

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО”**

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

До захисту допущено:

В.о. завідувача кафедри

Михайло НОВОТАРСЬКИЙ

(підпис)

“ ” _____ 2025 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою “Комп’ютерні системи та мережі”

спеціальності 123 “Комп’ютерна інженерія”

на тему: Інтерактивна інформаційно-довідкова система з військової історії
України

Виконав : студент 4 курсу, групи ІО-15
(шифр групи)

Поляков Валентин Олегович

(прізвище, ім’я, по батькові)

(підпис)

Керівник доцент Павлов В. Г.

(посада, науковий ступінь, вчене звання, прізвище та ініціали)

(підпис)

Консультант (нормоконтроль) асистент Нікольський С.С.,

(назва розділу)

(посада, вчене звання, науковий ступінь, прізвище та ініціали)

(підпис)

Рецензент д.т.н., проф. кафедри ІСТ Корнієнко Б. Я.

(посада, науковий ступінь, вчене звання, прізвище та ініціали)

(підпис)

Засвідчую, що у цьому дипломному
проєкті немає запозичень з праць інших
авторів без відповідних посилань.

Студент _____

(підпис)

Київ – 2025 р.

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО”**

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Рівень вищої освіти – перший (бакалавр)

Освітньо-професійна програма

“Комп’ютерні системи та мережі”

спеціальності 123 “Комп’ютерна інженерія”

ЗАТВЕРДЖУЮ

В.о завідувача кафедри

_____ **Михайло НОВОТАРСЬКИЙ**

(підпис)

“__” _____ 2025 р.

ЗАВДАННЯ

на бакалаврський дипломний проєкт студента

Полякова Валентин Олеговича

1. Тема проєкту Інтерактивна інформаційно-довідкова система з військової історії України
керівник проєкту Павлов Валерій Георгійович, доцент,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від 23 травня 2025 року №1705-с
2. Термін здачі студентом закінченого проєкту 02.06.2025
3. Вихідні дані до проєкту технічна документація, теоретичні дані.
4. Зміст розрахунково-пояснювальної записки (перелік питань, які розробляються): огляд картографічних платформ, обрані інструменти для розробки, програмна реалізація.
5. Перелік графічного матеріалу (з точним позначенням обов’язкових креслень) діаграма прецедентів, діаграма послідовності, діаграма класів.

6. Консультанта проєкту, з вказівкою розділів проєкту, які до них вносяться

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Нормоконтроль	Нікольський С. С.		

7. Дата видачі завдання «20» січня 2025 р.

Календарний план

№ п/п	Найменування етапів дипломного проєкту	Терміни виконання етапів проєкту	Примітки
1.	<i>Затвердження теми проєкту</i>	20.01.2025	
2.	<i>Вивчення та аналіз завдання</i>	07.02.2025	
3.	<i>Розробка архітектури та загальної структури системи</i>	06.04.2025	
4.	<i>Розробка структур окремих підсистем</i>	12.05.2025	
5.	<i>Програмна реалізація системи</i>	20.05.2025	
6.	<i>Оформлення пояснювальної записки</i>	01.05.2025	
7.	<i>Захист програмного продукту</i>	02.06.2025	
8.	<i>Передзахист</i>	04.06.2025	
9.	<i>Захист</i>	18.06.2025	

Студент-дипломник _____ Валентин ПОЛЯКОВ
(підпис)

Керівник проєкту _____ Валерій ПАВЛОВ
(підпис)

АНОТАЦІЯ

У цій роботі досліджено сучасні картографічні платформи та проаналізовано їх особливості й недоліки. На основі аналізу обрано інструменти для створення інформаційно-довідкової системи з військової історії України у вигляді інтерактивної карти. Проаналізовано архітектуру бази даних, серверної частини та веб-додатку, що забезпечують стабільність, продуктивність і зручність системи. Програмний продукт реалізовано на JavaScript із використанням React, Node.js і SQLite.

Ключові слова: інтерактивна карта, веб-додаток, військова історія, JavaScript, React, Mapbox.

ANNOTATION

In this project, modern cartographic platforms were explored and their features and limitations analyzed. Based on this analysis, tools were selected to develop an informational-reference system on Ukraine's military history in the form of an interactive map. The architecture of the database, server-side, and web application was analyzed, ensuring the system's stability, performance, and usability. The software product was developed using JavaScript with React, Node.js, and SQLite.

Keywords: interactive map, web application, military history, JavaScript, React, Mapbox.

[illegible]

ТЕХНІЧНЕ ЗАВДАННЯ
ДО ДИПЛОМНОГО ПРОЄКТУ

на тему: «Інтерактивна інформаційно-довідкова система з військової
історії України»

Київ – 2025

ЗМІСТ

НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ	4
ПІДСТАВИ ДЛЯ РОЗРОБКИ.....	4
МЕТА ТА ПРИЗНАЧЕННЯ РОЗРОБКИ.....	4
ДЖЕРЕЛА РОЗРОБКИ	5
ТЕХНІЧНІ ВИМОГИ	5
Вимоги до розробленого продукту.....	5
Вимоги до програмного забезпечення	5
Вимоги до апаратної частини	5
ЕТАПИ РОЗРОБКИ	6

					ІАЛЦ.467800.002 ТЗ		
Зм.	Арк.	№ докум.	Підпис	Дата			
Розробив		Поляков В.О.			Інтерактивна інформаційно-довідкова система з військової історії України Технічне завдання	Літ.	Аркуш
Перевірив		Павлов В.Г.					1
							4
Н. Контр.		Нікольський С.С.				КПІ ім. Ігоря Сікорського, ФІОТ, ІО-15	
Затвердив		Новотарський М.А.					

1 НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ

Дане технічне завдання поширюється на розробку інтерактивної інформаційно-довідкової системи з військової історії України у вигляді веб-додатку.

Областю застосування цієї системи є освітні та науково-дослідницькі проекти, а також повсякденне використання для ефективного сприйняття широкою аудиторією користувачів

2 ПІДСТАВИ ДЛЯ РОЗРОБКИ

Підставою для розробки даної системи є завдання для виконання роботи кваліфікаційно-освітнього рівня «бакалавр комп'ютерної інженерії», який був затверджений факультетом “Інформатики та обчислювальної техніки” кафедрою обчислювальної техніки Національного технічного Університету України «Київський Політехнічний інститут ім. Ігоря Сікорського».

3 МЕТА ТА ПРИЗНАЧЕННЯ РОЗРОБКИ

Метою та призначенням цієї роботи є створення інтерактивної інформаційно-довідкової системи з військової історії України у форматі веб-додатку, який забезпечить зручний доступ до інформації та її ефективне представлення через інтерактивну карту. Робота включає порівняльний аналіз сучасних картографічних платформ для вибору оптимального інструменту, обґрунтування технологічного стеку з урахуванням функціональних можливостей, а також програмну реалізацію проекту, розробку архітектури, визначення основних компонентів, їхньої взаємодії та створення зручного інтерфейсу користувача.

					ІАЛЦ.467800.003 ТЗ	Арк.
						2
Зм.	Арк.	№ докум.	Підпис	Дата		

4 ДЖЕРЕЛА РОЗРОБКИ

Джерелом розробки даного дипломного проекту є офіційні документації, публікації та статті в мережі Інтернет на дану тему, науково-технічна література.

5 ТЕХНІЧНІ ВИМОГИ

5.1. Вимоги до розробленого продукту

Розроблена система має виконувати такі вимоги:

- Простий і інтуїтивно-зрозумілий інтерфейс системи.
- Надати можливість користувачам здійснювати інтерактивну навігацію по карті з можливістю масштабування
- Надати можливість користувачам додавати свою інформацію до системи
- Надати можливість користувачам редагувати чи видаляти інформацію системи
- Надати можливість користувачам здійснювати пошук та фільтрацію за ключовими параметрами (дати, типи подій)
- Надати вичерпну та зрозумілу документацію для розробленого продукту.

5.2. Вимоги до програмного забезпечення

- ОС Windows, Mac, Linux, Android чи iOS.
- Веб-браузер із підтримкою HTML5, CSS3, JavaScript (наприклад, Google Chrome, Mozilla Firefox, Safari, Microsoft Edge)
- Visual Studio 2017 IDE версії або вище.

5.3. Вимоги до апаратної частини

- Пристрій із доступом до Інтернету (ПК, планшет або смартфон).
- Роздільна здатність екрану: не менше ніж 480x800
- ЦП не нижче Intel® Core (TM) i3-2100T.
- ROM не менше ніж 64 ГБ.
- RAM не менше ніж 4 ГБ.

					ІАЛЦ.467800.002 ТЗ	Арк.
						3
Зм.	Арк.	№ докум.	Підпис	Дата		

6 ЕТАПИ РОЗРОБКИ

Назва етапів виконання	Термін виконання
Затвердження теми роботи	20.01.2025
Вивчення та аналіз завдання	07.02.2025
Розробка архітектури та загальної структури системи	06.04.2025
Розробка структур окремих частин системи	12.05.2025
Програмна реалізація системи	20.05.2025
Виправлення помилок	28.05.2025
Оформлення пояснювальної записки	02.06.2025

					ІАЛЦ.467800.002 ТЗ	Арк.
						3
Зм.	Арк.	№ докум.	Підпис	Дата		

**ПОЯСНЮВАЛЬНА ЗАПИСКА
ДО ДИПЛОМНОГО ПРОЄКТУ**

на тему: «Інтерактивна інформаційно-довідкова система з військової
історії України»

Київ – 2025

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ	3
ВСТУП.....	4
РОЗДІЛ 1. ОГЛЯД КАРТОГРАФІЧНИХ ПЛАТФОРМ	6
1.1 Загальні відомості.....	6
1.2 Google Maps	6
1.3 Mapbox	7
1.4 Here Maps	9
1.5 OpenStreetMap.....	9
1.6 Bing Maps	10
1.7 Порівняльний аналіз платформ	11
ВИСНОВОК ДО РОЗДІЛУ 1	15
РОЗДІЛ 2. ОБРАНІ ІНСТРУМЕНТИ ДЛЯ РОЗРОБКИ	16
2.1 Основні інструменти програмування	16
2.2 Технології для бекенд частини	18
2.2.1. Node.js	18
2.2.2. npm	19
2.2.3 Express.js	19
2.2.4 REST API.....	20
2.3 Технології для фронтенд частини.....	21
2.3.1 React	21
2.3.2 mapbox-gl	22
2.3.3 ReactMarkdown	23
2.3.5 lucide-react	26
2.4. Система управління базою даних	27
ВИСНОВОК ДО РОЗДІЛУ 2.....	29
РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ.....	30

					ІАЛЦ.467800.003 ПЗ								
Зм.	Арк.	№ докум.	Підпис	Дата									
Розробив		Поляков В.О.			Інтерактивна інформаційно-довідкова система з військової історії України Пояснювальна записка				Літ.	Аркуш	Аркушів		
Перевірив		Павлов В.Г.									1	32	
									КПІ ім. Ігоря Сікорського, ФІОТ, ІО-15				
Н. Контр.		Нікольський С.С.											
Затвердив		Новотарський М.А.											

3.1 Специфікація вимог.....	30
3.1.1 Функціональні вимоги.....	30
3.1.2 Нефункціональні вимоги.....	30
3.2 Проектування системи	31
3.2.1 Використання UML під час розробки системи.....	31
3.2.2 Діаграма варіантів використання.....	32
3.2.3 Діаграма послідовності	36
3.3 Реалізація фронтенд частини	38
3.3.1 Сторінка реєстрації та авторизації.....	38
3.3.2 Інтерфейс користувача	39
3.3.3 Інтерфейс адміністратора.....	40
3.3.4 Сторінка модерації	41
3.3.5 Вікно додавання маркера	41
3.3.6 Панель маркера	45
3.3.7 Вікно редагування маркера	46
3.3.8 Панель пошуку та фільтрації маркерів.....	47
3.3.9 Панель легенди мапи	47
3.4 Реалізація бекенд частини	48
3.4.1 База даних	48
3.4.2 Схема авторизації	50
3.4.3 Реалізація взаємодії між клієнтом і сервером	51
ВИСНОВОК ДО РОЗДІЛУ 3.....	53
ВИСНОВКИ	54
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	55

ПЕРЕЛІК СКОРОЧЕНЬ

DOM	(Document Object Model) Об'єктна модель документа
HTML	(HyperText Markup Language) Мова гіпертекстової розмітки
API	(Application Programming Interface) Прикладний програмний інтерфейс
SQL	(Structured Query Language) Мова структурованих запитів
JSON	(JavaScript Object Notation) Запис об'єктів JavaScript
URL	(Uniform Resource Locator) Уніфікований локатор ресурсів
OSM	OpenStreetMap
СУБД	Система управління базами даних

					ІАЛЦ.467800.003 ПЗ	Арк.
						3
Зм.	Арк.	№ докум.	Підпис	Дата		

ВСТУП

У сучасному світі інформаційні технології відіграють ключову роль у збереженні, обробці та поширенні знань у різноманітних сферах діяльності, включно з освітою, наукою та культурою. Зокрема, цифрові рішення для збереження та представлення історичної інформації дозволяють зробити доступ до цінних даних більш швидким, зручним і інтерактивним. Особливо важливим є створення таких систем для популяризації національної історії, адже це сприяє не лише збереженню культурної спадщини, а й підвищенню рівня освіченості суспільства.

Військова історія України — складна та багатогранна тема, що потребує систематизації, зручного доступу та візуалізації. Традиційні джерела інформації у вигляді текстових документів або книг часто мають обмежену доступність, що ускладнює процес пошуку та аналізу потрібних даних. Використання інтерактивних карт та веб-додатків відкриває нові можливості для візуалізації подій, персоналій і географії бойових дій, роблячи інформацію більш наочною та зрозумілою для широкої аудиторії.

Розробка інтерактивної інформаційно-довідкової системи з військової історії України у форматі веб-додатку має на меті забезпечити користувачам зручний інструмент для пошуку, перегляду та аналізу історичних даних. Важливою складовою є створення інтуїтивного інтерфейсу, що дозволяє ефективно взаємодіяти з системою, а також використання сучасних веб-технологій для забезпечення високої продуктивності, масштабованості та доступності інформації з будь-яких пристроїв.

Крім того, впровадження таких інтерактивних систем сприяє інтеграції історичних даних з іншими інформаційними ресурсами, що дозволяє створювати багат шарові карти з різними рівнями деталізації та контексту. Використання сучасних технологій візуалізації та аналітики відкриває широкі можливості для подальшого розвитку цифрової історіографії, покращуючи

					ІАЛЦ.467800.003 ПЗ	Арк.
						4
Зм.	Арк.	№ докум.	Підпис	Дата		

доступність інформації не лише для фахівців, а й для широкого загалу користувачів. Таким чином, розробка подібної системи не лише задовольняє поточні потреби, але й створює базу для майбутніх інновацій у сфері цифрової культури і освіти.

Метою та призначенням цієї роботи є створення інтерактивної інформаційно-довідкової системи з військової історії України у форматі веб-додатку, який забезпечить зручний доступ до інформації та її ефективне представлення через інтерактивну карту. Робота включає порівняльний аналіз сучасних картографічних платформ для вибору оптимального інструменту, обґрунтування технологічного стеку з урахуванням функціональних можливостей, а також програмну реалізацію проекту, розробку архітектури, визначення основних компонентів, їхньої взаємодії та створення зручного інтерфейсу користувача.

					ІАЛЦ.467800.003 ПЗ	Арк.
						5
Зм.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 1. ОГЛЯД КАРТОГРАФІЧНИХ ПЛАТФОРМ

1.1 Загальні відомості

Інтерактивні картографічні платформи є ключовими елементами сучасних цифрових систем подання інформації. Серед найбільш поширених рішень — Google Maps, Mapbox, OpenStreetMap, HERE Maps та Bing Maps, які відрізняються за функціональністю, підтримкою офлайн-режиму, типом карт (растрові або векторні) та можливістю інтеграції власних даних. У цьому розділі здійснюється огляд основних характеристик цих платформ, їх продуктивності, рівня кастомізації та ліцензійних умов з метою вибору найбільш підходящої бази для створення інтерактивної інформаційно-довідкової системи, присвяченої військовій історії України.

1.2 Google Maps

Протягом багатьох років Google Maps мав статус провідного сервісу серед картографічних API. Однак в останні роки ринок змінився – з’явилися нові API карт, що значно посилили конкуренцію в цій сфері.

Google Maps і досі є лідером серед картографічних сервісів, проте після впровадження платної моделі доступу до свого API та підвищення його ціни більше ніж на 1400% у 2018 році, багато клієнтів були змушені відмовитись від використання сервісу та шукати альтернативні рішення [1].

Google Maps має низку суттєвих переваг у порівнянні з конкурентами. По-перше, сервіс відзначається високою точністю та актуальністю картографічних даних, що забезпечується регулярними оновленнями на основі якісних супутникових знімків та інформації з різних джерел. По-друге, платформа охоплює широкий географічний спектр — від великих міст до віддалених населених пунктів, що робить її універсальним рішенням для різних регіонів.

					ІАЛЦ.467800.003 ПЗ	Арк.
						6
Зм.	Арк.	№ докум.	Підпис	Дата		

Крім того, Google Maps пропонує розвинений функціонал API, який підтримує побудову маршрутів, інтеграцію з місцями (POI), а також можливості стилізації карт. Важливою перевагою є інтеграція з іншими сервісами Google, що розширює можливості платформи. Google Maps також відзначається високою стабільністю та зручністю використання, що робить її придатною як для невеликих проєктів, так і для масштабних корпоративних рішень.

Google Maps підтримує інтерактивну візуалізацію великих обсягів просторових даних та пропонує базові інструменти для стилізації карт, що дає змогу частково налаштувати зовнішній вигляд карти відповідно до завдань проєкту.

Користування платформою доступне як у межах умовно-безкоштовного тарифу з лімітом \$200, так і на платній основі. Для роботи з API необхідно зареєструватися на платформі та отримати унікальний ключ доступу [1].

Водночас слід враховувати певні обмеження платформи. По-перше, після переходу на платну модель використання API, вартість сервісу може суттєво зрости при масштабуванні проєкту, що може стати викликом для систем з обмеженим бюджетом. По-друге, хоча платформа надає можливості для стилізації карт, використання базового шару за замовчуванням обмежує повну свободу візуального оформлення, що може бути критично для створення унікального дизайну, адаптованого під специфіку тематики проєкту.

1.3 Mapbox

Mapbox — одна з провідних платформ для створення кастомізованих картографічних рішень, заснована у 2013 році. Вона використовує відкриті джерела даних, зокрема OpenStreetMap, що забезпечує високу деталізацію та актуальність картографічної інформації.

Mapbox надає гнучкі засоби кастомізації карт через інструмент Mapbox Studio, що дозволяє змінювати стилі, кольори, шари та елементи інтерфейсу

					ІАЛЦ.467800.003 ПЗ	Арк.
						7
Зм.	Арк.	№ докум.	Підпис	Дата		

відповідно до тематики проєкту. Це особливо актуально для відображення історичних об'єктів, де важливо виділити ключові візуальні елементи. У цьому аспекті платформа забезпечує вищу ступінь контролю над дизайном порівняно з Google Maps [2].

Завдяки використанню векторних плиток і ефективній системі кластеризації маркерів (SuperCluster), Mapbox забезпечує високу продуктивність навіть при великій кількості позначок на карті, що особливо важливо для інтерактивної довідкової системи з численними об'єктами.

Mapbox підтримує функцію пошуку об'єктів за координатами або назвами, що дозволяє користувачам швидко знаходити потрібні локації. Варто відзначити, що ця функція у Mapbox є безкоштовною у межах тарифних планів, на відміну від Google Maps, де пошук після перевищення певного ліміту може стати додатковою платною послугою.

Mapbox пропонує безкоштовний тариф із лімітом у 50 тисяч запитів на місяць, що дозволяє розробляти та тестувати систему без значних витрат. Оплата за перевищення ліміту здійснюється за моделлю pay-as-you-go, що означає, що користувач сплачує лише за фактичну кількість додаткових запитів понад безкоштовний тариф, без фіксованих щомісячних платежів. Такий підхід дозволяє гнучко керувати витратами залежно від реального використання сервісу [3].

Разом із перевагами платформи необхідно врахувати і деякі її обмеження. Наприклад, платформа також має обмежену підтримку офлайн-режиму — реалізація локального зберігання карт потребує додаткових рішень та інфраструктури, що впливає на гнучкість використання в середовищах без стабільного доступу до інтернету. До цього додаються вимоги до обов'язкового брендунгання: навіть при кастомізації дизайну, платформа зобов'язує відображати логотип Mapbox, що може суперечити вимогам візуального оформлення деяких проєктів. Варто також мати на увазі, що цінова політика сервісу не є стабільною — у минулому тарифні умови змінювалися, що створює

					ІАЛЦ.467800.003 ПЗ	Арк.
						8
Зм.	Арк.	№ докум.	Підпис	Дата		

додаткові ризики при масштабуванні системи або у випадках непередбачуваного зростання кількості запитів.

1.4 Here Maps

Платформа Here Maps переважно орієнтована на транспортну логістику та управління автопарками, що варто враховувати при розробці. Основний функціонал Here Maps орієнтований на навігацію та управління автопарками, що робить платформу менш гнучкою для кастомізації карт і відображення різноманітних об'єктів, зокрема історичних.

У порівнянні з Google Maps, яка має широкий функціонал для різних сценаріїв використання і кращу інтеграцію з іншими сервісами, Here Maps пропонує більш точне покриття у Європі, проте поступається за можливостями кастомізації і варіативності використання. Google Maps, як і Mapbox, надають більше інструментів для адаптації карт під специфічні потреби, що важливо для створення тематичних проєктів.

Mapbox, своєю чергою, пропонує розширені можливості для тонкої стилізації і кастомізації карт, що дозволяє максимально адаптувати візуалізацію під специфіку військової тематики, зокрема акцентуючи увагу на історичних об'єктах і маршрутах. На відміну від Here Maps, Mapbox підтримує більш гнучку систему налаштувань і краще підходить для інтерактивних довідкових систем з великою кількістю об'єктів.

1.5 OpenStreetMap

OpenStreetMap (OSM) — це безкоштовний проєкт із відкритим кодом, який забезпечує точні та детальні картографічні дані завдяки внескам спільноти користувачів. Як краудсорсингова платформа, OSM дозволяє спільно редагувати та оновлювати інформацію, що гарантує високу актуальність карт.

					ІАЛЦ.467800.003 ПЗ	Арк.
						9
Зм.	Арк.	№ докум.	Підпис	Дата		

API OpenStreetMap надає базові інструменти для роботи з картами, переважно орієнтовані на їх оновлення та редагування. Водночас, через обмежену функціональність, API не підходить для розробки складних картографічних сервісів без додаткових інструментів.

З цієї причини багато розробників використовують платформи, побудовані на основі даних OSM, наприклад, Mapbox. Вони надають розширений функціонал, інструменти для стилізації та кращу продуктивність, що значно спрощує створення кастомізованих і масштабованих картографічних рішень.

Відкритість даних OSM і активність спільноти створюють унікальне середовище для постійного вдосконалення картографічних ресурсів. Це особливо важливо для проєктів, які потребують швидких оновлень та локальної деталізації, що може бути критично для військової історичної тематики або оперативних завдань.

Проте, безпосереднє використання OpenStreetMap для створення власного проєкту має і певні недоліки. Основними є необхідність самостійно розробляти додаткові функції для стилізації, кешування та обробки даних, а також залежність від якості внесків спільноти, що може впливати на точність і повноту інформації в окремих регіонах.

1.6 Bing Maps

Bing Maps — це картографічна платформа від корпорації Microsoft, яка надає широкий спектр сервісів для відображення карт, побудови маршрутів, геокодування, аналізу просторових даних та інтеграції з іншими продуктами Microsoft. Сервіс підтримує різні типи карт: дорожні, супутникові та гібридні, що дозволяє обирати найкращий спосіб візуалізації в залежності від цілей проєкту.

Однією з ключових особливостей Bing Maps є тісна інтеграція з екосистемою Microsoft, зокрема з Azure, Power BI та іншими хмарними й

					ІАЛЦ.467800.003 ПЗ	Арк.
						10
Зм.	Арк.	№ докум.	Підпис	Дата		

аналітичними інструментами. Це робить платформу привабливою для корпоративного сегменту, де важливим є поєднання картографічного функціоналу з аналітикою, візуалізацією даних та автоматизацією процесів. З точки зору технічної реалізації, Bing Maps надає REST API, JavaScript API та підтримку геокодування, маршрутизації та просторового аналізу, що дозволяє реалізувати як базові, так і складні сценарії роботи з геоданими.

