НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра інформаційних систем та технологій

До захисту допущено:

Завідувач кафедри

_____ Олександр РОЛІК

«___»_____2025 р.

Дипломний проєкт

на здобуття ступеня бакалавра
за освітньо-професійною програмою «Інформаційні управляючі системи та технології»
спеціальності 126 «Інформаційні системи та технології»
на тему: «Рекомендаційна система туристичних маршрутів в українських Карпатах»

Виконав:
студент IV курсу, групи ІС-12
Василик Арсеній Віталійович _____

Керівник:
Асистент кафедри ІСТ,
Лесик Валентин Олександрович _____

Рецензент:
Асистент кафедри ІПІ,
Зарічковий Олександр Анаталійович _____

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

Київ – 2025 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 126 «Інформаційні системи та технології»

Освітньо-професійна програма «Інформаційні управляючі системи та технології»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Олександр РОЛІК
«___»_____2025 р.

ЗАВДАННЯ
на дипломний проєкт студенту
Василику Арсенію Віталійовичу

1. Тема проєкту «Рекомендаційна система туристичних маршрутів в українських Карпатах», керівник проєкту Лесик Валентин Олександрович, затверджені наказом по університету від «23» травня 2025 р. № 1705-с

2. Термін подання студентом проєкту: 9 червня 2025

3. Вихідні дані до проєкту: система повинна підтримувати можливість додавати та переглядати туристичні маршрути, фільтрувати їхні параметри.
4. Зміст пояснювальної записки: 1. Опис предметної області, 2. Аналіз готових рішень, 3. Формування вимог до системи, 4. Вибір технологій розроблення, 5. Розроблення інформаційної системи, 6. Математичне забезпечення, 7. Тестування системи, Висновки, Перелік посилань.

5. Перелік графічного матеріалу (із зазначенням обов’язкових креслеників, плакатів, презентацій тощо)
1. Діаграма прецедентів;
2. Діаграма компонентів клієнтської частини;
3. Діаграма компонентів серверної частини;
4. Діаграма послідовності створення маршруту;
5. Блок-схема алгоритму знаходження відстані.

6. Дата видачі завдання 25 березня 2025 р.

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
	Аналіз предметної області	18.04.2025	
	Огляд існуючих аналогів	22.04.2024	
	Розроблення функціональної моделі	30.04.2025	
	Формування вимог до системи	03.05.2025	
	Вибір технології розроблення	08.05.2025	
	Розроблення програмного забезпечення	10.05.2025	
	Опис технічного рішення	17.05.2025	
	Дослідження і розроблення математичного забезпечення	22.05.2025	
	Тестування системи	28.05.2025	

Студент Арсеній ВАСИЛИК

Керівник Валентин ЛЕСИК

АНОТАЦІЯ

Рекомендаційна система туристичних маршрутів в українських Карпатах. Проєкт містить 72 с. тексту, 19 рисунків, 13 таблиць, посилання на 23 літературні джерела, додатки та 5 конструкторських документи.

ТУРИСТИЧНІ МАРШРУТИ, ГЕОІНФОРМАЦІЙНІ СИСТЕМИ, REACT, NODE.JS, MONGODB.

Об'єктом розроблення є планування туристичних маршрутів у гірському регіоні українських Карпат.

Мета розроблення – підвищення безпеки гірського туризму через створення системи автоматизованої оцінки складності маршрутів. У дипломному проєкті розроблено вебдодаток для створення, аналізу та управління туристичними маршрутами в Карпатах. Система включає клієнтську частину на React з інтерактивними картами Leaflet, серверну частину на Node.js з Express та базу даних MongoDB. Розроблено математичне забезпечення для обчислення відстані маршрутів за формулою гаверсинусів, визначення набору висоти через інтеграцію з Open-Elevation API та алгоритм оцінки складності маршрутів. Система підтримує створення маршрутів вручну на карті та імпорт GPX-файлів, забезпечує фільтрацію та сортування за параметрами довжини, набору висоти та складності.

Отримані результати можуть бути корисними для туристів при плануванні безпечних походів у гірських регіонах та для туристичних організацій при розробці маршрутної мережі.

SUMMARY

Recommendation system for tourist routes in the Ukrainian Carpathians.

The project contains 72 pages of text, 19 figures, 13 tables, references to 23 literary sources, appendices and 5 design documents.

Keywords: tourist routes, geographic information systems, react, node.js, mongodb.

The object of development is planning tourist routes in the mountain region of the Ukrainian Carpathians.

The purpose of the development is to improve the safety of mountain tourism through creating a system for automated assessment of route difficulty.The graduation project developed a web application for creating, analyzing and managing tourist routes in the Carpathians. The system includes a client part on React with interactive Leaflet maps, a server part on Node.js with Express and a MongoDB database. Mathematical support has been developed for calculation of route distances using the Haversine formula, determination of elevation gain through integration with the Open-Elevation API and an algorithm for assessing route difficulty. The system supports manual route creation on the map and GPX file import, provides filtering and sorting by length, elevation gain and difficulty parameters.

The results obtained can be useful for tourists when planning safe hikes in mountainous regions and for tourism organizations when developing route networks.

№ рядка	Формат	Позначення	Найменування	Кіл. аркушів	№ екз.	Примітка
1			Документація загальна			
2						
3			Знову розроблена			
4						
5	А4	ІС12.060БАК.005 ПЗ	Рекомендаційна система	72		
6			туристичних маршрутів в			
7			українських Карпатах.			
8			Пояснювальна записка			
9	А3	ІС12.060БАК.005 Д1	Рекомендаційна система	1		
10			туристичних маршрутів в			
11			українських Карпатах. Діаграма			
12			прецедентів			
13	А3	ІС12.060БАК.005 Д2	Рекомендаційна система	1		
14			туристичних маршрутів в			
15			українських Карпатах. Діаграма			
16			компонентів клієнтської частини			
17	А3	ІС12.060БАК.005 Д3	Рекомендаційна система	1		
18			туристичних маршрутів в			
19			українських Карпатах. Діаграма			
20			компонентів серверної частини			
21	А3	ІС12.060БАК.005 Д4	Рекомендаційна система	1		
22			туристичних маршрутів в			
23			українських Карпатах. Діаграма			
24			послідовності створення маршруту			
25	А3	ІС12.060БАК.005 Д5	Рекомендаційна система	1		
26			туристичних маршрутів в			
27			українських Карпатах. Блок-схема			
28			алгоритму знаходження відстані			

Зм.	Лист	№ докум.	Підпис		ІС12.060БАК.005 ТП		
Розробив		Василик			Рекомендаційна система туристичних маршрутів в українських Карпатах. Відомість дипломного проєкту	Літ.	Арк. 1 / Аркушів 1
Перевірив		Лесик				Т	
Затв.						КПІ ім. Ігоря Сікорського Група ІС-12	

Пояснювальна записка
до дипломного проєкту
на тему: «Рекомендаційна система туристичних маршрутів в українських Карпатах»

Київ – 2025 року

ЗМІСТ

ВСТУП

Туристична індустрія в Україні, зокрема в Карпатському регіоні, є однією з найперспективніших галузей економіки, що демонструє стабільне зростання протягом останніх років. Карпати, завдяки своїм унікальним природним ландшафтам, багатій культурній спадщині та різноманітним можливостям для активного відпочинку, щороку приваблюють тисячі туристів як з України, так і з інших країн. Гірський туризм, піші походи, велотуризм та екотуризм стали провідними напрямками розвитку регіону, сприяючи економічному зростанню та підвищенню привабливості українських Карпат на міжнародному рівні.

Проте інтенсивний розвиток гірського туризму виявив серйозну проблему безпеки – значна частина туристів не має достатньої теоретичної підготовки чи фізичної готовності для проходження складних гірських маршрутів. Це призводить до зростання кількості нещасних випадків у гірській місцевості, що створює додаткове навантаження на рятувальні служби та негативно впливає на репутацію туристичної галузі [1]. Статистика показує, що більшість інцидентів у гірській місцевості пов'язана саме з неправильним вибором маршруту, недооцінкою його складності або недостатньою підготовкою туристів до специфічних умов гірського середовища.

Сучасний стан інформаційного забезпечення туристичної діяльності в українських Карпатах характеризується наявністю численних ресурсів класичних паперових мап та путівників, туристичних веб-сайтів, електронних карт, та мобільних додатків. Однак аналіз показує, що користувачі часто стикаються з труднощами при виборі маршруту, що відповідає їхнім фізичним можливостям та рівню підготовки. Непідготовлений турист зазвичай не здатний самостійно об'єктивно оцінити складність маршруту, і відповідно правильно розрахувати необхідні фізичні зусилля та час для його проходження.

Особливо проблематичною є ситуація із суб'єктивними оцінками складності маршрутів, що надаються іншими користувачами в існуючих туристичних додатках. Досвідчені туристи часто применшують реальну складність маршрутів

відповідно до власного високого рівня підготовки, що може ввести в оману менш досвідчених мандрівників. Це створює потребу в розробці об'єктивних, автоматизованих методів оцінки складності туристичних маршрутів, що ґрунтуються на географічних параметрах, як основи для підвищення безпеки гірського туризму.

Розвиток сучасних інформаційних технологій відкрив нові можливості для вирішення зазначених проблем безпеки. Геоінформаційні системи (ГІС), технології веброзробки, алгоритми машинного навчання та методи обробки великих масивів географічних даних дозволяють створювати інтелектуальні рекомендаційні системи для туристичної галузі. Такі системи здатні автоматизувати процеси аналізу географічних даних, обчислення параметрів маршрутів та надання об'єктивних оцінок складності маршрутів користувачам, що є критично важливим для забезпечення їх комфорту та безпеки.

Рекомендаційні системи в туризмі є перспективним напрямком досліджень. Особливо актуальним є розвиток таких систем для гірських регіонів, де правильний вибір маршруту є критично важливим для безпеки туристів та може запобігти нещасним випадкам.

Об'єктом розроблення є процес планування та вибору туристичних маршрутів в українських Карпатах з урахуванням підготовки користувачів та об'єктивних характеристик маршрутів.

Предметом розроблення є алгоритми автоматизованого розрахунку географічних параметрів туристичних маршрутів, методи оцінки їх складності на основі цих параметрів, а також технології веброзробки для створення інтерактивних геоінформаційних систем планування маршрутів..

Метою розроблення проєкту є підвищення безпеки гірського туризму в українських Карпатах через розроблення системи автоматизованої оцінки складності туристичних маршрутів.

1 ОПИС ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Опис процесу діяльності

Туристична індустрія в Україні, зокрема в Карпатському регіоні, протягом останніх десятиліть демонструє стабільне зростання, залучаючи дедалі більше місцевих та іноземних мандрівників. Карпати, з їхніми мальовничими пейзажами, багатими природними та культурними ресурсами, стали популярним напрямком для активного відпочинку, піших походів, велотуризму та екотуризму [2]. Однак зростання попиту на туристичні послуги супроводжується підвищенням вимог до інструментів планування подорожей. Сучасні туристи прагнуть швидко знаходити маршрути, які відповідають їхнім інтересам, фізичним можливостям і часовими обмеженнями, уникаючи тривалого пошуку інформації. Традиційні засоби, такі як паперові карти, друковані путівники чи загальні туристичні вебсайти, часто не відповідають цим потребам, оскільки не надають детальних даних про параметри маршрутів, їхню складність чи актуальні умови.

