б інтегрувала ключові функції всіх категорій, необхідні для комплексного супроводу соціальних ініціатив.

Один з найнеобхідніших інструментів для соціальних ініціатив це геолокаційні функції. Для таких проєктів географічний простір є не просто фоном, а основним організаційним принципом. Ефективність таких ініціатив залежить від можливості швидко визначити, де потрібна допомога.

Тому для розробки платформи варто вивчити досвід не лише соціальних, а й комерційних проєктів, які довели ефективність використання геолокаційних функцій.

Логістика – Bolt, Glovo побудували свої бізнес моделі на алгоритмах геопросторового поєднання клієнтів (пасажири, замовлення) та виконавців (водії, кур'єри). Їхні системи враховують місцезнаходження, маршрути та час оптимального розподілу ресурсів, як це продемонстровано на рисунку 1.6. Цю модель можна адаптувати для створення системи сповіщень про волонтерські можливості в певному радіусі від користувача.

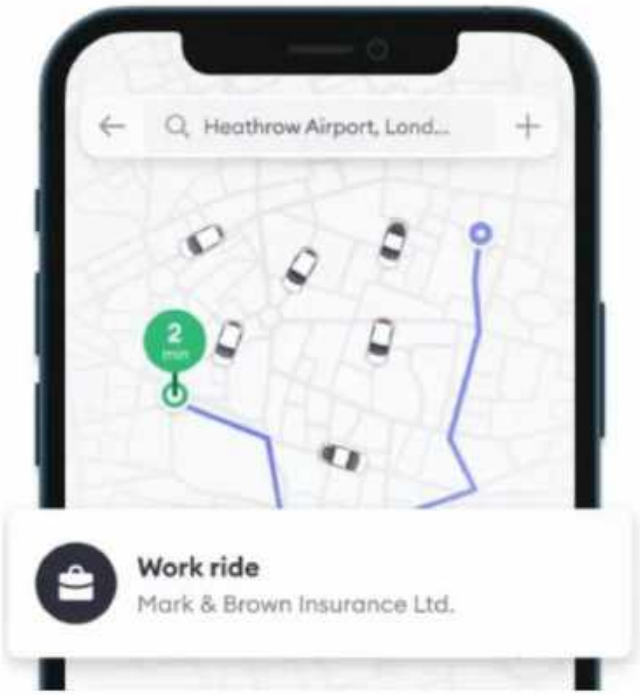

Рисунок 1.6 – Інтерфейс Bolt, що показує водіїв поруч [18]

Airbnb Experience – сервіс, що використовує геолокацію не тільки як інструмент навігації, а як засіб для створення унікального досвіду та розповіді історій. Кожна «Experience» не просто подія, а певна активність, яка проведена місцевим експертом, що дозволяє гостям зануритися в атмосферу та історію міста, як продемонстровано на рисунку 1.7. Карта на сторінці може візуалізувати маршрут екскурсії чи унікальність локації [19]. Подібний підхід можна перенести і до соціальних ініціатив, для прикладу розповідати історію місця, де відбувається подія, показувати фото перед початком та по завершенню події, відгуки учасників.

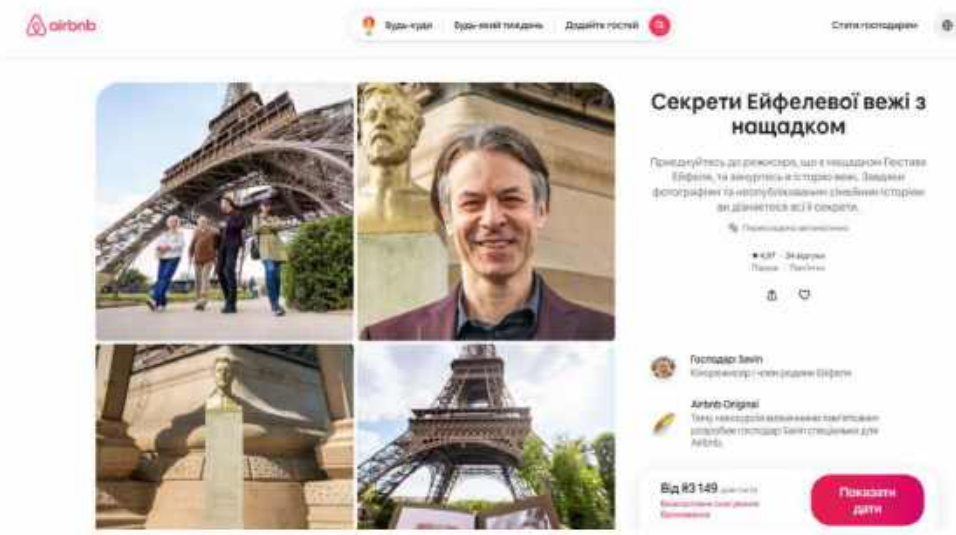

Рисунок 1.7 – Приклад події на Airbnb Experience [20]

Для ефективної підтримки соціальних ініціатив платформа повинна реалізовувати набір специфічних геолокаційних функцій:

– інтерактивна візуалізація на карті, а саме відображення ініціатив (наприклад точки збору допомоги, місця проведення толок) у вигляді позначок на мапі, це надає користувачам миттєвий просторовий контекст. На відміну від текстового списку, карта дозволяє інтуїтивно оцінити щільність ініціатив, їх розподіл по території та відстань до них;

– просторовий пошук поруч з користувачем – можливість фільтрувати та шукати ініціативи в певному радіусі від місцезнаходження користувача є ключовою для стимулювання спонтанного волонтерства;

– гео-таргетовані сповіщення, що інформують користувачів про нові ініціативи або термінові потреби виключно в межах їхнього географічного інтересу (наприклад в місті, де проживає користувач). Це дозволяє уникнути інформаційного перевантаження та доставляти релевантну інформацію тим, хто з найбільшою ймовірністю може відгукнутися.

Після проведеного аналізу предметної області проблеми, огляду існуючих платформ для підтримки соціальних ініціатив та рішень на базі геолокації, на ринку відсутній проєкт, який інтегрував спеціалізований волонтерський функціонал разом з інтуїтивним геолокаційним ядром. Це дозволяє створити

унікальний інструмент, який надасть ефективне рішення для координації локальних ініціатив в Україні.

1.2 Огляд і аналіз методів та засобів розробки вебплатформи для підтримки соціальних ініціатив із геолокаційними функціями.

Після огляду та аналізу предметної області та існуючих рішень, варто зупинитися на аналізуванні методів та засобів для технічної реалізації вебплатформи для підтримки соціальних ініціатив із геолокаційними функціями.

Вибір правильного архітектурного підходу, стеку технологій та інструментів є критично важливим для забезпечення надійності, масштабованості, безпеки та функціональності платформи. Неправильні рішення, прийняті на ранніх етапах проєктування, можуть призвести до значних технічних обмежень, високих витрат на підтримку та неможливості реалізації запланованого функціоналу в майбутньому.

Основа всього проєкту – архітектурна модель, вибір якої визначає, як компоненти системи будуть взаємодіяти між собою, як система буде реагувати на зростання навантаження та як легко буде впроваджувати нові функції в майбутньому. Є два рівні архітектурних рішень – клієнт-серверна модель, що лежить в основі всіх вебзастосунків, та вибір між монолітною та мікросервісною архітектурами, який визначає внутрішню структуру серверної частини.

Клієнт-серверна модель розділяє систему на два основні компоненти:

– клієнт (frontend) – виступає браузером користувача, який відповідає за представлення користувацького інтерфейсу, взаємодію з користувачем та запити до сервера [21];

– сервер (backend) – відповідає за обробку бізнес-логіки, управління даними, автентифікацію користувачів та надання інформації у відповідь на запити клієнта [22].

Зв'язок між клієнтом та сервером здійснюється через мережу за допомогою протоколів HTTP або HTTPS. Клієнт-серверна модель використовується для

переважної більшості сайтів та сервісів.

Для вебплатформи для підтримки соціальних ініціатив, клієнт-серверна модель має наступні переваги:

– всі дані користувачів, інформація про соціальні ініціативи, координати зберігаються та контролюються в одному місці, що значно спрощує адміністрування (оновлення, створення резервних копій);

– серверну частину можна масштабувати шляхом модернізації обладнання або додаванням нових серверів для розподілу навантаження, що дозволяє підтримувати стабільну роботи вебплатформи;

– центральний сервер є єдиною точкою впровадження механізмів захисту, що дозволяє централізовано контролювати доступ та захищати дані від кібератак.

Також клієнт-серверна модель має певні ризики, які теж необхідно враховувати:

– єдина точка відмови – це найбільший недолік, бо вся система залежить від центрального сервера, якщо він з якихось причин виходить з ладу, то вся платформа стає недоступною для користувачів;

– ризик перевантаження сервера під час пікових навантажень, що може значно сповільнити роботу платформи для користувачів.

Вибір між монолітною та мікросервісною архітектурою уже визначає як саме буде організована серверна частина. Це рішення має великий вплив на весь життєвий цикл розробки продукту.

У монолітній архітектурі весь застосунок розробляється, тестується та розгортається як єдина система. Усі його компоненти, наприклад автентифікація користувачів, геолокаційні функції, тісно пов'язані між собою в одній кодовій базі та працюють в єдиному процесі [23]. Переваги монолітної архітектури:

– розробникам простіше додавати новий функціонал, так як вся кодова база знаходиться в одному місці, що дозволяє швидше створити робочу версію продукту;

– всі компоненти взаємодіють в одному середовищі, що дозволяє легше знаходити та виправляти помилки;

– взаємодія між різними компонентами системи відбувається всередині одного процесу, що виключає будь-які затримки.

Недоліки монолітної архітектури:

– у процесі масштабування системи та додавання нового функціоналу кодова база стає об'ємною та складною, що сповільнює подальшу розробку, ускладнює впровадження змін та відстеження помилок;

– помилка в одній частині застосунку може призвести до відмови всієї платформи.

У мікросервісній архітектурі весь застосунок розбивається на набір невеликих, майже не пов'язаних між собою сервісів. Кожен сервіс відповідає за конкретну функцію, наприклад сервіс користувачів, карт, сповіщень, і може бути розроблений, розгорнутий незалежно від інших, комунікуючи між собою через чітко визначені API [24]. Переваги мікросервісної архітектури:

– можливість масштабування лише тих компонентів системи, які зазнають найбільшого навантаження;

– збій в одному компоненті не призведе до відмови всієї платформи, тому інший функціонал залишиться доступним для користувачів;

– кожен компонент системи має невелику кодову базу, тому його простіше зрозуміти, тестувати та оновлювати.

Недоліки мікросервісної архітектури:

– управління десятками незалежних компонентів системи, їх розгортання, моніторинг та логування є набагато складнішим, ніж адміністрування одного монолітного застосунку;

– мікросервіси зазвичай взаємодіють між собою через API, це створює додаткові затримки та ризики мережевих збоїв.

Вибір між backend технологіями визначає продуктивність, безпеку, швидкість розробки та ступінь підтримки бізнес-логіки системи. Є три провідні технологічні екосистеми, кожна з яких пропонує унікальний набір переваг та підходів до розробки: Python з фреймворком Django, PHP з фреймворком Laravel та Node.js з фрейворком Express.js.

Django – фреймворк на мові програмування Python, який заохочує швидку розробку, адже постачається з величезною кількістю вбудованих компонентів, які вирішують більшість поширених завдань веброзробки [25]. Це дозволяє розробникам зосередитись на унікальній бізнес-логіці застосунку, а не на переписуванні базових функцій. Особливості Django:

– ORM (Object-Relational Mapper), що дозволяє розробникам взаємодіяти з базою даних використовуючи об'єкти Python замість написання SQL-запитів;

– після визначення моделі даних, розробник отримує адміністративну панель – це інтерфейс для створення, читання, оновлення та видалення записів у базі даних;

– надає вбудовані механізми для захисту від більшості поширених веб-атак, а саме SQL-ін'єкції, XSS, CSRF.

Laravel – є найпопулярнішим PHP фреймворком, який має виразний синтаксис, що спрощує розробникам написання зручного для підтримки та читабельного коду [26]. Подібно до Django, надає великий набір інструментів для швидкого створення сучасних вебзастосунків. Особливості Laravel:

– Eloquent ORM – подібно до Django, дозволяє легко виконувати складні запити до бази даних за допомогою PHP-об'єктів;

– Artisan CLI та Blade – це інструменти командного рядка для автоматизації одноманітних завдань (створення міграцій, моделей) та шаблонізатор для створення динамічних інтерфейсів користувача;

– надає механізми для захисту від більшості поширених веб-атак, та включає готові функції автентифікації та авторизації користувача.