Щодо точності картографічних даних, Bing Maps демонструє високий рівень деталізації для більшості регіонів, зокрема у США та Західній Європі, однак для деяких територій, зокрема Східної Європи, покриття та актуальність даних можуть бути дещо обмеженими. Це слід враховувати при розробці проєктів, пов'язаних з українською історичною тематикою.

Платформа надає можливості для стилізації карт, однак ці можливості суттєво поступаються таким рішенням, як Mapbox. Гнучкість у налаштуванні зовнішнього вигляду карти обмежена заздалегідь заданими стилями та меншими можливостями редагування. Це може стати обмеженням при створенні унікальних історичних візуалізацій, де важливо виокремити конкретні об'єкти або події. Також слід зауважити, що документація API є менш детальною порівняно з конкурентами, що може ускладнити швидку інтеграцію та зменшує кількість прикладів реального використання для розробників.

Щодо ліцензування, Bing Maps має безкоштовний рівень для невеликих проєктів, однак доступ до більшості комерційних функцій, включаючи великий обсяг запитів, можливий лише на умовах підписки або корпоративної ліцензії. Це може обмежити використання Bing Maps у проєктах з обмеженим бюджетом або високими вимогами до масштабованості.

1.7 Порівняльний аналіз платформ

Для вибору оптимальної картографічної платформи в рамках створення інтерактивної інформаційно-довідкової системи з військової історії України

					ІАЛЦ.467800.003 ПЗ	Арк.
						11
Зм.	Арк.	№ докум.	Підпис	Дата		

доцільно провести порівняльний аналіз найбільш популярних рішень. Аналіз здійснюється за технічними та комерційними критеріями, які є визначальними для реалізації проєктів подібного типу.

У таблиці 1.1 надані технічні характеристики картографічних платформ, розглянутих у попередніх підрозділах.

Таблиця 1.1 – Технічні характеристики картографічних платформ

Платформа	Тип карт	Кастомізація	Підтримка офлайн	Документація
Google Maps	Векторні та растрові	Обмежена стилізація, базові інструменти	Обмежена	Детальна
Mapbox	Векторні	Висока кастомізація	Обмежена	Детальна
Here Maps	Векторні	Обмежена, орієнтована на навігацію	Немає	Базова, фокус на B2B
OSM	Векторні	Відсутня	Так	Базова, спільнотна
Bing Maps	Векторні та растрові	Обмежена, фіксовані стилі	Немає	Застаріла

Як видно з таблиці, найвищий рівень кастомізації пропонує Mapbox, що є критично важливим для візуалізації історичних об'єктів у специфічному стилі. Google Maps та Bing Maps надають базову кастомізацію, проте візуальні можливості цих платформ обмежені попередньо визначеними стилями. OSM, хоч і має відкритий код, потребує додаткових бібліотек і рішень для реалізації кастомного вигляду. Підтримка офлайн-режиму реалізована лише частково

(Mapbox, Google Maps) або повністю відсутня (Here, Bing Maps), що може бути критичним через нестабільне з'єднання.

Вибір картографічної платформи значною мірою залежить від якості технічної документації, що безпосередньо впливає на швидкість і ефективність розробки та інтеграції функціоналу. Платформи Google Maps і Mapbox вирізняються високим рівнем деталізації документації, наявністю численних прикладів та чіткою структурою, що значно спрощує роботу розробників над проєктами різної складності. У випадку OpenStreetMap офіційна документація є менш повною, проте її недоліки компенсуються великою кількістю спільнотних ресурсів і зовнішніх бібліотек, що допомагають у роботі з платформою. Натомість Here і Bing Maps мають менш докладні інструкції, обмежену кількість прикладів і менш зручний інтерфейс документації, що може ускладнити їх інтеграцію.

У таблиці 1.2 приведені вартість, ключові особливості та обмеження платформ.

Mapbox демонструє найбільш збалансований підхід між кастомізацією, вартістю та можливостями масштабування. У той же час, Google Maps є лідером за точністю даних, але має високу ціну при зростанні кількості запитів. Here і Bing Maps вигідні для корпоративних сценаріїв, однак мають менше гнучкості для реалізації спеціалізованих історичних проєктів. OpenStreetMap — найкраще рішення для проєктів з обмеженим бюджетом, але вимагає додаткової технічної реалізації.

					ІАЛЦ.467800.003 ПЗ	Арк.
						13
Зм.	Арк.	№ докум.	Підпис	Дата		

Таблиця 1.2 – Комерційні та функціональні аспекти

Платформа	Вартість/Ліцензія	Особливості	Обмеження
Google Maps	Платна, \$200 безкоштовно	Висока точність і актуальність даних, інтеграція з іншими сервісами Google	Висока ціна API при масштабуванні
Mapbox	Безкоштовно + pay-as-you-go	Підтримка пошуку, гнучкі тарифи	Обов'язкове брендуння, змінна тарифна політика
Here Maps	Комерційна та корпоративна ліцензія	Детальний у логістиці, найкраще покриття в Європі	Орієнтований на транспорт
OSM	Безкоштовна (open source)	Відкритий код, активна спільнота	Потреба в додаткових інструментах для повноцінної роботи
Bing Maps	Безкоштовний для малих проєктів, комерційна ліцензія – для великих	Інтеграція з екосистемою Microsoft	Слабке покриття Східної Європи, слабка документація

ВИСНОВОК ДО РОЗДІЛУ 1

У першому розділі були розглянуті сучасні картографічні платформи, їх переваги та обмеження. Для розробки інтерактивної інформаційно-довідкової системи з військової історії України ключовими критеріями є: високий рівень кастомізації карт, доступність тарифів та якість документації.

Marbox вирізняється як найоптимальніше рішення, завдяки гнучкій стилізації, ефективній роботі з векторними даними та прийнятній ціновій моделі, що дозволяє масштабувати проєкт без значних витрат на початкових етапах. Google Maps, хоч і забезпечує високу точність та стабільність, виявився менш придатним через обмеження в дизайні та високу вартість при активному використанні. OpenStreetMap є привабливою безкоштовною альтернативою, однак потребує значних додаткових технічних зусиль для реалізації повноцінної системи. Платформи Here та Bing Maps менш придатні для реалізації тематичних довідкових проєктів через обмежену гнучкість, застарілу або неповну документацію, а також орієнтацію переважно на бізнес- або логістичні сценарії.

Таким чином, з огляду на технічні, функціональні та економічні аспекти, платформу Marbox доцільно розглядати як базове технологічне рішення для реалізації проєкту візуалізації військової історії України.

					ІАЛЦ.467800.003 ПЗ	Арк.
						15
Зм.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 2. ОБРАНІ ІНСТРУМЕНТИ ДЛЯ РОЗРОБКИ

2.1 Основні інструменти програмування

Було обрано три мови програмування, що можуть використовуватися для написання проекту: Python, JavaScript та PHP. Переваги та недоліки даних мов неведені у таблиці 2.1.

Таблиця 2.1 – Порівняння мов програмування

	Python	JavaScript	PHP
Фреймворк	Django	React	Laravel
Продуктивність	Середня (повільніше через інтерпретатор)	Висока (швидкий рендеринг у браузері)	Середня (швидше за Python, але залежить від сервера)
Інтеграція з Mapbox	Хороша (через API, але складніше з UI)	Відмінна (нативна підтримка через Mapbox GL JS)	Хороша (через API, потребує JS для карти)
Спільнота та підтримка	Велика, багато бібліотек	Велика, активна екосистема	Велика, але менш активна

Отже, зваживши всі за і проти, було обрано JavaScript, оскільки найкраще інтегрується з мапою Mapbox та має велику екосистему.

JavaScript — це високорівнева, інтерпретована мова програмування, яка відіграє ключову роль у розробці інтерактивної інформаційно-довідкової системи з військової історії України. Ця мова була обрана як основна для

реалізації проєкту завдяки своїй універсальності, широкій підтримці в веб-розробці та здатності забезпечувати інтерактивність, що є критично важливим для створення динамічної карти з маркерами історичних подій. Також JavaScript є найвикористанішою мовою у світі (рис. 2.1).

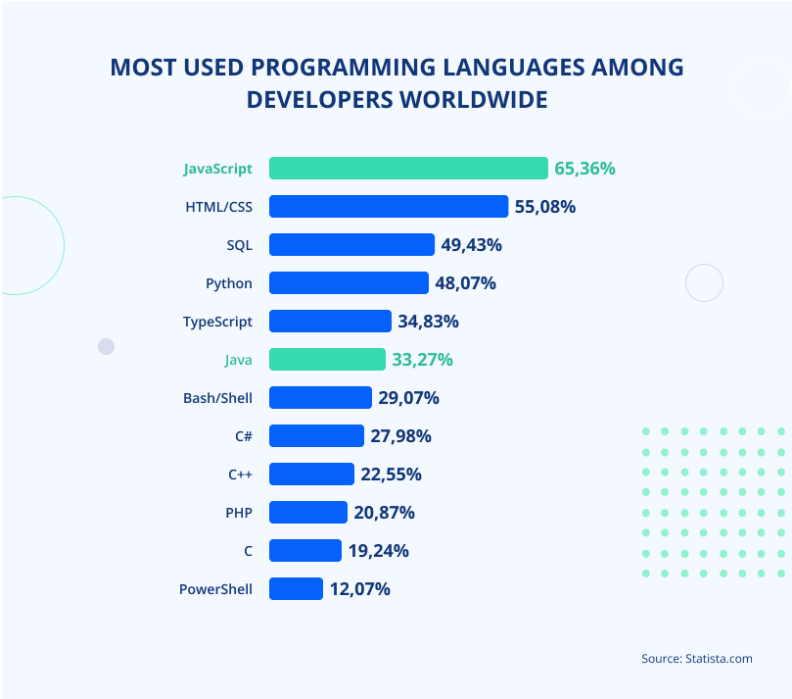


Рисунок 2.1 – Найбільш використовувані мови програмування у світі [10]

Мова підтримує кілька парадигм програмування, включаючи об’єктно-орієнтоване, функціональне та імперативне програмування. У контексті проєкту активно використовується функціональний підхід, зокрема через використання бібліотеки React, яка спирається на функціональні компоненти та концепцію імутабельності даних.

Виконання JavaScript можливе у будь-якому сучасному браузері і є основною мовою для фронтенд та бекенд частин проєкту, що дозволяє реалізувати інтерактивні елементи інтерфейсу користувача та обробку серверної логіки в єдиному технологічному стеку [4].

За допомогою React.js, що базується на JavaScript, реалізовано основні компоненти UI-інтерфейсу: інтерактивна карта, список подій, фільтри, форми

для додавання нових маркерів та інші. React дозволяє будувати складні веб-додатки.

На бекенді використовується середовище Node.js, що також працює на основі JavaScript. Node.js забезпечує роботу серверної частини і надає доступ до широкого спектру бібліотек та модулів, що застосовуються для обробки HTTP-запитів, взаємодії з базою даних, керуванням авторизації користувачів, забезпечує збереження та обробку інформації.

2.2 Технології для бекенд частини

2.2.1. Node.js

Node.js — це середовище виконання JavaScript поза браузером, яке базується на рушії V8 від Google Chrome. Воно дозволяє запускати JavaScript-код на сервері, що дає змогу використовувати одну мову програмування як для фронтенду, так і для бекенду.

Головною особливістю Node.js є його подієво-орієнтована, неблокуюча модель введення-виведення (I/O), що забезпечує високу продуктивність і масштабованість при роботі з великою кількістю одночасних з'єднань. Це особливо актуально для веб-додатків, які обробляють численні HTTP-запити або працюють із реальним часом.

Node.js має багату екосистему модулів, керованих через менеджер пакетів npm (Node Package Manager). Завдяки цьому розробники можуть швидко інтегрувати готові бібліотеки для реалізації різноманітних функцій, від обробки HTTP-запитів і маршрутизації до роботи з базами даних, до аутентифікації, логування та ін. [5]

У рамках реалізації проєкту Node.js виконує роль серверної платформи, яка приймає та обробляє запити від клієнтського додатку, взаємодіє з базою даних SQLite, забезпечує авторизацію користувачів та контролює бізнес-логіку

					ІАЛЦ.467800.003 ПЗ	Арк.
						18
Зм.	Арк.	№ докум.	Підпис	Дата		

системи. Для реалізації серверної логіки у проєкті використовується фреймворк Express.js — легковажна і гнучка бібліотека для побудови REST API, що значно полегшує розробку маршрутів та обробку HTTP-запитів.

Використання Node.js у проєкті також забезпечує можливість запуску серверної частини на різних платформах — Windows, Linux, macOS — без необхідності змінювати код, що підвищує портативність системи. Завдяки асинхронній природі Node.js сервер може ефективно обробляти запити з мінімальними затримками, що особливо важливо для підтримки інтерактивності і швидкодії веб-застосунку.

2.2.2. npm

npm (Node Package Manager) — це найпопулярніший менеджер пакетів для мови програмування JavaScript, який широко використовується в середовищі Node.js. Він забезпечує розробникам зручний та ефективний спосіб пошуку, встановлення, оновлення та керування сторонніми бібліотеками та інструментами, необхідними для створення веб-додатків, серверних сервісів і різноманітного програмного забезпечення.

За допомогою npm можна легко управляти залежностями проєкту: при встановленні певного пакета автоматично завантажуються і встановлюються всі необхідні для його роботи додаткові бібліотеки. Це значно спрощує підтримку і розвиток складних проєктів, адже npm відповідає за узгодження версій пакетів, що забезпечує стабільність і передбачуваність роботи програми.

2.2.3 Express.js

Express.js — це мінімалістичний і гнучкий веб-фреймворк для Node.js, який надає набір інструментів для розробки веб-додатків та API. Він спрощує створення серверної частини додатків, дозволяючи швидко налаштовувати

					ІАЛЦ.467800.003 ПЗ	Арк.
						19
Зм.	Арк.	№ докум.	Підпис	Дата		

маршрутизацію, обробку запитів і відповіді, а також інтегрувати проміжне програмне забезпечення (middleware) для розширення функціональності.

Ключовою особливістю Express.js є підтримка middleware — послідовних функцій для обробки HTTP-запитів, що дозволяє реалізовувати аутентифікацію, логування, парсинг запитів та обробку помилок. Це сприяє створенню модульних і масштабованих серверних застосунків.

Express.js підтримує роботу з шаблонізаторами для створення динамічних веб-сторінок і легко інтегрується з базами даних та іншими сервісами. Активна спільнота і широкий набір плагінів розширюють можливості фреймворка.

2.2.4 REST API

REST API (Representational State Transfer Application Programming Interface) — це архітектурний стиль для створення вебсервісів, який базується на використанні стандартних HTTP-протоколів і методів. Головна ідея REST полягає в тому, що сервер надає клієнту доступ до ресурсів (даних чи сервісів) через унікальні URL, а клієнт взаємодіє з цими ресурсами за допомогою стандартних HTTP-запитів: GET, POST, PUT, DELETE та ін (рис. 2.2). Кожен такий запит відповідає певній операції над ресурсом — отриманню, створенню, оновленню або видаленню.

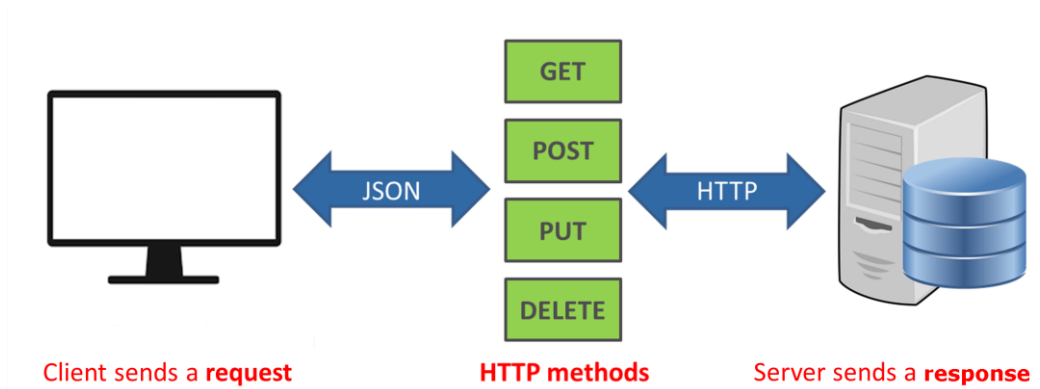


Рисунок 2.2 – Взаємодія клієнта та сервера при використанні REST [14]

Однією з головних переваг REST API є простота та універсальність. Вона не накладає жорстких обмежень на технології, які використовуються на стороні клієнта чи сервера, що робить REST сумісним із різноманітними платформами та мовами програмування. Завдяки цьому REST API широко застосовується у веброзробці, мобільних додатках, мікросервісних архітектурах та інтеграції між різними системами.

REST API використовує стандартизовані формати даних для обміну інформацією, найчастіше JSON або XML. JSON, завдяки своїй легкості та читабельності, є найбільш популярним вибором, що сприяє швидкому і зручному опрацюванню даних на стороні клієнта. Завдяки структурованості запитів і відповідей, REST API забезпечує прозору і зрозумілу комунікацію між клієнтом і сервером.

Організація REST API базується на понятті ресурсів — логічних сутностей системи, кожна з яких має унікальний URI. Операції над ресурсами відбуваються через HTTP-методи: GET для отримання даних, POST для створення, PUT або PATCH для оновлення і DELETE для видалення.

Безпека REST API зазвичай реалізується через механізми аутентифікації та авторизації, наприклад за допомогою токенів (JWT), OAuth або Basic Auth. Це забезпечує контроль доступу і захист даних від несанкціонованого використання. Крім того, для підвищення надійності часто використовують шифрування трафіку через HTTPS.

2.3 Технології для фронтенд частини

2.3.1 React

React — це сучасна і популярна бібліотека JavaScript, створена для розробки користувацьких інтерфейсів. Її основною особливістю є компонентний підхід, що дозволяє розбивати інтерфейс на невеликі, ізольовані

					ІАЛЦ.467800.003 ПЗ	Арк.
						21
Зм.	Арк.	№ докум.	Підпис	Дата		

і багаторазово використовувані частини — компоненти. Кожен компонент управляє власним станом і логікою, що робить код більш структурованим і зрозумілим. React працює на основі декларативного опису інтерфейсу, завдяки чому розробник описує, як має виглядати UI у певний момент часу, а бібліотека сама оптимально оновлює DOM, використовуючи концепцію віртуального DOM. Це значно підвищує продуктивність і забезпечує плавну взаємодію з користувачем навіть у складних додатках [6].

Ключовою перевагою React є однонаправлений потік даних, що полегшує відстеження змін у стані програми і контроль за оновленням інтерфейсу. Такий підхід робить додатки більш передбачуваними і полегшує процес налагодження. Окрім того, екосистема React надзвичайно велика — існує багато додаткових бібліотек для управління станом (наприклад, Redux або Context API), маршрутизації (React Router) та взаємодії з бекендом. Це дозволяє будувати масштабовані та підтримувані вебзастосунки, що відповідають сучасним вимогам.

Важливою перевагою React є широкий набір бібліотек і інструментів, що значно полегшують розробку додатків. Наприклад, React Router — популярна бібліотека для управління маршрутизацією, яка дозволяє створювати односторінкові застосунки з пивною навігацією без перезавантаження сторінки. Для стилізації компонентів часто використовують бібліотеки Styled Components і Emotion, які впроваджують підхід CSS-in-JS, що забезпечує кращу модульність коду та підтримку темізованих інтерфейсів [8].

2.3.2 mapbox-gl

mapbox-gl — це потужна JavaScript-бібліотека для створення інтерактивних, високопродуктивних вебкарт із використанням технології WebGL. Вона дозволяє відображати векторні карти, які можна масштабувати, панорамувати

					ІАЛЦ.467800.003 ПЗ	Арк.
						22
Зм.	Арк.	№ докум.	Підпис	Дата		

та кастомізувати у реальному часі, забезпечуючи плавну і якісну візуалізацію навіть при роботі з великими обсягами геопросторових даних.

Основною особливістю `mapbox-gl` є використання WebGL — апаратного прискорення графіки в браузері, що дає змогу рендерити карти як векторні шари, а не як статичні растрові зображення. Це означає, що карти можна масштабувати без втрати якості, а також застосовувати складні візуальні ефекти та анімації. Такий підхід значно покращує користувацький досвід і дозволяє створювати гнучкі, динамічні картографічні інтерфейси.

`mapbox-gl` підтримує роботу з різноманітними джерелами геоданих, включно з GeoJSON, векторними тайлами, растровими шарами та іншими форматами. Це дає змогу інтегрувати на карту різні типи інформації – від простих точок і ліній до складних полігонів і багат шарових структур. Бібліотека надає широкий набір інструментів для кастомізації стилю карти, зокрема можливість змінювати кольори, товщину ліній, прозорість, шрифти та інші параметри відображення.

Для взаємодії з картою `mapbox-gl` надає зручний API, який дозволяє обробляти події користувача, такі як кліки, наведення, масштабування та панорування. Це дає змогу створювати інтерактивні елементи, наприклад, спливаючі вікна з інформацією, динамічні підсвітки об'єктів або адаптивні панелі керування.

`mapbox-gl` сумісний з усіма сучасними браузерами і мобільними платформами, що забезпечує широку доступність створених картографічних додатків. Крім того, бібліотека активно розвивається, регулярно отримуючи оновлення і нові функції від спільноти розробників та компанії Mapbox [9].

2.3.3 ReactMarkdown

ReactMarkdown — це потужна та легковагова бібліотека для JavaScript, спеціально розроблена для інтеграції з React, яка дозволяє рендерити вміст,

					ІАЛЦ.467800.003 ПЗ	Арк.
						23
Зм.	Арк.	№ докум.	Підпис	Дата		

написаний у форматі Markdown, безпосередньо як React-компоненти (рис. 2.3). Вона є популярним вибором серед розробників веб-додатків, де необхідно відображати структурований текст, створений користувачами або збережений у форматі Markdown, наприклад, для описів подій, документації, статей, блогів чи коментарів. Бібліотека забезпечує простий спосіб перетворення Markdown-розмітки у валідний HTML, який інтегрується в React-додаток як JSX, зберігаючи при цьому гнучкість і безпеку.

ReactMarkdown підтримує розширення функціоналу через плагіни екосистем remark і rehype. Наприклад, за допомогою remark-gfm можна додати підтримку GitHub Flavored Markdown, що дозволяє рендерити таблиці, списки завдань чи зачеркнутий текст. Плагіни rehype дають змогу обробляти HTML, додавати підсвітку синтаксису для коду (наприклад, через rehype-highlight) або підтримувати математичні формули через remark-math і rehype-katex. Це робить бібліотеку надзвичайно гнучкою для специфічних потреб проєкту.

Ще однією сильною стороною ReactMarkdown є можливість кастомізації рендерингу. Розробники можуть перевизначати, як саме відображаються елементи Markdown (наприклад, заголовки, абзаци, списки чи посилання), замінюючи їх власними React-компонентами через проп components. Це дозволяє створювати унікальний вигляд контенту, що відповідає дизайну додатка. Наприклад, можна замінити стандартний тег `<a>` на компонент із підтримкою роутінгу, як `Link` із бібліотеки `react-router`.

Однією з ключових переваг ReactMarkdown є її безпечність за замовчуванням: бібліотека автоматично екранує HTML, запобігаючи виконанню потенційно небезпечних скриптів чи XSS-атак, що особливо важливо при роботі з контентом від користувачів. ReactMarkdown є легковаговою, що робить її ідеальною для проєктів, де важлива продуктивність. Вона не залежить від великих сторонніх бібліотек і легко інтегрується в будь-який React-проєкт. Крім того, бібліотека активно підтримується спільнотою.

					ІАЛЦ.467800.003 ПЗ	Арк.
						24
Зм.	Арк.	№ докум.	Підпис	Дата		

Завдяки своїм можливостям, ReactMarkdown широко застосовується в різних сценаріях: від простих блогів і CMS до складних документаційних платформ, таких як Docusaurus, де потрібно відображати Markdown-файли з високим рівнем кастомізації та безпеки. Бібліотека також підходить для проєктів, де контент динамічно генерується користувачами, наприклад, у форумах чи системах коментарів, забезпечуючи баланс між простотою використання та гнучкістю конфігурації [11].