Розвиток інформаційних технологій, зокрема геоінформаційних систем (ГІС), вебтехнологій та мобільних технологій, а також алгоритмів обробки великих масивів даних, створив нові можливості для подальшого вдосконалення туристичних сервісів [3]. Рекомендаційні системи в туризмі, які поєднують аналіз географічних даних із персоналізацією, стали важливим інструментом для задоволення потреб сучасних мандрівників. Такі системи дозволяють автоматизувати процеси збору, обробки та аналізу інформації про маршрути, пропонуючи користувачам варіанти подорожей із урахуванням їхніх уподобань. У контексті Карпат, де складний гірський рельєф, мінливі погодні умови та віддаленість багатьох локацій створюють додаткові виклики, подібні системи відіграють ключову роль у забезпеченні безпеки, комфорту та якості туристичного досвіду.

Процес планування туристичних маршрутів у гірських регіонах є багатогранним і включає кілька етапів. На першому етапі користувачі визначають свої цілі та можливості, доступний час, рівень власної фізичної підготовки,

можливості добирання до різних стартових локацій. Далі необхідно зібрати географічні дані, включаючи координати стартових та кінцевих точок, відстані, перепади висот і характеристики місцевості. На основі цих даних система оцінює складність маршрутів. Нарешті, рекомендаційна система фільтрує та ранжує маршрути, пропонуючи користувачам ті, що найкраще відповідають їхнім критеріям [4].

Варто зазначити, що в загальному випадку складність різних гірських маршрутів важко порівняти, через те що кожна гірська система відрізняється одна від одної. До прикладу, в українській частині Карпат гори менш скелясті та більш пологі, на відміну від Високих Татр [5]. Відповідно складний маршрут в перших може, бути легким в других. Тому складність маршрутів необхідно порівнювати відносно гірського регіону де вони знаходяться.

Технологічна основа таких систем базується на інтеграції геоінформаційних сервісів і сучасних програмних рішень. Наприклад, API для отримання даних про висоту дозволяють точно визначати топографічні характеристики маршрутів, а бібліотеки для обробки геоданих спрощують обчислення відстаней і траєкторій. Крім того, підтримка імпорту даних із GPS-пристроїв, в форматі GPX-файли, розширює можливості для користувачів, дозволяючи адаптувати вже створені маршрути до їхніх потреб. Важливим є також забезпечення доступу до актуальної інформації, наприклад, про погодні умови чи стан стежок, що може впливати на безпеку та комфорт подорожі.

Інтерфейси рекомендаційних систем у туризмі повинні бути інтуїтивними та адаптивними, щоб відповідати потребам широкої аудиторії – від початківців до досвідчених мандрівників. Сучасні технології веброзробки дозволяють створювати інтерактивні карти, на яких користувачі можуть малювати маршрути, переглядати деталі, додавати описи, зображення, власні мітки на позначення якихось локацій. Функції фільтрації за різноманітними параметрами, такими як довжина, складність, тип активності чи наявність якихось особливих туристично привабливих локацій на маршрутах, а також збереження маршрутів для подальшого використання, підвищують зручність і залученість користувачів.

Масштабованість і гнучкість є ключовими вимогами до розроблення таких систем. Туристичні потреби постійно змінюються, а нові технології відкривають нові можливості для вдосконалення. Наприклад, у майбутньому системи можуть інтегрувати дані про погоду в реальному часі, наприклад штурмові попередження, що даються метеослужбами, попередження рятувальних служб про небезпеку, тип поверхні маршруту, транспортну доступність чи туристичні об’єкти. Модульна архітектура дозволяє додавати ці функції без значних змін у базовому коді, забезпечуючи довгострокову актуальність системи.

Розроблення рекомендаційних систем у туризмі є перспективною галуззю, яка поєднує технологічні інновації з практичними потребами користувачів. У Карпатах такі системи мають особливе значення, адже вони допомагають туристам відкривати нові маршрути, планувати свої мандрівки безпечніше та отримувати кращі враження від відпочинку в горах. Сфера застосування може подібних систем може поширюватися не тільки на самостійних туристів, а й на туристичні фірми, рятувальні служби, які можуть оцінити складність тих чи інших маршрутів, та навіть екологічні установи, що можуть за даними про популярність різних маршрутів оцінити навантаження на екосистему.

1.2 Постановка задачі

Система призначення для підбору туристичних маршрутів в Карпатах.

Ціллю розроблення є створення рекомендаційної системи туристичних маршрутів. Така система повинна надавати користувачам можливість знаходити маршрут, що відповідатиме їхнім потребам та підготовці.

Для досягнення поставлених цілей необхідно вирішити такі задачі:

1) провести аналіз наявних на ринку рекомендаційних систем для туризму, зокрема тих, що працюють із гірськими маршрутами;

2) розробити інформаційну систему для створення, імпорту та управління туристичними маршрутами;

3) розробити математичне забезпечення для автоматичної оцінки складності маршрутів на основі відстані та набору висоти;

4) провести тестування розробленої системи.

Висновки до розділу 1

Під час роботи над розділом було проведено дослідження предметної області та встановлено, що рекомендаційні системи для туризму є перспективною галуззю, яка поєднує геоінформаційні технології, обробку даних і сучасні інтерфейси користувача. Було описано процес діяльності, який охоплює планування маршрутів, аналіз географічних даних і рекомендацію підходящих варіантів для туристів у Карпатах. Визначено цілі розроблення та задачі, які необхідно вирішити для їх досягнення. Розроблення таких систем має значний потенціал для вдосконалення туристичної індустрії України, сприяючи безпечним і комфортним подорожам у гірських регіонах.

2 АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ

Оскільки предметом розроблення є рекомендаційна система для туристичних маршрутів в українських Карпатах, у цьому розділі розглянуто найвідоміші сервіси та додатки, які надають користувачам можливість планувати маршрути в гірських регіонах, аналізуючи їхні параметри, такі як відстань, набір висоти та складність. Кожен із розглянутих продуктів має унікальні особливості, які забезпечують зручність для туристів, але водночас має певні обмеження, що впливають на їхню ефективність у контексті Карпат.

Найпопулярнішими сервіси для планування туристичних маршрутів у гірських регіонах на сьогодні є Komoot, AllTrails і Wikiloc.

Komoot – це один із провідних додатків для планування маршрутів, випущений у 2010 році, який підтримує піші походи, велотуризм і біг [6]. Основною метою сервісу є створення туристичних маршрутів із урахуванням типу активності, рівня фізичної підготовки користувача та його вподобань. Komoot використовує дані OpenStreetMap (OSM) і власні алгоритми для побудови маршрутів, надаючи детальну інформацію про відстань, набір висоти та тип поверхні. На рисунку 2.1 зображено інтерфейс головної сторінки Komoot.

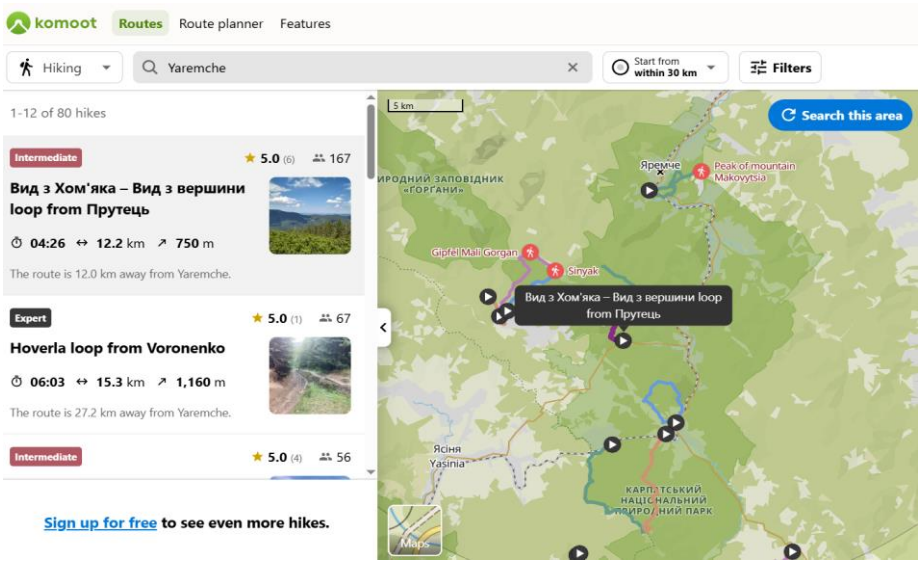

Рисунок 2.1 – Знімок екрану додатку Komoot

Перевагою Komoot є інтуїтивний інтерфейс і можливість створювати маршрути як вручну, так і автоматично, з урахуванням популярних стежок. Система пропонує профіль висот, що дозволяє оцінити складність маршруту, а також підтримує імпорт і експорт GPX-файлів. Крім того, Komoot має функцію спільноти, де користувачі діляться маршрутами та відгуками. Недоліком є платний доступ до багатьох функцій, таких як офлайн-карти чи детальніші рекомендації, що може обмежувати використання для туристів із невеликим бюджетом. Через малу популярність даного додатку в Україні він дає невеликий вибір маршрутів.

AllTrails – це популярна платформа, запущена у 2010 році, яка спеціалізується на піших походах і пропонує базу даних із мільйонами маршрутів по всьому світу [7]. Сервіс дозволяє користувачам шукати маршрути за параметрами, такими як довжина, складність, тип місцевості чи наявність туристичних об'єктів. AllTrails використовує дані від користувачів і топографічні карти для надання інформації про перепади висот і характеристики маршрутів. На рисунку 2.2 зображено інтерфейс головної сторінки AllTrails.

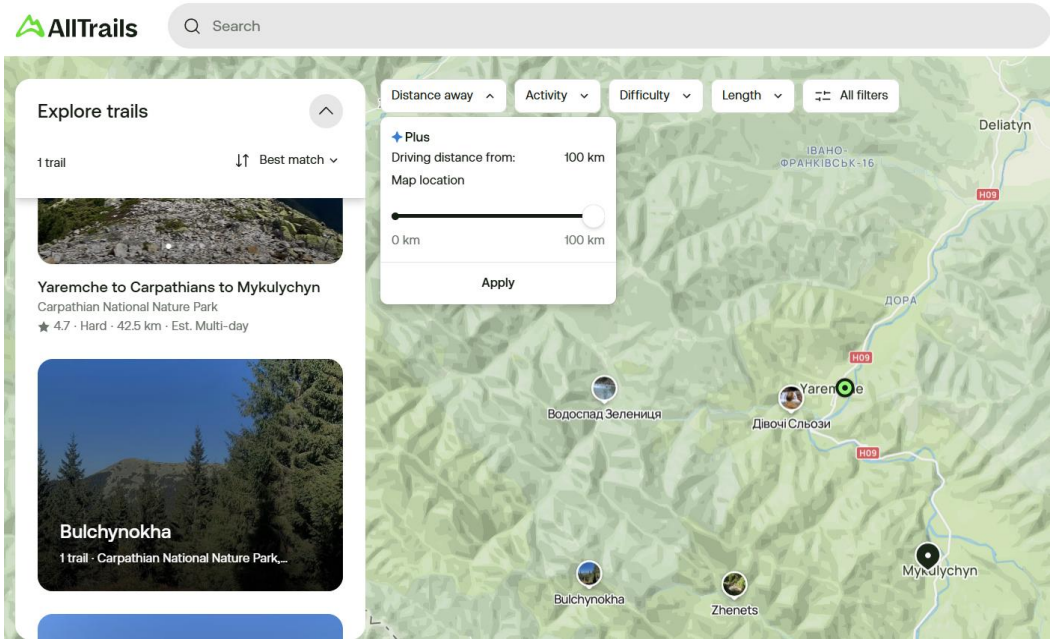

Рисунок 2.2 – Знімок екрана додатку AllTrails

Перевагами AllTrails є велика кількість маршрутів, детальні відгуки користувачів і можливість завантажувати офлайн-карти (у преміум-версії).