Node.js – не є фреймворком, а середовищем виконання JavaScript. Його архітектура кардинально відрізняється від традиційних багатопотокових моделей, які використовуються в Python та PHP. Node.js працює в одному потоці та використовує подійно-орієнтовану модель з неблокуючим вводом-виводом.

Особливості Node.js [27]:

– асинхронна, неблокуюча модель вводу-виводу, що забезпечує високу продуктивність для застосунків з великою кількістю одночасних підключень;

– має багато модулів та бібліотек, доступних через NPM (Node Package Manager), що пропонує рішення майже для кожної проблеми.

Вибір між frontend технологіями визначає наскільки інтерактивним, швидким та зручним буде користувацький досвід. Сучасна розробка інтерфейсу користувача використовує компонентний підхід, де все будується з незалежних та повторно використовуваних елементів. Є три ключових рішення – React, Vue, Angular.

React – JavaScript-бібліотека, зосереджується тільки на інтерфейсі користувача і розробники часто добавляють інші бібліотеки для різних функцій, наприклад маршрутизація чи керування станом [28]. Переваги React:

– розробнику надається можливість самостійно обирати необхідні інструменти та бібліотеки для їх інтеграції в проєкт;

– має розгалужену екосистему, що пропонує найбільшу кількість готових рішень та бібліотек для пришвидшення розробки;

– використання технології Virtual DOM (vDOM) забезпечує високу швидкодію та ефективне оновлення інтерфейсу користувача.

Недоліки React:

– не є повноцінним фреймворком, через що потребує багато бібліотек для створення застосунку;

– складність для новачків, оскільки через гнучкість екосистеми розробникам без достатнього досвіду складно обрати оптимальний набір інструментів та бібліотек.

Vue – JavaScript-фреймворк, який легко інтегрувати в існуючі проєкти, Може бути розширеним за допомогою Vue Router (маршрутизація), Vuex (керування станом), тому його функціоналу достатньо для побудови складних односторінкових застосунків (SPA) [29]. Переваги Vue:

– документація, яка є детальною та зрозумілою;

– пропонує офіційні рішення для типових задач;

– висока продуктивність, як і React, також використовує vDOM;

– вважається найпростішим для вивчення серед інших frontend рішень.

Недоліки Vue:

– менша екосистема – менше готових рішень;

– менш популярний серед великих та складних застосунків.

Angular – також є фреймворком, який розроблений та підтримуваний Google, для створення динамічних односторінкових застосунків за допомогою JavaScript та TypeScript [30]. На відміну від React та Vue, уже включає важливі інструменти як маршрутизація та керування станом. Переваги Angular:

– чітка архітектура робить проєкти більш передбачуваними та легшими для підтримки великою командою;

– використання TypeScript робить код надійнішим та зменшує кількість помилок;

– через свою структуру ідеально підходить для масштабних корпоративних застосунків.

Недоліки Angular:

– найскладніший для вивчення через велику кількість концепцій (строга типізація, впровадження залежностей, модулі, спостерігачі);

– обмежена гнучкість архітектури через строгу стандартизацію, що ускладнює інтеграцію нестандартних рішень;

– реалізація навіть простих функцій потребує написання більшого обсягу коду порівняно з іншими frontend рішеннями.

Вибір між системами керування базами даних (СКБД) визначає не лише інструмент зберігання даних (для вебплатформи це користувачі та ініціативи), але й ефективність їх обробки. Проведено огляд двох реляційних систем: MySQL та PostgreSQL, які можуть працювати з геопросторовими даними.

PostgreSQL є об’єктно-реляційною СКБД, що означає підтримку складних типів даних, такі як масиви, JSON, XML, а також дозволяє створювати власні об’єкти з наслідуванням. Це робить модель даних в PostgreSQL більш гнучкою, тоді як MySQL лишається простою реляційною СКБД.

Розширення PostGIS для PostgreSQL додає підтримку стандартних географічних об’єктів (точки, лінії, полігони тощо), а також багато функцій

просторового аналізу, включаючи вимірювання відстаней та площ, буферизацію, кластеризацію тощо [31]. Тому з цим розширенням, база даних перетворюється на повноцінну геоінформаційну систему.

MySQL надає набір вбудованих функцій (MySQL Spatial), що включає підтримку стандартних географічних об'єктів та дозволяє виконувати розрахунок відстані між об'єктами [32]. Хоча цей функціонал значно поступається розширенню PostGIS, але цього достатньо для реалізації базових геолокаційних завдань.

Головна перевага використання спеціальних геолокаційних функцій в обох СКБД полякає в просторовому індексуванні. Звичайні індекси в базі даних ефективні для сортування чисел чи тексту, але не підходять для двовимірних географічних даних. Просторові індекси (R-tree в MySQL або GiST в PostgreSQL) організовують дані так, щоб швидко відповідати на геолокаційні запити, наприклад знайти всі події в певному радіусі, пошук стає в кілька разів ефективнішим.

Вибір рішення для візуалізації інтерактивної карти представляє геопросторові дані, від його швидкості, інтерактивності та візуальної привабливості залежить зручність використання основного функціоналу платформи, а це пошук та взаємодія з ініціативами на мапі. Є три найпопулярніші рішення в сфері картографічних сервісів: комерційні платформи Google Maps та Mapbox, open-source бібліотека Leaflet.js з даними від OpenStreetMap.

Google Maps – набір інструментів та сервісів Google для роботи з геоданими, надає розробниками доступ до високоякісних карт, а також багато функцій для реалізації геолокаційних завдань (геокодинг, прокладання маршруту, тощо).

Особливості та нюанси Google Maps:

– працює за моделлю «pay-as-you-go», ціна виставляється за кожну 1000 запитів, зі зростанням кількості користувачів будуть і рости витрати;

– дані, які надає Google Maps є найповнішими тай найактуальнішими, проте на збереження цих даних накладаються суворі обмеження;

– створити повністю унікальний дизайн мапи не вийде, можна працювати тільки в рамках візуального стилю Google;

– для базових геолокаційних завдань, інтеграція дуже проста та добре задокументована.

Mapbox – платформа для розробки унікальних карт, надає розробниками контроль над візуальним стилем та даними, які використовуються з OpenStreetMap. Також даючи великий спектр функцій для реалізації різноманітних геолокаційних завдань.

Особливості та нюанси Mapbox:

– працює за моделлю «pay-as-you-go», також витрати будуть рости із зростанням кількості користувачів;

– дані базуються на OpenStreetMap, тобто в деяких регіонах буде поступатися точністю і якістю даним Google Maps. Зберігати отриману інформацію можна за платні запити;

– за допомогою інструментарію Mapbox, можна створити унікальний дизайн інтерактивної карти;

– використовує векторні тайли, що робить карти дуже швидкими та плавними, тобто краща продуктивність.

Leaflet.js – невелика open-source бібліотека, яка призначена тільки для відображення мапи OpenStreetMap та її елементів.

Особливості та нюанси Leaflet.js:

– повністю безкоштовне рішення з відкритим кодом;

– надає функціонал тільки для відображення мапи, якщо потрібно більше, то розробник повинен додавати сторонні плагіни або сервіси (наприклад Nominatim для геокодингу);

– для кастомізації потрібно працювати з CSS та плагінами.

Проведений огляд та аналіз засобів розробки вебплатформи для підтримки соціальних ініціатив з геолокаційними функціями демонструє, що існує широкий спектр технологічних рішень, кожне з яких має свої переваги, недоліки та особливості, попри те, що розглянута невелика частка технологій.

1.3 Постановка завдання на кваліфікаційну роботу магістра

На основі аналізу предметної області, соціальні ініціативи сприяють соціальному розвитку та забезпечують стійкість суспільства, а створення вебплатформи для їх підтримки – слугуватиме цифровим інструментом, який значно спрощує взаємодію між людьми, які потребують та надають допомогу. Для ефективної координації волонтерів потрібно створити ресурс, який інтегруватиме ключові переваги існуючих категорій платформ для ініціатив та враховувати український контекст. У свою чергу, аналіз засобів та методів розробки підтверджує можливість технічної реалізації такого рішення завдяки широкому спектру сучасних веб-технологій.

Тому вебплатформа для підтримки соціальних ініціатив з геолокаційними функціями повинна створюватися навколо наступних елементів:

– інтерактивна карта, має стати основою сайту. Використання картографічного сервісу для візуалізації подій з можливістю визначення власної геолокації та розрахунку відстані до події або ініціативи;

– система облікових записів – реєстрація та авторизація користувачів, наявність особистого кабінету з можливістю редагування профілю, перегляду інформації, налаштування приватних полів (email, телефон тощо);

– управління подіями – функціонал для створення подій із зазначенням необхідної інформації, завантаженням зображення та вибором локації безпосередньо на інтерактивній мапі;

– адміністративна панель – окремий блок сторінок для адміністратора із виведенням статистики, управлінням скаргами, подіями, користувачами;

– інформативність та UX – зручне відображення деталей подій та взаємодія з ними через «sidebar» без перезавантаження сторінки.

Після визначення елементів майбутньої вебплатформи, сформовано бачення необхідного обсягу робіт, тому можна скласти перелік цілей, які повинні бути завершені під час цієї роботи.

Перелік цілей, які розкривають суть завдання і повинні бути досягнені в

результаті виконання кваліфікаційної роботи магістра:

– провести аналіз особливостей волонтерської діяльності, включно з українським контекстом, та існуючих цифрових рішень для виявлення недоліків поточних платформ і формування вимог для нового ресурсу;

– здійснити порівняльний аналіз архітектурних підходів та сучасних веб-технологій для подальшого вибору оптимального стеку розробки;

– спроєктувати структуру бази даних та схему розмежування прав користувачів для забезпечення надійності та безпеки вебплатформи;

– розробити програмну реалізацію клієнтської та серверної частин вебплатформи відповідно до сформованих функціональних вимог;

– інтегрувати геолокаційний функціонал з використанням інтерактивної карти для візуалізації подій, кластеризації маркерів та розрахунку відстаней;

– реалізувати адміністративний модуль для управління контентом, модерації подій;

– провести експериментальне дослідження розробленої вебплатформи для оцінки її продуктивності, швидкодії та ефективності алгоритмів обробки даних, зокрема геопросторового пошуку та кластеризації.

РОЗДІЛ 2
ТЕОРЕТИЧНЕ ДОСЛІДЖЕННЯ ТА ПРАКТИЧНА РЕАЛІЗАЦІЯ ВЕБПЛАТФОРМИ ДЛЯ ПІДТРИМКИ СОЦІАЛЬНИХ ІНІЦІАТИВ

2.1 Обґрунтування вибору шляхів, технологій, алгоритмів і засобів вирішення поставленого завдання

На основі проведеного аналізу предметної області та існуючих рішень, було визначено, що для розробки вебплатформи необхідне рішення, яке забезпечить високу інтерактивність, масштабованість та надійну роботу з геопросторовими даними. Для цього обрано архітектуру односторінкового застосунку (SPA) на базі розділеної клієнт-серверної моделі. Архітектура вебплатформи складатиметься з двох незалежних проєктів (frontend та backend), які будуть «спілкуватися» між собою через API.

Стек технологій, що використовується в процесі розробки, наведено в таблиці 2.1.

Таблиця 2.1 – Обраний стек технологій

Компонент	Технологія	Аргументація вибору
Серверна частина (backend)	PHP з фреймворком Laravel	Обрано завдяки високій швидкості розробки, наявності потужних інструментів роботи з БД, вбудованим механізмам безпеки та авторизації. Буде виконувати роль API-сервера, надаючи кінцеві точки для роботи з даними
Клієнтська частина (frontend)	React з інструментом збірки Vite	Забезпечує високу інтерактивність, необхідну для динамічної карти та складних компонентів. Розділення на frontend та backend дозволяє незалежно масштабувати компоненти, а Vite гарантує швидке локальне розгортання та збірку
База даних (СКБД)	MySQL	Обрано як надійну, високопродуктивну реляційну СКБД. Для роботи з геопросторовими даними будуть використовуватись вбудовані просторові функції та індекси
Картографічний сервіс	Leaflet.js з OpenStreetMap (OSM)	Обрано через простоту та гнучкість кастомізації. Використання OSM дозволяє уникнути комерційних обмежень та витрат

Основою функціоналу платформи є здатність швидко і точно знаходити ініціативи для користувача. Для вирішення цього завдання розроблено алгоритм фільтрації, який поєднує семантичні параметри події (категорія, статус, часові рамки, в певному радіусі від користувача).

Алгоритм фільтрації реалізовано на рівні побудови SQL-запиту за допомогою Laravel Eloquent. Це дозволяє динамічно додавати умови до запиту залежно від параметрів, які обрав користувач.