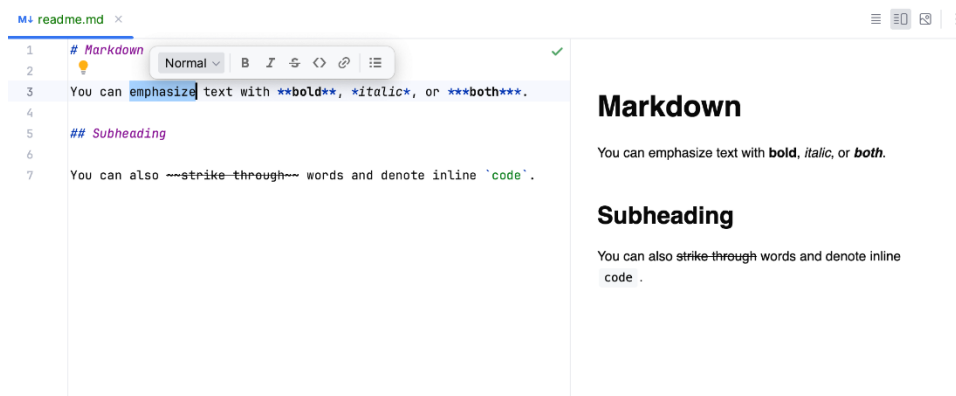


Рисунок 2.3 – Приклад тексту у форматі Markdown

2.3.4 react-router-dom

react-router-dom — це офіційна бібліотека маршрутизації для веб-додатків на базі React. Вона дозволяє створювати односторінкові додатки (SPA), де перемикання між сторінками відбувається без перезавантаження браузера, що забезпечує кращу швидкодію та плавність взаємодії.

Завдяки react-router-dom, розробники можуть визначати маршрути, компоненти сторінок, вкладені шляхи, редиректи, навігацію через кнопки чи посилання, а також керувати історією браузера. Це є критично важливо для побудови структурованих та масштабованих інтерфейсів у сучасних веб-додатках [15].

					ІАЛЦ.467800.003 ПЗ	Арк.
						25
Зм.	Арк.	№ докум.	Підпис	Дата		

2.3.5 axios

axios — це популярна JavaScript-бібліотека для здійснення HTTP-запитів до серверу. Вона побудована на основі Promise і надає простий API для надсилання та отримання даних, що робить її особливо зручною для використання у React-додатках та в середовищі Node.js.

Axios підтримує всі основні HTTP-методи (GET, POST, PUT, DELETE), автоматично обробляє JSON-дані, дозволяє встановлювати заголовки, обробку помилок та скасування запитів. Це робить бібліотеку універсальним інструментом для взаємодії з REST API або бекенд-сервісами.

2.3.5 lucide-react

lucide-react — це сучасна бібліотека векторних іконок, оптимізована для використання в середовищі React. Вона є React-обгорткою для набору відкритих іконок Lucide.

Бібліотека надає простий інтерфейс для імпорту іконок як React-компонентів, що дозволяє легко змінювати їхній розмір, колір, розташування та інтегрувати в будь-який UI (рис. 2.4). Вона ідеально підходить для створення чистого й інтуїтивно зрозумілого інтерфейсу користувача у сучасних веб-додатках [12].

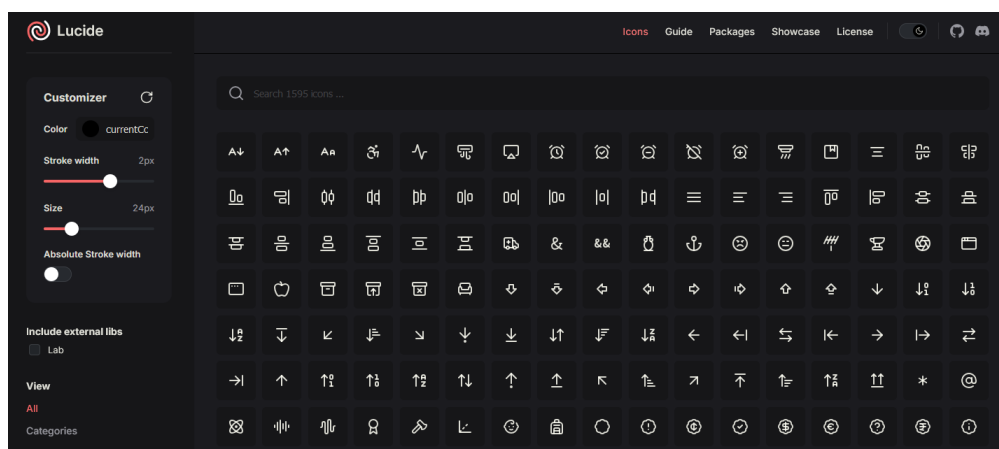


Рисунок 2.4 – Іконки Lucide

					ІАЛЦ.467800.003 ПЗ	Арк.
						26
Зм.	Арк.	№ докум.	Підпис	Дата		

2.4. Система управління базою даних

Для реалізації серверної частини проєкту було обрано систему управління базами даних SQLite, що зумовлено невисоким навантаженням, обмеженою кількістю підключень та потребою у простому розгортанні. SQLite є вбудованою реляційною СУБД, яка працює без окремого серверного процесу — уся база зберігається в одному файлі, що спрощує розгортання, резервне копіювання та міграцію між середовищами.

SQLite обробляє запити безпосередньо з файлу, не потребуючи окремих мережових з'єднань чи адміністрування користувачів. Це робить її особливо зручною для локальних і автономних застосунків, а також невеликих веб-систем, які не вимагають складної інфраструктури. Простота налаштування обмежується лише підключенням відповідної бібліотеки (наприклад, `sqlite3` у `Node.js`) і вказанням шляху до файлу бази.

Однією з ключових переваг є ефективна робота з невеликими обсягами даних. У проєкті, де основною задачею є читання історичних подій, фільтрація за категоріями чи відображення на карті, швидкий локальний доступ до даних є цілком достатнім. SQLite підтримує стандарт SQL-92, що дозволяє реалізувати всі базові операції з даними — створення таблиць, фільтрацію, агрегацію, об'єднання тощо.

СУБД не потребує адміністрування — немає потреби у створенні користувачів, налаштуванні прав доступу чи логуванні підключень. Усі запити відбуваються через API застосунку, що забезпечує централізований контроль доступу до даних і підвищує безпеку.

Ще однією перевагою є низькі системні вимоги. SQLite можна запускати навіть на малопотужних пристроях або недорогих віртуальних серверах без втрати продуктивності. Це особливо зручно для розробки, тестування або експлуатації в умовах обмежених ресурсів.

					ІАЛЦ.467800.003 ПЗ	Арк.
						27
Зм.	Арк.	№ докум.	Підпис	Дата		

У межах проєкту SQLite використовується для зберігання даних про події: координати, дати, описи, категорії, а також — за потреби — інформацію про користувачів. Структура бази реалізована як набір пов'язаних таблиць з підтримкою зовнішніх ключів і транзакцій, що гарантує цілісність та надійність збереження даних.

Інтеграція з Node.js відбувається через популярні бібліотеки (sqlite3, better-sqlite3), які забезпечують повноцінну підтримку SQL-запитів. Це дозволяє реалізовувати і клієнтську, і серверну логіку однією мовою програмування — JavaScript, що спрощує розробку та підтримку системи.

					ІАЛЦ.467800.003 ПЗ	Арк.
						28
Зм.	Арк.	№ докум.	Підпис	Дата		

ВИСНОВОК ДО РОЗДІЛУ 2

У даному розділі було здійснено огляд та обґрунтування вибору технологій для реалізації серверної та клієнтської частин дипломного проєкту.

Для серверної частини було обрано мову програмування JavaScript із середовищем виконання Node.js, що забезпечує ефективну обробку запитів та масштабованість системи. Реалізація серверного API виконана у вигляді REST сервісу, що гарантує стандартизовану взаємодію з клієнтською частиною.

Для організації бази даних використано СУБД SQLite, яка є оптимальним рішенням для зберігання структурованих даних у межах проєкту завдяки своїй легкості, простоті налаштування та достатній функціональності для роботи з обмеженими обсягами інформації.

Клієнтська частина розроблена з використанням JavaScript та бібліотеки React, що дозволяє створити інтуїтивно зрозумілий, адаптивний та високопродуктивний інтерфейс користувача. Застосування React забезпечує компонентний підхід до розробки та полегшує подальше масштабування проєкту.

Для реалізації інтерактивної карти використано бібліотеки react-map-gl і Mapbox GL JS, які забезпечують зручне відображення геопросторових даних та інтерактивність, що є ключовими для функціональності інформаційно-довідкової системи.

Таким чином, обраний технологічний стек повністю відповідає вимогам проєкту, забезпечуючи стабільність, продуктивність і зручність у подальшій експлуатації системи.

					ІАЛЦ.467800.003 ПЗ	Арк.
						29
Зм.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Специфікація вимог

3.1.1 Функціональні вимоги

Відповідно до обраного способу реалізації, переходимо до розробки системи. Інтерактивна інформаційно-довідкова система буде реалізована у вигляді інтерактивної карти з маркерами, які містять інформацію про історичні події. Робота системи повинна включати в себе авторизацію користувача, дозволяти зареєстрованим користувачам створювати, переглядати, редагувати та видаляти маркери, надавати адміністратору розширені права для управління вмістом і користувачами системи. Для зручності є необхідним пошук і фільтрування записів за ключовими словами.

3.1.2 Нефункціональні вимоги

Система повинна залишатися швидкою при завантаженні, пошуку та фільтрації маркерів, а також, при одночасному використанні системи великою кількістю користувачів, незалежно від кількості даних або кількості обробляючих запитів.

Система має бути доступною, з підтримкою роботи у сучасних браузерях (Google Chrome, Mozilla Firefox, Microsoft Edge, Safari) без необхідності встановлювати додаткове програмне забезпечення, що є важливим для зручності широкого кола користувачів.

Інтерфейс має бути інтуїтивно зрозумілим, з зрозумілими позначеннями елементів керування та підказками для нових користувачів.

					ІАЛЦ.467800.003 ПЗ	Арк.
						30
Зм.	Арк.	№ докум.	Підпис	Дата		

3.2 Проектування системи

3.2.1 Використання UML під час розробки системи

UML (Unified Modeling Language) є стандартною мовою моделювання, яка широко застосовується для візуалізації, специфікації, конструювання та документування компонентів програмного забезпечення. Його використання є ключовим під час проектування складних програмних систем, оскільки UML забезпечує єдину мову комунікації між аналітиками, розробниками, тестувальниками та іншими учасниками проєкту.

UML-моделі дозволяють створити формалізоване уявлення про структуру і поведінку майбутньої системи ще до початку її реалізації. Завдяки наявності різних типів діаграм (діаграм випадків використання, класів, діяльності, послідовностей тощо), розробники можуть чітко окреслити функціональні можливості системи, встановити зв'язки між компонентами, визначити їхню відповідальність та взаємодію.

У процесі розробки даного онлайн-сервісу UML застосовувався для побудови діаграм, які описують архітектуру компонентів, взаємодію користувача із системою, внутрішні бізнес-процеси та логіку обробки запитів. Це дозволило підвищити якість технічного проєктування, зменшити ризики на етапі реалізації та полегшити подальший супровід проєкту.

Значення UML як інструменту ефективного моделювання підтверджується численними науковими і навчальними джерелами. Зокрема, у посібнику О. М. Кудінова зазначено, що застосування UML сприяє стандартизації процесів аналізу та проєктування програмних систем, дозволяє зменшити кількість помилок при розробці та забезпечує ефективну комунікацію між усіма учасниками життєвого циклу програмного забезпечення [13].

					ІАЛЦ.467800.003 ПЗ	Арк.
						31
Зм.	Арк.	№ докум.	Підпис	Дата		

3.2.2 Діаграма варіантів використання

Діаграма прецедентів використання (англ. Use Case Diagram) є одним із ключових інструментів методології UML, що дозволяє моделювати взаємодію користувачів або зовнішніх систем з програмним продуктом. Основною метою цієї діаграми є формалізації вимог до системи, що виражені через типові сценарії її використання. Це дає змогу розробникам, замовникам і тестувальникам однаково інтерпретувати функціональні можливості системи.

На діаграмі зображуються актори (користувачі, адміністратори, зовнішні сервіси) та варіанти використання, які відображають окремі функції або сценарії взаємодії. Такий підхід дозволяє виявляти як основні, так і допоміжні процеси, що відбуваються у межах системи, а також підкреслити взаємозв'язки між ними.

У системі визначено два основних актори: «Користувач» та «Адміністратор». Перший взаємодіє із сервісом через веб-інтерфейс. Основні дії, доступні користувачу, включають реєстрацію в системі, авторизацію, додавання маркерів на карту, перегляд інформації про маркери, редагування власних маркерів та їх видалення. Користувач також може здійснювати пошук та фільтрацію маркерів за різними критеріями, такими як ключові слова, теги або дати.

«Адміністратор», у свою чергу, має доступ до функцій, що забезпечують контроль якості контенту та стабільну роботу сервісу. Зокрема, він може здійснювати модерацію маркерів, що надходять від користувачів, редагувати будь-які маркери в системі та за потреби видаляти їх. Адміністратор також має доступ до всіх функцій звичайного користувача.

Таким чином, діаграма прецедентів використання дозволяє систематизувати функціональні можливості системи маркерів на карті, чітко розмежувати права доступу між ролями та забезпечити узгодження логіки роботи системи на етапі розробки.

					ІАЛЦ.467800.003 ПЗ	Арк.
						32
Зм.	Арк.	№ докум.	Підпис	Дата		

Тепер стисло опишемо прецеденти використання. Діаграма прецедентів використання наведена у Додатку А.

Реєстрація користувачів

Актор: користувач

Попередні умови: користувач має надати необхідні дані, наприклад, адресу електронної пошти, яка буде використовуватися для реєстрації.

Мета: створення облікового запису для користувача.

Сценарій:

- Користувач відкриває систему. Система відображає форму реєстрації. Користувач вводить дані. Система створює обліковий запис та надає доступ до системи.

Результат: користувач зареєстрований та може увійти у систему.

Авторизація користувачів

Актор: користувач

Попередні умови: користувач має зареєстрований обліковий запис.

Мета: доступ до облікового запису.

Сценарій:

- Користувач відкриває систему. Система відображає форму входу. Користувач вводить дані. Система перевіряє дані та надає доступ до облікового запису.

Результат: користувач має доступ до свого облікового запису та функціоналу системи.

Додавання маркеру

Актор: користувач

Попередні умови: користувач авторизований

Мета: додати маркер з інформацією.

Сценарій:

					ІАЛЦ.467800.003 ПЗ	Арк.
						33
Зм.	Арк.	№ докум.	Підпис	Дата		

- Користувач відкриває систему. Натискає кнопку для додавання мітки. Система відображає форму для заповнення. Користувач заповнює форму та підтверджує. Маркер додається до системи та відправляється на модерацію.

Результат: маркер створений та очікує на підтвердження адміністрацією.

Перегляд маркеру

Актор: користувач

Попередні умови: користувач авторизований

Мета: переглянути інформацію маркеру, створеного ним або іншими користувачами.

Сценарій:

- Користувач натискає на маркер. Система відображає форму маркера.

Результат: користувач ознайомлений з інформацією маркеру.

Редагування власного маркеру

Актор: користувач

Попередні умови: користувач авторизований, маркер був створений ним.

Мета: редагувати свій маркер

Сценарій:

- Користувач натискає на свій маркер. Система відображає форму маркера. Користувач вносить зміни та підтверджує. Нова версія маркеру відправляється на підтвердження адміністрацією.

Результат: маркер відредагований та очікує підтвердження адміністрацією.

Видалення власного маркеру

Актор: користувач

Попередні умови: користувач авторизований, маркер був створений ним.

Мета: видалити свій маркер

Сценарій:

- Користувач натискає на свій маркер. Система відображає форму маркера. Користувач натискає на кнопку видалення. Маркер видаляється з карти.

Результат: маркер видалений.

					ІАЛЦ.467800.003 ПЗ	Арк.
						34
Зм.	Арк.	№ докум.	Підпис	Дата		

Пошук та фільтрація маркерів

Актор: користувач

Попередні умови: користувач авторизований.

Мета: знайти або відфільтрувати маркери

Сценарій:

- Користувач відкриває панель пошуку. Вводить ключові слова, теги, дати або діапазон дат. Система відображає маркери, що відповідають критеріям пошуку.

Результат: користувач бачить маркери, що відповідають критеріям пошуку.

Адміністрування маркерів

Модерація маркерів

Актор: адміністратор

Попередні умови: адміністратор авторизований

Мета: перевірити та підтвердити маркери, що були додані користувачами.

Сценарій:

- Адміністратор відкриває панель модерації. Система відображає список маркерів, що потребують перевірки. Адміністратор переглядає маркер, підтверджує чи видаляє його.

Результат: маркер стає доступний на карті або видається.

Редагування будь-якого маркеру

Актор: адміністратор

Попередні умови: адміністратор авторизований

Мета: редагувати будь-якого маркеру

Сценарій:

- Адміністратор натискає на маркер. Система відображає форму маркера. Адміністратор вносить зміни та підтверджує.

Результат: маркер відредагований

					ІАЛЦ.467800.003 ПЗ	Арк.
						35
Зм.	Арк.	№ докум.	Підпис	Дата		

Видалення будь-якого маркеру

Актор: адміністратор

Попередні умови: користувач авторизований, маркер був створений ним.

Мета: видалити свій маркер

Сценарій:

- Адміністратор натискає на маркер. Система відображає форму маркера.

Адміністратор натискає на кнопку видалення. Маркер видаляється з карти.

Результат: маркер видалений.

3.2.3 Діаграма послідовності

Діаграма послідовності (англ. Sequence Diagram) – це від діаграм UML, який використовується для моделювання динамічної поведінки системи, демонструючи порядок обміну повідомлення між об'єктами або компонентами в межах певного сценарію. Вона відображає послідовність взаємодії у часі, допомагаючи зрозуміти, як саме відбувається обробка запитів та виконання операцій у системі [15].

Діаграма демонструє послідовність викликів методів між компонентами інтерактивної карти: від додавання нового маркера користувачем, через авторизацію та збереження даних у базі, до завантаження маркерів для видимої області карти. Система забезпечує швидкий доступ до геопросторових даних через прямі запити до бази даних з фільтрацією за координатами. Взаємодія відбувається в реальному часі, що дозволяє користувачам миттєво бачити нові маркери та переглядати детальну інформацію про них при кліку на карті.

Діаграма послідовності є візуальним підтвердженням архітектурних рішень, описаних у попередніх розділах, і служить інструментом для аналізу і оптимізації процесів у системі, дозволяючи чітко відстежувати кожен крок від запиту користувача до отримання готового результату. Діаграма послідовності наведена у Додатку Б.

					ІАЛЦ.467800.003 ПЗ	Арк.
						36
Зм.	Арк.	№ докум.	Підпис	Дата		

3.2.3 Діаграма класів

Діаграма класів є однією з найважливіших діаграм в UML, що використовується для статичного моделювання структури об'єктно-орієнтованих систем. Вона показує класи системи, їх атрибути, методи та взаємозв'язок між ними.

Клас в об'єктно-орієнтованому програмуванні представляє собою шаблон або «креслення» для створення об'єктів, що описує їх структуру (атрибути) та поведінку (методи). Класи забезпечують інкапсуляцію даних та функціональність, дозволяючи створювати модульні та повторно використовувані компоненти системи.

Діаграма класів наведена у Додатку В.

Опис основних компонентів, що зображені на діаграмі класів:

MapView – головний компонент інтерактивної карти, що координує взаємодію між усіма підсистемами. Відповідає за ініціалізацію карти, обробку подій користувача та синхронізацію стану маркерів з UI компонентами.

useMapbox – хук для управління картою Mapbox, включаючи ініціалізацію, налаштування мови інтерфейсу та обробку помилок завантаження. Забезпечує адаптивність карти при зміні розміру панелей та автоматичне перемальовування при потребі.

useMapMarkers – керує життєвим циклом маркерів на карті, включаючи їх створення, оновлення та видалення. Обробляє модерацію маркерів, встановлює прозорість для немодерованих об'єктів та забезпечує інтерактивність через обробники подій кліку.

useMapInteractions – обробляє взаємодію користувача з картою, включаючи режим додавання нових маркерів, обробку кліків по карті та керування модальними вікнами створення маркерів. Координує зміну курсору та стану інтерфейсу.

					ІАЛЦ.467800.003 ПЗ	Арк.
						37
Зм.	Арк.	№ докум.	Підпис	Дата		

ApiClient – централізований HTTP клієнт з автоматичним додаванням токенів авторизації та обробкою помилок. Включає інтерсептори для автоматичного оновлення токенів та перенаправлення на сторінку входу при втраті авторизації.

authAPI, markersAPI, moderationAPI – спеціалізовані сервіси для роботи з різними аспектами системи: аутентифікація користувачів, CRUD операції з маркерами та процеси модерації контенту відповідно. Забезпечують типізовані методи для всіх API операцій.

MapInfo та **CreateMarkerModal** – UI компоненти для відображення інформації про карту та створення нових маркерів. MapInfo показує статистику та деталі вибраних маркерів, а CreateMarkerModal забезпечує інтерфейс для додавання нових точок на карту.

Auth та **Marker** – допоміжні утіліти для роботи з авторизацією та візуальним представленням маркерів. Auth керує токенами та сесіями користувачів, а Marker створює HTML елементи маркерів з відповідними іконками та обробниками подій.

3.3 Реалізація фронтенд частини

3.3.1 Сторінка реєстрації та авторизації

Перед початком роботи з системою користувач повинен зареєструватися в системі, якщо в нього немає облікового запису, в іншому випадку – авторизуватися.

Сторінки реєстрації та авторизації мають форми зі схожою структурою: поля для введення електронної адреси та паролю. Форма реєстрації додатково включає поле для підтвердження паролю. Для зручності додана можливість швидко перемикатися між сторінками реєстрації та авторизації.

Вигляд сторінок зображено на рис. 3.1 та 3.2.