Система оцінює складність маршрутів, але часто покладається на суб'єктивні оцінки користувачів, що може бути менш точним порівняно з автоматизованими алгоритмами. Недоліком безпосередньо для Карпат є обмежена кількість маршрутів, а також платний доступ до багатьох функцій.

Wikiloc – це додаток і вебплатформа, запущена у 2006 році, призначена для планування та обміну туристичними маршрутами для піших походів, велотуризму, бігу та інших активностей [8]. Wikiloc має велику базу даних із понад 59 мільйонами маршрутів, створених спільнотою користувачів, і дозволяє планувати власні маршрути за допомогою функції "Plan your trail", яка використовує популярні ділянки стежок для створення безпечних і перевірених маршрутів. Сервіс підтримує топографічні карти, профілі висот і GPS-навігацію. Інтерфейс головної сторінки Wikiloc зображено на рисунку 2.3.

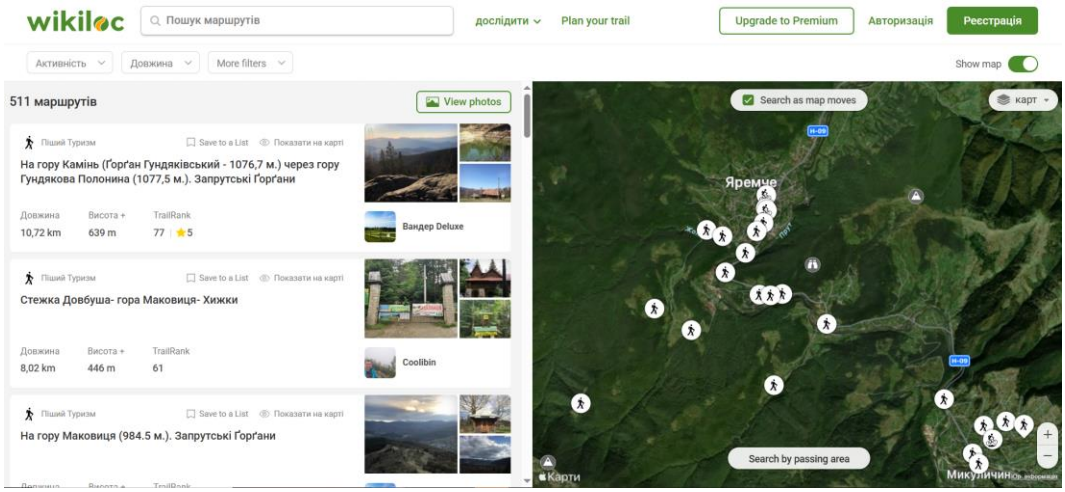

Рисунок 2.3 – Знімок екрану додатку Wikiloc

Перевагами Wikiloc є величезна спільнота користувачів, яка забезпечує різноманітність маршрутів, і функція планування маршрутів, що дозволяє вибирати ключові точки, через які має проходити шлях, а система автоматично створює маршрут із використанням даних спільноти. Wikiloc підтримує імпорт і експорт GPX-файлів, офлайн-навігацію (у преміум-версії) і надає детальну інформацію про відстань, набір висоти та користувацькі відгуки. У Карпатах Wikiloc може бути корисним завдяки великій кількості маршрутів, створених

місцевими туристами. Недоліком є те, що деякі функції, такі як офлайн-карти чи розширений функціонал планування маршрутів, доступні лише в платній версії. Крім того, якість маршрутів залежить від діяльності спільноти, що може призводити до неточностей у менш популярних регіонах Карпат, де дані можуть бути застарілими, неповними чи взагалі відсутніми.

Слід зазначити що існують і інші сервіси та додатки для планування маршрутів в гірській місцевості та навігації, наприклад популярні Osmand та Mapy.cz. Проте їх ключовим аспектом є саме навігація на місцевості, а всі маршрути чи стежки одразу відображені на картах, а не списком з можливістю фільтрації за складністю, набором висоти чи загальної відстанню, що очевидно не підходить туристам з малим досвідом або взагалі без нього.

Висновки до розділу 2

У цьому розділі були описані найбільш відомі сервіси для планування туристичних маршрутів у гірських регіонах, що є в тій чи іншій мірі аналогами рекомендаційної системи, що розробляється в рамках цього проєкту. Було проаналізовано їхні функціональні можливості, зокрема створення маршрутів, оцінку параметрів, підтримку геоданих і користувацький досвід. Кожен із розглянутих продуктів – Komoot, AllTrails і Wikiloc – має свої переваги, такі як інтуїтивний інтерфейс, велика база маршрутів чи активна спільнота користувачів, але також обмеження, пов’язані з платним доступом, неточностями даних у Карпатах чи залежністю даних від суб’єктивної оцінки користувачів. Цей аналіз підкреслює актуальність розроблення спеціалізованої системи для українських Карпат, яка враховує регіональні особливості та сучасні потреби туристів.

3 ФОРМУВАННЯ ВИМОГ ДО СИСТЕМИ

Загальні вимоги:

1) сумісність із веббраузерами: система повинна працювати у сучасних ве-браузерах (Google Chrome, Mozilla Firefox, Safari) і підтримувати їхні останні стабільні версії для забезпечення широкої доступності;

2) продуктивність: система повинна забезпечувати швидке завантаження карт, обробку геоданих і відображення маршрутів із мінімальною затримкою, навіть для маршрутів із великою кількістю точок;

3) масштабованість: система має підтримувати додавання нових функцій, таких як інтеграція з іншими API (наприклад, погодними сервісами) або підтримка додаткових форматів даних, без значних змін у коді;

4) надійність: система повинна бути стійкою до помилок, зокрема при невдалих запитах до зовнішніх API (наприклад, Open-Elevation), із механізмами повторних спроб і обробки виключень.

Інтерфейс користувача:

1) інтуїтивність: інтерфейс має бути простим і зрозумілим для користувачів із різним рівнем досвіду, включаючи туристів-початківців і досвідчених мандрівників, із чіткими елементами управління для створення, перегляду та фільтрації маршрутів;

2) візуалізація: система повинна надавати інтерактивну карту з можливістю вибору шарів і відображенням маршрутів, маркерів початку/кінця.

3.1 Вимоги до функціональних характеристик

Управління маршрутами:

1) створення маршрутів: система повинна дозволяти користувачам створювати маршрути вручну шляхом малювання на карті;

2) імпорт GPX-файлів: система має підтримувати завантаження маршрутів у форматі GPX, автоматично обробляючи координати та обчислюючи

їхні параметри (відстань, набір висоти);

3) збереження маршрутів: система повинна забезпечувати збереження маршрутів на сервері через REST API із можливістю додавання назви, опису, зображень і категорії складності;

4) фільтрація та сортування: система має надавати функції фільтрації маршрутів за параметрами (довжина, набір висоти, складність) і сортування за назвою, довжиною, висотою чи складністю.

Аналіз і оцінка маршрутів:

1) автоматична оцінка складності: система повинна автоматично обчислювати складність маршрутів на основі відстані та набору висоти, отриманих із геоданих, із відображенням результату в read-only полі;

2) обчислення параметрів: система має визначати відстань маршруту і набір висоти із точністю, достатньою для адекватного планування проходження маршруту.

3.2 Вимоги до видів забезпечення

Технічне забезпечення:

1) інтеграція з вебтехнологіями: система має бути побудована на основі сучасних фреймворків та бібліотек для забезпечення інтерактивного інтерфейсу та роботи з картами;

2) оптимізація продуктивності: використання ефективних алгоритмів обробки геоданих і механізмів повторних спроб для запитів до Open-Elevation API для зменшення навантаження на мережу.

Програмне забезпечення:

1) модульність: система має модульну архітектуру з окремими компонентами для полегшення підтримки та розширення функціоналу;

2) обробка помилок: система повинна включати механізми обробки помилок, такі як валідація координат і повторні запити до API у разі збоїв;

3) тестування: усі компоненти системи мають бути протестовані на коректність обробки геоданих, імпорту GPX-файлів і збереження маршрутів на сервері;

Інформаційне забезпечення:

1) документація для розробників: надання документації з описом структури компонентів;

2) керівництво для користувачів: включення інструкцій для кінцевих користувачів із поясненням, як створювати маршрути, імпортувати GPX-файли, фільтрувати та зберігати маршрути;

3) приклади даних: надання тестових GPX-файлів і прикладів маршрутів у Карпатах для демонстрації можливостей системи.

Висновки до розділу 3

Під час роботи над розділом було проведено аналіз рекомендаційної системи туристичних маршрутів в українських Карпатах, що є предметом розроблення, і визначено такі категорії вимог: загальні, функціональні та вимоги до видів забезпечення. Вимоги, зазначені у цьому розділі, спрямовані на створення надійної, масштабованої та зручної системи для планування туристичних маршрутів, яка забезпечує автоматизовану оцінку складності, інтерактивний інтерфейс, створення нових маршрутів, їх фільтрацію. Ці вимоги допоможуть розробникам і користувачам ефективно використовувати систему для створення безпечних і комфортних подорожей у Карпатах.

4 ВИБІР ТЕХНОЛОГІЙ РОЗРОБЛЕННЯ

Розроблення рекомендаційної системи туристичних маршрутів в українських Карпатах, яка є вебдодатком для створення, обробки та оцінки складності маршрутів, потребує технологій, здатних забезпечити інтерактивний інтерфейс, коректну роботу з географічними даними, збереження інформації, адаптивність до різних пристроїв, можливість масштабування та подальшого розвитку. Система повинна підтримувати відображення карт, обчислення параметрів маршрутів, таких як відстань і набір висоти, а також обробку GPX-файлів і запитів до серверу. У цьому розділі розглянуто технології для клієнтської та серверної частин, їхні можливості, переваги та обмеження, з урахуванням потреб проєкту, зокрема специфіки гірських регіонів Карпат, де важливі швидке завантаження і доступність.

4.1 Технології для розроблення клієнтської частини

Клієнтська частина вебдодатка відповідає за інтерфейс користувача, відображення карт і взаємодію з сервером. Для цього потрібен фреймворк, який забезпечує модульність, швидке оновлення даних і сумісність із бібліотеками для геоінформаційних даних.