Для фільтрації подій, які знаходяться в певному радіусі від користувача, використовується вбудована функція MySQL («ST_Distance_Sphere») для розрахунку відстані між користувачем та ініціативою, вона враховує сферичну форму Землі та працює зі збереженими координатами події.

Реалізація динамічного фільтра ініціатив наведено в лістингу 2.1, як фрагмент одного з параметрів фільтрації, а саме пошуком подій в певному радіусі.

Лістинг 2.1 – Реалізація динамічного фільтра (фрагмент)

```
// Додавання умови гео-пошуку лише за наявності координат користувача у
запиті
$query->when(request()->filled('coords'), function ($q) {
    [$lat, $lng] = explode(',', request('coords'));

    // Розрахунок відстані (в км) за сферичною моделлю Землі
    $q->selectRaw(
        'ST_Distance_Sphere(point(longitude, latitude), point(?, ?)) /
1000 as distance, events.*',
        [$lng, $lat]);

    // Фільтрація за радіусом (якщо вказано параметр radius)
    $q->when(request()->filled('radius') && request('radius') !== 'all',
function ($q_radius) use ($lat, $lng) {
        $radius = (int) request('radius') * 1000; // Конвертація км в метри
        $q_radius->whereRaw(
            'ST_Distance_Sphere(point(longitude, latitude), point(?, ?))
<= ?',
            [$lng, $lat, $radius]);});});});

//Аналогічно реалізовано динамічні фільтри за статусом, категоріями та
часом
```

Кінець лістингу 2.1

Для покращення користувацького досвіду при відображенні великої кількості ініціатив на карті використовується кластеризація маркерів, для цього застосовується бібліотека «Leaflet.markercluster». Цей алгоритм динамічно групує близько розташовані маркери в єдиний кластер (маркер з числом). При масштабуванні карти кластери розпадаються на окремі маркери. Цей підхід забезпечує оптимізацію продуктивності (зменшує кількість елементів) та покращує сприйняття карти.

Враховуючи обрану архітектуру SPA через API, для забезпечення безпечного обміну даними між frontend та backend використовується аутентифікація на основі Laravel Sanctum. Для розділених застосунків Sanctum використовує аутентифікацію на основі «cookie» файлів та сесій користувача (Cookie based session), постачається з механізмом XSRF-токенів для захисту від міжсайтової підробки запитів (CSRF).

Послідовність алгоритму аутентифікації реалізована наступним чином:

– ініціалізація – клієнт (React) викликає кінцеву точку «/sanctum/csrf-cookie» для отримання CSRF-токену та встановлення сесійної «cookie». HTTP-клієнт Axios налаштований для роботи з «cookie», його наведено в лістингу 2.2;

– вхід – користувач надсилає свої облікові дані на «/api/login». Laravel перевіряє дані, якщо вони правильні, то створює сесію та встановлює спеціальну «cookie»;

– авторизація – кожен наступний запит, що надходить від React до Laravel, буде містити ту «cookie», за якою автоматично буде перевірятись сесія користувача та токен.

Лістинг 2.2 – Налаштований Axios

```
import axiosLib from 'axios'
const axios = axiosLib.create({
  baseURL: import.meta.env.VITE_BACKEND_URL, //Посилання на backend
  withCredentials: true, }) //Дозволяє надсилати кукі та XSRF-токен
export default axios
```

Кінець лістингу 2.2

Фрагмент контролера, який відповідає за аутентифікацію, наведено в лістингу 2.3, як функції для «/api/login» та «/api/logout».

Лістинг 2.3 – Фрагмент контролера аутентифікації

```
    public function login(UserLoginRequest $request): JsonResponse
    {
        // Спроба автентифікації за email та паролем
        if (!Auth::attempt($request->only('email', 'password'))) {
            return response()->json(['errors' => ['email' => ['Невірні
дані']]], 401);
        }

        // Регенерація ID сесії для захисту від фіксації сесії
        $request->session()->regenerate();

        return response()->json(['user' => new UserResource(auth()-
>user())], 200);
    }

    public function logout(Request $request): JsonResponse
    {
        // Інвалідація сесії та токена при виході
        $request->session()->invalidate();
        $request->session()->regenerateToken();

        return response()->json(['message' => "Logged out"], 200);
    }
```

Кінець лістингу 2.3

Для забезпечення передбачуваності структури та оптимізації передачі даних на клієнтському рівні використовується ресурсно-орієнтований підхід та стандартизація JSON-відповідей за допомогою Laravel Resources.

Важливою особливістю реалізації цього є фільтрація вихідних даних, тобто на клієнт передається не повна модель об'єкта з усіма системними полями, а лише необхідний набір даних. Це зменшує розмір навантаження, для прикладу для візуалізації подій використовуються тільки дані, які беруть участь в фільтрації та координати ініціативи, такий ресурс наведено в лістингу 2.4.

За замовчування Laravel Resources повертає колекції у форматі «{data:[…]}». Це забезпечує передбачуваність при роботі з колекціями на клієнті.

Лістинг 2.4 – Ресурс з необхідним набор даних для ініціатив

```
class EventSmallResource extends JsonResource
{
    public function toArray(Request $request): array
    {
        return [
            'id' => $this->id,
            'title' => $this->title,
            'latitude' => $this->latitude,
            'longitude' => $this->longitude,
            'status' => $this->status,
               'categories' => CategoryResource::collection($this-
>whenLoaded('categories')),
            $this->mergeWhen(isset($this->distance), [
                'distance' => round($this->distance, 2)
            ]),
        ];
    }
}
```

Кінець лістингу 2.4

Для забезпечення конфіденційності та розмежування повноважень використовується метод «Role-Based Access Control» (RBAC), реалізований через зв'язок таблиць користувачів та ролей і механізм Laravel Policies.

Вебплатформа підтримує ієрархію ролей:

– гість (неавторизований користувач), який має доступ до перегляду ініціатив на мапі та профілів інших користувачів;

– учасник (тільки зареєстрований користувач), який може брати участь в ініціативах, залишати коментарі, подавати скарги на контент та редагувати власний профіль;

– організатор – до можливостей учасника додається створення, редагування власних ініціатив;

– адміністратор, який має повний доступ до всіх даних, модерації, управління користувачами, подіями та скаргами.

Кожна дія, яка відправляється на backend (наприклад редагування профіля) перевіряється через свою Policy. Дія виконується, якщо перевірка успішна, якщо ні, то повертається HTTP-помилка «403». Для прикладу, реалізація Policy для

оновлення профілю наведено в лістингу 2.5, де відбувається перевірка чи користувач володіє цим профілем або має роль адміністратора.

Лістинг 2.5 – Policy для профіля користувача

```
class ProfilePolicy
{
    public function update(User $user, Profile $profile): bool
    {
        return $user->id === $profile->user_id ||
          $user->roles()->where('name', User::ROLE_ADMIN)->exists();
    }
}
```

Кінець лістингу 2.5

2.2 Практична реалізація об'єкта проєктування

Практична реалізація вебплатформи розпочинається з проєктування архітектури даних, оскільки база даних є фундаментом роботи інформаційної системи. Цей етап включає визначення сутностей, встановлення зв'язків між ними та вибір оптимальних типів даних для зберігання інформації. Подальша розробка базується на створеній базі даних та включає реалізацію програмних компонентів серверної та клієнтської частин відповідно до обраного технологічного стеку (Laravel, React).

Було спроєктовано ER-діаграму, її наведено в рисунку 2.1, яка відображає структуру бази даних, що складається з 11 взаємопов'язаних сутностей:

– «users» – містить лише критично важливі дані для входу (email, пароль, ім'я) та системні поля (час створення, час оновлення). Для зберігання розширеної інформації про користувача використовується окрема таблиця «profiles», яка пов'язана зв'язком один-до-одного;

– «profiles» – містить додаткову інформацію про користувача (зображення, телефон, соціальні мережі, інтереси та налаштування профілю);

– «roles» та «user_role» – містять ролі та прив'язку ролей до користувачів через зв'язок багато-до-багатьох;

– «events» – містить повну інформацію про ініціативу (назва, зображення, опис, статус, дата і час початку та кінця). Координати зберігаються в окремих полях «latitude» та «longitude». Поле «location» використовується для зберігання структурованої текстової адреси (область, місто, вулиця, будинок);

– «categories» та «category_event» – забезпечують категоризацію ініціатив через зв'язок багато-до-багатьох, що дозволяє одній ініціативі належати до кількох категорій;

– «event_user» – забезпечує фіксацію участі користувачів у ініціативах, реалізуючи зв'язок багато-до-багатьох між подіями та користувачами;

– «comments» та «complaints» – реалізовані з використанням поліморфних зв'язків. Поля «commentable» та «complaintable» дозволяють прив'язувати коментарі та скарги до кількох сутностей системи (ініціатива, профіль користувача);

– «role_upgrade_requests» – відповідає за модерацію запитів користувачів на отримання статусу організатора.

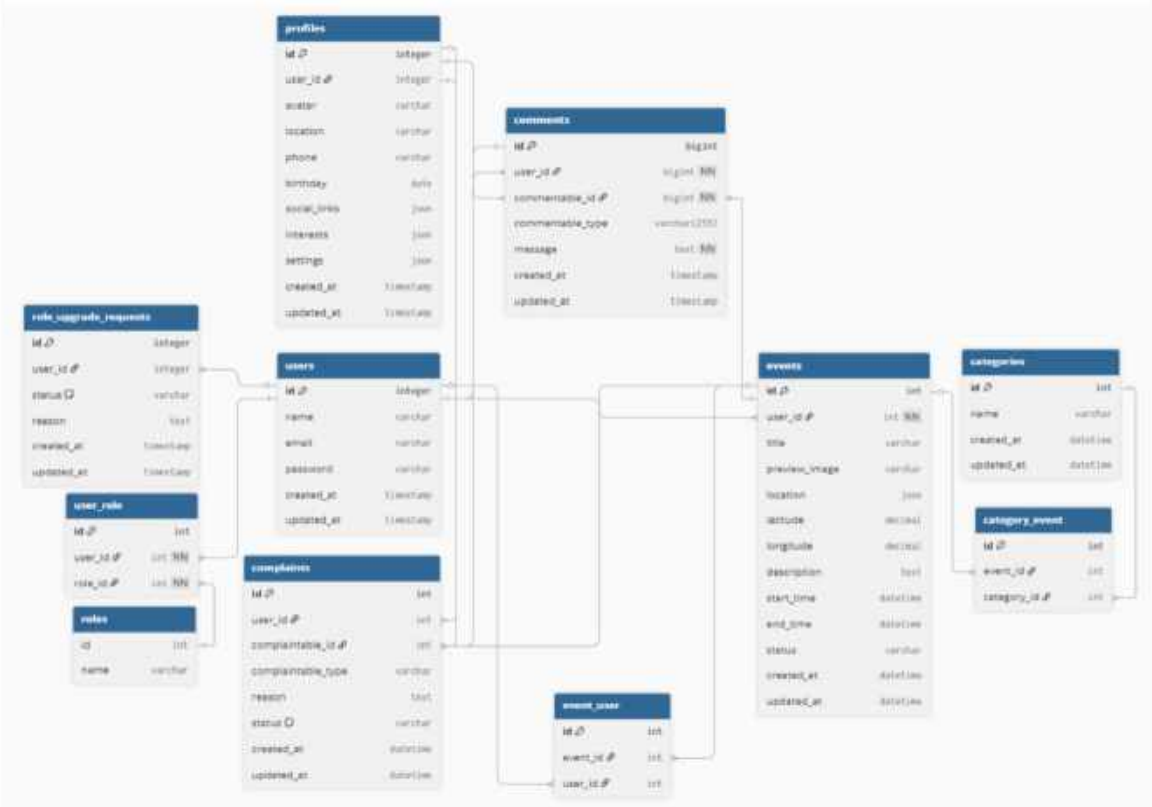

Рисунок 2.1 – ER-діаграма бази даних

Проєктування та створення фізичної структури бази даних реалізовано за допомогою механізму міграцій Laravel (Laravel Migrations). Це дозволяє описувати схему даних безпосередньо в коді, що забезпечує контроль версій структури БД, легкість розгортання на нових серверах та можливість відкату змін.

Для демонстрації логіки навігації та роботи системи захисту доступу розроблено схему розмежування прав доступу до функціональних компонентів вебплатформи, її зображено на рисунку 2.2. Вона відображає залежність доступності сторінок та елементів інтерфейсу від ролі користувача та прав володіння об'єктами.