					ІАЛЦ.467800.003 ПЗ	Арк.
						38
Зм.	Арк.	№ докум.	Підпис	Дата		

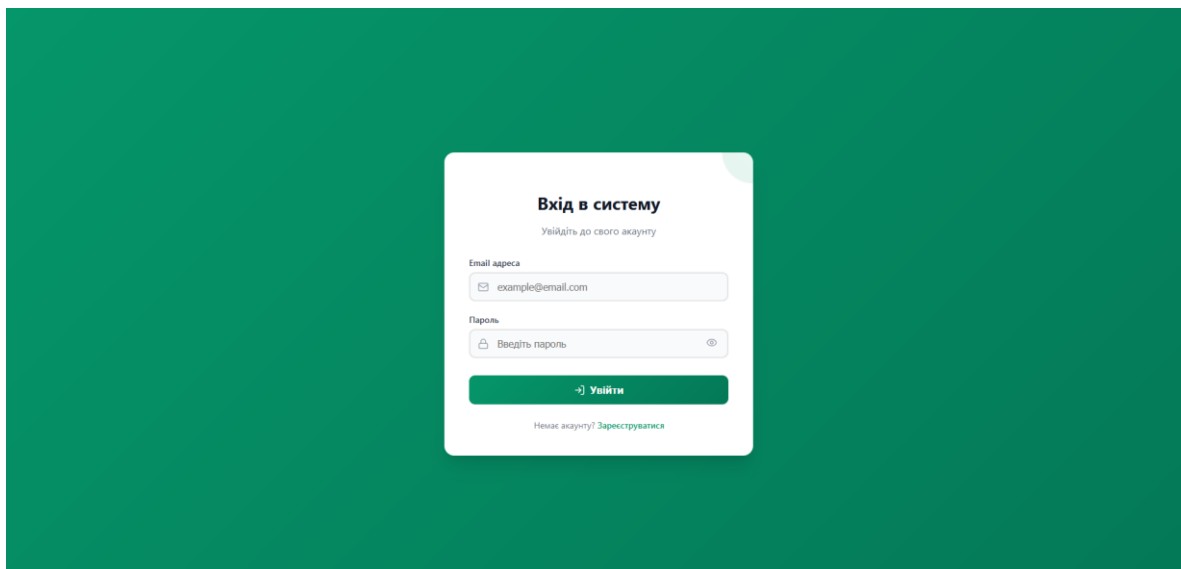


Рисунок 3.1 – Сторінка авторизації

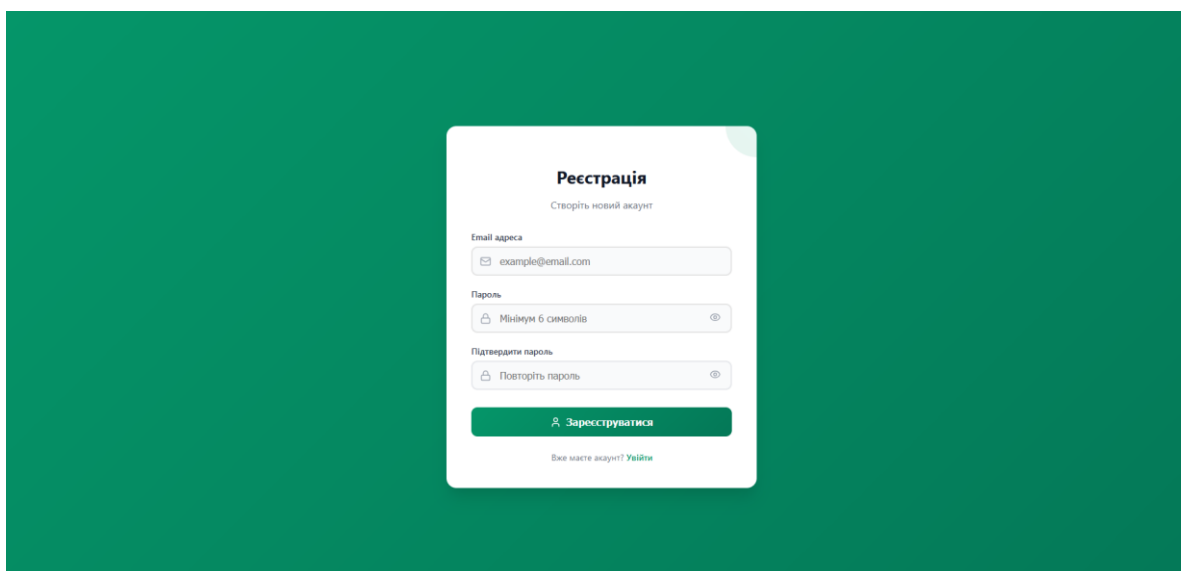


Рисунок 3.2 – Сторінка реєстрації

3.3.2 Інтерфейс користувача

Після успішної авторизації користувач потрапляє до основного інтерфейсу системи. Центральним елементом інтерфейсу є інтерактивна карта, що реалізована за допомогою Mapbox. На мапі знаходяться маркери, додані іншими користувачами у систему. Інтерфейс має функціональні елементи: кнопки «Додати мітку» для створення нового маркера і «Вийти» для виходу з

поточного облікового запису, дві згорнуті панелі «Легенда мапи» з допоміжною інформацією про типи маркерів і «Пошук і фільтри» для пошуку і фільтрації існуючих маркерів. Вигляд інтерфейсу користувача представлено на рис. 3.3.

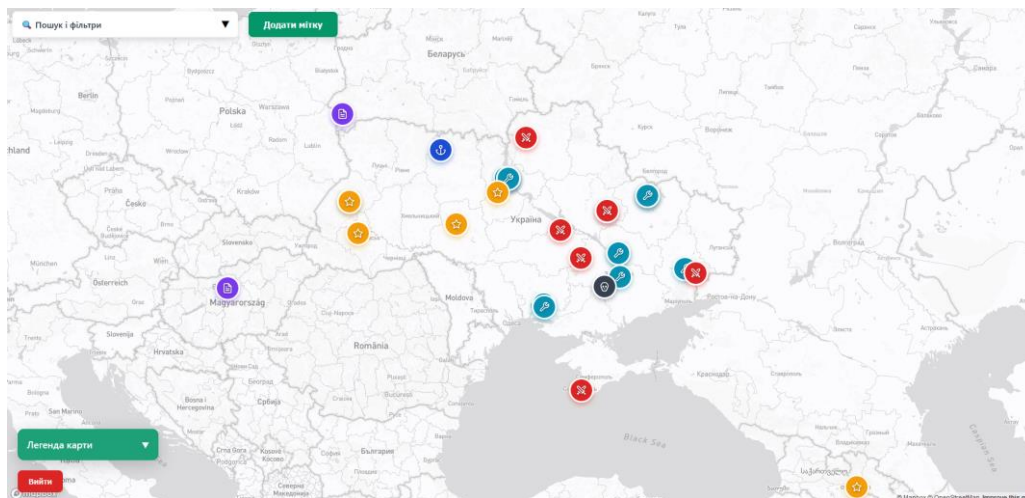


Рисунок 3.3 – Інтерфейс користувача

3.3.3 Інтерфейс адміністратора

Інтерфейс адміністратора ідентичний інтерфейсу користувача, за винятком додаткової кнопки «Модерація», яка з'являється після авторизації з облікового запису адміністратора. Ця кнопка відкриває сторінку модерації. Вигляд інтерфейсу адміністратора представлено на рис. 3.4.

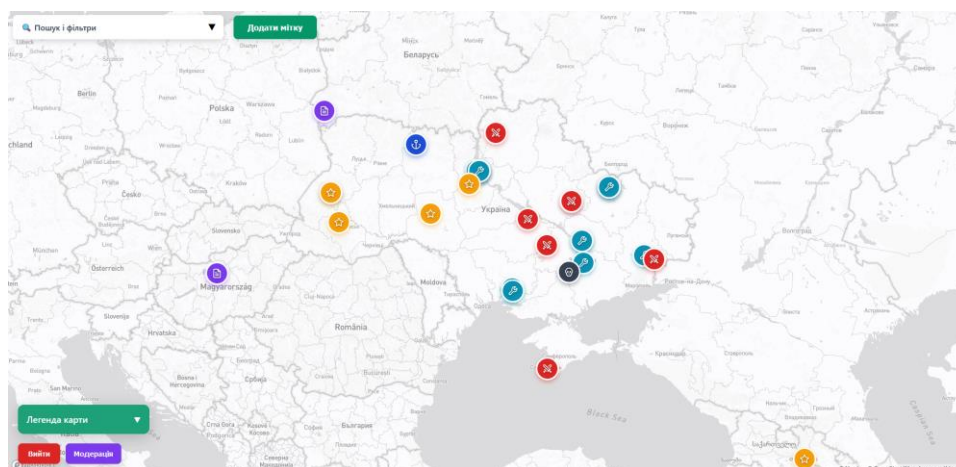


Рисунок 3.4 – Інтерфейс адміністратора

					ІАЛЦ.467800.003 ПЗ	Арк.
						40
Зм.	Арк.	№ докум.	Підпис	Дата		

3.3.4 Сторінка модерації

Сторінка модерації доступна тільки адміністраторам та відображає всі мітки, що очікують на перевірку. Вигляд сторінки модерації зображено на рис. 3.5.

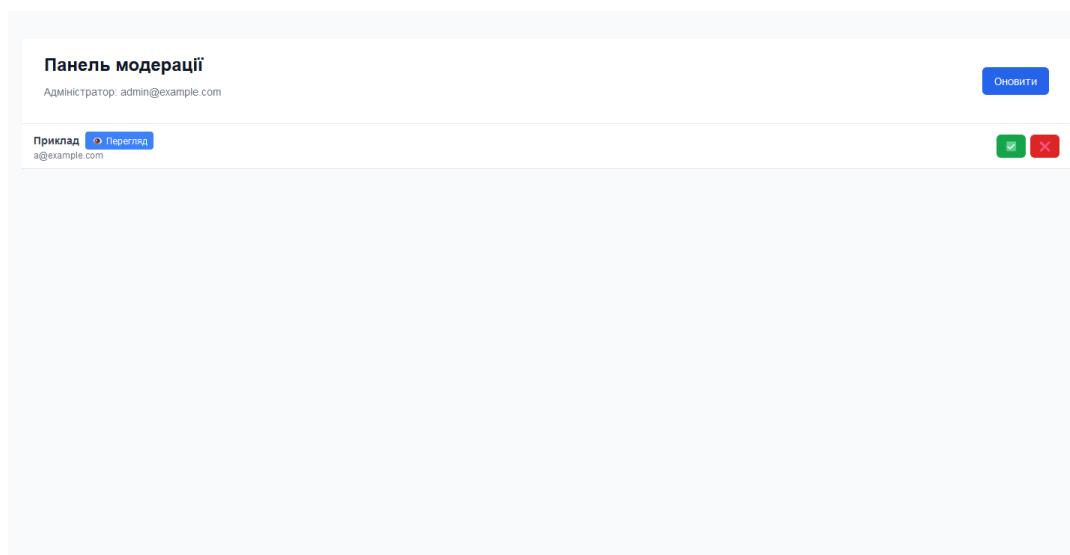


Рисунок 3.5 – Сторінка модерації

На сторінці також присутня кнопка «Оновити», що дозволяє оновити список маркерів, що ще не пройшли модерацію.

3.3.5 Вікно додавання маркера

Натискання на кнопку «Додати мітку» активує режим додавання маркера на мапу. Курсор змінюється на хрестик, а кнопка «Додати мітку» трансформується в кнопку «Скасувати». Після натиснення на вибраному місці, відкриється вікно додавання маркера, що містить:

- координати маркера (широта та довгота, визначаються автоматично)
- поле для вибору дати події
- текстові поля для введення назви, тегів та опису маркера

					ІАЛЦ.467800.003 ПЗ	Арк.
						41
Зм.	Арк.	№ докум.	Підпис	Дата		

- вибір із десяти кастомізованих іконок маркера
- функціональні кнопки «Зберегти» та «Скасувати»

Вікно додавання маркера зображено на рис. 3.6.

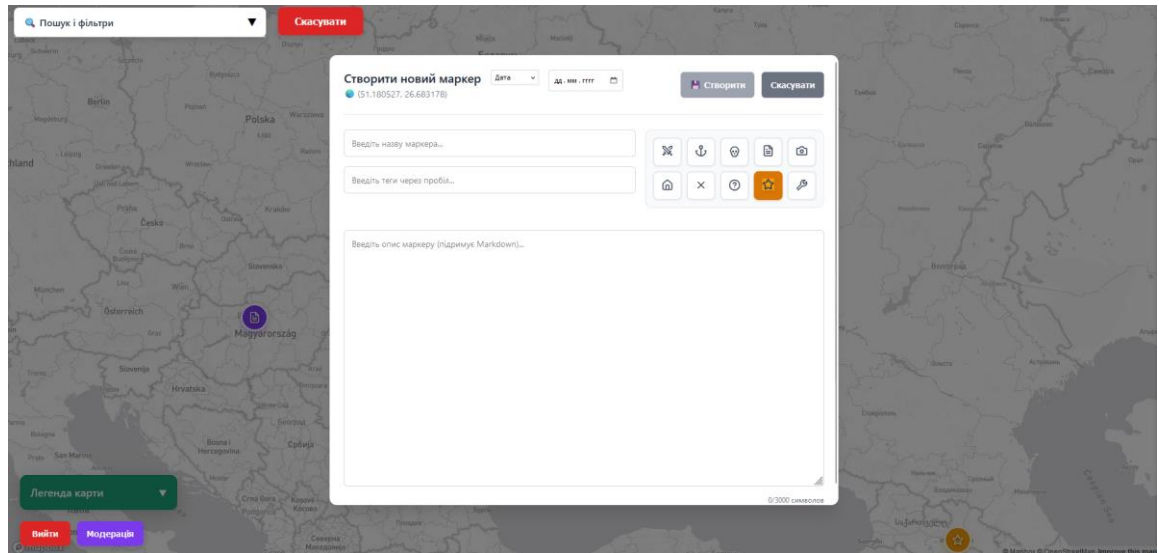


Рисунок 3.6 – Вікно додавання маркера

Вибір дати події

Так як історичні події можуть мати різний формат запису дат, то було реалізовано гнучкий вибір формату дат: «Без дати», «Рік», «Дата», «Роки», «Дати». Всі варіанти вигляду поля для дати зображено на рис. 3.7. При виборі формату «Дата» або «Дати», користувач може обрати дату, натиснувши на іконку календаря праворуч поля (рис. 3.8).

Без дати ▾	Роки ▾	от ▾
Рік ▾	1990 ▾	до ▾
Дата ▾	ДД . ММ . ГГГГ 📅	Дати ▾
		ДД . ММ . ГГГГ 📅

Рисунок 3.7 – Варіанти вигляду поля для введення дати залежно від формату

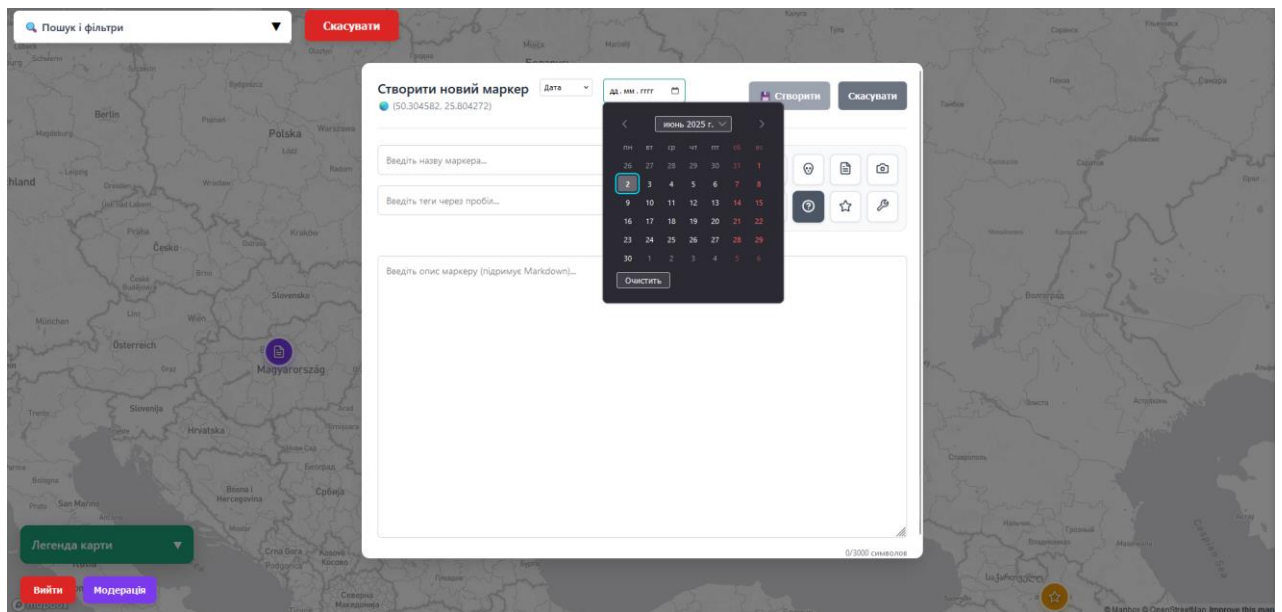


Рисунок 3.8 – Вибір дати за допомогою календарю

Текстові поля

Кожен маркер має три текстові поля: для назви, тегів та опису. Назва та опис є обов'язковими, теги можна залишити порожніми. Поле опису має обмеження до 3000 символів, назви і тегів – 100. Поле опису підтримує форматування за допомогою Markdown, що дозволяє додавати структурований текст, списки, заголовки, тощо.

Markdown — це легковагова мова розмітки, призначена для створення структурованого тексту з мінімальними зусиллями та подальшого його перетворення у формати, придатні для відображення в веб-інтерфейсах, зокрема HTML. Вона забезпечує простий синтаксис для форматування тексту, включаючи заголовки, списки, посилання, цитати, таблиці та інші елементи, за допомогою текстових позначок, таких як # для заголовків або * для маркованих списків. У рамках проекту Markdown використовується для створення та форматування описів міток, що додаються користувачами на інтерактивну карту, забезпечуючи зручність введення інформації та її структуроване представлення в інтерфейсі системи.

					ІАЛЦ.467800.003 ПЗ	Арк.
						43
Зм.	Арк.	№ докум.	Підпис	Дата		

Застосування Markdown у проєкті сприяє спрощенню процесу внесення контенту користувачами, які не володіють технічними навичками, та забезпечує уніфіковане представлення даних. Завдяки компактності формату та підтримці бібліотеками, такими як marked або showdown, текст у Markdown легко конвертується в HTML для відображення на веб-сторінках. Це дозволяє ефективно обробляти описи міток на серверній частині системи, забезпечуючи їх читабельність і структурованість. Крім того, використання Markdown підтримує процес модерації, оскільки адміністратори можуть легко перевіряти та редагувати форматований контент, що відповідає вимогам до якості та достовірності інформації в системі. Таким чином, інтеграція Markdown підвищує зручність користування системою та сприяє її доступності для широкої аудиторії, включаючи освітні та науково-дослідницькі спільноти.

Іконки

Маркер може мати одну з 10 варіантів іконок. Кожна іконка відповідає певному типу історичної події. Список іконок вказані на рис. 3.9.

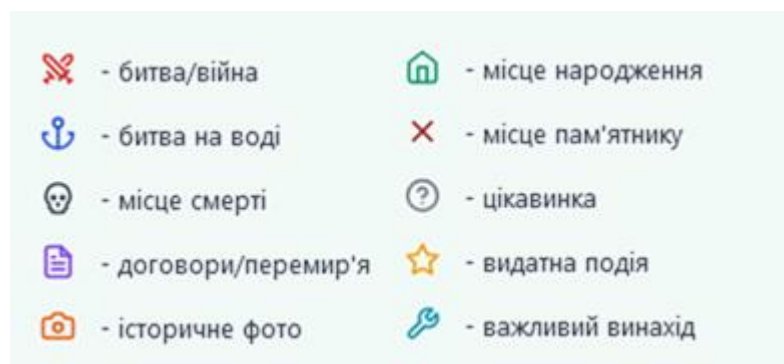


Рисунок 3.9 – Список доступних іконок

Функціональні кнопки

Кнопка «Зберегти» не активна, доки користувач не заповнить всі обов'язкові поля. Після збереження маркер записується у базу даних з усіма вказаними параметрами. На карті маркер відображається сірим кольором і має

підказку «Ця мітка очікує на модерацію» поки адміністратор не підтвердить його. На рис. 3.10 зображений маркер, що ще не пройшов модерацію.

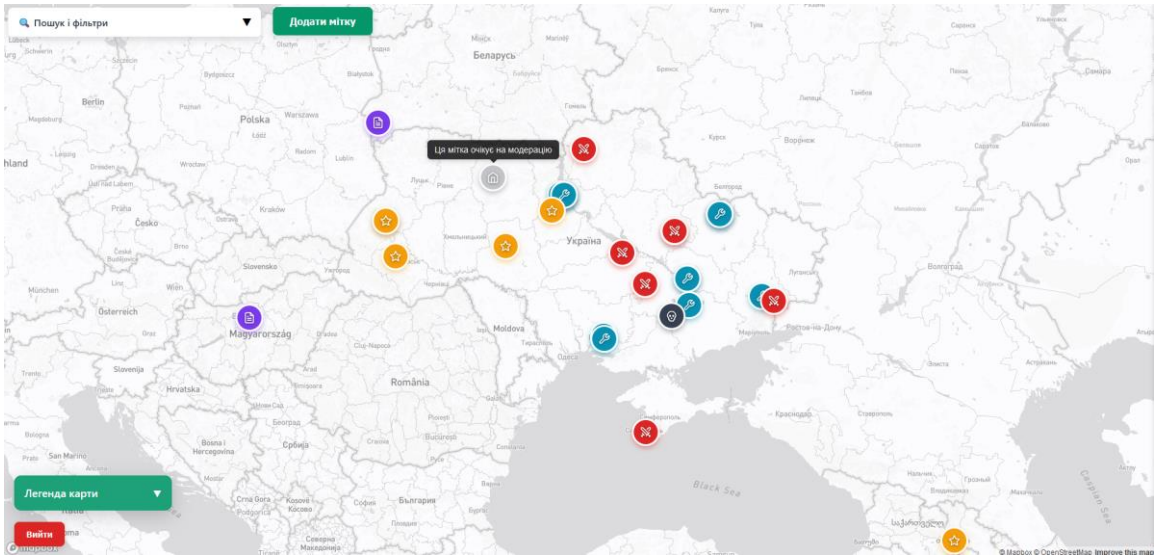


Рисунок 3.10 – Вигляд маркера, що не пройшов модерацію

3.3.6 Панель маркера

Панель маркера можна відкрити, натиснувши на будь-який маркер, який вже є погоджений адміністратором. Зовнішній вигляд панелі представлений на рис. 3.11.

Панель маркера отримує з бази даних інформацію про маркер: рік, назву, теги, опис, координати, автора маркеру, дату створення маркеру. Також, якщо маркер був створений поточним користувачем або користувач є адміністратором, наявні кнопки редагування та видалення маркеру. Натискання на кнопку редагування відкриває вікно редагування маркера, а на видалення – маркер видаляється з мапи та бази даних.

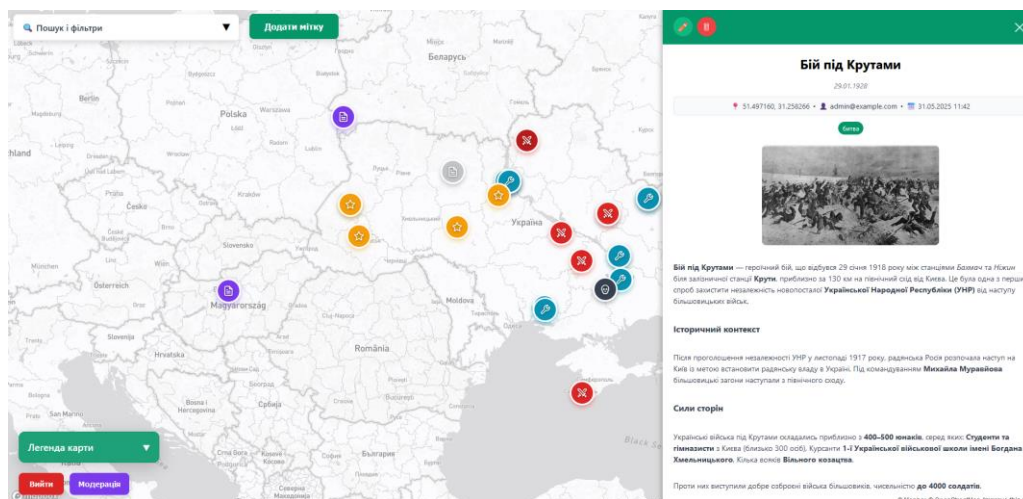


Рисунок 3.11 – Панель маркера

3.3.7 Вікно редагування маркера

Вікно редагування маркера дозволяє внести зміни до даних вже існуючого маркера. Для відкриття вікна редагування маркера потрібно натиснути на кнопку редагування маркера, що знаходиться або у панелі маркера, або на сторінці модератора.

Вікно редагування маркера ідентичне вікну додавання маркера, але поля вже заповненні даними з бази даних. При натисненні кнопки «Зберегти», дані маркера перезаписуються у базі даних.

На рис. 3.12 зображено вигляд вікна.

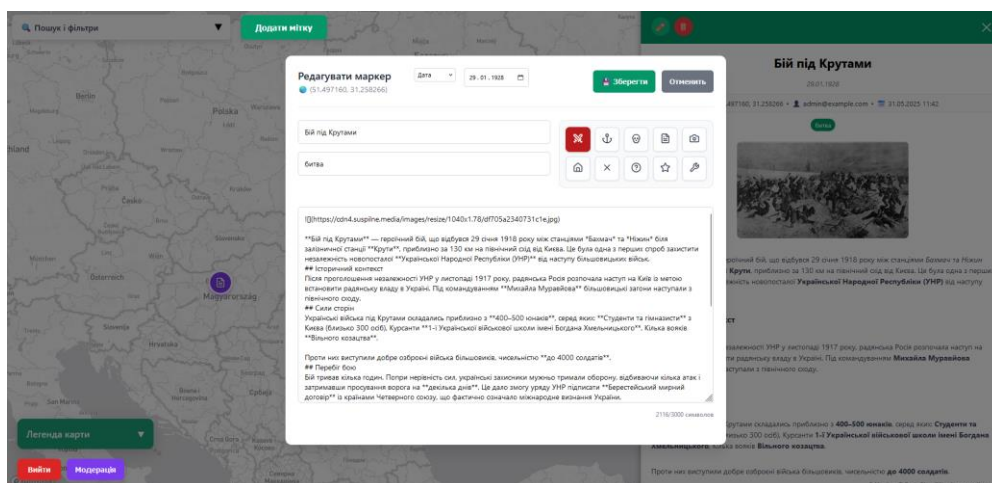


Рисунок 3.12 – Вікно редагування маркера

3.3.8 Панель пошуку та фільтрації маркерів

Панель пошуку та фільтрації за замовчуванням згорнута і розгортається після натискання по неї.