Технології для створення інтерфейсів почали активно розвиватися у 2000-х роках із появою складних вебдодатків. У 2010-х роках односторінкові додатки (SPA) стали популярними завдяки фреймворкам, які дозволяють створювати динамічні інтерфейси без перезавантаження сторінок. Такі інструменти забезпечують компонентну архітектуру, що спрощує розроблення модульних систем, і підтримують крос-браузерність для роботи на десктопах і мобільних пристроях. Для проєкту важлива інтеграція з картами, API для висот і інструментами для адаптивного дизайну, а також простота освоєння для невеликих командх.

Розглянемо два найпопулярніші фреймворки для розроблення вебдодатків із підтримкою інтерактивних інтерфейсів і геоданих: React і Vue.js. Також коротко

оцінимо бібліотеки для роботи з картами та геоданими, які інтегруються з цими фреймворками.

React – це бібліотека для створення інтерфейсів користувача, розроблена Facebook і вперше випущена у 2013 році. React базується на компонентному підході, де інтерфейс поділяється на незалежні компоненти, які можна повторно використовувати. React використовує віртуальний DOM для оптимізації оновлень сторінки, що забезпечує високу продуктивність навіть для складних додатків [9].

Перевагами React є гнучкість, велика екосистема бібліотек і активна спільнота. React легко інтегрується з бібліотеками для роботи з картами, такими як Leaflet (через react-leaflet) і Mapbox GL JS, а також із бібліотеками для геообчислень, наприклад, Turf.js. React підтримує асинхронні запити через Axios або Fetch, що спрощує взаємодію з API, такими як Open-Elevation. Крім того, React має інструменти для створення адаптивних інтерфейсів, наприклад, Bootstrap або Material-UI, які забезпечують коректне відображення на різних пристроях.

React написаний на JavaScript, що є універсальною мовою для веброзробки. Застосування JSX дозволяє поєднувати HTML-подібний синтаксис із JavaScript-логікою, спрощуючи створення компонентів. Недоліками React є відносно висока складність для початківців через необхідність налаштування інструментів (наприклад, Webpack) і залежність від сторонніх бібліотек для додаткового функціоналу, таких як маршрутизація чи управління станом.

Vue.js – це сучасний фреймворк для створення інтерфейсів, випущений у 2014 році Евеном Ю. Vue.js також базується на компонентному підході, проте має простіший синтаксис у порівнянні з React. Vue.js включає вбудовані інструменти для управління станом і маршрутизації, що зменшує потребу в сторонніх бібліотеках [10].

Перевагами Vue.js є легкість освоєння, компактність і вбудована реактивність, що спрощує оновлення даних у реальному часі. Vue.js інтегрується з бібліотеками для карт, такими як Leaflet або OpenLayers, і підтримує обчислення геоінформаційних даних через Turf.js. Однак екосистема Vue.js менша, ніж у React,

що може обмежувати вибір готових рішень для специфічних задач, таких як складні географічні обчислення.

4.2 Технології для розроблення серверної частини

Серверна частина системи відповідає за збереження маршрутів, обробку файлів, фільтрацію даних і надання API. Для цього потрібні технології, що забезпечують REST API, базу даних і обробку мультимедійного контенту. Розглянуто два підходи: Node.js із Express і MongoDB та Django із PostgreSQL.

Node.js із Express і MongoDB – серверне середовище та фреймворк, випущені у 2009 і 2010 роках, разом із базою даних MongoDB, представленою у 2009 році. Node.js працює на JavaScript, забезпечуючи асинхронну обробку запитів, що підходить для API із високим навантаженням [11]. Express спрощує створення REST API для операцій із даними, такими як збереження маршрутів чи фільтрація за параметрами [12]. MongoDB – це NoSQL-база, яка зберігає дані як JSON-подібні документи, що зручно для GeoJSON-структур із координатами маршрутів [13]. Для моделювання даних використовується Mongoose, що забезпечує схеми та валідацію (наприклад, обов’язкові поля для назви чи зображень) [14]. Через слабку типізацію JavaScript для обробки файлів необхідно застосовувати додаткові бібліотеки, наприклад Multer, випущений у 2014 році, який зберігає зображення локально з унікальними іменами [15]. dotenv, випущений у 2013 році, керує змінними середовища, такими як підключення до бази.

Переваги цього підходу включають швидкість розроблення, уніфікацію з JavaScript-клієнтом і гнучкість MongoDB для роботи з геоданими. Обмеженнями є слабка типізація JavaScript, що може ускладнити масштабування та призводить до необхідності використання додаткових бібліотек, і ризик неконсистентності даних у MongoDB при великих обсягах.

Django із PostgreSQL – фреймворк, випущений у 2005 році, на мові Python, разом із реляційною базою даних PostgreSQL, представленою у 1986 році. Django пропонує ORM, маршрутизацію, адмін-панель і вбудовані механізми безпеки [16].

REST API створюється через Django REST Framework, випущений у 2011 році, який підтримує операції з маршрутами та фільтрацію. PostgreSQL із розширенням PostGIS дозволяє зберігати геодані (наприклад, координати як тип GEOMETRY) і виконувати просторові запити, такі як пошук маршрутів у регіоні [17]. Django має власну систему обробки файлів, подібну до Multer, для завантаження зображень.

Переваги включають строгі типи Python, безпеку Django (захист від SQL-ін'єкцій) і консистентність PostgreSQL. Обмеженнями є більша складність для невеликих проєктів і менша швидкість асинхронної обробки порівняно з Node.js.

4.3 Бібліотеки для роботи з картами та геоданими

Для відображення карт, побудови маршрутів і обчислень розглянуто чотири інструменти: Leaflet, Mapbox GL JS, Turf.js і Open-Elevation API.

Leaflet – бібліотека для створення інтерактивних карт, випущена у 2011 році. Вона підтримує шари OpenStreetMap і OpenTopoMap, які відображають топографію, важливу для Карпат. Leaflet дозволяє імпортувати GPX-файли та малювати маршрути через плагін Leaflet Draw. Її переваги включають простоту інтеграції, низькі вимоги до ресурсів і безкоштовність, що забезпечує швидке завантаження на пристроях із слабким зв'язком. Обмеженням є слабка підтримка 3D-візуалізації, що не є критичним для 2D-карт у туристичних додатках [18].

Mapbox GL JS – бібліотека, випущена у 2015 році, що використовує векторні тайли та WebGL для створення карт із 3D-рендерингом. Вона підтримує кастомізацію стилів і обробку великих наборів геоданих. Переваги включають високу якість відображення і можливість інтеграції з API для обчислень. Основним недоліком є платний доступ до API після перевищення ліміту запитів (наприклад, 200 000 на місяць для тайлів), а також вищі вимоги до ресурсів, що може вплинути на продуктивність у віддалених регіонах [19].

Turf.js – бібліотека для геообчислень, представлена у 2013 році. Вона обробляє GeoJSON-дані, обчислюючи відстані, площі та параметри маршрутів. Turf.js сумісна з Leaflet, дозволяючи визначати довжину маршруту за формулою

гаверсинусів [20]. Переваги включають модульність, безкоштовність і легкість інтеграції. Основним недоліком є недостатній функціонал для складних топографічних обчислень, таких як аналіз рельєфу.

Open-Elevation API – інструмент для отримання висот за координатами, що використовує дані SRTM [21]. Він підтримує пакетні запити для обробки кількох точок. Переваги включають простоту використання та відкритий код. Недоліком є ліміт на кількість запитів і можливі неточності в гірських регіонах, що потребує механізмів повторних спроб.

Висновки до розділу 4

У процесі виконання розділу було проаналізовано різноманітні технологій розроблення для створення рекомендаційної системи туристичних маршрутів. Для клієнтської частини розглянуто фреймворки React і Vue.js, бібліотеки для роботи з картами (Leaflet, Mapbox GL JS), геообчислень (Turf.js, Open-Elevation API), а також інструменти для стилізації (Bootstrap) і запитів (Axios). Для серверної частини оцінено можливості Node.js із Express і MongoDB, а також Django із PostgreSQL. У результаті для реалізації проєкту обрано технології React для фронтенду, бібліотеки Leaflet, Turf.js, Open-Elevation API для роботи з геоінформаційними даними, а також Node.js із Express, MongoDB, Mongoose, Multer і dotenv для бекенду. Ці технології забезпечують інтерактивність, зрозумілість та ефективну обробку геоінформаційних даних.

5 РОЗРОБЛЕННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

У цьому розділі описано розроблення рекомендаційної системи туристичних маршрутів в українських Карпатах, яка є вебдодатком для створення, аналізу, оцінки та збереження туристичних маршрутів. Система дозволяє туристам планувати гірські походи, обираючи серед існуючих маршрутів, чи створюючи нові маршрути безпосередньо на карті або імпортуючи GPX-файлами, при цьому система здатна самостійно обчислювати географічні параметри отриманих маршрутів, такі як відстань і загальний набір висоти, та на їх основі визначає складність маршрутів. Розглянуто загальну структуру системи, її функціональну модель, організацію даних, методи їхньої обробки та передачі, архітектуру програмного забезпечення, та надано керівництво користувача. Посилання на репозиторій код проєкту наведено в додатку А.

5.1 Архітектура інформаційної системи

Реалізована рекомендаційна система туристичних маршрутів в українських Карпатах базується на клієнт-серверній архітектурі, що складається з трьох основних рівнів: презентаційного, логічного та рівня даних. Така архітектура забезпечує чіткий розподіл обов'язків між компонентами системи та дозволяє незалежно розвивати кожен рівень. Багаторівнева структура також сприяє покращенню безпеки системи через ізоляцію бізнес-логіки від прямого доступу користувачів.

Презентаційний рівень реалізовано як одно-сторінковий вебдодаток на основі вебфреймворку React. Цей рівень відповідає за забезпечення інтуїтивно зрозумілого та адаптивного вебінтерфейсу для взаємодії з системою, візуалізацію мапи через інтеграцію з бібліотекою Leaflet для відображення інтерактивних карт з підтримкою різних картографічних шарів, обробку користувацьких дій при створенні нових маршрутів чи редагуванні, фільтрації та сортуванні існуючих, а

також клієнтську валідацію для перевірки коректності введених даних перед відправкою на сервер.

Серверна частина реалізована на платформі Node.js з використанням фреймворку Express.js і формує логічний рівень системи. Цей рівень включає REST API для обробки HTTP-запитів від клієнтської частини, бізнес-логіку з алгоритмами обчислення відстані маршруту з застосуванням бібліотеки turf.js, обчислення загального набору висоти на маршруті через інтеграцію з Open-Elevation API та автоматичну оцінку складності маршрутів. Також на цьому рівні здійснюється обробка файлів, включаючи завантаження та обробку GPX-файлів і збереження зображень маршрутів, а також управління сесіями для обробки запитів користувачів та забезпечення безпеки даних.

Для зберігання даних використовується документо-орієнтована база даних MongoDB, яка формує рівень даних системи. MongoDB забезпечує гнучку схему даних для зберігання маршрутів з кількома різноманітними атрибутами без жорстких обмежень схеми, нативну підтримку GeoJSON-форматів для збереження координат маршрутів, масштабованість з можливістю горизонтального масштабування при зростанні обсягу даних та високу продуктивність завдяки оптимізованим запитам для фільтрації та сортування маршрутів.