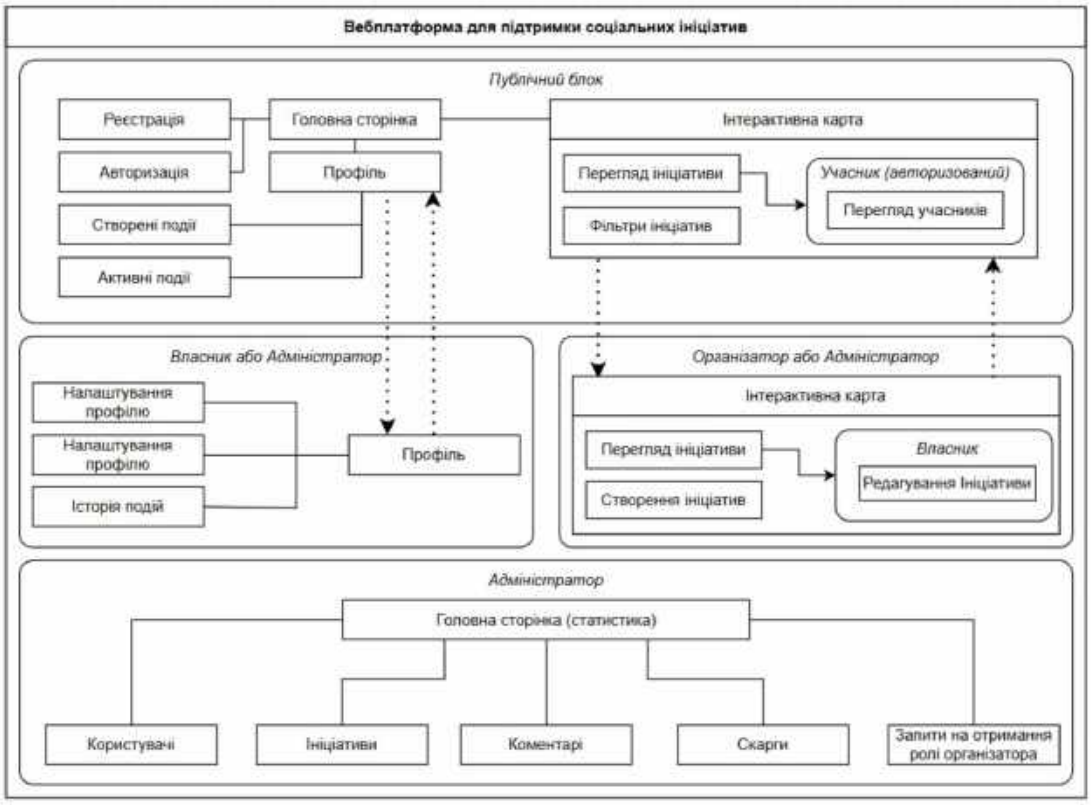

Рисунок 2.2 – Схема вебплатформи

Так як використовується розділена клієнт-серверна архітектура, було встановлено, налаштовано та розгорнуто два проєкта власне клієнтська частина (React) та серверна (Laravel) разом з докер-контейнером для БД. Першим етапом взаємодії користувача з вебплатформою є головна сторінка.

Основою структури клієнстького інтерфейсу є створений компонент-шаблон «MainLayout», який забезпечує єдиний стиль та навігацію для всіх сторінок, де «header» та «footer» є статичними елементами, а основний контент динамічно змінюється залежно від сторінки.

Компонент «header» виступає ключовим елементом навігації. Його логіка реалізована з використанням умовного рендерингу, що дозволяє адаптувати інтерфейс під різні випадки:

– стан користувача – якщо неавторизований – відображаються посилання на логін та реєстрацію, якщо авторизований – події та профіль користувача;

– контекст сторінки – якщо користувач знаходиться поза головною сторінкою, то з'являється кнопка «Назад». На сторінці з інтерактивною картою, з'являється кнопка виклику фільтрів ініціатив.

Візуалізація різних станів «header» наведено на рисунку 2.3.

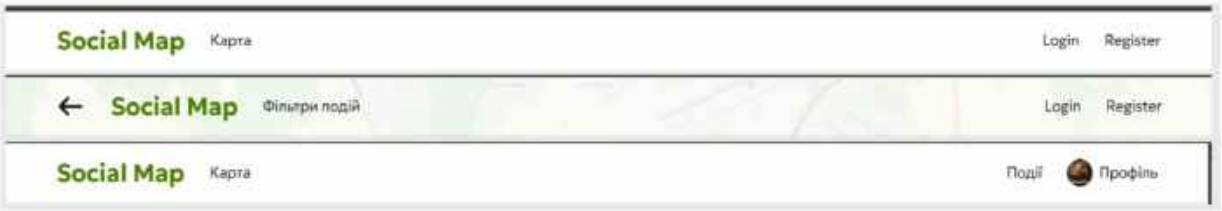

Рисунок 2.3 – Стани компонента «header»

Компонент «footer», зображений на рисунку 2.4, виконує інформаційну та правову функції. Він містить обов'язкові атрибуції використаних зовнішніх сервісів.

Рисунок 2.4 – Компонент «footer»

Головна сторінка виконує презентаційну функцію та більше слугує переходом до інтерактивної карти. Структура сторінки складається з трьох секцій.

Перша секція містить заголовок з коротким описом, основна мета – заклик до дії, що наведено на рисунку 2.5.

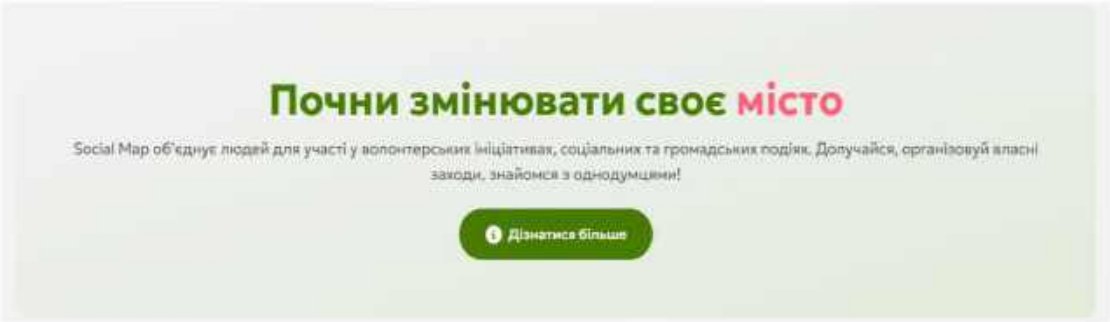

Рисунок 2.5 – Перша секція головної сторінки

Друга секція фокусує увагу користувача на ключовій функції системи, а саме інтерактивній карті. Вона містить статичне зображення та кнопку, яка перенаправляє користувача на «/map», де ініціалізується компонент з картою. Друга секція наведена на рисунку 2.6.

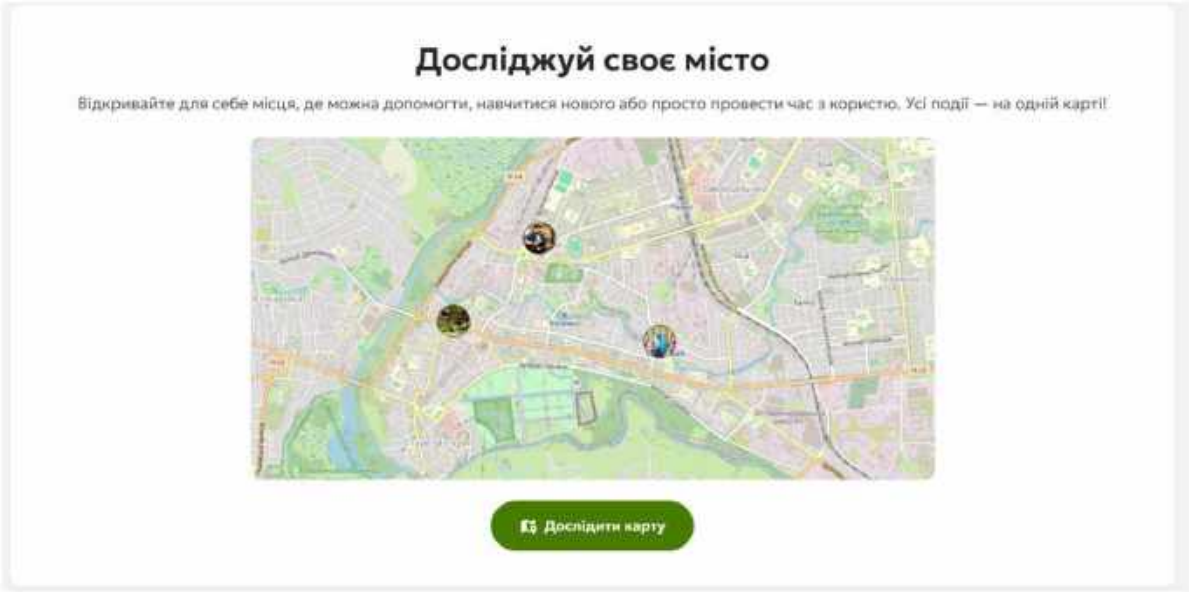

Рисунок 2.6 – Друга секція головної сторінки

Третя секція головної сторінки – блок динамічного контенту, що відображає картки трьох останніх ініціатив. Використовується «useEffect», щоб при завантаженні сторінки отримати з БД події для наповнення за кінцевою

точкою «/api/events/latest», наведено в лістингу 2.6. В контролері, що відповідає за події (EventController), написана функція, яка відповідає за передачу останніх подій.

Лістинг 2.6 – Отримання останніх ініціатив на клієнті

```
import { useEffect, useState } from "react"
import axios from "@plugin/axios"
export default function Home() {
    const [events, setEvents] = useState([])
    useEffect(() => {
        const getEvents = async () => {
            await axios.get('/api/events/latest').then(res => {
                setEvents(res.data.data)
            })
        }
        getEvents()
    }, [])
return (...)//Головна сторінка
```

Кінець лістингу 2.6

Вигляд третьої секції, з уже отриманими останніми подіями, наведено на рисунку 2.7.

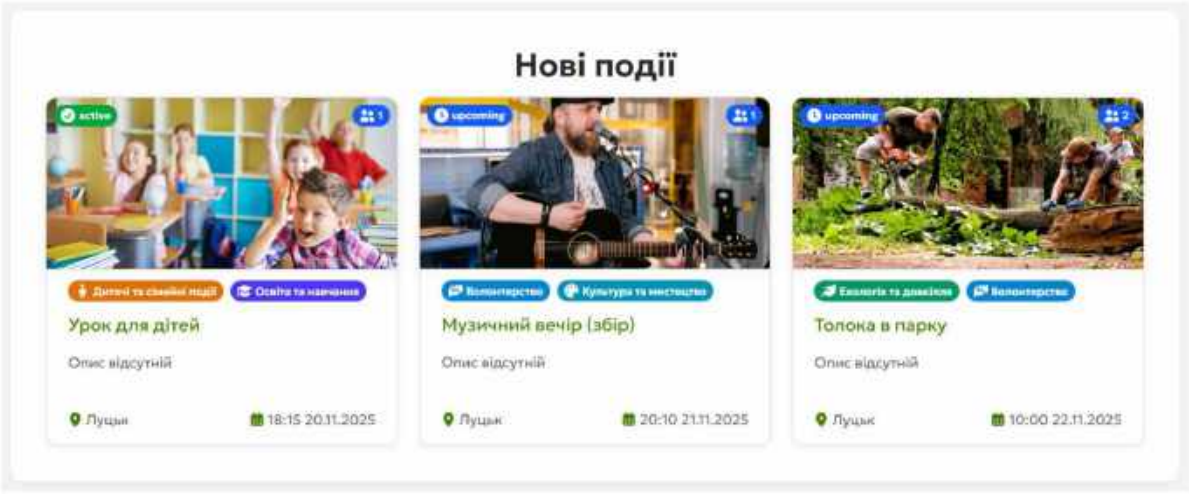

Рисунок 2.7 – Третя секція головної сторінки

Картка події була створена як окремий компонент «EventCard», щоб уникнути дублювання коду, бо використовується на інших сторінках. Для

візуальної ідентифікації стану та категорії ініціатив, цей елемент реалізовано як динамічну стилізацію. Колір та іконки бібліотеки «FontAwesome» винесено в окремий файл з константами, який імпортується в основний компонент та динамічно змінює елемент в залежності від назви статусу чи категорії. Фрагмент реалізації відображення статусу наведено в лістингу 2.7.