Панель складається з двох вкладок: «Фільтри» та «Результати». В «Фільтрах» можна вказати параметру пошуку, в «Результатах» - переглядати відфільтровані маркери у вигляді списку.

Фільтрувати маркери можна за текстовим записом, за часовим проміжком дати події, за часовим проміжком дати створення маркеру або за автором маркера. Також можна виділити маркери, що були створенні поточним користувачем. Для скидання всіх фільтрів передбачено кнопку «Скинути все».

Маркери, що не відповідають умовам, будуть зменшувати свою прозорість, що робить їх менш помітними. Результати пошуку по слову «битва» зображені на рис. 3.13.

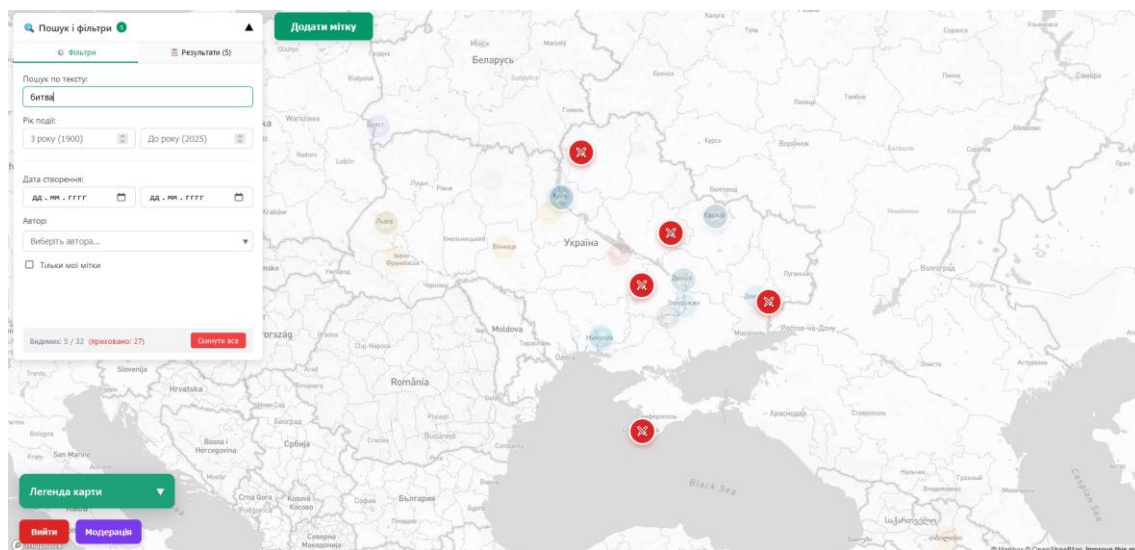


Рисунок 3.13 – Вигляд результату пошуку по слову «битва»

3.3.9 Панель легенди мапи

Панель легенди мапи згорнута за замовчуванням та розгортається після натискання. Вона є підказкою для нових користувачів. В панелі вказані всі

існуючі типи маркерів, їх значення, кількість маркерів у системі та кількість маркерів, що були додані поточним користувачем. На рис. 3.14 зображено вигляд панелі.

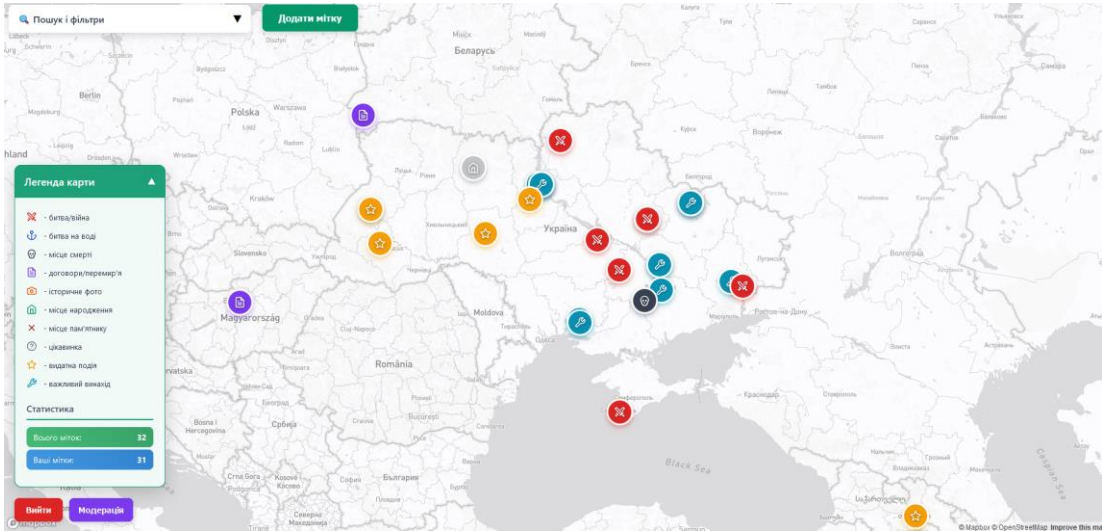


Рисунок 3.14 – Панель легенди мапи

3.4 Реалізація бекенд частини

3.4.1 База даних

База даних складається з двох основних таблиць: users та markers (рис. 3.15)

Таблиця users зберігає всіх користувачів системи. Призначення полів:

- email – електронна пошта (логін)
- password – хешований пароль
- created_at – дата та час реєстрації

Таблиця markers зберігає всі маркери мапи. Призначення полів:

- latitude/longitude – координати
- title – назва маркера
- description – опис маркера
- icon – іконка маркера
- user_email – автор маркера, зв'язок з users

- event_date – дата історичної події
- tags – теги
- is_moderated – статус модерації
- created_at – дата створення мітки

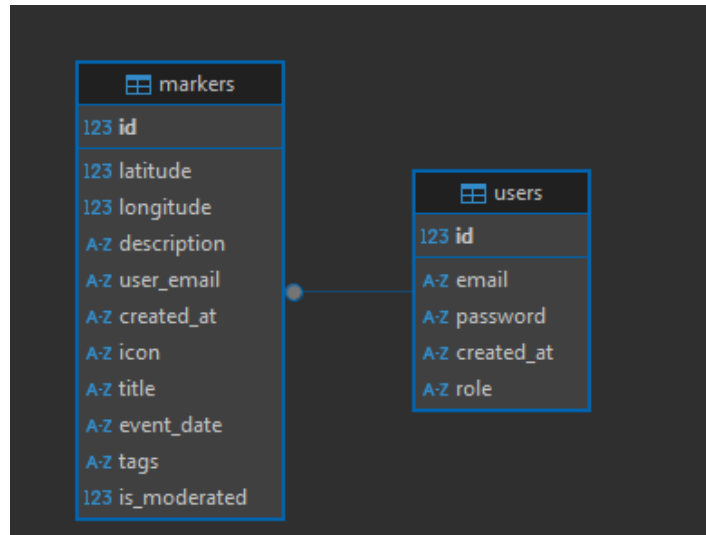


Рисунок 3.15 – Структура бази даних

Спосіб ініціалізації бази даних вказаний на рис. 3.16.

```

1  const sqlite3 = require('sqlite3').verbose();
2  const path = require('path');
3
4  const dbPath = path.join(__dirname, 'users.sqlite');
5
6  const db = new sqlite3.Database(dbPath, (err) => {
7    if (err) {
8      console.error('Помилка підключення до бази даних:', err.message);
9    } else {
10     console.log('Підключено до SQLite бази даних.');

```

Рисунок 3.16 – Ініціалізація бази даних

3.4.2 Схема авторизації

Система авторизації базується на JWT токенах.

JWT – це стандарт токена доступу на основі JSON. Як правило, використовується для передачі даних для авторизації в клієнт-серверних програмах. Токени створюються сервером, підписуються секретним ключем і передаються клієнту, який надалі використовує цей токен для підтвердження своєї особи.

При реєстрації після валідації даних, що ввів користувач, відбувається хешування паролів за допомогою bcrypt (рис. 3.17) та новий користувач додається в базу даних з хешованим паролем.

```
const { email, password } = req.body;

// Валідація даних
if (!email || !password) {
  return res.status(400).json({ error: 'Email та пароль обов\'язкові' });
}

if (password.length < 6) {
  return res.status(400).json({ error: 'Пароль повинен містити мінімум 6 символів' });
}

db.get('SELECT id FROM users WHERE email = ?', [email], async (err, row) => {
  if (err) {
    return res.status(500).json({ error: 'Помилка бази даних' });
  }

  if (row) {
    return res.status(400).json({ error: 'Користувач з таким email вже існує' });
  }

  // Хешування пароля
  const hashedPassword = await bcrypt.hash(password, 10);

  db.run('INSERT INTO users (email, password) VALUES (?, ?)',
    [email, hashedPassword],
    function(err) {
      if (err) {
        return res.status(500).json({ error: 'Помилка створення користувача' });
      }

      res.status(201).json({
        message: 'Користувач успішно зареєстрований',
        userId: this.lastID
      });
    });
});
```

Рис. 3.17 – Процес реєстрації та хешуванням пароля

					ІАЛЦ.467800.003 ПЗ	Арк.
						50
Зм.	Арк.	№ докум.	Підпис	Дата		

При вході користувача, відбувається валідація даних, порівнюється введений пароль з хешованим і генерується JWT токен (рис. 3.18). Токен має термін дії 1 година.

```
if (!email || !password) {
  return res.status(400).json({ error: 'Email та пароль обов\'язкові' });
}

db.get('SELECT * FROM users WHERE email = ?', [email], async (err, user) => {
  if (err) {
    return res.status(500).json({ error: 'Помилка бази даних' });
  }

  if (!user) {
    return res.status(401).json({ error: 'Неправильний email або пароль' });
  }

  const validPassword = await bcrypt.compare(password, user.password);

  if (!validPassword) {
    return res.status(401).json({ error: 'Неправильний email або пароль' });
  }

  // Створення JWT токена
  const token = jwt.sign({ email: user.email, role: user.role }, JWT_SECRET, { expiresIn: '1h' });

  res.json({
    message: 'Вхід успішний',
    token: token,
    user: {
      id: user.id,
      email: user.email
    }
  });
});
```

Рис. 3.18 – Використання секретного ключу для підпису JWT-токенів

3.4.3 Реалізація взаємодії між клієнтом і сервером

Сервер обробляє HTTP запити від клієнта, авторизує користувачів, управляє даними в базі даних і повертає структуровані JSON-відповіді. Клієнтська частина ініціює запити та обробляє отримані відповіді для відображення інтерфейсу.

Детальний список endpoints:

Аутентифікація:

- POST /api/auth/register – призначений для реєстрації нового користувача в системі.

					ІАЛЦ.467800.003 ПЗ	Арк.
						51
Зм.	Арк.	№ докум.	Підпис	Дата		

- POST /api/auth/login – авторизація користувача та отримання JWT токена.

Операції з маркерами:

- GET /api/markers – отримання списку всіх маркерів.
- POST /api/markers – створення нового маркеру.
- PUT /api/markers/:id – оновлення існуючого маркеру.
- DELETE /api/markers/:id – видалення маркеру.

Модерація:

- GET /api/markers/unmoderated – отримання списку немодерованих маркерів (тільки для адміністраторів).
- PUT /api/markers/:id/moderate – модерація маркера (схвалення або відхилення)

					ІАЛЦ.467800.003 ПЗ	Арк.
						52
Зм.	Арк.	№ докум.	Підпис	Дата		

ВИСНОВОК ДО РОЗДІЛУ 3

У даному розділі було реалізовано ключовий функціонал системи, що забезпечує інтерактивну взаємодію користувача з картою та можливість додавання, редагування й перегляду маркерів з описом історичних подій.

Додавання маркерів реалізовано за допомогою інтеграції з Mapbox GL JS, що дозволяє фіксувати координати обраної точки на карті. Кожен маркер містить метадані, зокрема назву, опис, дату події та належність до певного періоду історії. Ці дані надсилаються на сервер через HTTP-запит методом POST, де обробляються API, реалізованим на Node.js, та зберігаються у базі даних SQLite.

Редагування маркерів передбачає двосторонню взаємодію з сервером: при відкритті інформаційної панелі здійснюється GET-запит для отримання актуальних даних, а зміни передаються назад на сервер через PUT-запит, після чого оновлюються у базі. Таким чином забезпечується цілісність і актуальність даних.

Для зручності користувача реалізовано інформаційну панель, створену у вигляді окремого React-компонента. Вона динамічно відкривається при натисканні на маркер та дозволяє здійснювати редагування даних у зручному інтерфейсі.

Таким чином, реалізований функціонал маркерів забезпечує повноцінну взаємодію з картою, зберігаючи історичну інформацію та створюючи гнучку основу для розширення можливостей системи.

					ІАЛЦ.467800.003 ПЗ	Арк.
						53
Зм.	Арк.	№ докум.	Підпис	Дата		

ВИСНОВКИ

У ході виконання дипломної роботи було досягнуто поставленої мети — розроблено інтерактивну інформаційно-довідкову систему з військової історії України у вигляді веб-додатку, що забезпечує зручний доступ до структурованої інформації та її ефективне візуальне представлення за допомогою інтерактивної карти.

Було здійснено порівняльний аналіз сучасних картографічних сервісів, на основі якого обрано оптимальне рішення — платформу Mapbox, що поєднує гнучкість налаштувань, продуктивність і широкі можливості інтерактивності.

Було обґрунтовано вибір технологічного стеку, до якого увійшли JavaScript, Node.js, React і SQLite. Обрані інструменти забезпечують ефективну побудову клієнт-серверної архітектури, швидку розробку, масштабованість і зручність підтримки проєкту.

Програмна реалізація включала розробку REST API, створення інтерфейсу користувача з можливістю перегляду, додавання та редагування маркерів, створення взаємодії з базою даних та інтеграцію з картографічною платформою. Особливу увагу приділено зручності користування, адаптивності інтерфейсу та збереженню цілісності даних.

Таким чином, результати роботи підтверджують ефективність обраних підходів та технологій для створення інформаційно-довідкової системи історичного спрямування. Реалізоване програмне забезпечення може бути використане як основа для подальшого розширення функціоналу, зокрема додавання пошуку, фільтрації за історичними періодами, можливості мультимедійного супроводу та підтримки багатокористувацького доступу.

					ІАЛЦ.467800.003 ПЗ	Арк.
						54
Зм.	Арк.	№ докум.	Підпис	Дата		

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Google Maps Platform [Електронний ресурс]. – 2025. – Режим доступу до ресурсу: <https://mapsplatform.google.com/>
2. Mapbox vs Google Maps — What are the differences?. [Електронний ресурс]. – Режим доступу до ресурсу – <https://www.softkraft.co/mapbox-vs-google-maps>
3. Mapbox [Електронний ресурс]. – 2025. – Режим доступу до ресурсу: <https://www.mapbox.com/>
4. JavaScript from Beginner to Professional. archive. [Електронний ресурс]. Режим доступу до ресурсу: https://dn721809.ca.archive.org/0/items/Vismay/1425_JavaScript-from-Beginner-to-Professional.pdf
5. Introduction to Node.js [Електронний ресурс] // Node. – 2023. – Режим доступу до ресурсу: <https://nodejs.org/en/learn/getting-started/introduction-to-nodejs>.
6. ReactDOM [Електронний ресурс] // React. – 2024. – Режим доступу до ресурсу: <https://legacy.reactjs.org/docs/react-dom.html>.
7. About SQLite [Електронний ресурс] // sqlite. – 2024. – Режим доступу до ресурсу: <https://www.sqlite.org/about.html>.
8. Посібник: знайомство з React [Електронний ресурс]. – Режим доступу до ресурсу: <https://uk.reactjs.org/tutorial/tutorial.html>
9. Использование mapbox-gl в React и Next.js [Електронний ресурс] – // habr. – 2021. – Режим доступу до ресурсу: <https://habr.com/ru/articles/565636/>
10. Java vs JavaScript [Електронний ресурс] // Jaydevs. – Режим доступу до ресурсу: <https://jaydevs.com/java-vs-javascript/>.
11. React Markdown [Електронний ресурс] // GitHub / remarkjs. – Режим доступу до ресурсу: <https://github.com/remarkjs/react-markdown>.

					ІАЛЦ.467800.003 ПЗ	Арк.
						55
Зм.	Арк.	№ докум.	Підпис	Дата		

12. Lucide Icons [Електронний ресурс] // lucide.dev. – Режим доступу до ресурсу: <https://lucide.dev/icons/>.
13. Кудінов О. М. Об'єктно-орієнтоване моделювання програмних систем з використанням UML. Харків: ХНУРЕ, 2017. 152 с.
14. Fowler M. *UML Distilled: A Brief Guide to the Standard Object Modeling Language. 3rd Edition*. Addison-Wesley Professional, 2003. 208 с.
15. What is REST API [Електронний ресурс] // PHPenthusiast. – Режим доступу до ресурсу: <https://phpenthusiast.com/blog/what-is-rest-api>.
16. React Router [Електронний ресурс] // W3Schools. – Режим доступу до ресурсу: https://www.w3schools.com/react/react_router.asp.
17. Axios Documentation [Електронний ресурс] // [axios-http](https://axios-http.com/ru/docs/intro). – Режим доступу до ресурсу: <https://axios-http.com/ru/docs/intro>.

					ІАЛЦ.467800.003 ПЗ	Арк.
						56
Зм.	Арк.	№ докум.	Підпис	Дата		

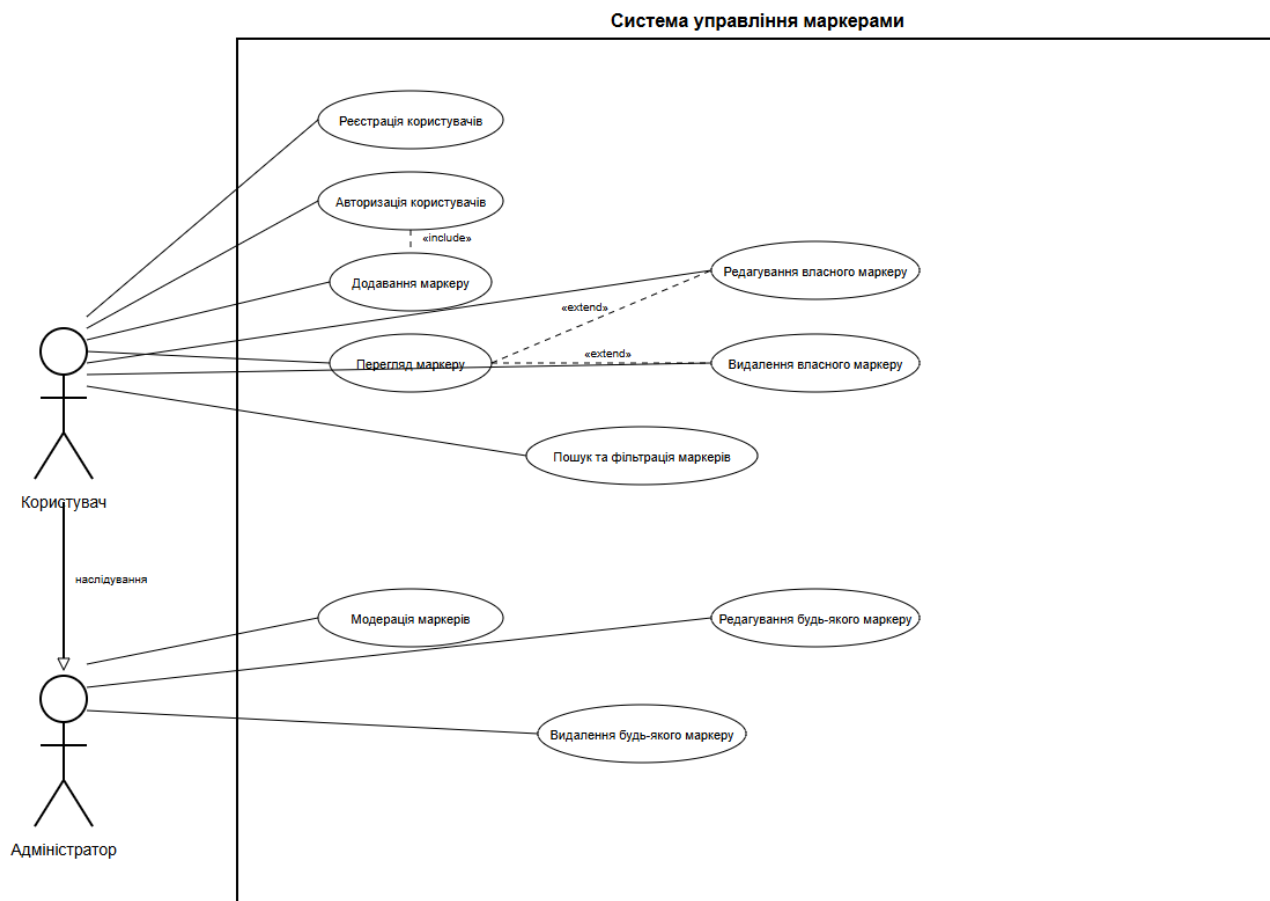
ДОДАТОК А

**Інтерактивна інформаційно-довідкова система з військової
історії України**

**Діаграма прецедентів
ІАЛЦ.467800.004 Д1**

Аркушів 1

Київ 2025 р



					ІАЛЦ.467800.004 Д1								
Зм.	Арк.	№ докум.	Підпис	Дата									
Розробив		Поляков В.О.			Інтерактивна інформаційно-довідкова система з військової історії України Діаграма прецедентів				Літ.	Аркуш	Аркушів		
Перевірив		Павлов В.Г.									1	1	
									КПІ ім. Ігоря Сікорського, ФІОТ, ІО-15				
Н. Контр.		Нікольський С.С.											
Затвердив		Новотарський М.А.											

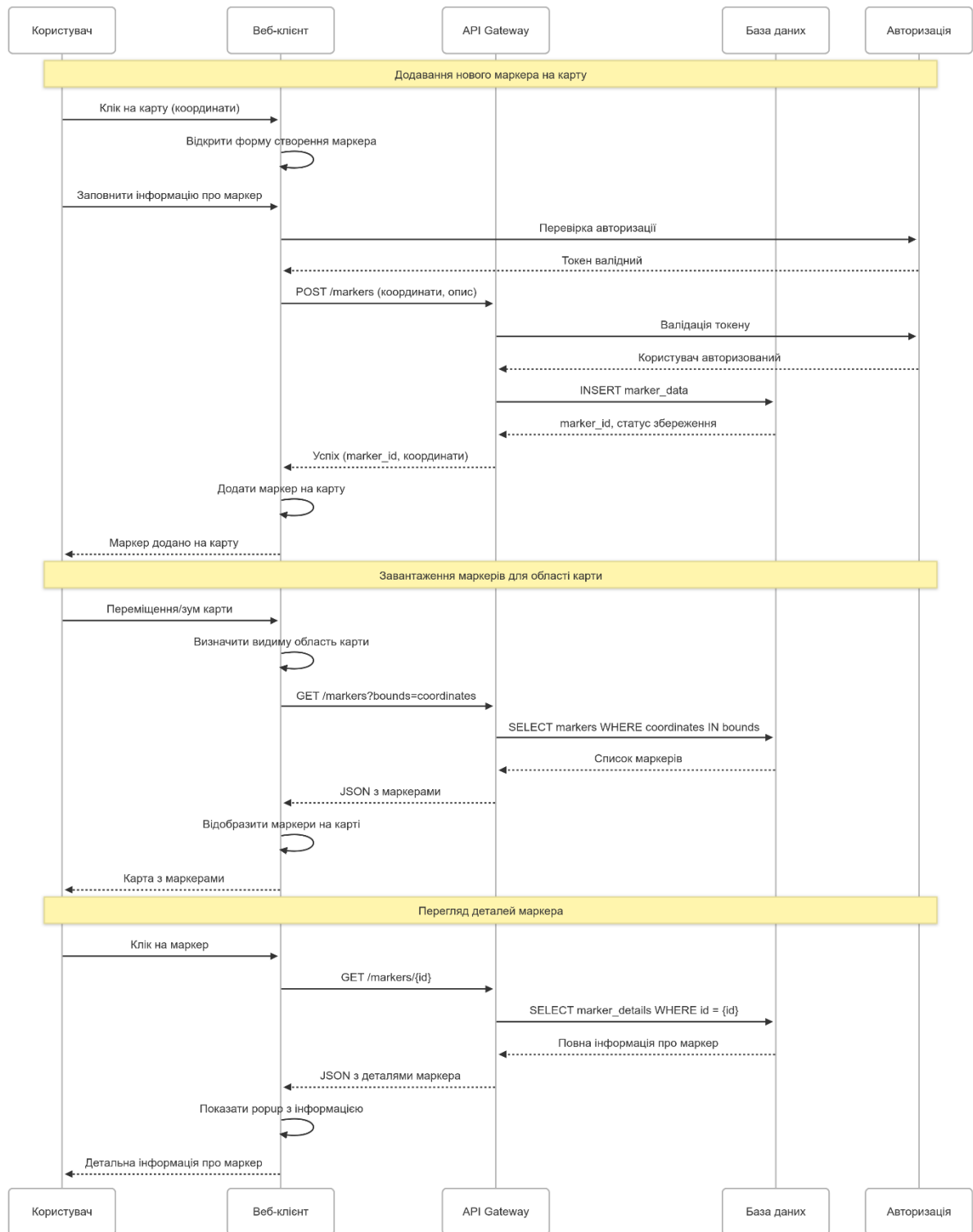
ДОДАТОК Б

**Інтерактивна інформаційно-довідкова система з військової
історії України**

Діаграма послідовності
ІАЛЦ.467800.005 Д2

Аркушів 1

Київ 2025 р



					ІАЛЦ.467800.005 Д2			
Зм.	Арк.	№ докум.	Підпис	Дата				
Розробив	Поляков В.О.				Інтерактивна інформаційно-довідкова система з військової історії України Діаграма послідовності		Літ.	Аркуш
Перевірив	Павлов В.Г.							1
Н. Контр.	Нікольський С.С						КПІ ім. Ігоря Сікорського, ФІОТ, ІО-15	
Затвердив	Новотарський М.А.							