Система інтегрується з кількома зовнішніми сервісами, зокрема з Open-Elevation API для отримання даних про висоти точок маршруту, OpenStreetMap та OpenTopoMap як джерелами картографічних даних для відображення різних шарів карт, а також з бібліотекою Leaflet для візуалізації карт та інтерактивної роботи з геоданими. Архітектура системи забезпечує модульність, що дозволяє легко додавати новий функціонал, наприклад інтеграція з сервісами прогнозу погоди, системами бронювання місць в готелях або соціальними мережами. Використання стандартних протоколів HTTP/HTTPS та REST, а також форматів даних JSON і GeoJSON забезпечує сумісність з широким спектром сторонніх сервісів та можливість їх подальшої інтеграції. На рисунку 5.1 наведено загальну архітектуру системи.

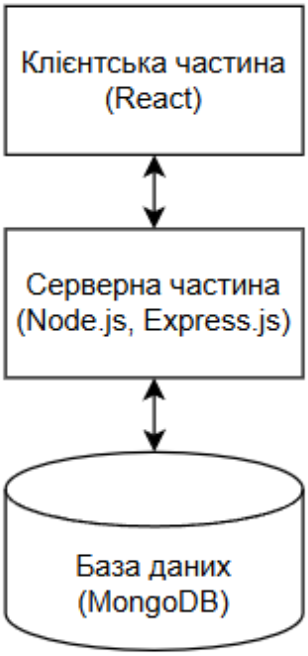

Рисунок 5.1 – Загальна архітектура системи

5.2 Функціональна модель системи

Функціональна модель рекомендаційної системи туристичних маршрутів описує взаємодію між різними акторами системи та основні бізнес-процеси. Модель побудована на основі аналізу потреб користувачів та специфіки туристичної діяльності в українських Карпатах.

Основним актором системи є користувач, який представляє туриста, що планує маршрути в Карпатах. Користувачі можуть мати різний рівень досвіду: від початківців, які потребують простих та безпечних маршрутів, до досвідчених туристів, які шукають триваліші та складніші маршрути. Додатковим актором є сама система, яка автоматично виконує обчислення та аналіз геоданих без втручання користувача.

Варіант використання "Створення маршруту" має на меті створити новий туристичний маршрут для планування походу. При цьому користувач має доступ до вебінтерфейсу системи та обирає спосіб створення маршруту, який може включати малювання на карті або імпорт GPX-файлу. При малюванні на карті

користувач послідовно встановлює точки маршруту, починаючи зі стартової позиції, а при імпорті GPX користувач завантажує файл через інтерфейс системи. Система автоматично обчислює параметри маршруту, включаючи відстань, набір висоти та складність, після чого користувач може переглянути створений маршрут на карті. Результатом є створення маршруту з відображенням його в інтерфейсі разом з розрахованими параметрами.

Варіант використання "Збереження маршруту" спрямований на збереження створеного маршруту у базі даних для подальшого використання. Передумовою є створений раніше маршрут, після чого користувач заповнює форму з інформацією про маршрут, включаючи назву та опис, завантажує зображення маршруту до п'яти файлів, а система валідує введені дані та зберігає маршрут у базі даних з усіма атрибутами. Результатом є додавання маршруту до загального каталогу системи.

Варіант використання "Перегляд списку маршрутів" дозволяє користувачеві ознайомитися з доступними маршрутами в системі за умови наявності збережених маршрутів. Користувач відкриває сторінку зі списком маршрутів, система відображає картки маршрутів з основною інформацією, включаючи назву, довжину, складність та зображення, а користувач може переглянути опис обраного маршруту. Результатом є отримання користувачем можливості огляду доступних маршрутів.

Варіант використання "Фільтрація та сортування маршрутів" має на меті знайти та відобразити маршрути, що відповідають критеріям користувача. За наявності збережених маршрутів користувач встановлює фільтри за параметрами відстані, набору висоти та рівня складності, або обирає критерій сортування за назвою, відстанню, набором висоти або складністю, а система застосовує фільтри чи сортування та відображає відфільтровані результати. Результатом є отримання користувачем списку маршрутів, що відповідають його потребам та можливостям.

Система автоматично виконує кілька важливих функцій. Варіант використання "Обчислення параметрів маршруту" спрямований на автоматичний розрахунок характеристик маршруту та спрацьовує при створенні нового маршруту. Система отримує координати точок маршруту, обчислює загальну

відстань за формулою гаверсинусів, запитує дані про висоти точок через Open-Elevation API, розраховує загальний набір висоти маршруту та застосовує алгоритм оцінки складності. Результатом є доповнення маршруту розрахованими параметрами.

Варіант використання "Оцінка складності маршруту" має на меті автоматично визначити рівень складності маршруту та спрацьовує після обчислення параметрів маршруту. Система використовує спеціальну формулу для розрахунку складності, що враховує відстань та набір висоти, класифікує результат за рівнями легкого, помірного, складного, дуже складного та екстримального, і присвоює маршруту відповідний рівень складності.

Система підтримує два основні бізнес-процеси. Процес створення та збереження маршруту включає створення треку маршруту, автоматичне обчислення географічних параметрів, валідацію даних та збереження у базі даних. Процес пошуку та відбору маршруту включає перегляд каталогу, застосування фільтрів чи сортування результатів та вибір підходящого маршруту. Ця функціональна модель забезпечує повне покриття потреб користувачів системи та створює основу для подальшого розвитку функціоналу, такого як рейтингова система, коментарі користувачів або інтеграція з GPS-навігацією. Діаграму прецедентів наведено на кресленику ІС12.060БАК.005 Д1. В таблиці 5.1 наведено опис варіантів використання системи.

Таблиця 5.1 – Опис варіантів використання

Актор	Варіант використання	Опис
Користувач	Перегляд маршрутів	Користувач може переглядати уже існуючі маршрути
Користувач	Фільтрація маршрутів	Користувач може фільтрувати маршрути за параметрами: відстань, загальний набір висоти та складність маршруту

Актор	Варіант використання	Опис
Користувач	Створення треку маршруту	Користувач може намалювати вручну маршрут на карті чи імпортувати з зовнішніх джерел як GPX-файлу.
Користувач	Додавання маршруту до бази даних	Користувач після створення чи імпорту геоданих про маршрут, може додати його до списку маршрутів, додавши зображення та його опис.
Система	Визначення відстані та загального набору висоти маршруту	Система запитує значення висот точок для обчислення загального набору висоти, а також обчислює відстань між ними
Система	Оцінка складності маршруту	Система оцінює складність маршруту на основі даних про відстань та загальний набір висоти на маршруті

5.3 Розроблення клієнтської частини

Клієнтська частина рекомендаційної системи туристичних маршрутів в українських Карпатах реалізована як односторінковий вебзастосунок на основі бібліотеки React. Основною метою клієнтської частини є надання користувачеві інтуїтивного та функціонального інтерфейсу для створення, перегляду, фільтрації та управління туристичними маршрутами.

5.3.1 Архітектура клієнтської частини

Клієнтська частина побудована на компонентній архітектурі React, що забезпечує модульність та можливість повторного використання компонентів. Застосунок складається з головного компонента App, який містить всю бізнес-

логіку та керує глобальним станом системи через React hooks, зокрема useState та useEffect. Така архітектура дозволяє ефективно управляти даними та забезпечує реактивність інтерфейсу при зміні стану.

Система використовує декларативний підхід до управління станом, де кожна зміна в даних автоматично відображається в інтерфейсі користувача. Це особливо важливо для роботи з картами та динамічними списками маршрутів, де потрібно миттєво реагувати на дії користувача. Архітектура базується на принципах компонентного підходу, де кожен функціональний блок винесений в окремий компонент, декларативного управління станом через React hooks, односторінкового застосування без перезавантаження сторінки та реактивного інтерфейсу з автоматичним оновленням UI. Діаграму компонентів клієнтської частини системи наведено в кресленику ІС12.060БАК.005 Д2.

5.3.2 Структура основних компонентів

Компонент App є центральним елементом застосунку та виконує роль координатора всієї системи. Він відповідає за управління глобальним станом застосунку, координацію взаємодії між дочірніми компонентами, обробку HTTP-запитів до серверної частини та управління модальними вікнами для створення та редагування маршрутів. Головний компонент керує численними станами застосунку. Зокрема, він підтримує масив routes для зберігання списку всіх маршрутів, отриманих з сервера. Фільтри lengthFilter, elevationFilter та difficultyFilter дозволяють користувачам налаштовувати критерії пошуку маршрутів за максимальною довжиною, максимальним набором висоти та рівнем складності відповідно. Стан selectedRoute зберігає поточно вибраний для відображення на карті маршрут, а sortBy визначає критерій сортування списку маршрутів. Компонент також управляє станами модальних вікон showAddModal та showEditModal, які контролюють відображення діалогових вікон для додавання нових та редагування існуючих маршрутів. Об'єкти newRoute та editRoute містять

дані, що вводяться користувачем при створенні або модифікації маршрутів, включаючи назву, опис, координати, зображення та інші атрибути.

MapView є основним компонентом для відображення картографічної інформації та взаємодії з маршрутами. Компонент інтегрується з бібліотекою React Leaflet для створення інтерактивних карт та забезпечує візуалізацію маршрутів у зручному для користувача вигляді. Компонент підтримує відображення інтерактивної карти з можливістю вибору між різними картографічними шарами. Користувач може переключатися між стандартною картою OpenStreetMap, яка надає загальну географічну інформацію, та топографічною картою OpenTopoMap, що особливо важливо для планування гірських маршрутів через детальне відображення рельєфу місцевості. MapView візуалізує маршрути у вигляді полілінії синього кольору, що з'єднує всі точки треку. На початку та в кінці маршруту встановлюються маркери з вспливаючими підказками, які інформують користувача про стартову та фінішну точки обраного маршруту. Компонент автоматично налаштовує масштаб карти для зручного відображення всього маршруту.

MapDrawing забезпечує функціональність створення маршрутів безпосередньо на карті через інтеграцію з бібліотекою Leaflet Draw. Цей компонент є ключовим для інтерактивного планування маршрутів користувачами системи. Він ініціалізує контролер Leaflet Draw з налаштуваннями, що дозволяють малювати тільки полілінії, оскільки інші геометричні фігури не релевантні для представлення туристичних маршрутів. При активації режиму малювання користувач може послідовно натискати на певні точки на карті, які автоматично з'єднуються лініями, створюючи трек маршруту. Після завершення малювання маршруту компонент автоматично витягує координати всіх точок з створеного layer та формує масив географічних координат. Ці координати відразу відправляються на серверну частину через спеціальний API endpoint для обчислення географічних параметрів маршруту. Сервер повертає розраховані значення загальної відстані, набору висоти та автоматично визначеної складності маршруту. MapDrawing підтримує два режими роботи: створення нового маршруту та редагування існуючого. У режимі

редагування компонент дозволяє модифікувати координати вже збереженого маршруту, автоматично перераховуючи всі параметри при внесенні змін.

MapBounds є допоміжним компонентом, що автоматично налаштовує масштаб та здійснює центрування карти для зручного відображення вибраного маршруту. Компонент аналізує координати маршруту та викликає метод fitBounds карти Leaflet, що забезпечує відображення всього треку маршруту в межах видимої області карти з зручним масштабом.