Лістинг 2.7 – Реалізація візуалізації статусів

```
// У файлі з константами
export const statusColors = {
    "upcoming": "bg-blue-600",
    "active": "bg-green-600",
    "finished": "bg-gray-600",
    "canceled": "bg-red-600"
};

export const statusIcons = {
    "upcoming": "fa-clock",
    "active": "fa-check-circle",
    "finished": "fa-flag-checkered",
    "canceled": "fa-times-circle"
};
// У компоненті EventCard
//...
<span className={`status-badge ${statusColors[event.status]}`}>
    <FontAwesomeIcon icon={statusIcons[event.status]} className="icon" />
    {event.status}
</span>
//...
```

Кінець лістингу 2.7

Очікуваною дією нового користувача, після відвідування головної сторінки, є перехід до інтерактивної карти, яка знаходиться за маршрутом «/map». Її реалізація базується на бібліотеці «leaftet», яка забезпечує візуалізацію даних OpenStreetMap. Використовується «useEffect», щоб при першому завантаженні сторінки отримати актуальний список подій за адресою «/api/events». Після отримання ініціатив, вони передаються до компонента з мапою, для подальшої візуалізації об'єктів.

Для покращення інформативності інтерфейсу, стандартні маркери бібліотеки «leaflet» було замінено на кастомізований компонент, який

реалізується функцією «createEventDivIcon». Вона генерує HTML-розмітку маркера, вбудовуючи в неї зображення ініціативи. Додано обробник події «click», який визначає активну подію і виконує центрування карти на обраній точці. Фрагмент реалізації створення маркерів наведено в лістингу 2.8.

Лістинг 2.8 – Створення маркерів

```
const createEventDivIcon = (event) => {
       const imageUrl = event.preview_image;
       const iconHtml = `<img src="${imageUrl}" alt="${event.title}" />`;

        return L.divIcon({
            className: 'custom-event-marker-container',
            html: iconHtml,
        });
    }
useEffect(() => {
     if (mapRef.current) {
            mapRef.current._eventMarkersMap = {};
            const markers = events.map(event => {
                const customIcon = createEventDivIcon(event);
                const marker = L.marker([event.latitude,
event.longitude], { icon: customIcon })
                if (event.distance) {
                    marker.bindPopup(`Відстань: ${event.distance} км`);
                }
                marker.on('click', () => {
                    onSelectEvent(event);
                    mapRef.current.setView([parseFloat(event.latitude),
parseFloat(event.longitude) + 0.0025], 17);
                });
                mapRef.current._eventMarkersMap[event.id] = marker;
                return marker
            });
        }
    }, [events, onSelectEvent]);
```

Кінець лістингу 2.8

Для оптимізації відображення великої кількості об’єктів, до карти додано шар кластеризації за допомогою бібліотеки «leaflet.markercluster». Після ініціалізації маркерів вони автоматично об’єднуються в групи залежно від масштабу карти. Вигляд сторінки інтерактивної мапи з отриманими ініціативами та роботою алгоритму кластеризації зображено на рисунку 2.8.

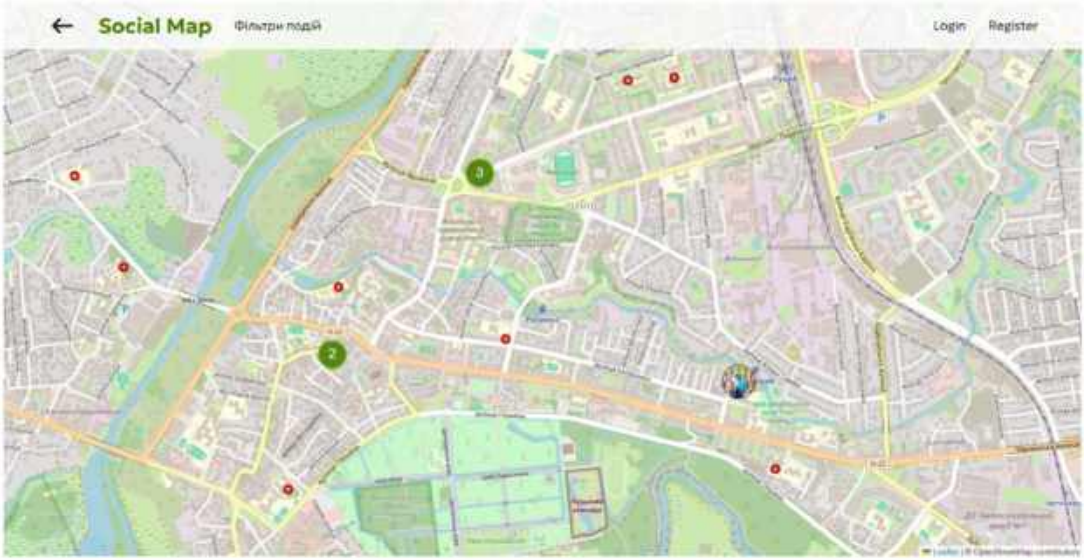

Рисунок 2.8 – Сторінка інтерактивної мапи

Для забезпечення безшовної взаємодії користувача з ініціативами, перегляд детальної інформації про подію реалізовано у вигляді бічної панелі (Sidebar). Таке рішення дозволяє ознайомлюватись з контентом, не втрачаючи візуального контакту з картою та уникнути перезавантаження сторінки.

Логіка роботи бічної панелі базується на управлінні локальним станом («useState»), для цього створено змінні стану, які відповідають за видимість компонента та режим відображення (деталі події, редагування, фільтрація). Завдяки використанню механізму умовного рендерингу, наповнення панелі змінюється динамічно.

При кліку на маркер ініціативи викликається функція, яка встановлює видимість бічної панелі, тип контенту, синхронізує стан інтерфейсу з URL-адресою браузера, що дозволяє користувачам ділитися посиланням на конкретну подію. Також ініціює асинхронний запит до API для отримання додаткової інформації про подію. Реалізація функції вибору активної події наведено в лістингу 2.9.

Для обробки прямих переходів за посиланням використано хук «useEffect». При завантаженні сторінки, «URLSearchParams» аналізує адресу, і якщо там присутній ідентифікатор ініціативи, то автоматично відкривається бічна панель з відповідним контентом.

Лістинг 2.9 – Функція вибору активної події

```
const handleSelectEvent = useCallback((event) => {
        // Робота з параметрами URL
        const params = new URLSearchParams(location.search);
        // Встановлення ID події
        params.set('event', event.id);
        // Оновлення URL без перезавантаження сторінки
        navigate({ search: params.toString() }, { replace: true });
       // Оновлення локального стану інтерфейсу
        setShowSidebar(true);
        setSidebarMode('view');
       // Завантаження детальних даних події
        getEvent(event.id);
    }, [navigate, getEvent]);
```

Кінець лістингу 2.9

Після кліку на маркер, відкривається бічна панель з повною інформацією про ініціатив, її зображено на рисунку 2.9.

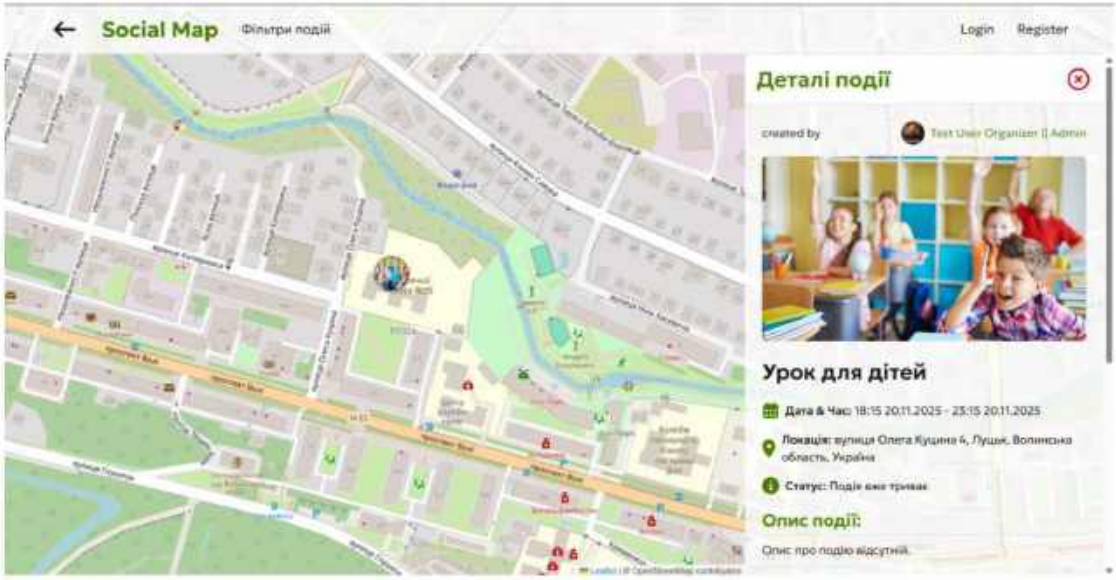

Рисунок 2.9 – Відкрита бічна панель з ініціативою

Інтерфейс фільтрації реалізовано через набір полів вводу («Input»). Для параметрів «статус» та «категорія» використано компонент випадаючого списку («Select») з підтримкою вибору кількох позицій. При зміні значень фільтрів спрацьовує хук «useEffect», він ініціює асинхронний запит «/api/events», передаючи обрані критерії через параметри запиту. Сервер обробляє запит та

повертає на клієнт відфільтрований масив ініціатив. Інтерфейс панелі фільтрів зображено на рисунку 2.10.

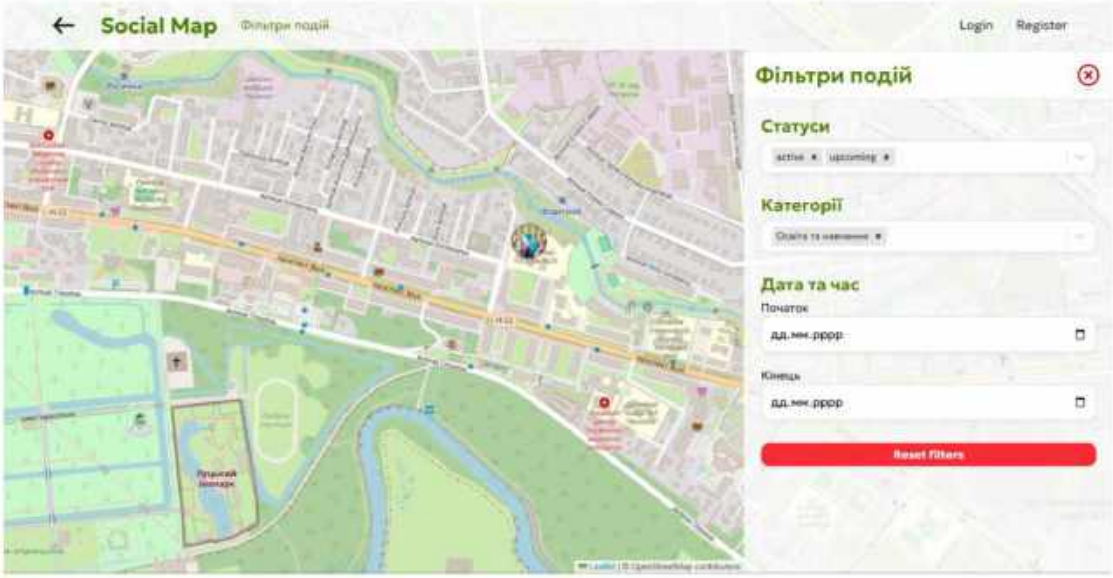

Рисунок 2.10 – Фільтри ініціатив

Після того як новий користувач ознайомиться з інтерактивною картою, для подальших дій потрібно авторизуватися. Тому розроблено окремі форми входу та реєстрації, їх зображено на рисунку 2.11.