ДОДАТОК В

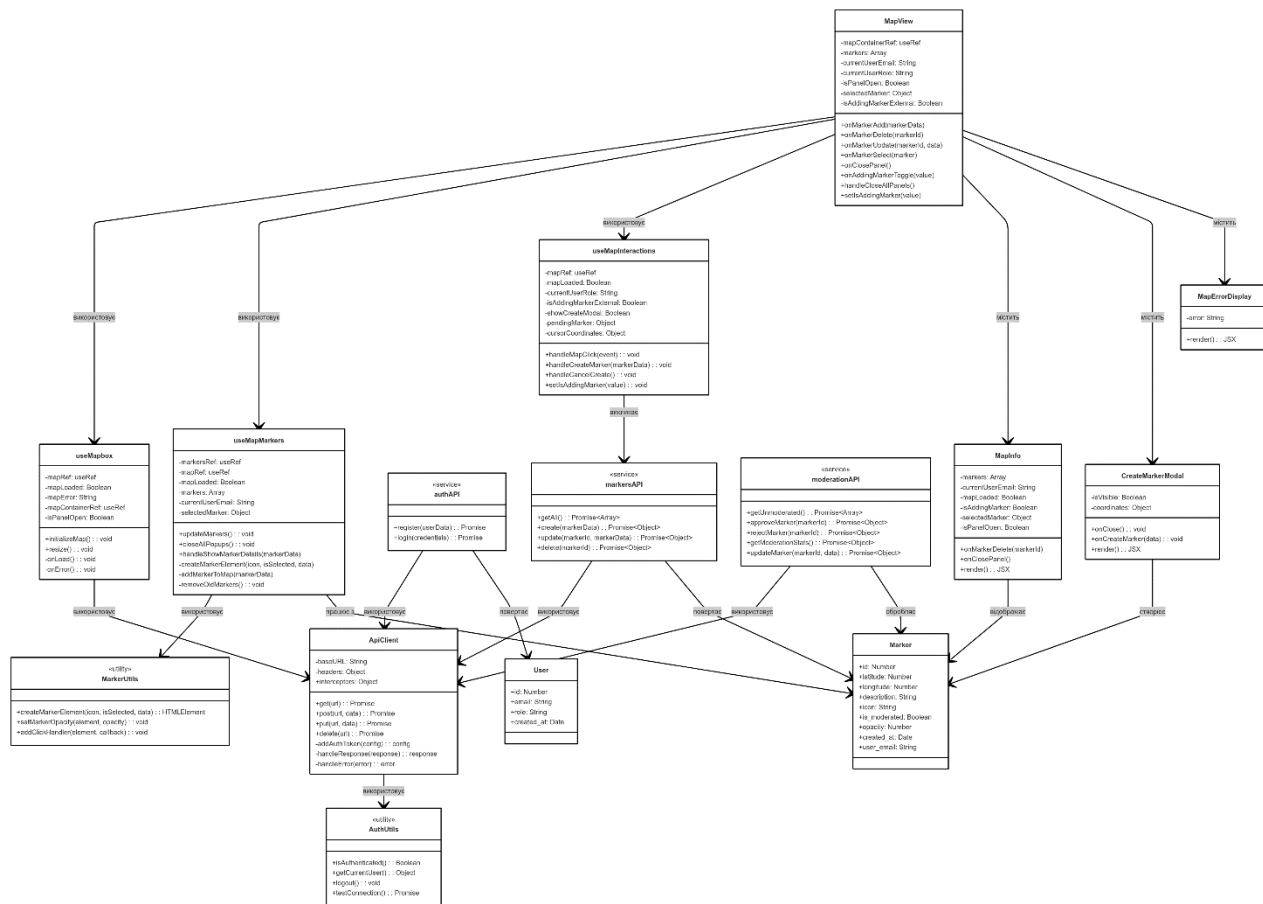
**Інтерактивна інформаційно-довідкова система з військової
історії України**

Діаграма класів

ІАЛЦ.467800.006 ДЗ

Аркушів 1

Київ 2025 р



					ІАЛЦ.467800.006 ДЗ							
Зм.	Арк.	№ докум.	Підпис	Дата	Інтерактивна інформаційно-довідкова система з військової історії України Діаграма класів			Літ.	Аркуш	Аркушів		
Розробив		Поляков В.О.									1	1
Перевірив		Павлов В.Г.										
Н. Контр.		Нікольський С.С.										
Затвердив		Новотарський М.А.						КПІ ім. Ігоря Сікорського, ФІОТ, ІО-15				

ДОДАТОК Г

**Інтерактивна інформаційно-довідкова система з військової
історії України**

Фрагмент програмного коду

ІАЛЦ.467800.007 Д4

Аркушів 1

Київ 2025 р

Код з файлу src/pages/MapPage.js

```
import React, { useState, useEffect } from 'react';
import { useNavigate } from 'react-router-dom';
import MapView from '../components/MapView';
import InfoPanel from '../components/InfoPanel';
import { MapInfo } from '../components/MapView/MapInfo';
import AdminModerationButton from '../components/ui/AdminModerationButton';
import MarkerFilterSearchPanel from '../components/MarkerFilterSearchPanel';
import { useMarkerFilterSearch } from '../hooks/useMarkerFilterSearch';
import { markersAPI, isAuthenticated, logout, getCurrentUser } from '../utils/api';
import { getCurrentUserEmail, getCurrentUserRole } from '../utils/auth';
import { mapPageStyles, injectSpinnerStyles } from '../styles/MapPageStyles';

export default function MapPage() {
  const [markers, setMarkers] = useState([]);
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState(null);
  const [selectedMarker, setSelectedMarker] = useState(null);
  const [showInfoPanel, setShowInfoPanel] = useState(false);
  const [currentUser, setCurrentUser] = useState(null);
  const [isAddingMarker, setIsAddingMarker] = useState(false);
  const navigate = useNavigate();

  const {
    searchQuery,
    setSearchQuery,
    dateFrom,
    setDateFrom,
    dateTo,
    setDateTo,
    eventDateFrom,
    setEventDateFrom,
    eventDateTo,
    setEventDateTo,
    selectedTags,
    selectedAuthors,
    showOnlyMyMarkers,
    setShowOnlyMyMarkers,
    availableTags,
    availableAuthors,
    markerVisibilityMap,
    filteredMarkers,
    hasActiveFilters,
    filterStats,
    clearAllFilters,
    toggleTag,
    clearTags,
    toggleAuthor,
    clearAuthors
  } = useMarkerFilterSearch(markers, getCurrentUserEmail());
  useEffect(() => {
    injectSpinnerStyles();

    if (!isAuthenticated()) {
      navigate('/login');
    }
  }, []);
```

					ІАЛЦ.467800.007 Д4									
Зм.	Арк.	№ докум.	Підпис	Дата	Інтерактивна інформаційно-довідкова система з військової історії України Фрагмент програмного коду				Літ.		Аркуш	Аркушів		
Розробив	Поляков В.О.										1		38	
Перевірив	Павлов В.Г.													
Н. Контр.	Нікольський С.С.										КПІ ім. Ігоря Сікорського, ФІОТ, ІО-15			
Затвердив	Новотарський М.А.													

```

    return;
  }

  const user = getCurrentUser();
  setCurrentUser(user);

  const emailFromToken = getCurrentUserEmail();

  loadMarkers();
}, [navigate]);

const loadMarkers = async () => {
  try {
    setLoading(true);
    setError(null);

    const response = await markersAPI.getAll();
    setMarkers(response.markers || []);

  } catch (error) {
    console.error('Помилка завантаження міток:', error);
    setError('Помилка завантаження міток: ' + (error.response?.data?.error || error.message));

    if (error.response?.status === 401 || error.response?.status === 403) {
      handleLogout();
    }
  } finally {
    setLoading(false);
  }
};

const handleMarkerAdd = async (markerData) => {
  try {

    const response = await markersAPI.create(markerData);

    setMarkers(prevMarkers => [response.marker, ...prevMarkers]);

    return { success: true, message: 'Мітка успішно додана!' };
  } catch (error) {
    console.error('Помилка додавання мітки:', error);

    if (error.response?.status === 401 || error.response?.status === 403) {
      handleLogout();
    }

    throw error;
  }
};

const handleMarkerUpdate = async (markerId, updateData) => {
  try {

    const response = await markersAPI.update(markerId, updateData);

    setMarkers(prevMarkers =>
      prevMarkers.map(marker =>
        marker.id === markerId ? response.marker : marker
      )
    );
  }
};

```

					ІАЛЦ.467800.007 Д4	Арк.
						2
Зм.	Арк.	№ докум.	Підпис	Дата		

```

    if (selectedMarker && selectedMarker.id === markerId) {
      setSelectedMarker(response.marker);
    }

    return { success: true, message: 'Мітка успішно оновлена!' };
  } catch (error) {
    console.error('Помилка оновлення мітки:', error);

    if (error.response?.status === 401 || error.response?.status === 403) {
      handleLogout();
    }

    throw error;
  }
};

const handleMarkerDelete = async (markerId) => {
  try {

    await markersAPI.delete(markerId);

    setMarkers(prevMarkers => prevMarkers.filter(marker => marker.id !== markerId));

    if (selectedMarker && selectedMarker.id === markerId) {
      setSelectedMarker(null);
      setShowInfoPanel(false);
    }

    return { success: true, message: 'Мітка успішно видалена!' };
  } catch (error) {
    console.error('Помилка видалення мітки:', error);

    if (error.response?.status === 401 || error.response?.status === 403) {
      handleLogout();
    }

    throw error;
  }
};

const handleMarkerSelect = (marker) => {
  setSelectedMarker(marker);
  setShowInfoPanel(true);
};

const handleFilterPanelMarkerSelect = (marker) => {
  handleMarkerSelect(marker);
};

const handleCloseInfoPanel = () => {
  setShowInfoPanel(false);
  setSelectedMarker(null);
};

const handleLogout = () => {
  if (window.confirm('Ви впевнені, що хочете вийти?')) {
    logout();
    navigate('/login');
  }
};

```

					ІАЛЦ.467800.007 Д4	Арк.
						3
Зм.	Арк.	№ докум.	Підпис	Дата		

```

const handleToggleAddMarker = () => {
  const newState = !isAddingMarker;
  setIsAddingMarker(newState);

  if (newState && showInfoPanel) {
    setShowInfoPanel(false);
    setSelectedMarker(null);
  }
};

const canAddMarkers = () => {
  const userRole = getCurrentUserRole();
  return userRole === 'admin' || userRole === 'editor';
};

const finalMarkers = markers.map(marker => ({
  ...marker,
  opacity: markerVisibilityMap.get(marker.id) || 1
})));

const getAddMarkerButtonStyle = () => {
  const baseStyle = { ...mapPageStyles.addMarkerButton };

  if (loading) {
    return { ...baseStyle, ...mapPageStyles.addMarkerButtonDisabled };
  }

  if (isAddingMarker) {
    return { ...baseStyle, ...mapPageStyles.addMarkerButtonActive };
  }

  return { ...baseStyle, ...mapPageStyles.addMarkerButtonInactive };
};

if (loading) {
  return (
    <div style={mapPageStyles.loadingContainer}>
      <div style={mapPageStyles.loadingSpinner}></div>
      <div>Завантаження карти...</div>
    </div>
  );
}

if (error) {
  return (
    <div style={mapPageStyles.errorContainer}>
      <div style={mapPageStyles.errorMessage}>
        {error}
      </div>
      <div style={mapPageStyles.errorButtons}>
        <button
          onClick={loadMarkers}
          style={mapPageStyles.buttonRetry}
        >
           Спробувати знову
        </button>
        <button
          onClick={handleLogout}
          style={mapPageStyles.buttonLogoutError}
        >

```

					ІАЛЦ.467800.007 Д4	Арк.
						4
Зм.	Арк.	№ докум.	Підпис	Дата		

```

        <button>Вийти</button>
    </div>
</div>
);
}

return (
<div style={mapPageStyles.container}>
    <MapView
        markers={finalMarkers}
        onMarkerAdd={handleMarkerAdd}
        onMarkerDelete={handleMarkerDelete}
        onMarkerUpdate={handleMarkerUpdate}
        onMarkerSelect={handleMarkerSelect}
        currentUserEmail={getCurrentUserEmail()}
        currentUserRole={getCurrentUserRole()}
        isPanelOpen={showInfoPanel}
        onClosePanel={handleCloseInfoPanel}
        selectedMarker={selectedMarker}
        onAddingMarkerToggle={setIsAddingMarker}
        isAddingMarkerExternal={isAddingMarker}
    />

    <MarkerFilterSearchPanel
        searchQuery={searchQuery}
        onSearchQueryChange={setSearchQuery}
        dateFrom={dateFrom}
        onDateFromChange={setDateFrom}
        dateTo={dateTo}
        onDateToChange={setDateTo}
        eventDateFrom={eventDateFrom}
        onEventDateFromChange={setEventDateFrom}
        eventDateTo={eventDateTo}
        onEventDateToChange={setEventDateTo}
        selectedTags={selectedTags}
        availableTags={availableTags}
        onToggleTag={toggleTag}
        onClearTags={clearTags}
        selectedAuthors={selectedAuthors}
        availableAuthors={availableAuthors}
        onToggleAuthor={toggleAuthor}
        onClearAuthors={clearAuthors}
        showOnlyMyMarkers={showOnlyMyMarkers}
        onShowOnlyMyMarkersChange={setShowOnlyMyMarkers}
        onClearFilters={clearAllFilters}
        hasActiveFilters={hasActiveFilters}
        filterStats={filterStats}
        filteredMarkers={filteredMarkers}
        onMarkerSelect={handleFilterPanelMarkerSelect}
    />

    <InfoPanel
        selectedMarker={selectedMarker}
        isVisible={showInfoPanel}
        onClose={handleCloseInfoPanel}
        onMarkerUpdate={handleMarkerUpdate}
        onMarkerDelete={handleMarkerDelete}
        currentUserEmail={getCurrentUserEmail()}
        currentUserRole={getCurrentUserRole()}
    />

```

					ІАЛЦ.467800.007 Д4	Арк.
						5
Зм.	Арк.	№ докум.	Підпис	Дата		

```

<AdminModerationButton />

<div style={mapPageStyles.addMarkerContainer}>
  <button
    onClick={handleToggleAddMarker}
    disabled={loading}
    style={getAddMarkerButtonStyle()}
    onMouseEnter={e => {
      if (!loading) {
        Object.assign(e.target.style, mapPageStyles.addMarkerButtonHover);
      }
    }}
    onMouseLeave={e => {
      if (!loading) {
        const baseStyle = getAddMarkerButtonStyle();
        Object.assign(e.target.style, baseStyle);
      }
    }}
  >
    <span>{isAddingMarker ? 'Скасувати' : 'Додати мітку'}</span>
  </button>
</div>

<button
  onClick={handleLogout}
  style={mapPageStyles.logoutButton}
  onMouseEnter={e => {
    Object.assign(e.target.style, mapPageStyles.logoutButtonHover);
  }}
  onMouseLeave={e => {
    Object.assign(e.target.style, mapPageStyles.logoutButton);
  }}
>
  <span>Вийти</span>
</button>
</div>
);
}

```

Код з файлу src/pages/AdminModerationPage.jsx

```

import React, { useState, useEffect } from 'react';
import { getCurrentUserEmail } from '../utils/auth';
import { moderationAPI } from '../utils/api';
import { moderationPageStyles as styles } from '../styles/moderationPageStyles';
import EditMarkerModal from '../components/EditMarkerModal';

```

```

const compactMarkerStyle = {
  display: 'flex',
  alignItems: 'center',
  justifyContent: 'space-between',
  padding: '10px 16px',
  background: '#fff',
  borderBottom: '1px solid #e5e7eb',
  fontSize: '14px',
  color: '#374151'
};

```

```

const AdminModerationPage = () => {

```

					ІАЛЦ.467800.007 Д4	Арк.
						6
Зм.	Арк.	№ докум.	Підпис	Дата		


```

const [unmoderatedMarkers, setUnmoderatedMarkers] = useState([]);
const [stats, setStats] = useState(null);
const [loading, setLoading] = useState(true);
const [error, setError] = useState(null);
const [processingMarkers, setProcessingMarkers] = useState(new Set());
const [selectedMarker, setSelectedMarker] = useState(null);
const [isModalVisible, setIsModalVisible] = useState(false);

const loadData = async () => {
  try {
    setLoading(true);
    setError(null);

    const [markersResponse, statsResponse] = await Promise.all([
      moderationAPI.getUnmoderated(),
      moderationAPI.getModerationStats()
    ]);

    setUnmoderatedMarkers(markersResponse.markers || []);
    setStats(statsResponse);
  } catch (err) {
    setError(err.response?.data?.error || err.message || 'Помилка завантаження даних');
  } finally {
    setLoading(false);
  }
};

useEffect(() => {
  loadData();
}, []);

const handleApproveMarker = async (markerId) => {
  if (processingMarkers.has(markerId)) return;

  try {
    setProcessingMarkers(prev => new Set([...prev, markerId]));
    await moderationAPI.approveMarker(markerId);

    setUnmoderatedMarkers(prev => prev.filter(marker => marker.id !== markerId));
    setStats(prev => prev ? {
      ...prev,
      moderated: prev.moderated + 1,
      unmoderated: prev.unmoderated - 1,
      moderationPercentage: Math.round(((prev.moderated + 1) / prev.total) * 100)
    } : null);

    if (selectedMarker?.id === markerId) {
      setSelectedMarker(null);
      setIsModalVisible(false);
    }
  } catch (err) {
    alert('Помилка підтвердження маркера: ' + (err.response?.data?.error || err.message));
  } finally {
    setProcessingMarkers(prev => {
      const newSet = new Set(prev);
      newSet.delete(markerId);
      return newSet;
    });
  }
};

```

					ІАЛЦ.467800.007 Д4	Арк.
						7
Зм.	Арк.	№ докум.	Підпис	Дата		

```

const handleRejectMarker = async (markerId) => {
  if (processingMarkers.has(markerId)) return;

  if (!window.confirm('Ви впевненні що хочете видалити цей маркер?')) return;

  try {
    setProcessingMarkers(prev => new Set([...prev, markerId]));
    await moderationAPI.rejectMarker(markerId);

    setUnmoderatedMarkers(prev => prev.filter(marker => marker.id !== markerId));
    setStats(prev => prev ? {
      ...prev,
      total: prev.total - 1,
      unmoderated: prev.unmoderated - 1,
      moderationPercentage: prev.total > 1 ? Math.round((prev.moderated / (prev.total - 1)) * 100) : 100
    } : null);

    if (selectedMarker?.id === markerId) {
      setSelectedMarker(null);
      setIsModalVisible(false);
    }
  } catch (err) {
    alert('Ошибка видалення маркеру: ' + (err.response?.data?.error || err.message));
  } finally {
    setProcessingMarkers(prev => {
      const newSet = new Set(prev);
      newSet.delete(markerId);
      return newSet;
    });
  }
};

const handleViewMarker = (marker) => {
  setSelectedMarker(marker);
  setIsModalVisible(true);
};

const handleCloseModal = () => {
  setIsModalVisible(false);
  setSelectedMarker(null);
};

const handleUpdateMarker = async (markerId, updatedData) => {
  try {
    await moderationAPI.updateMarker(markerId, updatedData);

    setUnmoderatedMarkers(prev =>
      prev.map(marker =>
        marker.id === markerId
          ? { ...marker, ...updatedData }
          : marker
        )
    );

  } catch (error) {
    throw error;
  }
};

if (loading) {
  return <div style={styles.wrapper}>Завантаження...</div>;
}

```


					ІАЛЦ.467800.007 Д4	Арк.
						8
Зм.	Арк.	№ докум.	Підпис	Дата		

```

}

if (error) {
  return (
    <div style={styles.wrapper}>
      <div style={{ color: 'red', marginBottom: 20 }}>{error}</div>
      <button style={styles.button} onClick={loadData}>Повторить</button>
    </div>
  );
}

return (
  <div style={styles.wrapper}>
    <header style={styles.header}>
      <div>
        <h1 style={styles.headerTitle}>Панель модерации</h1>
        <p style={styles.subTitle}>Адміністратор: {getCurrentUserEmail()}</p>
      </div>
      <button style={styles.button} onClick={loadData}>Оновити</button>
    </header>

    <div>
      {unmoderatedMarkers.map(marker => (
        <div key={marker.id} style={compactMarkerStyle}>
          <div style={{ flex: 1, overflow: 'hidden' }}>
            <div style={{ display: 'flex', alignItems: 'center', gap: '8px' }}>
              <strong>{marker.title}</strong>
              <button
                style={{
                  background: '#3b82f6',
                  color: 'white',
                  border: 'none',
                  borderRadius: '4px',
                  padding: '4px 8px',
                  fontSize: '12px',
                  cursor: 'pointer',
                  display: 'flex',
                  alignItems: 'center',
                  gap: '4px'
                }}
                onClick={() => handleViewMarker(marker)}
                title="Подивитися деталі маркеру"
              >
                 Перегляд
              </button>
            </div>
            <div style={{ fontSize: '12px', color: '#6b7280' }}>
              {marker.user_email || 'Автор невідомий'}
            </div>
          </div>
          <div style={{ display: 'flex', gap: '6px' }}>
            <button
              style={{ ...styles.button, background: '#16a34a', padding: '6px 10px' }}
              disabled={processingMarkers.has(marker.id)}
              onClick={() => handleApproveMarker(marker.id)}
              title="Погодити маркер"
            >
              
            </button>
            <button
              style={{ ...styles.button, background: '#dc2626', padding: '6px 10px' }}

```

					ІАЛЦ.467800.007 Д4	Арк.
						9
Зм.	Арк.	№ докум.	Підпис	Дата		

```

        disabled={processingMarkers.has(marker.id)}
        onClick={() => handleRejectMarker(marker.id)}
        title="Отклонить метку"
      >
        ✕
      </button>
    </div>
  </div>
))}
</div>

<EditMarkerModal
  isVisible={isModalVisible}
  onClose={handleCloseModal}
  onUpdateMarker={handleUpdateMarker}
  selectedMarker={selectedMarker}
/>
</div>
);
};

```

```
export default AdminModerationPage;
```

Код з файлу App.js

```

import React from 'react';
import { BrowserRouter as Router, Routes, Route, Navigate } from 'react-router-dom';
import RegisterPage from './pages/RegisterPage';
import LoginPage from './pages/LoginPage';
import MapPage from './pages/MapPage';
import AdminModerationPage from './pages/AdminModerationPage';
import ProtectedRoute from './components/common/ProtectedRoute';
import AdminRoute from './components/common/AdminRoute';