5.3.3 Система управління станом та взаємодія з API

Застосунок використовує локальний стан React компонентів для управління даними. Всі критичні дані зберігаються в головному компоненті App та передаються до дочірніх компонентів через властивості props. Такий підхід забезпечує централізоване управління станом та спрощує відстеження змін даних. Реактивні оновлення забезпечуються механізмом React, де зміни в стані автоматично викликають повторний рендеринг відповідних компонентів. Це означає, що коли користувач змінює фільтри або додає новий маршрут, інтерфейс миттєво оновлюється, не потребуючи додаткового коду з боку розробника.

Всі операції з серверною частиною виконуються асинхронно з використанням бібліотеки Axios. Функція fetchRoutes відповідає за отримання списку маршрутів з сервера з урахуванням поточних налаштувань фільтрів та сортування. Ця функція автоматично викликається при зміні будь-якого з фільтрів завдяки useEffect hook.

Функція calculateRouteParams здійснює POST-запит до серверного endpoint для обчислення параметрів маршруту. Вона отримує масив координат та повертає об'єкт з розрахованими значеннями відстані, загального набору висоти та складності маршруту. Функція parseGpxFile обробляє завантажені користувачем GPX-файли, відправляючи їх на сервер для парсингу та отримуючи готові до використання координати та параметри маршруту.

5.3.4 Структура інтерфейсу користувача

Інтерфейс побудований з використанням бібліотеки React Bootstrap та має чітку трьохколонкову структуру. У верхній частині розташована навігаційна панель з логотипом "Карпатські маршрути" та основними розділами застосунку. Основна робоча область поділена на три функціональні секції, що забезпечують ергономічну організацію інформації.

Ліва колонка містить панель фільтрів з елементами управління для налаштування критеріїв пошуку маршрутів. Центральна колонка відображає список знайдених маршрутів у вигляді карток з основною інформацією та кнопками управління. Права колонка займає область карти для візуалізації вибраних маршрутів.

Панель фільтрів надає користувачам гнучкі можливості для налаштування критеріїв пошуку маршрутів. Фільтр за максимальною довжиною реалізований через range input з діапазоном від 0 до 100 кілометрів, що покриває практично всі можливі туристичні маршрути в Карпатах. Поточне значення фільтра динамічно відображається поряд з повзунком, що покращує зручність використання.

Аналогічно працює фільтр за максимальним набором висоти з діапазоном від 0 до 3500 метрів, що враховує специфіку українських Карпат з їхніми найвищими вершинами. Фільтр за складністю представлений dropdown меню з варіантами: Усі, Легка, Помірна, Складна, Дуже складна та Екстремальна, що відповідає автоматично розрахованій системою класифікації складності.

Функція сортування дозволяє упорядковувати маршрути за різними критеріями через dropdown селектор. Користувач може сортувати маршрути за назвою в алфавітному порядку, за довжиною від найкоротших до найдовших, за набором висоти від найменшого до найбільшого, або за рівнем складності від найлегших до найскладніших.

Кожен маршрут у списку представлений у вигляді картки Bootstrap Card з горизонтальним розташуванням контенту. Ліва частина картки містить зображення маршруту, причому для маршрутів з кількома фотографіями використовується

React Bootstrap Carousel, що дозволяє переглядати всі доступні зображення. Зображення автоматично масштабуються до фіксованої висоти 150 пікселів зі збереженням пропорцій.

Права частина картки містить текстову інформацію: назву маршруту як заголовок, основні параметри (відстань в кілометрах, набір висоти в метрах) та рівень складності з кольоровим кодуванням. Рівень складності відображається з використанням спеціальних CSS класів, що забезпечують візуальне розрізнення різних категорій складності.

Під інформаційним блоком розташовані кнопки управління маршрутом. Головна кнопка "Показати на карті" дозволяє візуалізувати маршрут в правій частині інтерфейсу. Додатково присутні кнопки "Редагувати" та "Видалити" для управління маршрутами. При наведенні курсора на картку відображається tooltip з повним описом маршруту.

5.3.5 Модальні вікна для управління маршрутами

В даній програмній реалізації застосовуються два модальних вікна для створення нового маршруту та редагування вже існуючого маршруту.

Модальне вікно додавання нового маршруту реалізовано з використанням React Bootstrap Modal компонента та забезпечує комплексний інтерфейс для введення всієї необхідної інформації. Вікно містить поля для введення назви та опису маршруту, які є обов'язковими для заповнення.

Ключовою особливістю є інтегрована мініатюрна карта висотою 300 пікселів, на якій користувач може безпосередньо намалювати маршрут. Карта підтримує ті ж картографічні шари, що і основна карта застосунку. Після завершення малювання система автоматично обчислює параметри маршруту та заповнює відповідні поля, які позначені як read-only для запобігання ручному втручанню.

Альтернативним способом створення маршруту є завантаження GPX-файлу через спеціальне поле file input. При виборі файлу система відправляє його на сервер для обробки, витягує координати треку та автоматично заповнює всі поля

маршруту. Цей підхід особливо зручний для користувачів, які вже мають готові треки з GPS-пристроїв або інших туристичних застосунків.

Вікно також містить поле для завантаження зображень з підтримкою множинного вибору файлів. Система приймає до п'яти зображень на маршрут, що дозволяє користувачам документувати ключові точки або краєвиди маршруту. Всі зображення автоматично валідуються на відповідність дозволеним форматам.

Модальне вікно редагування надає можливість модифікувати існуючі маршрути з максимальним збереженням наявних даних. При відкритті вікна всі поля автоматично заповнюються поточними значеннями маршруту, що дозволяє користувачеві бачити наявну інформацію та вносити тільки необхідні зміни.

Особливістю редагування є відображення поточних зображень маршруту у вигляді мініатюр розміром 60x60 пікселів. Користувач може переглянути всі наявні фотографії та прийняти рішення щодо необхідності їх збереження або заміни. Чекбокс "Зберегти існуючі зображення" дозволяє контролювати, чи будуть збережені поточні зображення при додаванні нових.

Вікно містить ту ж інтегровану карту для малювання нового треку маршруту, що дозволяє користувачеві коригувати його при необхідності. Також підтримується завантаження нового GPX-файлу для повної заміни координат маршруту. При внесенні змін до треку маршруту система автоматично перераховує всі його параметри.

5.3.6 Обробка файлів та мультимедіа

Система підтримує імпорт туристичних маршрутів у стандартному форматі GPX, що є найпоширенішим форматом для обміну GPS-треками між різними пристроями та застосунками. При завантаженні GPX-файлу клієнтська частина створює FormData об'єкт та відправляє файл на серверний endpoint для обробки. Функція handleGpxUpload обробляє події завантаження файлів для створення нових маршрутів, а handleEditGpxUpload виконує аналогічну функцію для режиму редагування. Обидві функції включають комплексну обробку помилок з

інформативними повідомленнями для користувача у випадку проблем з форматом файлу або серверною обробкою. Після успішної обробки GPX-файлу сервер повертає масив координат, розраховану відстань, набір висоти та рівень складності маршруту. Ці дані автоматично заповнюють відповідні поля інтерфейсу, що значно спрощує процес додавання маршрутів для користувачів, які вже мають готові GPS-треки.

Система забезпечує гнучке управління зображеннями маршрутів з підтримкою завантаження множинних файлів. File input елементи налаштовані з атрибутом multiple, що дозволяє користувачам обирати кілька зображень одночасно. Перевірка типів файлів здійснюється через атрибут accept з значенням "image/*", що обмежує вибір тільки файлами зображень. При завантаженні зображень система використовує метод Array.from для перетворення FileList об'єкта у звичайний масив, який потім зберігається в стані компонента. Для відправки на сервер зображення додаються до FormData об'єкта за допомогою методу forEach, що забезпечує коректну передачу файлів через HTTP.

У режимі редагування система надає додаткові можливості управління зображеннями. Поточні зображення відображаються у вигляді мініатюр, що дозволяє користувачеві оцінити наявний контент. Чекбокс keepExistingImages дає можливість контролювати збереження існуючих зображень при додаванні нових, що запобігає випадковій втраті цінного візуального контенту.

5.3.7 Валідація даних та обробка помилок

Система включає комплексну перевірку даних на правильність на клієнтській стороні для забезпечення цілісності інформації перед відправкою на сервер. Функція handleAddRoute перевіряє заповнення всіх обов'язкових полей: назви, довжини, набору висоти, складності, опису маршруту, наявності координат треку та хоча б одного зображення. Валідація виконується перед кожною спробою збереження маршруту, та у випадку якщо будь-яке з обов’язкових полів незаповнене користувач отримує зрозуміле повідомлення з переліком необхідних

дій. Такий підхід запобігає непотрібним запитам до сервера та покращує користувацький досвід через миттєвий зворотний зв'язок.

Всі асинхронні операції в застосунку загорнуті в конструкції try-catch для перехоплення та належної обробки помилок. Функції для роботи з API включають детальне логування помилок у консоль розробника для діагностики проблем та показ зрозумілих повідомлень користувачеві через alert діалоги. При помилках обчислення параметрів маршруту система встановлює значення за замовчуванням (нульова довжина та набір висоти, легка складність) та інформує користувача про проблему з пропозицією повторити операцію. Такий підхід забезпечує стабільність роботи застосунку навіть при тимчасових проблемах з серверною частиною або зовнішніми API.

5.3.8 Адаптивний дизайн та оптимізація

Інтерфейс розроблений з використанням Bootstrap Grid System для забезпечення коректного відображення на пристроях з різними розмірами екранів. На десктопних комп'ютерах застосунок використовує трьох-колонковий макет з фільтрами, списком маршрутів та картою, розташованими горизонтально для максимальної ефективності використання простору екрана.

На планшетах система адаптивно змінює ширину колонок, зберігаючи всю функціональність, але оптимізуючи розташування елементів для сенсорного управління. На мобільних пристроях макет трансформується у вертикальне розташування блоків, що забезпечує зручність використання на невеликих екранах.

Система використовує ефективні методи відображення для забезпечення високої продуктивності. Списки маршрутів формуються з унікальними ключами React для оптимізації процесу оновлення DOM. Умовне відображення застосовується для показу компонентів тільки за необхідності, що зменшує навантаження на браузер.

UseEffect hook налаштований з масивом залежностей для автоматичного оновлення списку маршрутів тільки при зміні релевантних параметрів фільтрації

або сортування. Це запобігає непотрібним запитам до сервера та забезпечує позитивний користувацький досвід.

Карти Leaflet налаштовані з обмеженнями мінімального та максимального масштабу для запобігання надмірному навантаженню при завантаженні тайлів. Зображення в картках маршрутів автоматично оптимізуються за розміром з використанням CSS властивостей object-fit для збереження якості при мінімальному використанні пам'яті браузера.

5.4 Розроблення серверної частини

Серверна частина рекомендаційної системи туристичних маршрутів в українських Карпатах реалізована на платформі Node.js з використанням вебфреймворку Express.js. Архітектура серверної частини побудована за принципами модульності, що забезпечує легкість підтримки, масштабування та тестування системи.

5.4.1 Архітектурна організація серверної частини

Серверна частина організована за багаторівневою архітектурою, де кожен рівень відповідає за специфічні аспекти обробки даних та бізнес-логіки. Головний файл server.js виконує роль точки входу та координує роботу всіх інших компонентів системи, включаючи налаштування мідлверів, підключення до бази даних та маршрутизацію запитів.