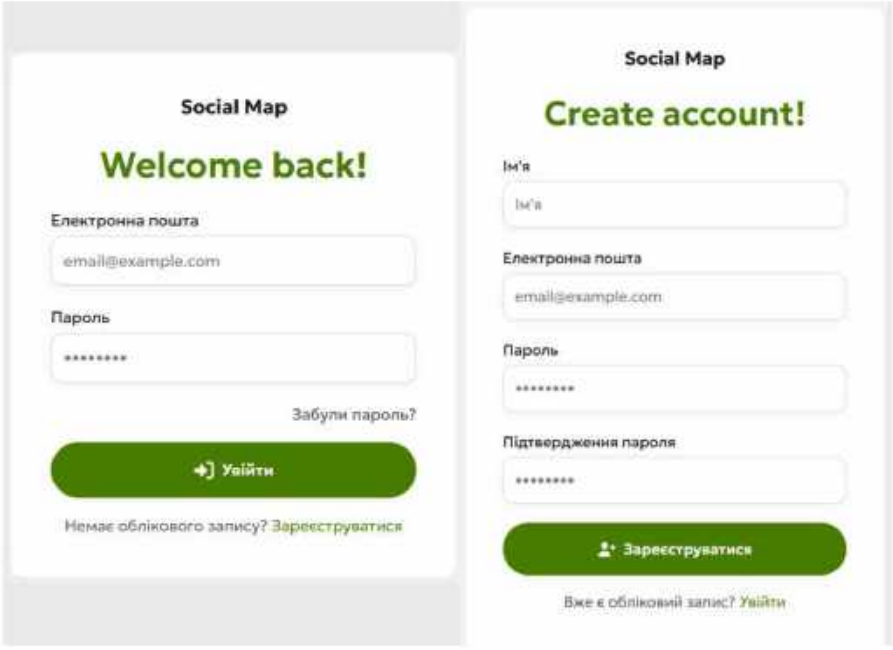

Рисунок 2.11 – Форми входу та реєстрації

Ключовим елементом клієнтської архітектури є управління глобальним станом авторизації. Інформація про поточного користувача повинна бути доступна для всіх компонентів системи, для вирішення цього завдання реалізовано провайдер, що використовує React Context. Він зберігає об'єкт користувача, стан завантаження, а також надає методи для виходу з акаунта та оновлення даних. Реалізацію цього провайдера наведено в лістингу 2.10.

Лістинг 2.10 – Провайдер контексту авторизації

```
export default function AppProvider({ children }) {

    const { user: initialUser, loading: initialLoading } = useAuth();
    const [user, setUser] = useState(initialUser);
    const [loading, setLoading] = useState(initialLoading);

    const refreshUser = async () => {
        const res = await axios.get('/user');
        setUser(res.data.data);
    }

    const logout = async () => {
        await axios.post('/logout');
        setUser(null);
    }

    useEffect(() => {
        setUser(initialUser);
        setLoading(initialLoading);
    }, [initialUser, initialLoading]);

    return (
       <AppContext.Provider value={{user, loading, refreshUser, logout}}>
           {children}
       </AppContext.Provider>
    )
}
```

Кінець лістингу 2.10

Після успішної авторизації, користувач отримує доступ до особистого профілю. Візуально інтерфейс цієї сторінки розділено на три секції.

Перша секція – ідентифікаційний блок, зображений на рисунку 2.12. Тут відображається зображення, ім'я користувача, а також набір кнопок, що слугують переходом до сторінок редагування, налаштування та виходу з вебплатформи.

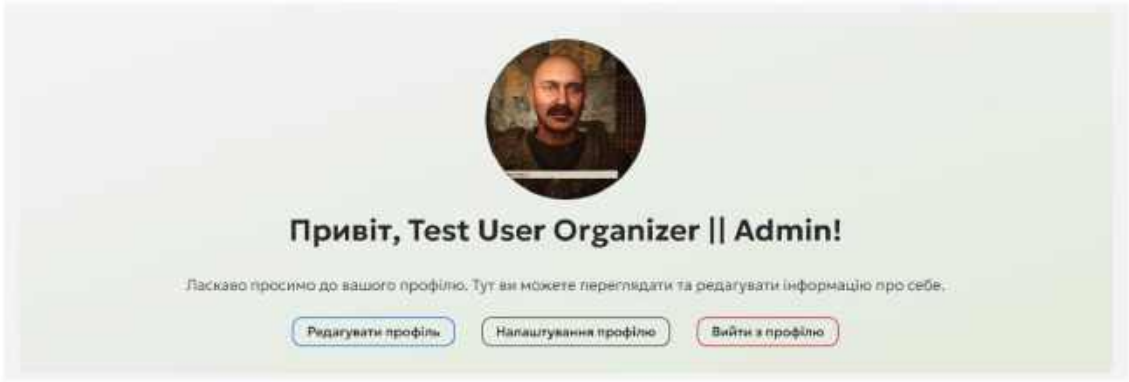

Рисунок 2.12 – Перша секція профілю

Друга секція – інформаційний блок, зображений на рисунку 2.13. Тут міститься інформація про користувача, перелік соцмереж та інтересів. Також є динамічна картка з активністю (дані обчислюються на сервері) та посилання на списки ініціатив: створених користувачем та тих, у яких він бере участь.

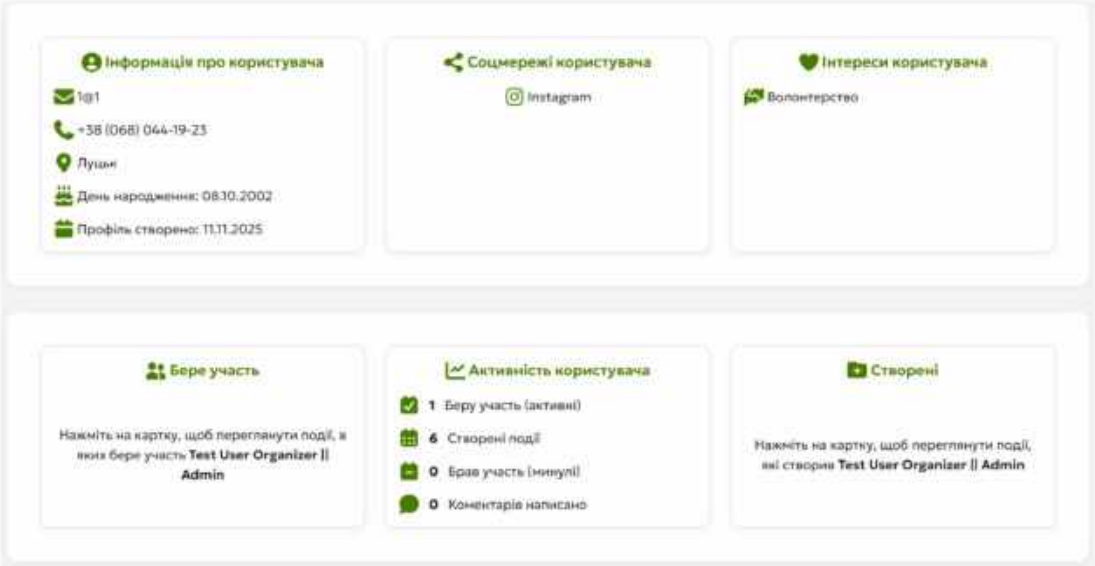

Рисунок 2.13 – Друга секція профілю

Третя секція – блок коментарів, наведено на рисунку 2.14. Цей компонент побудований на основі поліморфних зв'язків, що дозволяє використовувати його як на сторінці профілю, так і в бічній панелі подій. Для оптимізації роботи з великою кількістю коментарів, реалізовано кастомний хук «useInfiniteScroll», що забезпечує динамічне завантаження нових об'єктів при прокручуванні сторінки.

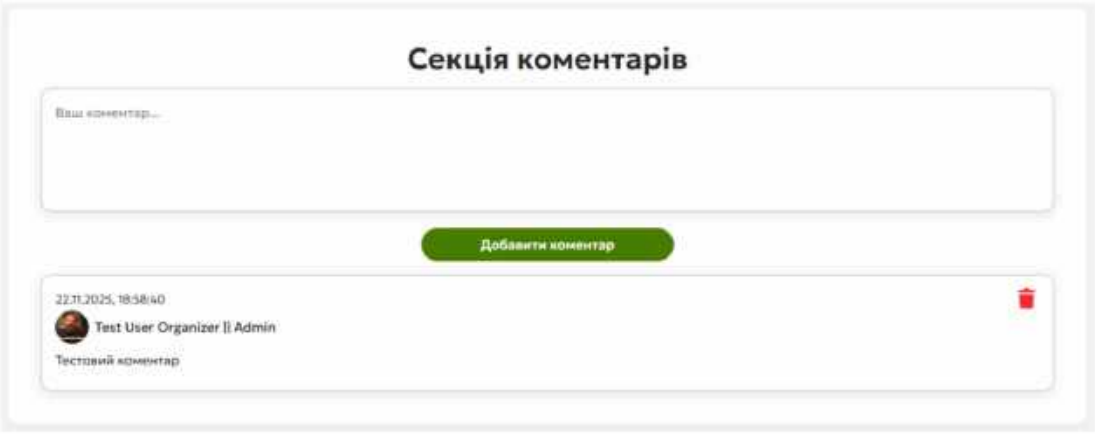

Рисунок 2.14 – Третя секція профілю

Редагування облікового запису відбувається на окремій сторінці за маршрутом «/profile/id/edit», доступ сюди сурово обмежено через «ProtectedRoutes» на клієнті та Policy на сервері – право внесення змін мають виключно власник профілю або адміністратор. Інтерфейс редагування реалізовано через форму. Особливу увагу приділено механізму оновлення аватара користувача, замість стандартного поля, розроблено інтерактивне поле з «drag-and-drop». Інтерфейс сторінки редагування профілю наведено на рисунку 2.15.

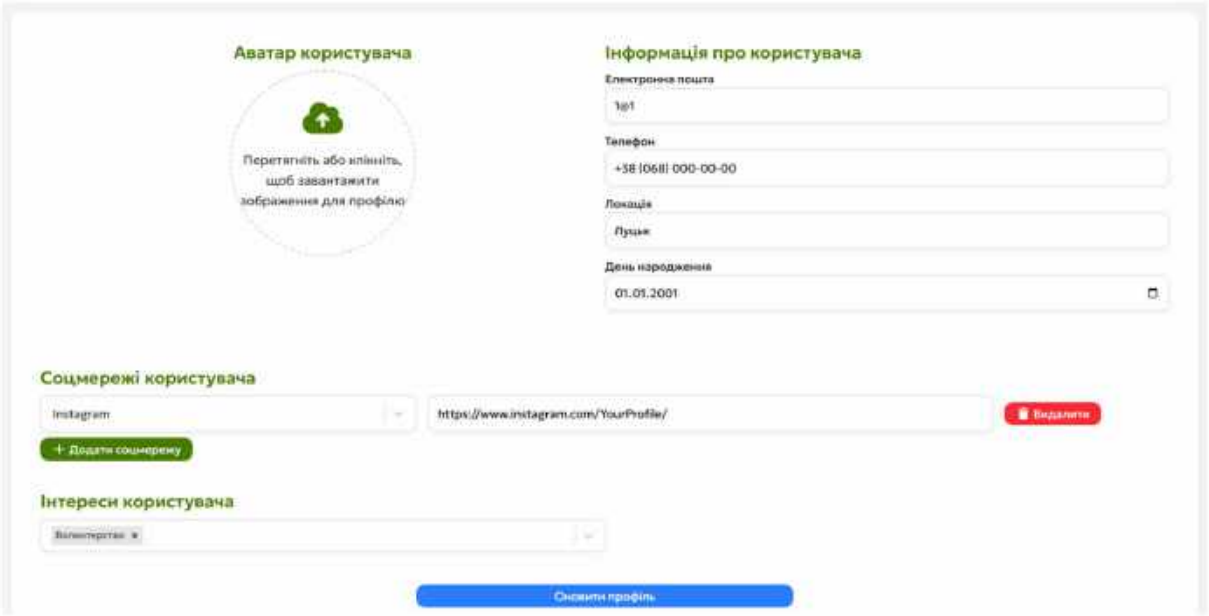

Рисунок 2.15 – Інтерфейс сторінки редагування профілю

В налаштуваннях профілю, за маршрутом «/profile/id/settings», можна вибрати які поля показувати іншим користувачам. Реалізовано через форму з полями типу «checkbox». Інтерфейс налаштувань зображено на рисунку 2.16.

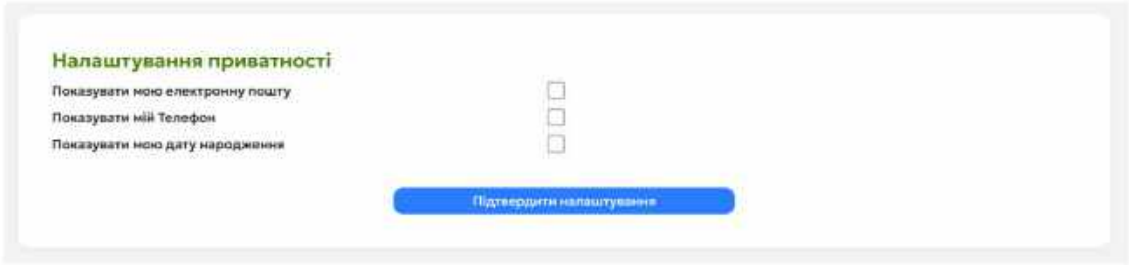

Рисунок 2.16 – Інтерфейс налаштування профілю

Збережені налаштування записуються в БД у форматі JSON. Логіка приватності даних реалізована на рівні API-ресурсу («ProfileResource»). Перед формуванням відповіді, сервер перевіряє права доступу: якщо запит робить власник профілю, то повертаються всі дані, а якщо сторонній користувач – застосовуються збережені налаштування видимості. Фрагмент коду, що реалізує перевірку налаштувань видимості, наведено в лістингу 2.11.