```

```

function App() {
  const token = localStorage.getItem('token');

  return (
    <Router
      future={{
        v7_startTransition: true,
        v7_relativeSplatPath: true
      }}
    >
      <Routes>
        <Route path="/register" element={<RegisterPage />} />
        <Route path="/login" element={<LoginPage />} />
        <Route
          path="/map"
          element={
            <ProtectedRoute>
              <MapPage />
            </ProtectedRoute>
          }
        />
        <Route
          path="/admin/moderation"
          element={
            <AdminRoute>
              <AdminModerationPage />
            </AdminRoute>
          }
        />
      </Routes>
    </Router>
  );
}

```

					ІАЛЦ.467800.007 Д4	Арк.
						10
Зм.	Арк.	№ докум.	Підпис	Дата		

```

    }
  />
  <Route
    path="/"
    element={
      token ? <Navigate to="/map" replace /> : <Navigate to="/login" replace />
    }
  />
  <Route
    path="*"
    element={
      token ? <Navigate to="/map" replace /> : <Navigate to="/login" replace />
    }
  />
</Routes>
</Router>
);
}

```

export default App;

Код з файлу src/components/MapView.jsx

```

import React, { useEffect, useRef } from 'react';
import mapboxgl from 'mapbox-gl';
import 'mapbox-gl/dist/mapbox-gl.css';
import { MapInfo } from './MapInfo';
import { MapErrorDisplay } from './MapErrorDisplay';
import { useMapbox } from '../hooks/useMapbox';
import { useMapMarkers } from '../hooks/useMapMarkers';
import { useMapInteractions } from '../hooks/useMapInteractions';
import CreateMarkerModal from './CreateMarkerModal';

const MapView = ({
  markers,
  onMarkerAdd,
  onMarkerDelete,
  onMarkerUpdate,
  onMarkerSelect,
  currentUserEmail,
  currentUserRole,
  isPanelOpen,
  onClosePanel,
  selectedMarker,
  onAddingMarkerToggle,
  isAddingMarkerExternal,
}) => {
  const mapContainerRef = useRef(null);

  const {
    mapRef,
    mapLoaded,
    mapError,
    initializeMap,
  } = useMapbox(mapContainerRef, isPanelOpen);

  const {
    updateMarkers,
    closeAllPopups,
  } = useMapMarkers(mapRef, mapLoaded, markers, currentUserEmail, selectedMarker, onMarkerSelect, onMarkerDelete);

```

					ІАЛЦ.467800.007 Д4	Арк.
						11
Зм.	Арк.	№ докум.	Підпис	Дата		

```

const handleCloseAllPanels = () => {
  if (onClosePanel) {
    onClosePanel();
  }
  closeAllPopups();
};

const setIsAddingMarker = (value) => {
  if (onAddingMarkerToggle) {
    onAddingMarkerToggle(value);
  }
};

const {
  handleMapClick,
  showCreateModal,
  pendingMarker,
  handleCreateMarker,
  handleCancelCreate,
  cursorCoordinates,
} = useMapInteractions(
  mapRef,
  mapLoaded,
  onMarkerAdd,
  handleCloseAllPanels,
  currentUserRole,
  isAddingMarkerExternal,
  setIsAddingMarker
);

useEffect(() => {
  initializeMap();
}, []);

useEffect(() => {
  updateMarkers();
}, [markers, selectedMarker]);

useEffect(() => {
  if (!mapRef.current || !mapLoaded) return;

  const handleClick = (e) => {
    console.log('Клік по карті, isAddingMarker:', isAddingMarkerExternal);
    handleMapClick(e);
  };

  mapRef.current.on('click', handleClick);

  return () => {
    if (mapRef.current) {
      mapRef.current.off('click', handleClick);
    }
  };
}, [isAddingMarkerExternal, mapLoaded, handleMapClick]);

if (mapError) {
  return <MapErrorDisplay error={mapError} />;
}

return (

```

					ІАЛЦ.467800.007 Д4	Арк.
						12
Зм.	Арк.	№ докум.	Підпис	Дата		

```

<div style={ {
  position: 'relative',
  width: '100vw',
  height: '100vh',
  overflow: 'hidden',
} }>
  <MapInfo
    markers={markers}
    currentUserEmail={currentUserEmail}
    mapLoaded={mapLoaded}
    isAddingMarker={isAddingMarkerExternal}
    selectedMarker={selectedMarker}
    isPanelOpen={isPanelOpen}
    onMarkerDelete={onMarkerDelete}
    onClosePanel={onClosePanel}
  />
  <CreateMarkerModal
    isVisible={showCreateModal}
    onClose={handleCancelCreate}
    onCreateMarker={handleCreateMarker}
    coordinates={pendingMarker}
  />
  <div
    ref={mapContainerRef}
    style={ {
      width: '100%',
      height: '100vh',
      cursor: isAddingMarkerExternal ? 'crosshair' : 'default',
    } }
  />
</div>
);
};

```

export default MapView;

Код з файлу src/components/InfoPanel.jsx

```

import React, { useState, useEffect } from 'react';
import ReactMarkdown from 'react-markdown';
import { useInfoPanel } from '../hooks/useInfoPanel.js';
import { infoPanelStyles } from '../styles/infoPanelStyles.js';
import EditMarkerModal from '../EditMarkerModal.jsx';

const InfoPanel = ({
  selectedMarker,
  isVisible,
  onClose,
  currentUserEmail,
  currentUserRole,
  onMarkerDelete,
  onMarkerUpdate,
}) => {
  const [isEditModalVisible, setIsEditModalVisible] = useState(false);
  const { error, isDeleting, handleDelete } = useInfoPanel(selectedMarker, null, onMarkerDelete, onClose);

  const style = document.createElement('style');
  style.textContent = markdownResetCSS;
  document.head.appendChild(style);

  return () => {

```

					ІАЛЦ.467800.007 Д4	Арк.
						13
Зм.	Арк.	№ докум.	Підпис	Дата		

```

    if (document.head.contains(style)) {
      document.head.removeChild(style);
    }
  };
}, []);

const formatDate = (dateString) => {
  try {
    return new Date(dateString).toLocaleString('uk-UA', {
      day: '2-digit',
      month: '2-digit',
      year: 'numeric',
      hour: '2-digit',
      minute: '2-digit',
      hour12: false,
    }).replace(',', ' ');
  } catch {
    return 'Невідома дата';
  }
};

const formatCoordinates = (lat, lng) => {
  return `${parseFloat(lat).toFixed(6)}, ${parseFloat(lng).toFixed(6)}`;
};

const formatTags = (tags) => {
  if (!tags || tags.trim() === '') return [];
  return tags.split(/\s+/).filter(tag => tag.length > 0);
};

const canEditMarker = selectedMarker && (
  selectedMarker.user_email === currentUserEmail || currentUserRole === 'admin'
);

const canDeleteMarker = canEditMarker;

const handleEditClick = () => {
  setIsEditModalVisible(true);
};

const handleEditModalClose = () => {
  setIsEditModalVisible(false);
};

const handleMarkerUpdate = async (markerId, updatedData) => {
  if (onMarkerUpdate) {
    await onMarkerUpdate(markerId, updatedData);
  }
};

console.log('InfoPanel Debug:', {
  selectedMarker: selectedMarker?.id,
  currentUserEmail,
  markerUserEmail: selectedMarker?.user_email,
  canEditMarker,
  canDeleteMarker
});

const markdownComponents = {
  img: ({ src, alt }) => (
    <img

```

					ІАЛЦ.467800.007 Д4	Арк.
						14
Зм.	Арк.	№ докум.	Підпис	Дата		


```

src={src}
alt={alt || 'Зображення'}
style={{
  maxWidth: '50%',
  height: 'auto',
  borderRadius: '8px',
  margin: '8px auto',
  display: 'block',
}}
/>
),
p: ({ children }) => (
  <p style={{
    margin: '0 0 6px 0',
    padding: '0',
    lineHeight: '1.5',
    fontSize: '14px',
    color: '#374151'
  }}>
    {children}
  </p>
),
h1: ({ children }) => (
  <h1 style={{
    margin: '0 0 8px 0',
    fontSize: '18px',
    fontWeight: 'bold',
    color: '#374151'
  }}>
    {children}
  </h1>
),
h2: ({ children }) => (
  <h2 style={{
    margin: '0 0 8px 0',
    fontSize: '16px',
    fontWeight: 'bold',
    color: '#374151'
  }}>
    {children}
  </h2>
),
h3: ({ children }) => (
  <h3 style={{
    margin: '0 0 6px 0',
    fontSize: '15px',
    fontWeight: 'bold',
    color: '#374151'
  }}>
    {children}
  </h3>
),
ul: ({ children }) => (
  <ul style={{
    margin: '0 0 6px 0',
    paddingLeft: '20px'
  }}>
    {children}
  </ul>
),
ol: ({ children }) => (

```

					ІАЛЦ.467800.007 Д4	Арк.
						15
Зм.	Арк.	№ докум.	Підпис	Дата		

```

<ol style={{
  margin: '0 0 6px 0',
  paddingLeft: '20px'
}}>
  {children}
</ol>
),
li: ({ children }) => (
  <li style={{
    margin: '0 0 2px 0'
  }}>
    {children}
  </li>
),
blockquote: ({ children }) => (
  <blockquote style={{
    margin: '0 0 6px 0',
    paddingLeft: '12px',
    borderLeft: '3px solid #d1d5db',
    fontStyle: 'italic',
    color: '#6b7280'
  }}>
    {children}
  </blockquote>
),
code: ({ children }) => (
  <code style={{
    backgroundColor: '#f3f4f6',
    padding: '2px 4px',
    borderRadius: '3px',
    fontSize: '13px',
    fontFamily: 'monospace'
  }}>
    {children}
  </code>
),
pre: ({ children }) => (
  <pre style={{
    backgroundColor: '#f3f4f6',
    padding: '12px',
    borderRadius: '6px',
    margin: '0 0 6px 0',
    overflow: 'auto',
    fontSize: '13px',
    fontFamily: 'monospace'
  }}>
    {children}
  </pre>
)
};

return (
  <>
    <div style={infoPanelStyles.container(isVisible)}>
      <div style={{ ...infoPanelStyles.header, justifyContent: 'space-between' }}>
        <div style={{ display: 'flex', gap: '8px' }}>
          {canEditMarker} && (
            <button
              onClick={handleEditClick}
              style={{
                width: '36px',

```

					ІАЛЦ.467800.007 Д4	Арк.
						16
Зм.	Арк.	№ докум.	Підпис	Дата		

```

        height: '36px',
        borderRadius: '50%',
        backgroundColor: '#10b981',
        border: 'none',
        color: 'white',
        fontSize: '18px',
        cursor: 'pointer',
        transition: 'background-color 0.2s',
    }}
    title="Редагувати мітку"
  >
    
  </button>
)}
{canDeleteMarker && (
  <button
    onClick={handleDelete}
    disabled={isDeleting}
    style={{
      width: '36px',
      height: '36px',
      borderRadius: '50%',
      backgroundColor: '#ef4444',
      border: 'none',
      color: 'white',
      fontSize: '18px',
      cursor: isDeleting ? 'not-allowed' : 'pointer',
      opacity: isDeleting ? 0.6 : 1,
      transition: 'background-color 0.2s',
    }}
    title="Видалити мітку"
    onMouseEnter={(e) => {
      if (!isDeleting) e.target.style.backgroundColor = '#b91c1c';
    }}
    onMouseLeave={(e) => {
      if (!isDeleting) e.target.style.backgroundColor = '#ef4444';
    }}
  >
    
  </button>
)}
</div>
<div>
  <button
    onClick={onClose}
    style={infoPanelStyles.closeButton}
    onMouseEnter={(e) => (e.target.style.backgroundColor = 'rgba(255,255,255,0.2)')}
    onMouseLeave={(e) => (e.target.style.backgroundColor = 'transparent')}
  >
    ✕
  </button>
</div>
</div>

<div style={infoPanelStyles.content}>
  {selectedMarker ? (
    <div style={infoPanelStyles.errorMessage}>
      {error}
    </div>
  )}

```

					ІАЛЦ.467800.007 Д4	Арк.
						17
Зм.	Арк.	№ докум.	Підпис	Дата		

```

    })

    <h2 style={infoPanelStyles.headerTitle}>
      {selectedMarker?.title || 'Без назви'}
    </h2>

    <div style={{
      textAlign: 'center',
      fontStyle: 'italic',
      fontSize: '13px',
      color: '#6b7280',
      marginBottom: '12px'
    }}>
      {selectedMarker.event_date || 'Немає дати'}
    </div>

    <div style={{
      ...infoPanelStyles.compactInfo,
      display: 'flex',
      justifyContent: 'center',
      alignItems: 'center',
      gap: '6px',
      padding: '8px',
      border: '1px solid #e5e7eb',
      borderRadius: '6px',
      backgroundColor: '#f9faff',
      marginBottom: '12px',
      fontSize: '13px',
      color: '#374151',
      flexWrap: 'wrap'
    }}>
      <span>📍 {formatCoordinates(selectedMarker.latitude, selectedMarker.longitude)}</span>
      <span>•</span>
      <span>👤 {selectedMarker.user_email || 'Анонім'}</span>
      <span>•</span>
      <span>📅 {formatDate(selectedMarker.created_at)}</span>
    </div>

    <div style={infoPanelStyles.tagsContainer}>
      {formatTags(selectedMarker.tags).length > 0 ? (
        formatTags(selectedMarker.tags).map((tag, index) => (
          <span key={index} style={infoPanelStyles.tagBadge}>
            {tag}
          </span>
        ))
      ) : (
        <span style={{ fontSize: '13px', color: '#6b7280' }}>
          Немає тегів
        </span>
      )}
    </div>

    <div style={infoPanelStyles.descriptionContainer}>
      <div style={{
        ...infoPanelStyles.descriptionValue,
        flex: 1
      }}>
        {selectedMarker.description ? (
          <div className="markdown-content">
            <ReactMarkdown components={markdownComponents}>
              {selectedMarker.description}
            </ReactMarkdown>
          </div>
        ) : (
          <div style={infoPanelStyles.descriptionValue}>
            Немає опису
          </div>
        )}
      </div>
    </div>
  </div>
)

```

					ІАЛЦ.467800.007 Д4	Арк.
						18
Зм.	Арк.	№ докум.	Підпис	Дата		

```

        </ReactMarkdown>
      </div>
    ) : (
      'Немає опису'
    )
  </div>
</div>
</>
) : (
  <div style={infoPanelStyles.emptyState}>
    Виберіть мітку на карті
  </div>
  )
</div>
</div>

<EditMarkerModal
  isVisible={isEditModalVisible}
  onClose={handleEditModalClose}
  onUpdateMarker={handleMarkerUpdate}
  selectedMarker={selectedMarker}
/>
</>
);
};

export default InfoPanel;

```

Код з файлу src/components/InfoPanel.jsx

```

import React, { useState, useCallback } from 'react';
import PropTypes from 'prop-types';
import { IconSelector } from '../ui/IconSelector';
import { modalStyles } from '../styles/createMarkerModalStyles';
import { infoPanelStyles } from '../styles/infoPanelStyles';
import { useMarkerForm } from '../hooks/useMarkerForm';

const DateTypeSelector = ({ dateType, setDateType, isLoading }) => (
  <select
    value={dateType}
    onChange={(e) => setDateType(e.target.value)}
    disabled={isLoading}
    style={modalStyles.dateTypeSelect}
  >
    <option value="none">Без дати</option>
    <option value="year">Рік</option>
    <option value="date">Дата</option>
    <option value="yearRange">Роки</option>
    <option value="dateRange">Дати</option>
  </select>
);

const DateInputs = ({ dateType, formState, setFormState, isLoading }) => {
  const inputStyles = {
    ...modalStyles.inputField,
    ...modalStyles[dateType.includes('year') ? 'yearInput' : 'dateInput'],
    backgroundColor: isLoading ? '#f9fafb' : 'white',
  };

  const commonProps = {
    disabled: isLoading,
  };

```

					ІАЛЦ.467800.007 Д4	Арк.
						19
Зм.	Арк.	№ докум.	Підпис	Дата		

```

style: inputStyles,
onFocus: (e) => (e.target.style.borderColor = '#059669'),
onBlur: (e) => (e.target.style.borderColor = '#d1d5db'),
};

switch (dateType) {
case 'year':
return (
<input
type="number"
min="1"
max={new Date().getFullYear() + 100}
value={formState.eventYear}
onChange={(e) => setFormState({ eventYear: e.target.value })}
placeholder="1990"
{...commonProps}
/>
);
case 'date':
return (
<input
type="date"
value={formState.eventDate}
onChange={(e) => setFormState({ eventDate: e.target.value })}
{...commonProps}
/>
);
case 'yearRange':
return (
<div style={modalStyles.dateRangeContainer}>
<input
type="number"
min="1"
max={new Date().getFullYear() + 100}
value={formState.eventYearFrom}
onChange={(e) => setFormState({ eventYearFrom: e.target.value })}
placeholder="от"
{...commonProps}
/>
<input
type="number"
min="1"
max={new Date().getFullYear() + 100}
value={formState.eventYearTo}
onChange={(e) => setFormState({ eventYearTo: e.target.value })}
placeholder="до"
{...commonProps}
/>
</div>
);
case 'dateRange':
return (
<div style={modalStyles.dateRangeContainer}>
<input
type="date"
value={formState.eventDateFrom}
onChange={(e) => setFormState({ eventDateFrom: e.target.value })}
{...commonProps}
/>
<input
type="date"

```

					ІАЛЦ.467800.007 Д4	Арк.
						20
Зм.	Арк.	№ докум.	Підпис	Дата		

```
        value={formState.eventDateTo}
        onChange={(e) => setFormState({ eventDateTo: e.target.value })}
        {...commonProps}
      />
    </div>
  );
  default:
    return null;
}
};

const CreateMarkerModal = ({ isVisible, onClose, onCreateMarker, coordinates }) => {
  const { formState, updateFormState, error, setError, isLoading, setIsLoading, resetForm, getEventDateString } =
useMarkerForm();

  const handleIconSelect = useCallback((iconName) => {
    try {
      if (typeof updateFormState === 'function') {
        updateFormState({ selectedIcon: iconName });
      } else {
        console.error('updateFormState is not a function');
      }
    } catch (err) {
      console.error('Error setting icon:', err);
    }
  }, [updateFormState]);

  const handleSave = useCallback(async () => {
    const { title, description, eventYear, eventYearFrom, eventYearTo, eventDateFrom, eventDateTo, dateType, tags } =
formState;

    // Валідація
    if (!title.trim()) return setError('Назва не може бути пустою');
    if (!description.trim()) return setError('Опис не може бути пустим');
    if (title.length > 100) return setError('Назва не може містити більше 100 символів');
    if (description.length > 3000) return setError('Опис не може містити більше 3000 символів');

    if (dateType === 'year' && eventYear && (eventYear < 1 || eventYear > new Date().getFullYear() + 100)) {
      return setError('Рік має бути від 1 до ' + (new Date().getFullYear() + 100));
    }
    if (dateType === 'yearRange' && eventYearFrom && eventYearTo && parseInt(eventYearFrom) >
parseInt(eventYearTo)) {
      return setError('Рік початку не може бути більше року закінчення');
    }
    if (dateType === 'dateRange' && eventDateFrom && eventDateTo && new Date(eventDateFrom) > new
Date(eventDateTo)) {
      return setError('Дата початку не може бути більше дати закінчення');
    }

    setIsLoading(true);
    setError(null);

    try {
      const tagList = tags.split(/[#\s]+/).map(tag => tag.trim()).filter(tag => tag.length > 0);

      await onCreateMarker({
        title: title.trim(),
        event_date: getEventDateString(),
        tags: tagList.join(' '),
        description: description.trim(),
        icon: formState.selectedIcon,

```

					ІАЛЦ.467800.007 Д4	Арк.
						21
Зм.	Арк.	№ докум.	Підпис	Дата		

```

    });

    resetForm();
  } catch (error) {
    console.error('Помилка створення маркеру:', error);
    setError(error.response?.data?.error || error.message || 'Помилка створення маркеру. Спробуйте ще раз.');
```

} finally {
 setIsLoading(false);
 }
}, [formState, onCreateMarker, getEventDateString, setIsLoading, setError, resetForm]));

```

const handleCancel = useCallback(() => {
  resetForm();
  onClose();
}, [resetForm, onClose]);

React.useEffect(() => {
  console.log('FormState:', formState);
  console.log('updateFormState type:', typeof updateFormState);
}, [formState, updateFormState]);

if (!isVisible) return null;

return (
  <div style={modalStyles.overlay} onClick={handleCancel}>
    <div style={modalStyles.modal} onClick={(e) => e.stopPropagation()}>
      <div style={modalStyles.header}>
        <div style={modalStyles.headerLeft}>
          <h2 style={modalStyles.title}>Створити новий маркер</h2>
          {coordinates && (
            <p style={modalStyles.subtitle}>
              📍 ({coordinates.latitude.toFixed(6)}, {coordinates.longitude.toFixed(6)})
            </p>
          )}
        </div>
        <div style={modalStyles.datePickerContainer}>
          <div style={modalStyles.dateTypeContainer}>
            <DateTypeSelector
              dateType={formState.dateType}
              setDateType={(value) => updateFormState({ dateType: value })}
              isLoading={isLoading}
            />
          </div>
          <div style={modalStyles.dateFieldsContainer}>
            <div style={modalStyles.dateInputsContainer}>
              <DateInputs
                dateType={formState.dateType}
                formState={formState}
                setFormState={updateFormState}
                isLoading={isLoading}
              />
            </div>
          </div>
        </div>
        <div style={modalStyles.headerRight}>
          <button
            onClick={handleSave}
            disabled={!isLoading || !formState.title.trim() || !formState.description.trim()}
            style={infoPanelStyles.button('#059669', isLoading, !formState.title.trim() || !formState.description.trim())}
          >
            {isLoading ? '💾' : '📄'} {isLoading ? 'Збереження...' : 'Створити'}
          </button>
        </div>
      </div>
    </div>
  </div>
);

```

					ІАЛЦ.467800.007 Д4	Арк.
						22
Зм.	Арк.	№ докум.	Підпис	Дата		


```

</button>
<button
  onClick={handleCancel}
  disabled={isLoading}
  style={infoPanelStyles.button('#6b7280', isLoading)}
>
  Скасувати
</button>
</div>
</div>

{error && <div style={modalStyles.errorMessage}>{error}</div>}

<div style={modalStyles.content}>
  <div style={modalStyles.titleRow}>
    <div style={modalStyles.textFieldsColumn}>
      <div style={modalStyles.fieldGroup}>
        <input
          type="text"
          value={formState.title}
          onChange={(e) => updateFormState({ title: e.target.value })}
          placeholder="Введіть назву маркера..."
          disabled={isLoading}
          style={{
            ...modalStyles.inputField,
            backgroundColor: isLoading ? '#f9fafb' : 'white',
          }}
          onFocus={(e) => (e.target.style.borderColor = '#059669')}
          onBlur={(e) => (e.target.style.borderColor = '#d1d5db')}
        />
      </div>
      <div style={modalStyles.fieldGroup}>
        <input
          type="text"
          value={formState.tags}
          onChange={(e) => updateFormState({ tags: e.target.value })}
          placeholder="Введіть теги через пробіл..."
          disabled={isLoading}
          style={{
            ...modalStyles.inputField,
            backgroundColor: isLoading ? '#f9fafb' : 'white',
          }}
          onFocus={(e) => (e.target.style.borderColor = '#059669')}
          onBlur={(e) => (e.target.style.borderColor = '#d1d5db')}
        />
      </div>
    </div>
    <div style={modalStyles.iconsContainer}>
      <IconSelector
        selectedIcon={formState.selectedIcon}
        onIconSelect={handleIconSelect}
        disabled={isLoading}
        isModerated={false}
      />
    </div>
  </div>
  <div style={modalStyles.fieldGroup}>
    <textarea
      value={formState.description}
      onChange={(e) => updateFormState({ description: e.target.value })}
      placeholder="Введіть опис маркера (підтримує Markdown)..."
    />
  </div>
</div>

```

					ІАЛЦ.467800.007 Д4	Арк.
						23
Зм.	Арк.	№ докум.	Підпис	Дата		

```

    disabled={isLoading}
    style={{
      ...modalStyles.inputField,
      backgroundColor: isLoading ? '#f9fafb' : 'white',
      minHeight: '400px',
      resize: 'vertical',
    }}
    onFocus={(e) => (e.target.style.borderColor = '#059669')}
    onBlur={(e) => (e.target.style.borderColor = '#d1d5db')}
  />
  <div style={infoPanelStyles.charCounter}>
    {formState.description.length}/3000 символів
  </div>
</div>
</div>
</div>
</div>
);
};