Основна бізнес-логіка винесена в окремі сервісні модулі, що дозволяє тримати код організованим, з можливістю підтримки, оновлення та додавання нових функцій, не вносячи зміни у весь код. Така структура також сприяє принципу єдиної відповідальності, де кожен модуль займається виключно своєю функціональною областю. Діаграму компонентів серверної частини наведено на кресленику ІС12.060БАК.005 Д3.

5.4.2 Організація REST API

REST API системи побудований навколо ресурсу "маршрути" та надає повний спектр CRUD операцій для управління туристичними маршрутами. Основний маршрутизатор routes.js містить всі endpoint для роботи з маршрутами та забезпечує стандартизований інтерфейс для взаємодії з клієнтською частиною.

Endpoint GET /api/routes забезпечує отримання списку маршрутів з підтримкою гнучкої системи фільтрації та сортування. Користувачі можуть фільтрувати маршрути за максимальною довжиною, максимальним набором висоти та рівнем складності. Система сортування підтримує упорядкування за назвою, довжиною, набором висоти та складністю маршруту. Реалізація фільтрації використовує MongoDB query операції, що забезпечує ефективну обробку запитів навіть при великому обсязі даних.

Endpoint POST /api/routes призначений для створення нових маршрутів та реалізує складну логіку обробки багатокомпонентних даних, включаючи зображення та метадані маршруту. При створенні маршруту система автоматично обчислює всі необхідні параметри, включаючи відстань, набір висоти та рівень складності, використовуючи спеціалізовані сервісні модулі.

Endpoint PUT /api/routes/:id забезпечує функціональність оновлення існуючих маршрутів з можливістю часткового оновлення полів. Особливістю цього endpoint є інтелектуальне управління зображеннями, де користувач може вибирати між заміною всіх зображень або додаванням нових до уже існуючих.

5.4.3 Сервісна архітектура

Бізнес-логіка системи організована в спеціалізовані сервісні модулі, кожен з яких відповідає за конкретну функціональну область.

Сервіс elevationService відповідає за отримання даних про висоти точок маршруту через інтеграцію з Open-Elevation API. Модуль реалізує надійну систему повторних спроб при невдалих запитах, валідацію координат та обробку помилок

мережі. Для кожної пари координт здійснюється до трьох спроб отримання даних з затримкою між спробами, що значно підвищує надійність системи в умовах нестабільного з'єднання чи помилок у роботі з зовнішнім API.

Функція calculateElevationGain обчислює загальний набір висоти маршруту, аналізуючи різниці висот між сусідніми точками та підсумовуючи лише позитивні значення, що відповідає фізичному визначенню загального набору висоти під час походу.

Сервіс routeCalculationService об'єднує різні аспекти обчислення параметрів маршруту в єдиний інтерфейс. Модуль використовує бібліотеку Turf.js для точного обчислення відстаней між точками маршруту за формулою гаверсинусів, що враховує сферичну геометрію Землі та забезпечує високу точність розрахунків для географічних координат.

Інтеграція з elevationService дозволяє отримувати дані про висоти та обчислювати загальний набір висоти, після чого система автоматично визначає рівень складності маршруту, використовуючи спеціалізований алгоритм оцінки.

Сервіс gpxService забезпечує обробку GPX-файлів, що є стандартним форматом для обміну GPS-даними між різними застосунками. Модуль використовує бібліотеки toGeoJSON та xmldom для парсингу XML-структури GPX-файлів та конвертації їх у зручний для обробки GeoJSON-формат.

Особливістю реалізації є інтелектуальна обробка даних про висоти: якщо GPX-файл вже містить інформацію про висоти точок, система використовує ці дані безпосередньо, що значно пришвидшує процес обробки. У випадку відсутності даних про висоти система автоматично запитує їх значення через Open-Elevation API.

5.4.4 Визначення складності маршруту

Модуль difficultyCalculator реалізує складний алгоритм оцінки складності маршрутів, який враховує не лише базові параметри відстані та набору висоти, але й їх співвідношення та специфічні характеристики гірської місцевості.

Формула розрахунку включає базовий компонент, що враховує відстань з ваговим коефіцієнтом 0.5, компонент набору висоти з діленням на базовий фактор 50, бонус за крутість маршруту, що обчислюється як співвідношення набору висоти до відстані, та фактор для високогірних маршрутів, що використовує логарифмічну шкалу для врахування експоненційного зростання складності на великих висотах.

Система класифікації включає п'ять рівнів складності: легкий (0-15 балів), помірний (15-30 балів), складний (30-50 балів), дуже складний (50-75 балів) та екстремальний (75+ балів). Така детальна градація дозволяє туристам точніше оцінювати свої можливості та обирати відповідні маршрути.

5.4.5 Модель даних та валідація

Модель Route.js визначає структуру даних маршрутів в MongoDB з використанням Mongoose ODM. Модель включає строгу типізацію всіх полів, валідацію вхідних даних та автоматичне відстеження часу створення та останнього оновлення записів.

Особливу увагу приділено валідації геоданих: поле path має валідатор, що перевіряє наявність мінімум двох точок для формування маршруту, а поле images включає валідацію на наявність принаймні одного зображення та обмеження максимальної кількості до десяти зображень.

Схема також використовує Mongoose middleware для автоматичного оновлення поля updatedAt при кожному збереженні документа, що забезпечує точне відстеження історії змін маршрутів.

Система використовує бібліотеку Multer для обробки завантаження файлів з клієнтської частини. Конфігурація включає налаштування дискового сховища з автоматичною генерацією унікальних імен файлів на основі timestamp, що запобігає конфліктам імен та забезпечує організовану структуру файлової системи.

Для GPX-файлів реалізована додаткова валідація типу файлу, що перевіряє як MIME-type, так і розширення файлу. Після обробки GPX-файли автоматично

видаляються з тимчасового сховища, що запобігає накопиченню непотрібних файлів на сервері.

Зображення маршрутів зберігаються в папці uploads з налаштованим статичним доступом через Express.js, що дозволяє клієнтській частині безпосередньо отримувати зображення за URL-адресами.

Серверна частина включає комплексну систему обробки помилок на всіх рівнях архітектури. Кожен API endpoint містить try-catch блоки для перехоплення та обробки виключень, а також валідацію вхідних даних для запобігання некоректним запитам.

Система використовує CORS middleware з налаштуванням для конкретного origin'у клієнтської частини, що забезпечує контрольований доступ до API. Всі помилки логуються в консоль сервера з детальною інформацією для полегшення діагностики проблем.

Валідація даних відбувається на кількох рівнях: на рівні HTTP-запитів перевіряється наявність обов'язкових полів, на рівні Mongoose схеми здійснюється типізована валідація, а на рівні бізнес-логіки перевіряється коректність географічних координат та інших спеціалізованих даних.

5.4.6 Оптимізація та продуктивність

Серверна частина оптимізована для ефективної роботи з географічними даними та великими обсягами запитів. MongoDB використовується з індексацією полів, що часто використовуються для фільтрації та сортування, що значно пришвидшує виконання запитів до бази даних та забезпечує швидку відповідь системи навіть при обробці великої кількості маршрутів.

Інтеграція з зовнішніми API реалізована з урахуванням можливих затримок та обмежень швидкості запитів. Система повторних спроб та timeout-ів забезпечує надійну роботу навіть при тимчасових проблемах з мережею або перевантаженні зовнішніх сервісів. Кешування результатів запитів до Open-Elevation API дозволяє

зменшити навантаження на зовнішній сервіс та прискорити обробку повторних запитів для однакових координат.

5.5 Керівництво користувача

В цьому розділі розглянуто функціонал, наявний у розробленій системі. Система надає можливість перегляду, фільтрації, додавання та редагування туристичних маршрутів з інтуїтивно-зрозумілим інтерфейсом.

5.5.1 Головна сторінка системи

При запуску системи користувач бачить її головну сторінку, наведеної на рисунку 5.2.

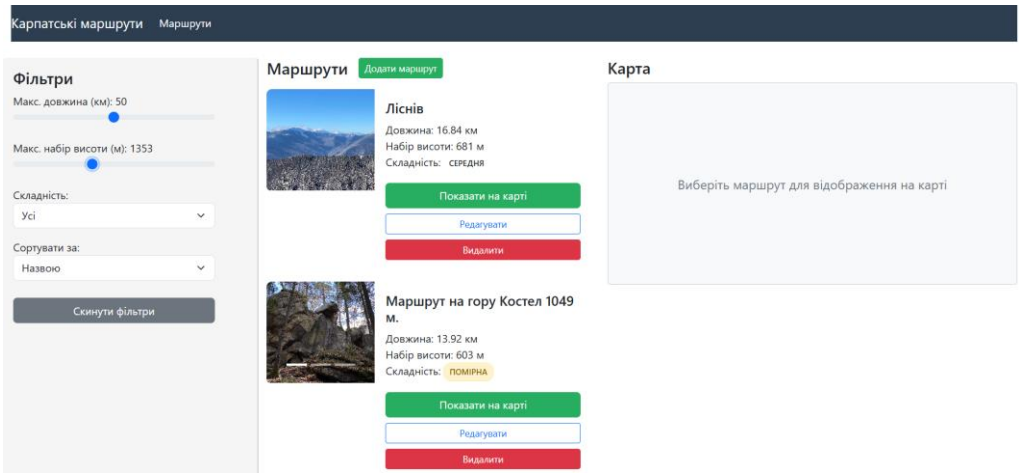

Рисунок 5.2 – Головна сторінка системи

Інтерфейс створено з урахуванням принципів зручності використання та забезпечує швидкий доступ до всіх основних функцій. Головна сторінка системи складається з трьох основних секцій:

1) панель фільтрів та сортувань (ліва частина);
2) список доступних маршрутів (центральна частина);
3) інтерактивна карта для відображення маршрутів (права частина).

5.5.2 Система фільтрації маршрутів

У лівій частині інтерфейсу знаходиться панель фільтрів та сортування, що дозволяє користувачам налаштовувати критерії пошуку маршрутів відповідно до їхніх потреб та фізичних можливостей. На рисунку 5.3 зображено панель фільтрів.

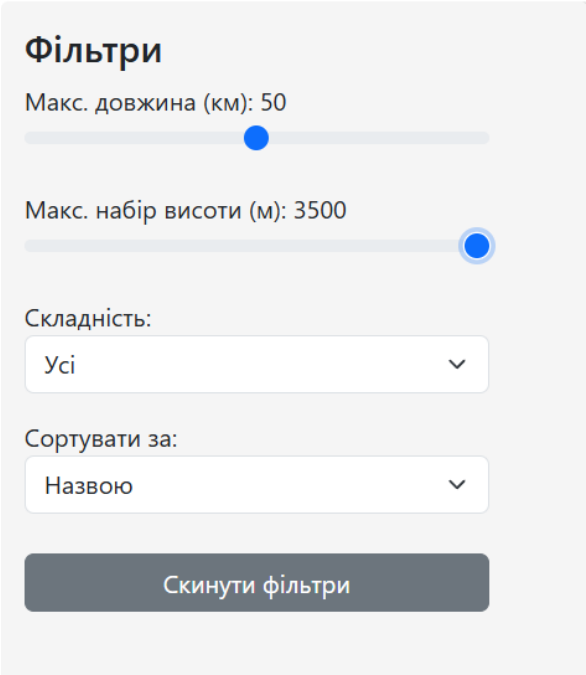

Рисунок 5.3 – Панель фільтрів

Доступні параметри фільтрації:

1) максимальна довжина маршруту – повзунок для встановлення максимально допустимої відстані від 0 до 100 км;

2) максимальний набір висоти – регулювання набору висоти маршруту від 0 до 3500 метрів;

3) рівень складності – випадаючий список з варіантами: "Усі", "Легка", "Помірна", "Складна", "Дуже складна", "Екстремальна";

4) сортування результатів – можливість упорядкування за назвою, довжиною, набором висоти або складністю.