Лістинг 2.11 – Перевірка прихованості інформації

```
    public function toArray(Request $request): array
    {
        $isCreator = $request->user() && $request->user()->can('update',
$this->resource);
        $defaults = ['is_email_public' => true, ...];
        $settings = array_merge($defaults, $this->settings ?? []);
        $userLoaded = $this->relationLoaded('user');

        return [
           ... // Інші дані
            'user' => ['email' =>
$this->when($userLoaded && ($isCreator || $settings['is_email_public']),
                    $this->user->email),],
            'settings' => $this->when(
                $isCreator,
                $this->settings),
        ];
    }
```

Кінець лістингу 2.11

Щоб завершити блок облікового запису, реалізовано три однотипні сторінки, які включають:

– сторінка «CreatedEvents» – містить події, які створив користувач;

– «HistoryEvents» – містить всі події в яких користувач брав участь, це ініціативи зі статусом «finished» та «canceled». До цієї сторінки має доступ тільки власник;

– «ParticipatingEvents» – це зворотна сторінка до історії подій, користувач бере участь (ініціативи зі статусом «upcoming» та «active»).

В кожної сторінки своя кінцева точка з функцією, де відфільтровується список подій, реалізацію наведено в лістингу 2.12, на прикладі створених користувачем подій.

Лістинг 2.12 – Функція «CreatedEvents»

```
public function CreatedEvents(Profile $profile)
    {
        $user = $profile->user;
        $events = $user->events()->with('categories')
            ->orderBy('created_at', 'desc')
            ->paginate(6);

        return EventResource::collection($events);
    }
```

Кінець лістингу 2.12

А інтерфейс сторінки зі створеними ініціативами користувача зображено на рисунку 2.17 (без «header» та «footer»).

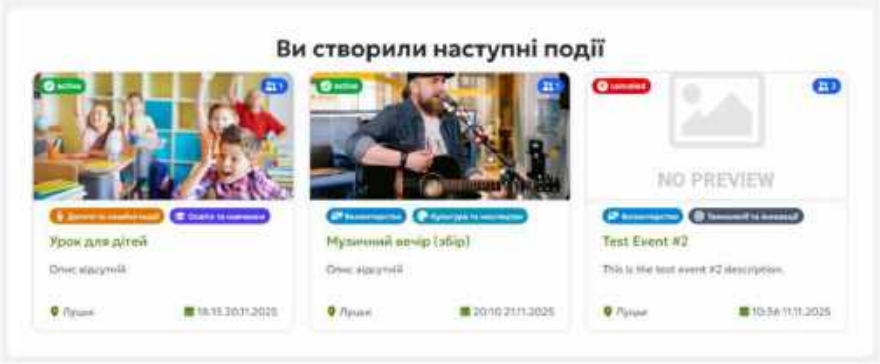

Рисунок 2.17 – Сторінка зі створеними подіями

Для авторизованого користувача додається функціонал і на інтерактивній мапі, а саме взаємодія з ініціативами. Тепер можна долучитися до події, переглянути учасників та залишати коментарі, це все зображено на рисунку 2.18.

Рисунок 2.18 – Додаткова взаємодія авторизованого користувача

Якщо користувач отримав роль «Організатор», він може створювати нові ініціативи. Реалізована форма в бічній панелі на карті, яка використовується як для створення, так і редагування. Інтерфейс форми створення події наведено на рисунку 2.19.

Рисунок 2.19 – Створення події

Для забезпечення контролю за контентом та управління обліковими записами розроблено модуль адміністратора. Доступ до цих сторінок суворо обмежений і реалізований на двох рівнях захисту: серверному та клієнтському.

На стороні сервера використано механізм Laravel Gates. Визначено правило «is_admin», яке перевіряє наявність відповідної ролі у користувача перед виконанням будь-якого запиту до адміністративних маршрутів API. Реалізацію наведено в лістингу 2.13.

Лістинг 2.13 – Перевірка доступу до адміністративних маршрутів

```
Gate::define('is_admin', function ($user) {
    // Перевірка наявності ролі 'admin' у зв'язаній таблиці
    return $user->roles()->where('name', User::ROLE_ADMIN)->exists();
});
```

Кінець лістингу 2.13

На стороні клієнта захист реалізовано на рівні маршрутизації. Для цього створено спеціальний компонент, який перевіряє наявність ролі адміністратора перед рендерингом дочірніх маршрутів. Якщо користувач без відповідної ролі намагається перейти за адресою «/admin/*», система автоматично перенаправляє його на головну сторінку. Фрагмент реалізації захищеного маршруту наведено в лістингу 2.14.

Лістинг 2.14 – Реалізація захищеного маршруту для «/admin/*»

```
export default function AdminRoute({ children }) {
    const { user, loading } = useContext(AppContext);
    const location = useLocation();
    if (loading) {
        return <div className="page"><Loader /></div>;
    }

    if (!user.roles.some(role => role === 'admin')) {
       return <Navigate to="/" state={{ from: location }} replace />;
    }

    return children;
}
```

Кінець лістингу 2.14

Якщо користувач має роль «Адміністратор» і переходить за маршрутом «/admin», рендериться головна сторінка панелі керування, яка виконує функцію моніторингу та виводить ключові статистичні показники системи. Інтерфейс цієї сторінки наведено на рисунку 2.20.

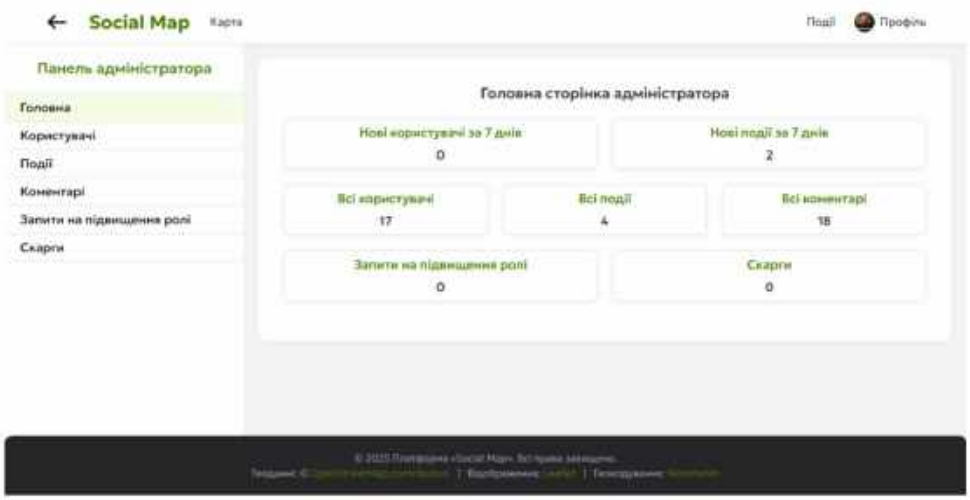

Рисунок 2.20 – Головна сторінка панелі адміністратора

Для управління сутностями системи розроблено уніфіковані табличні інтерфейси. Для прикладу, сторінка управління подіями дозволяє адміністратору переглядати список ініціатив, здійснювати пошук за назвою, фільтрувати за статусом. Доступні інструменти модерації – редагування або видалення подій. Інтерфейс таблиці управління подіями зображено на рисунку 2.21.

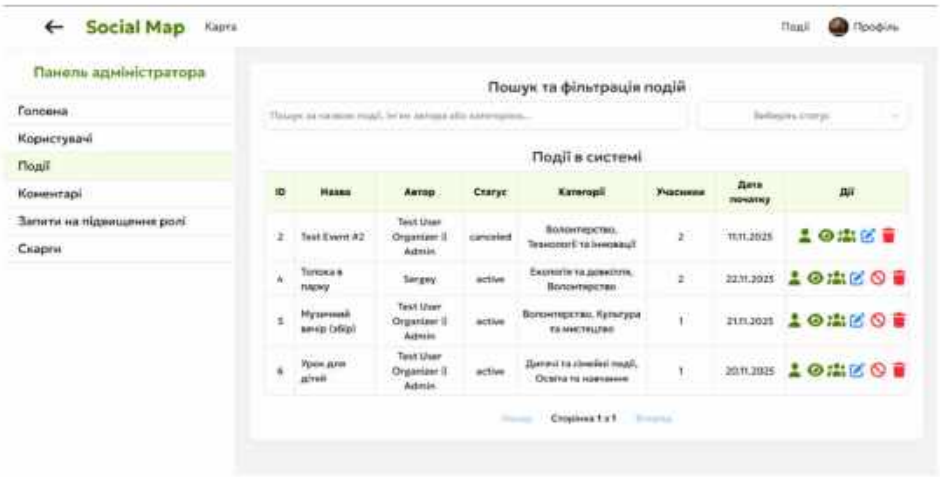

Рисунок 2.21 – Інтерфейс управління подіями

РОЗДІЛ 3
ЕКСПЕРЕМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ РЕЗУЛЬТАТИВНОСТІ ВЕБПЛАТФОРМИ ДЛЯ ПІДТРИМКИ СОЦІАЛЬНИХ ІНІЦІАТИВ

3.1 Методика проведення дослідження

Метою експериментального дослідження є комплексна перевірка результативності функціонування розробленої вебплатформи, оцінка продуктивності обраних архітектурних рішень та їх впливу на швидкодію системи під навантаженням.

Враховуючи, що система реалізована як односторінковий застосунок (SPA) із інтерактивною картою, дослідження проводитиметься у двох площинах:

– серверна частина, де оцінюється ефективність алгоритмів обробки даних та формування відповідей API;

– клієнтська частина, де оцінюється швидкость відображення графічних елементів та оптимізація мережевого трафіку.

Для фіксації метрик продуктивності обрано наступний набір програмних засобів:

– Postman використовується для виконання HTTP-запитів до API, вимірювання часу відповіді сервера та аналізу розміру отриманих пакетів даних;

– вбудовані інструменти розробника в браузері – для аналізу завантаження ресурсів та часу виконання JavaScript.

Методика дослідження включає проведення трьох ключових експериментів, спрямованих на підтвердження ефективності розробленого функціоналу.

Порівняльний аналіз методів геопросторового пошуку. Метою є обґрунтування доцільності використання вбудованої функції СКБД «ST_Distance_Sphere» для фільтрації подій за радіусом. Методика проведення базується на вимірюванні часу виконання SQL-запиту, спрямованого на вибірку об’єктів у фіксованому радіусі. У якості об’єктів порівняння обрано два підходи: запит із використанням вбудованої геопросторової функції СКБД та

альтернативний варіант із застосуванням тригонометричної формули Гаверсинуса реалізованої безпосередньо на рівні SQL. Для забезпечення достовірності результатів та оцінки масштабованості рішення, тестування проводитиметься на трьох згенерованих наборах даних різного об'єму, що містять 500, 1000, 5000, 10000 та 50000 записів ініціатив.

Дослідження ефективності оптимізації обміну даними. Метою є оцінка впливу застосування спеціалізованих API-ресурсів на обсяг переданих даних та швидкість завантаження клієнта. Методика проведення передбачає виконання HTTP-запитів на отримання списку ініціатив із подальшою фіксацією розміру отриманої JSON відповіді. У якості об'єктів порівняння розглядаються передача повної моделі даних, що включає всі поля таблиці, та використання оптимізованого API-ресурсу, який повертає лише критично важливі атрибути. Експеримент проводитиметься на фіксованій вибірці зі 100 об'єктів для визначення коефіцієнту стиснення трафіку.

Дослідження продуктивності відображення інтерактивної карти. Метою є перевірка ефективності алгоритму кластеризації маркерів для забезпечення плавності інтерфейсу. Методика базується на вимірюванні навантаження на клієнтський браузер під час рендеренгу масиву з 500 ініціатив. Порівняльний аналіз проводитиметься між режимом прямого відображення кожного маркера як окремого DOM-елемента та режимом з кластеризацією, що динамічно групує сусідні об'єкти. Критеріями оцінки виступатимуть суб'єктивна плавність сторінки та час виконання скриптів візуалізації.

Експериментальні дослідження проводитимуться у локальному середовищі розробки, що використовувалося для створення вебплатформи. Такий підхід забезпечує відтворюваність результатів та мінімізує вплив сторонніх факторів, що можуть спотворювати час виконання запитів або поведінку клієнтської частини.