```

```

CreateMarkerModal.propTypes = {
  isVisible: PropTypes.bool.isRequired,
  onClose: PropTypes.func.isRequired,
  onCreateMarker: PropTypes.func.isRequired,
  coordinates: PropTypes.shape({
    latitude: PropTypes.number,
    longitude: PropTypes.number,
  }),
};

```

```
export default CreateMarkerModal;
```

Код з файлу `src/components/MarkerFilterSearchPanel.jsx`

```

import React, { useState, useRef, useEffect } from 'react';
import { panelStyles } from '../styles/MarkerFilterSearchPanelStyles.js';

```

```

const MarkerFilterSearchPanel = ({
  searchQuery,
  onSearchQueryChange,
  dateFrom,
  onDateFromChange,
  dateTo,
  onDateToChange,
  eventDateFrom,
  onEventDateFromChange,
  eventDateTo,
  onEventDateToChange,
  selectedTags,
  availableTags,
  onToggleTag,
  onClearTags,
  selectedAuthors,
  availableAuthors,
  onToggleAuthor,
  onClearAuthors,
  showOnlyMyMarkers,
  onShowOnlyMyMarkersChange,
  onClearFilters,
  hasActiveFilters,
  filterStats,

```

					ІАЛЦ.467800.007 Д4	Арк.
						24
Зм.	Арк.	№ докум.	Підпис	Дата		

```

filteredMarkers,
onMarkerSelect
}) => {
  const [isExpanded, setIsExpanded] = useState(false);
  const [activeTab, setActiveTab] = useState('filters');
  const [authorSearchQuery, setAuthorSearchQuery] = useState("");
  const [isAuthorDropdownOpen, setIsAuthorDropdownOpen] = useState(false);

  const authorDropdownRef = useRef(null);
  const authorInputRef = useRef(null);

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

    document.addEventListener('mousedown', handleClickOutside);
    return () => {
      document.removeEventListener('mousedown', handleClickOutside);
    };
  }, []);

  const filteredAuthors = availableAuthors.filter(author =>
    author.toLowerCase().includes(authorSearchQuery.toLowerCase())
  );

  const handleAuthorInputClick = () => {
    setIsAuthorDropdownOpen(!isAuthorDropdownOpen);
  };

  const handleAuthorSearch = (e) => {
    setAuthorSearchQuery(e.target.value);
    if (!isAuthorDropdownOpen) {
      setIsAuthorDropdownOpen(true);
    }
  };

  const handleAuthorSelect = (author) => {
    onToggleAuthor(author);
    setAuthorSearchQuery("");
  };

  const removeAuthor = (author) => {
    onToggleAuthor(author);
  };

  const renderFiltersTab = () => (
    <div style={panelStyles.filtersScrollContainer}>
      <div style={panelStyles.inputGroup}>
        <label style={panelStyles.label}>Пошук по тексту:</label>
        <input
          style={panelStyles.input}
          type="text"
          placeholder="Назва, теги або опис..."
          value={searchQuery}
          onChange={(e) => onSearchQueryChange(e.target.value)}
          onFocus={(e) => e.target.style.borderColor = '#059669'}
          onBlur={(e) => e.target.style.borderColor = '#d1d5db'}
        />
      </div>
    </div>
  );

```

					ІАЛЦ.467800.007 Д4	Арк.
						25
Зм.	Арк.	№ докум.	Підпис	Дата		

```

</div>

<div style={panelStyles.inputGroup}>
  <label style={panelStyles.label}>Рік події:</label>
  <div style={panelStyles.dateRow}>
    <input
      style={panelStyles.input}
      type="number"
      placeholder="3 року (1900)"
      min="1900"
      max="2100"
      value={eventDateFrom}
      onChange={e => onEventDateFromChange(e.target.value)}
      onFocus={e => e.target.style.borderColor = '#059669'}
      onBlur={e => e.target.style.borderColor = '#d1d5db'}
    />
    <input
      style={panelStyles.input}
      type="number"
      placeholder="До року (2025)"
      min="1900"
      max="2100"
      value={eventDateTo}
      onChange={e => onEventDateToChange(e.target.value)}
      onFocus={e => e.target.style.borderColor = '#059669'}
      onBlur={e => e.target.style.borderColor = '#d1d5db'}
    />
  </div>
</div>

<div style={panelStyles.separator}></div>

<div style={panelStyles.inputGroup}>
  <label style={panelStyles.label}>Дата створення:</label>
  <div style={panelStyles.dateRow}>
    <input
      style={panelStyles.input}
      type="date"
      placeholder="3 дати"
      value={dateFrom}
      onChange={e => onDateFromChange(e.target.value)}
      onFocus={e => e.target.style.borderColor = '#059669'}
      onBlur={e => e.target.style.borderColor = '#d1d5db'}
    />
    <input
      style={panelStyles.input}
      type="date"
      placeholder="До дати"
      value={dateTo}
      onChange={e => onDateToChange(e.target.value)}
      onFocus={e => e.target.style.borderColor = '#059669'}
      onBlur={e => e.target.style.borderColor = '#d1d5db'}
    />
  </div>
</div>

<div style={panelStyles.inputGroup}>
  <div style={panelStyles.sectionHeader}>
    <label style={panelStyles.label}>Автор:</label>
    {selectedAuthors.length > 0 && (
      <span

```

					ІАЛЦ.467800.007 Д4	Арк.
						26
Зм.	Арк.	№ докум.	Підпис	Дата		

```

        style={panelStyles.clearSectionButton}
        onClick={onClearAuthors}
      >
        Очистити
      </span>
    )}
  </div>

  <div style={panelStyles.authorDropdownContainer} ref={authorDropdownRef}>
    <div style={panelStyles.authorInputWrapper}>
      <input
        ref={authorInputRef}
        style={{
          ...panelStyles.authorInput,
          ...panelStyles.authorInputFocused(isAuthorDropdownOpen)
        }}
        type="text"
        placeholder="Виберіть автора..."
        value={authorSearchQuery}
        onChange={handleAuthorSearch}
        onClick={handleAuthorInputClick}
        onFocus={(e) => e.target.style.borderColor = '#059669'}
        onBlur={(e) => {
          if (!isAuthorDropdownOpen) {
            e.target.style.borderColor = '#d1d5db';
          }
        }}
      />
      <span style={panelStyles.dropdownArrow(isAuthorDropdownOpen)}>
        ▼
      </span>
    </div>

    {isAuthorDropdownOpen && (
      <div style={panelStyles.authorDropdown}>
        {filteredAuthors.length > 0 ? (
          filteredAuthors.map((author) => (
            <div
              key={author}
              style={panelStyles.authorOption}
              onClick={() => handleAuthorSelect(author)}
              onMouseEnter={(e) => e.target.style.backgroundColor = '#f3f4f6'}
              onMouseLeave={(e) => e.target.style.backgroundColor = 'transparent'}
            >
              <input
                type="checkbox"
                checked={selectedAuthors.includes(author)}
                onChange={() => {}}
                style={{ pointerEvents: 'none' }}
              />
              <span>{author}</span>
            </div>
          ))
        ) : (
          <div style={panelStyles.authorNotFound}>
            {authorSearchQuery ? 'Авторів не знайдено' : 'Немає доступних авторів'}
          </div>
        )}
      </div>
    )}
  </div>
)
</div>

```

					ІАЛЦ.467800.007 Д4	Арк.
						27
Зм.	Арк.	№ докум.	Підпис	Дата		

```

{selectedAuthors.length > 0 && (
  <div style={panelStyles.selectedAuthors}>
    {selectedAuthors.map((author) => (
      <div key={author} style={panelStyles.selectedAuthorTag}>
        <span>{author}</span>
        <button
          style={panelStyles.removeAuthorButton}
          onClick={() => removeAuthor(author)}
          onMouseEnter={(e) => e.target.style.backgroundColor = 'rgba(255,255,255,0.2)'}
          onMouseLeave={(e) => e.target.style.backgroundColor = 'transparent'}
        >
          ×
        </button>
      </div>
    ))}
  </div>
)}
</div>

<label style={panelStyles.checkbox}>
  <input
    type="checkbox"
    checked={showOnlyMyMarkers}
    onChange={(e) => onShowOnlyMyMarkersChange(e.target.checked)}
  />
  ТІЛЬКИ МОЇ МІТКИ
</label>
</div>
);

const renderResultsTab = () => (
  <div style={panelStyles.resultsContainer}>
    {filteredMarkers.length > 0 ? (
      <div style={panelStyles.resultsList}>
        {filteredMarkers.slice(0, 50).map((marker) => (
          <div
            key={marker.id}
            style={panelStyles.resultItem}
            onClick={() => onMarkerSelect && onMarkerSelect(marker)}
            onMouseEnter={(e) => e.target.style.backgroundColor = '#e5e7eb'}
            onMouseLeave={(e) => e.target.style.backgroundColor = 'transparent'}
          >
            <div style={panelStyles.resultTitle}>
              {marker.title || 'Без назви'}
            </div>
            <div style={panelStyles.resultMeta}>
              {marker.user_email} • {marker.event_date || 'Без дати'}
              {marker.tags && ` ` * ${marker.tags} ``}
            </div>
          </div>
        ))}
        {filteredMarkers.length > 50 && (
          <div style={{
            ...panelStyles.resultItem,
            ...panelStyles.resultItemMoreIndicator
          }}>
            ... і ще {filteredMarkers.length - 50} меток
          </div>
        )}
      </div>
    )}
  </div>
);

```

					ІАЛЦ.467800.007 Д4	Арк.
						28
Зм.	Арк.	№ докум.	Підпис	Дата		

```

): (
  <div style={panelStyles.emptyState}>
    {hasActiveFilters ? (
      <div>🔍 Нічого не знайдено</div>
      <div style={{ marginTop: '8px', fontSize: '12px' }}>
        Спробуйте змінити критерії пошуку
      </div>
    ) : (
      <div>🏷 Всі мітки</div>
      <div style={{ marginTop: '8px', fontSize: '12px' }}>
        Використайте фільтри для пошуку
      </div>
    ) : (
      <div>🔍 Пошук і фільтри
      {hasActiveFilters && (
        <span style={panelStyles.headerBadge}>
          {filterStats.visible}
        </span>
      )}
    </h3>
    <span style={panelStyles.headerArrow(isExpanded)}>
      ▼
    </span>
  </div>

  {isExpanded && (
    <div style={panelStyles.tabs}>
      <div
        style={{
          ...panelStyles.tab,
          ...(activeTab === 'filters' ? panelStyles.activeTab : {})}
      >
        ⚙ Фільтри
      </div>
      <div
        style={{
          ...panelStyles.tab,
          ...(activeTab === 'results' ? panelStyles.activeTab : {})}
      >
        📄 Результати ({filterStats.visible})
      </div>
    </div>
  )}
)

```

					ІАЛЦ.467800.007 Д4	Арк.
						29
Зм.	Арк.	№ докум.	Підпис	Дата		

```

<div style={panelStyles.content(isExpanded)}>
  {activeTab === 'filters' ? renderFiltersTab() : renderResultsTab()}

  <div style={panelStyles.statsInfo}>
    <span>
      Видимих: {filterStats.visible} / {filterStats.total}
      {filterStats.hidden > 0 && (
        <span style={panelStyles.hiddenCount}>
          (приховано: {filterStats.hidden})
        </span>
      )}
    </span>
    {hasActiveFilters && (
      <button
        style={panelStyles.clearButton}
        onClick={onClearFilters}
        onMouseEnter={(e) => e.target.style.backgroundColor = '#b91c1c'}
        onMouseLeave={(e) => e.target.style.backgroundColor = '#ef4444'}
      >
        Скинути все
      </button>
    )}
  </div>
</div>
</div>
);
};

```

export default MarkerFilterSearchPanel;

Код з файлу src/hooks/useMapbox.js

```

import { useRef, useState, useEffect } from 'react';
import mapboxgl from 'mapbox-gl';

mapboxgl.accessToken = 'API_KEY_HERE';

export const useMapbox = (mapContainerRef, isPanelOpen) => {
  const mapRef = useRef(null);
  const [mapLoaded, setMapLoaded] = useState(false);
  const [mapError, setMapError] = useState(null);

  const initializeMap = () => {
    if (mapRef.current) return;

    try {
      mapRef.current = new mapboxgl.Map({
        container: mapContainerRef.current,
        style: 'mapbox://styles/mapbox/light-v11',
        center: [31.1656, 48.3794],
        zoom: 5,
        attributionControl: true,
        collectResourceTiming: false,
      });

      mapRef.current.on('load', () => {
        setMapLoaded(true);
        setMapError(null);
        const map = mapRef.current;
        console.log('Map loaded. Adding language control...');
      });
    } catch (error) {
      setMapError(error);
    }
  };
};

```

					ІАЛЦ.467800.007 Д4	Арк.
						30
Зм.	Арк.	№ докум.	Підпис	Дата		


```

    map.getStyle().layers.forEach((layer) => {
    if (layer.id.endsWith('-label')) {
        map.setLayoutProperty(layer.id, 'text-field', [
            'coalesce',
            ['get', 'name_uk-Hant'],
            ['get', 'name'],
        ]);
    }
    })
    })

    mapRef.current.on('error', (e) => {
        setMapError('Помилка завантаження карти: ' + e.error?.message || 'Помилка невідома');
    });

    } catch (error) {
        setMapError('Помилка ініціалізації карти: ' + error.message);
    }

    return () => {
        if (mapRef.current) {
            mapRef.current.remove();
            mapRef.current = null;
        }
    };
};

useEffect(() => {
    if (!mapRef.current || !mapLoaded) return;

    const timeoutId = setTimeout(() => {
        mapRef.current.resize();
    }, 350);

    return () => clearTimeout(timeoutId);
}, [isPanelOpen, mapLoaded]);

return { mapRef, mapLoaded, mapError, initializeMap };
};

```

Код з файлу **src/hooks/useMapMarkers.js**

```

import { useRef, useCallback, useEffect } from 'react';
import mapboxgl from 'mapbox-gl';
import { createMarkerElement } from '../utils/markerElement';

const useMapMarkers = (mapRef, mapLoaded, markers, currentUserEmail, selectedMarker, onMarkerSelect,
onMarkerDelete) => {
    const markersRef = useRef([]);

    const closeAllPopups = useCallback(() => {
    }, []);

    const handleShowMarkerDetails = useCallback((markerData) => {
        const isModerated = markerData && markerData.is_moderated !== undefined ? markerData.is_moderated : true;

        if (!isModerated) {

```

					ІАЛЦ.467800.007 Д4	Арк.
						31
Зм.	Арк.	№ докум.	Підпис	Дата		

```

    return;
  }

  if (onMarkerSelect) {
    onMarkerSelect(markerData);
  }
}, [onMarkerSelect]);

useEffect(() => {
  window.deleteMarker = async (markerId) => {
    if (window.confirm('Ви впевнені, що хочете видалити цю мітку?')) {
      try {
        await onMarkerDelete(markerId);
      } catch (error) {
        console.error('Помилка видалення мітки:', error);
        alert('Помилка видалення мітки: ' + (error.response?.data?.error || error.message));
      }
    }
  };

  return () => {
    delete window.deleteMarker;
  };
}, [onMarkerDelete]);

const updateMarkers = useCallback(() => {
  console.log('Оновлення маркерів на:', markers?.length || 0);

  markersRef.current.forEach(marker => {
    try {
      marker.remove();
    } catch (error) {
      console.warn('Помилка при видаленні маркера:', error);
    }
  });
  markersRef.current = [];

  markers.forEach(markerData => {
    try {
      const isSelected = selectedMarker && selectedMarker.id === markerData.id;

      const isModerated = markerData && markerData.is_moderated !== undefined ? markerData.is_moderated : true;

      const markerElement = createMarkerElement(markerData.icon || 'MapPin', isSelected, markerData);

      const opacity = markerData.opacity !== undefined ? markerData.opacity : 1;
      markerElement.style.opacity = opacity;

      const marker = new mapboxgl.Marker({
        element: markerElement
      })
        .setLngLat([markerData.longitude, markerData.latitude])
        .addTo(mapRef.current);

      marker.getElement().addEventListener('click', (e) => {
        e.stopPropagation();

        if (isModerated) {
          handleShowMarkerDetails(markerData);
        }
      });
    }
  });
}

```

					ІАЛЦ.467800.007 Д4	Арк.
						32
Зм.	Арк.	№ докум.	Підпис	Дата		

```

    } catch (error) {
      console.error('Помилка додавання маркера:', markerData.id, error);
    }
  });

}, [markers, mapLoaded, currentUserEmail, selectedMarker, handleShowMarkerDetails]);

useEffect(() => {
  updateMarkers();
}, [markers, selectedMarker, mapLoaded]);

return { updateMarkers, closeAllPopups };
};

export { useMapMarkers };

```

Код з файлу **src/hooks/useMapInteraction.js**

```

import { useState, useCallback, useEffect } from 'react';

const useMapInteractions = (mapRef, mapLoaded, onMarkerAdd, onCloseAllPanels, currentUserRole,
isAddingMarker, setIsAddingMarker) => {
  const [pendingMarker, setPendingMarker] = useState(null);
  const [showCreateModal, setShowCreateModal] = useState(false);
  const [cursorCoordinates, setCursorCoordinates] = useState(null);

  const handleMapClick = useCallback(async (e) => {
    if (isAddingMarker) {
      e.preventDefault();
      const { lng, lat } = e.lngLat;

      setPendingMarker({
        latitude: lat,
        longitude: lng
      });
      setShowCreateModal(true);
      return;
    }
    const target = e.originalEvent.target;
    const isMapClick = target.classList.contains('mapboxgl-canvas') ||
      target.closest('.mapboxgl-canvas');

    const isMarkerClick = target.closest('.custom-marker') ||
      target.classList.contains('custom-marker');

    if (isMapClick && !isMarkerClick && onCloseAllPanels) {
      onCloseAllPanels();
    }
  }, [isAddingMarker, onCloseAllPanels]);

  const handleCreateMarker = useCallback(async (markerData) => {
    if (!pendingMarker) return;
    try {
      const isModerated = currentUserRole === 'admin';

      await onMarkerAdd({
        ...pendingMarker,
        ...markerData,
        is_moderated: isModerated
      });
    }
  },

```

					ІАЛЦ.467800.007 Д4	Арк.
						33
Зм.	Арк.	№ докум.	Підпис	Дата		

```

    setShowCreateModal(false);
    setPendingMarker(null);
    setIsAddingMarker(false);
  } catch (error) {
    console.error('Помилка додавання мітки:', error);
    throw error;
  }
}, [pendingMarker, onMarkerAdd, currentUserRole, setIsAddingMarker]);

const handleCancelCreate = useCallback(() => {
  setShowCreateModal(false);
  setPendingMarker(null);
}, []);

const handleMouseMove = useCallback((e) => {
  if (isAddingMarker) {
    const { lng, lat } = e.lngLat;
    setCursorCoordinates({
      latitude: lat,
      longitude: lng
    });
  }
}, [isAddingMarker]);

useEffect(() => {
  if (!mapRef.current || !mapLoaded) return;

  const map = mapRef.current;
  const canvas = map.getCanvas();

  if (canvas) {
    canvas.style.cursor = isAddingMarker ? 'crosshair' : '';
  }

  const methods = ['dragPan', 'scrollZoom', 'boxZoom', 'keyboard', 'doubleClickZoom', 'touchZoomRotate'];

  methods.forEach(method => {
    if (isAddingMarker) {
      map[method].disable();
    } else {
      map[method].enable();
    }
  });

  if (isAddingMarker) {
    map.on('mousemove', handleMouseMove);
  } else {
    map.off('mousemove', handleMouseMove);
    setCursorCoordinates(null);
  }

  return () => {
    if (map.isStyleLoaded()) {
      map.off('mousemove', handleMouseMove);
    }
  };
}, [isAddingMarker, mapLoaded, handleMouseMove]);

return {
  handleMapClick,

```

					ІАЛЦ.467800.007 Д4	Арк.
						34
Зм.	Арк.	№ докум.	Підпис	Дата		

```

    showCreateModal,
    pendingMarker,
    handleCreateMarker,
    handleCancelCreate,
    cursorCoordinates
  };
};

export { useMapInteractions };

```

Код з файлу `src/utlils/api.js`

```

import axios from 'axios';

const API_BASE_URL = 'http://localhost:5000/api';

const api = axios.create({
  baseURL: API_BASE_URL,
  headers: {
    'Content-Type': 'application/json',
  },
});

api.interceptors.request.use(
  (config) => {
    const token = localStorage.getItem('token');
    if (token) {
      config.headers.Authorization = `Bearer ${token}`;
    }
    return config;
  },
  (error) => {
    return Promise.reject(error);
  }
);

api.interceptors.response.use(
  (response) => {
    return response;
  },
  (error) => {
    if (error.response?.status === 401 || error.response?.status === 403) {
      localStorage.removeItem('token');
      localStorage.removeItem('user');
      window.location.href = '/login';
    }
    return Promise.reject(error);
  }
);

export const authAPI = {
  register: async (userData) => {
    const response = await api.post('/auth/register', userData);
    return response.data;
  },

  login: async (credentials) => {
    const response = await api.post('/auth/login', credentials);

    if (response.data.token) {
      localStorage.setItem('token', response.data.token);
    }
  }
};

```

					ІАЛЦ.467800.007 Д4	Арк.
						35
Зм.	Арк.	№ докум.	Підпис	Дата		

```

    localStorage.setItem('user', JSON.stringify(response.data.user));
  }

  return response.data;
}
};

export const markersAPI = {
  getAll: async () => {
    const response = await api.get('/markers');
    return response.data;
  },

  create: async (markerData) => {
    const response = await api.post('/markers', markerData);
    return response.data;
  },

  update: async (markerId, markerData) => {
    const response = await api.put(`/markers/${markerId}`, markerData);
    return response.data;
  },

  delete: async (markerId) => {
    const response = await api.delete(`/markers/${markerId}`);
    return response.data;
  }
};

export const isAuthenticated = () => {
  const token = localStorage.getItem('token');
  if (!token) return false;

  try {
    const payload = JSON.parse(atob(token.split('.')[1]));
    const currentTime = Date.now() / 1000;

    if (payload.exp < currentTime) {
      localStorage.removeItem('token');
      localStorage.removeItem('user');
      return false;
    }

    return true;
  } catch (error) {
    console.error('Помилка при перевірці токена:', error);
    localStorage.removeItem('token');
    localStorage.removeItem('user');
    return false;
  }
};

export const getCurrentUser = () => {
  const userStr = localStorage.getItem('user');
  if (!userStr) return null;

  try {
    return JSON.parse(userStr);
  } catch (error) {
    console.error('Помилка при отриманні даних користувача:', error);
    return null;
  }
};

```

					ІАЛЦ.467800.007 Д4	Арк.
						36
Зм.	Арк.	№ докум.	Підпис	Дата		

```

    }
  };

export const logout = () => {
  localStorage.removeItem('token');
  localStorage.removeItem('user');
};

export const testConnection = async () => {
  try {
    const response = await api.get('/test');
    return response.data;
  } catch (error) {
    throw error;
  }
};

export const moderationAPI = {
  getUnmoderated: async () => {
    const response = await api.get('/markers/unmoderated');
    return response.data;
  },

  approveMarker: async (markerId) => {
    const response = await api.put(`/markers/${markerId}/moderate`, { is_moderated: 1 });
    return response.data;
  },

  rejectMarker: async (markerId) => {
    const response = await api.delete(`/markers/${markerId}`);
    return response.data;
  },

  getModerationStats: async () => {
    const response = await api.get('/markers/moderation-stats');
    return response.data;
  },

  updateMarker: async (markerId, markerData) => {
    const response = await api.put(`/markers/${markerId}`, markerData);
    return response.data;
  }
};

export default api;

```

Код з файлу src/utlis/auth.js

```

export const getCurrentUserEmail = () => {
  const token = localStorage.getItem('token');
  if (!token) return null;

  try {
    const payload = JSON.parse(atob(token.split('.')[1]));
    return payload.email;
  } catch {
    return null;
  }
};

export const getCurrentUserRole = () => {

```

					ІАЛЦ.467800.007 Д4	Арк.
						37
Зм.	Арк.	№ докум.	Підпис	Дата		

```
const token = localStorage.getItem('token');
if (!token) return null;

try {
  const payload = JSON.parse(atob(token.split('.')[1]));
  return payload.role || 'user';
} catch {
  return null;
}
};
```

					ІАЛЦ.467800.007 Д4	Арк.
						38
Зм.	Арк.	№ докум.	Підпис	Дата		