Отримані результати дозволять зробити висновок про якість оптимізації вебплатформи та її здатність працювати під навантаженням.

3.2 Обробка та аналіз отриманих результатів

Першим етапом дослідження стало порівняння швидкодії SQL-запитів для фільтрації подій у радіусі 5 кілометрів від центру мапи. Тестування проводилося на 5 наборах даних різного обсягу шляхом виконання 100 ітерацій запиту для кожного випадку з розрахунком середнього часу відповіді. Результати наведено в таблиці 3.1.

Таблиця 3.1 – Порівняння швидкості виконання алгоритмів геопошуку

Кількість записів (ініціативи)	ST_Distance_Sphere (мс)	Формула Гаверсинуса (мс)
500	16	46
1000	19	20
5000	47	77
10000	80	85
50000	370	398

Отримані дані демонструють перевагу використання вбудованої функції СКБД над математичними розрахунками SQL. При використанні «ST_Distance_Sphere» система показує стабільні результати, при збільшенні навантаження час виконання прогнозовано зростає, проте вбудований метод залишається ефективнішим. Хоча у відносному вимірюванні при великих обсягах даних різниця складає близько 7%, використання «ST_Distance_Sphere» є архітектурно доцільним рішенням, оскільки забезпечує кращу читабильність коду та переносить обчислювальне навантаження на алгоритми СКБД.

Другим етапом було дослідження впливу використання оптимізованих ресурсів Laravel на зменшення обсягу переданих даних. Порівнювався розмір JSON-відповіді при запиті зі 100 ініціатив. Результати наведено в таблиці 3.2.

Таблиця 3.2 – Ефективність оптимізації структури API-відповіді

Тип моделі даних	Середній розмір об’єкта	Загальний розмір відповіді	Час завантаження (мс)
Повна модель	1,49 КБ	148,57 КБ	47
Оптимізований ресурс	0,37 КБ	36,59 КБ	29

Як свідчать отримані дані, використання фільтрації полів на рівні API дозволило зменшити розмір відповіді у 4 рази. Також зафіксовано зменшення часу обробки та передачі запиту.

Така оптимізація є критично важливою для подальшого масштабування вебплатформи. Мінімізація обсягу трафіку суттєво розвантажує пропускну здатність, що дозволяє серверу обслуговувати більшу кількість одночасних користувачів без зниження продуктивності.

Третім етапом стала перевірка плавності роботи інтерфейсу (FPS – кадрів за секунду) при відображенні великої кількості маркерів. Порівнювалися режими з вимкненою та увімкненою кластеризацією для набору з 500 ініціатив. Результати наведено в таблиці 3.3.

Таблиця 3.3 – Показники продуктивності рендерингу карти

Режим відображення	Кількість DOM-елементів	FPS
Без кластеризації	500	117-150
З кластеризацією	39	156-165

Використання бібліотеки кластеризації дозволило скоротити кількість DOM-елементів на інтерактивній мапі з 500 до 39, тобто майже у 13 разів. Хоча потужна система, яка використовувалась для розробки і тестування, забезпечила високий рівень FPS в обох випадках, режим «без кластеризації» продемонстрував нестабільність кадрової частоти. Зменшення кількості DOM-елементів є важливим показником для пристроїв, де обробка 500 окремих маркерів призвела б до падіння продуктивності. Тому впровадження кластеризації є необхідною умовою для масштабування системи.

Отримані результати досліджень підтверджують ефективність архітектурних рішень та оптимізаційних методів. Використання вбудованих функцій СКБД, оптимізація структури API-відповідей та впровадження кластеризації маркерів забезпечують підвищення продуктивності вебплатформи та її здатність до масштабування. Практичне застосування отриманих результатів гарантує стабільну роботу системи під навантаженням та створює основу для подальшого розвитку і впровадження нових функціональних можливостей.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи магістра розроблена вебплатформа для підтримки соціальних ініціатив з геолокаційними функціями, яка може підвищити ефективність координації людей, які надають або потребують допомоги.

Проведено аналіз особливостей волонтерської діяльності та існуючих цифрових рішень. Встановлено, що в умовах сучасних викликів волонтерство стало ключовим елементом, особливо в кризових ситуаціях. Аналіз існуючих платформ виявив суттєвий недолік – відсутність або обмеженість геолокаційних функцій, необхідних для оперативної діяльності на локальному рівні. На основі цього сформовано функціональні вимоги до нового ресурсу, ключовими з яких стали: інтерактивна карта, система облікових записів, управління подіями, адміністративна панель, інформативність та UX.

Здійснено порівняльний аналіз архітектурних підходів та сучасних веб-технологій. Як архітектурні рішення розглянуто клієнт-серверна модель та монолітна або мікросервісна, з їхніми перевагами та недоліками. Проаналізовано backend та frontend технології, СКБД для роботи з геопросторовими даними та картографічні сервіси для їх відображення. Як оптимальний стек розробки було обрано Laravel, React, MySQL та Leaflet, що забезпечать високу інтерактивність, масштабованість та надійну роботу з координатами для SPA на базі розділеної клієнт-серверної архітектури.

Розроблено ER-діаграму бази даних, що складається з 11 взаємопов'язаних сутностей, які включають таблиці для користувачів, подій, категорій, ролей, коментарів та допоміжних, які забезпечать їх взаємодію. Для забезпечення безпеки спроєктовано рольову модель доступу, яка відображає залежність доступності сторінок та елементів інтерфейсу від ролі користувача та прав володіння об'єктами.

Створено повнофункціональний API на базі Laravel, який використовує ресурсно-орієнтований підхід для стандартизації відповідей (API Resources) та

Sanctum для аутентифікації. Клієнтську частину реалізовано на React із використанням компонентного підходу, налаштовано маршрутизацію та глобальне управління станом для користувача. Розгорнуто БД та впроваджено рольову модель доступу з використанням Laravel Policies та Gates, додано ієрархію ролей, що гарантує надійний захист даних та чітке розмежування повноважень.

Реалізовано інтерактивну карту на базі бібліотеки Leaflet.js та даних OpenStreetMap для візуалізації соціальних ініціатив. Розроблено алгоритм динамічної фільтрації подій, що крім звичайних параметрів (статус, категорія), включає пошук у заданому радіусі з використанням вбудованих функцій MySQL, які також розраховують відстань від користувача. Впроваджено механізм кластеризації маркерів за допомогою «Leaflet.markercluster».

Створено захищену панель адміністратора, доступ до якої обмежено на рівні серверних політик та клієнтської маршрутизації. Модуль включає інструменти перегляду статистики вебплатформи, управління обліковими записами користувачів, модерації подій та обробки скарг, що забезпечує повний контроль за контентом системи.

Проведено експериментальне дослідження для оцінки продуктивності обраних рішень та їх вплив на швидкодію вебплатформи під навантаженням. Експериментально підтверджено, що використання вбудованих функцій геопошуку MySQL показує стабільні результати та при збільшенні кількості запитів, залишається ефективнішим ніж обчислення за формулою. Оптимізація API-ресурсів дозволила зменшити обсяг даних у 4 рази, також зменшується час завантаження. Впровадження кластеризації скоротило кількість DOM-елементів на карті у 13 разів, забезпечивши стабільну плавність інтерфейсу.

Узагальнюючи отримані результати, можна стверджувати, що створена вебплатформа є готовим до впровадження програмним продуктом, який має високу практичну цінність для громадських організацій та волонтерських центрів. Запропоновані архітектурні рішення забезпечують масштабованість системи та створюють основу для подальшого розвитку функціоналу.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Сверстюк А. С., Гузоватий С. В. Порівняльний аналіз картографічних сервісів для інтеграції інтерактивних карт при веброзробці. Науковий журнал «Комп'ютерно-інтегровані технології: освіта, наука, виробництво». Луцьк: Луцький НТУ, 2025. Вип. №59. С. 221-227.

2. Сверстюк А. С., Андрущак І. Є., Гузоватий С. В. Геокодинг у веброзробці: аналіз методів та практична реалізація на базі Nominatim. Науковий журнал «Комп'ютерно-інтегровані технології: освіта, наука, виробництво». Луцьк: Луцький НТУ, 2025. Вип. № 61. С. 199-205.

3. Як фінансуються соціальні ініціативи Olantrans. URL: https://surl.lu/tpqhmr (дата звернення: 01.09.2025).

4. Захарчук М. Є. Волонтерство як соціальний феномен, інститут громадянського суспільства, економічна і соціальна категорія. Публічне управління та адміністрування в умовах війни і в поствоєнний період в Україні, 2022. Том 2. С. 129-133. URL: https://surl.li/anzooy (дата звернення: 01.09.2025).

5. Олександр Резнік. Залученість українців до волонтерської діяльності під час повномасштабної війни – Фонд «Демократичні ініціативи» ім. Ілька Кучеріва. URL: https://surl.li/ovbtsp (дата звернення: 02.09.2025).

6. Rudakova S., Shchetinina L., Danylevych N., Kozhara V. Development and digitalization of public organizations in Ukraine. URL: https://surl.lu/nvchej (дата звернення: 03.09.2025).

7. Press. Kickstarter. URL: https://surl.li/kiijvt (дата звернення: 04.09.2025).

8. ZOIC. Trading Card Game by David Silva. Kickstarter. URL: https://surl.li/cimdhc (дата звернення: 04.09.2025).

9. Як українські краудфандингові платформи допомагають військовим, волонтерам і не лише їм. Центр демократії та верховенства права. URL: https://surl.li/yrlizf (дата звернення: 05.09.2025).

10. Кожна вулична тварина має право на здорове життя! Спільнокошт – краудфандинг в Україні. Велика Ідея. URL: https://surl.lt/rwwidw (дата звернення: 05.09.2025).

11. VolunteerMatch, Powered by Idealist: Overview, LinkedIn. URL: https://surl.li/cvhrfg (дата звернення: 08.09.2025).

12. Best Virtual Volunteer Opportunities, On-Site Volunteering, and More – Idealist. URL: https://surli.cc/ijhqfq (дата звернення: 08.09.2025).

13. Волонтерска Платформа для пошуку волонтерів. URL: https://surl.li/haeqqe (дата звернення: 09.09.2025).

14. Можливості. Волонтерська платформа. URL: https://surl.li/ytoepa (дата звернення: 09.09.2025).

15. Meetup: Overview, LinkedIn. URL: https://surl.li/wxghic (дата звернення: 10.09.2025).

16. About Nextdoor. URL: https://surl.li/dlbqsw (дата звернення: 11.09.2025).

17. Nextdoor adds Help Maps and Groups to connect neighbors during the coronavirus outbreak. TechCrunch. URL: https://surl.lt/bnhgwn (дата звернення: 11.09.2025).

18. Ride booker. Bolt Business. URL: https://surl.li/vgtcue (дата звернення: 12.09.2025).

19. An overview to Airbnb experiences: Pros and Cons. Xola. URL: https://surl.li/uuzuso (дата звернення: 15.09.2025).

20. Секрети Ейфелевої вежі з нащадком. URL: https://surl.li/zwxfuo (дата звернення: 15.09.2025).

21. What Is Client-Server Architecture? Supermicro. URL: https://surl.li/scsopf (дата звернення: 16.09.2025).

22. Understanding Client-Server Architecture Basics – SynchroNet. URL: https://surl.li/boocdh (дата звернення: 16.09.2025).

23. Monolithic vs Microservices: Differences, Pros, Cons in 2025. URL: https://surl.li/zpelay (дата звернення: 17.09.2025).

24. Microservices vs monolithic architecture. Atlassian. URL: https://surl.li/ongssr (дата звернення: 17.09.2025).

25. Django, ImpiCode. URL: https://surl.lt/lxdmex (дата звернення: 18.09.2025).

26. Advantages And Disadvantages Of Laravel Development. URL: https://surl.li/bsulhw (дата звернення: 19.09.2025).

27. What are the Features of Node.js? GeeksforGeeks. URL: https://surl.li/lwbyew (дата звернення: 19.09.2025).

28. The Pros and Cons of Using React vs Vue.js vs Angular – DEV Community. URL: https://surl.li/ohllpj (дата звернення: 20.09.2025).

29. Angular Vs React Vs Vue: Which One To Choose. TatvaSoft Blog. URL: https://surl.li/cacosq (дата звернення: 20.09.2025).

30. Fastest Front-End Frameworks in 2025. URL: https://surl.li/njearj (дата звернення: 22.09.2025).

31. PostGIS. URL: https://postgis.net/ (дата звернення: 23.09.2025).

32. Working with Geospatial Features in MySQL – PlanetScale. URL: https://surl.li/poqyvo (дата зверення: 23.09.2025).