5.5.3 Перегляд списку маршрутів

Центральна частина інтерфейсу відведена для відображення списку маршрутів, що відповідають заданим критеріям фільтрації. Список складається з карток відповідних маршрутів Список маршрутів зображено на рисунку 5.4.

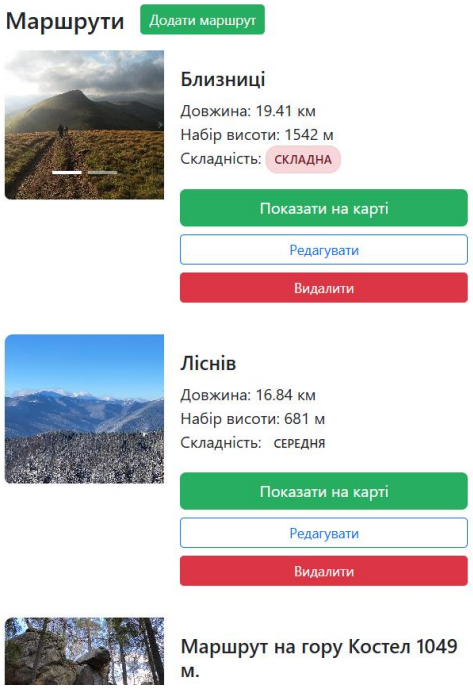

Рисунок 5.4 – Список маршрутів

Кожна картка маршруту містить наступну інформацію:

1) зображення маршруту – якщо завантажено кілька фотографій, відображається у вигляді каруселі з можливістю перегортання;
2) назва маршруту – чітко виділена назва для легкої ідентифікації;
3) основні характеристики(довжину, загальний набір висоти та складність маршруту);
4) кнопки дій(“Показати на карті”, “Редагувати”, “Видалити”).

При наведенні курсору на картку маршруту відображається спливаючий опис із детальною інформацією про маршрут.

5.5.4 Інтерактивна карта

Права частина інтерфейсу призначена для відображення обраних маршрутів на інтерактивній карті. Компонент карти зображено на рисунку 5.5

Рисунок 5.5 – Компонент карти

Функціонал карти включає:

1) вибір шару карти – перемикання між стандартною картою (OSM) та топографічною картою (OpenTopoMap);
2) відображення маршруту – синя лінія, що показує весь шлях маршруту;
3) маркери початку та кінця – позначки на початковій та кінцевій точках маршруту;
4) спливаючі вікна – інформаційні підказки при натисканні на маркери;
5) масштабування та навігація – стандартні елементи керування картою.

Карта автоматично налаштовує масштабується та центрується на обраному маршруті для зручного відображення його.

5.5.5 Додавання нового маршруту

Для додавання нового маршруту користувач натискає кнопку "Додати маршрут" у верхній частині списку маршрутів, наведеного раніше на рисунку 5.9, після чого відкривається модальне вікно “Додати новий маршрут”.

Процес додавання маршруту складається з декількох ключових етапів. Для початку необхідно ввести загальні дані про маршрут такі як назва та опис маршруту. На рисунку 5.10 зображено процес внесення назви та опису маршруту.

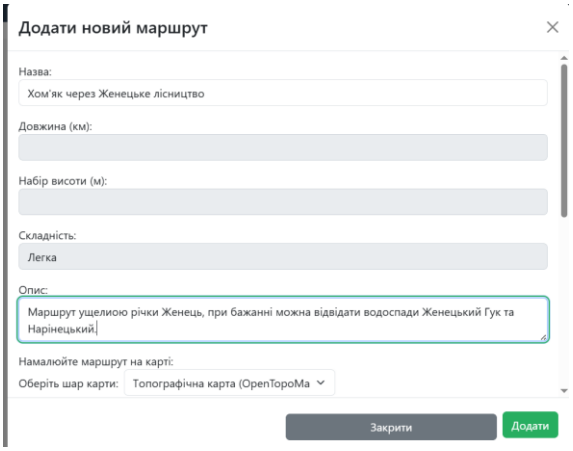

Рисунок 5.6 – внесення даних про назву та опис маршруту

Система не дає можливості вручну ввести відстань та загальний набір висоти, а складність автоматично визначає як легку. Для кращої візуалізації ці поля виділені іншим кольором. Система автоматично їх заповнює проводячи власні розрахунки, лише після отримання треку маршруту. Для цього необхідно додати GPX-файл або вручну намалювати маршрут на карті. Для того щоб вручну намалювати маршрут на карті необхідно натиснути кнопку “Draw” на панелі інструментів в правій частині карти та задати стартову точку маршруту. Після чого додаючи наступні проміжні точки створювати маршрут, при цьому система буде автоматично їх з’єднувати синьою полілінією. При неправильному виборі точки можна натиснути кнопку “Delete last point”, після чого остання обрана точка буде видалена. Для завершення малювання маршруту необхідно натиснути кнопку “Finish”, після чого система додасть трек маршруту до даних про нього та

відобразить його на карті. На рисунку 5.7 зображено процес малювання маршруту по карті вручну.

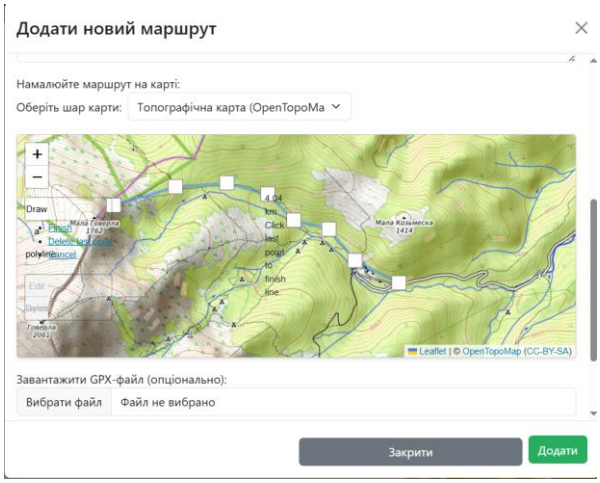

Рисунок 5.7 – Процес малювання маршруту по карті

Також є можливість завантажити існуючий трек маршруту, створений з допомогою інших додатків. Для цього необхідно натиснути кнопку “Вибрати файл” та обрати відповідний файл формату GPX. На рисунку 5.8 зображено додавання GPX-файлу до даних про маршрут.

Рисунок 5.8 – Відображення маршруту на карті після додавання GPX-файлу

Після отримання треку маршруту система автоматично обчислить такі параметри як відстань, загальний набір висоти та складність маршруту, та заповнить відповідні поля. Для завершення створення нового маршруту необхідно додати зображення до маршруту. Система дає можливість завантажити до 5 зображень маршруту, при цьому додавання хоча б одного зображення є обов’язковим. Для того щоб додати зображення необхідно натиснути кнопку

“Вибрати файл” у відповідному полі та обрати потрібне зображення. На рисунку 5.9 зображено процес додавання фото до маршруту.

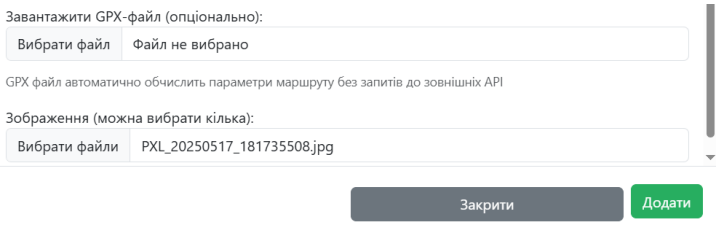

Рисунок 5.9 – Додавання зображень до даних про маршрут

Після виконання всіх кроків необхідно натиснути кнопку додати, та маршрут з’явиться в списку. Створений маршрут можна редагувати натиснувши кнопку видалити, або редагувати, процес чого буде описаний в наступному пункті. Діаграму послідовності створення маршруту наведено в кресленику ІС12.060БАК.005 Д4.

5.5.6 Редагування існуючого маршруту

При натисканні кнопки "Редагувати" відкривається аналогічне модальне вікно з попередньо заповненими полями. На рисунку 5.10 зображено вікно редагування

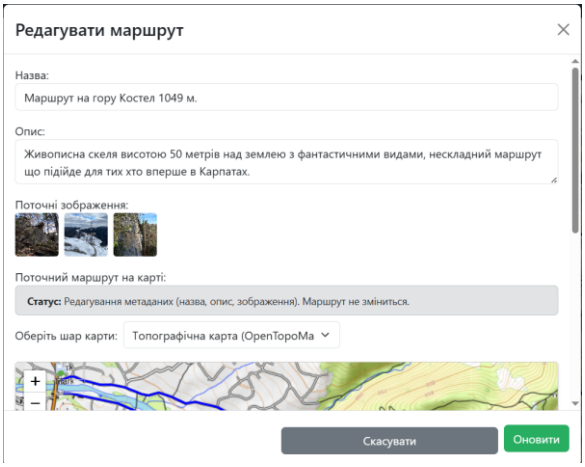

Рисунок 5.10 – вікно редагування маршруту

При редагуванні маршруту можна вносити зміни в назву та опис маршруту заповнивши відповідні поля заново. Також можливо переглянути уже існуючі

зображення та трек маршруту. За потреби можна додати новий GPX-файл треку маршруту чи намалювати вручну аналогічно як при створення нового маршруту. Також існує можливість додати нові зображення зберігши попередні. Редагування даних про маршрут зображено на рисунку 5.11.

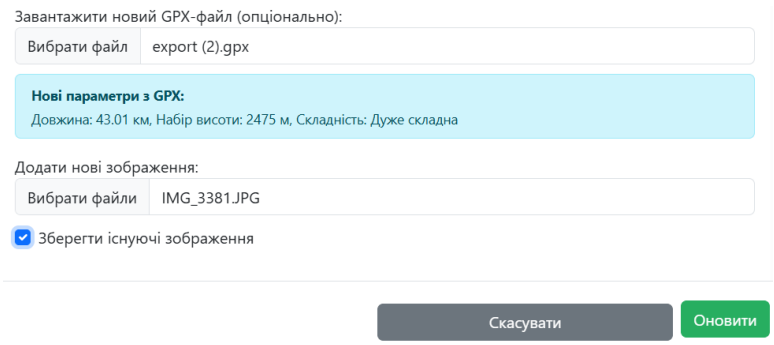

Рисунок 5.11 – Зміна даних про маршрут

Для кращої наочності було продемонстровано результат зміни даних про маршрут. На рисунку 5.12 зображено маршрут до редагування.

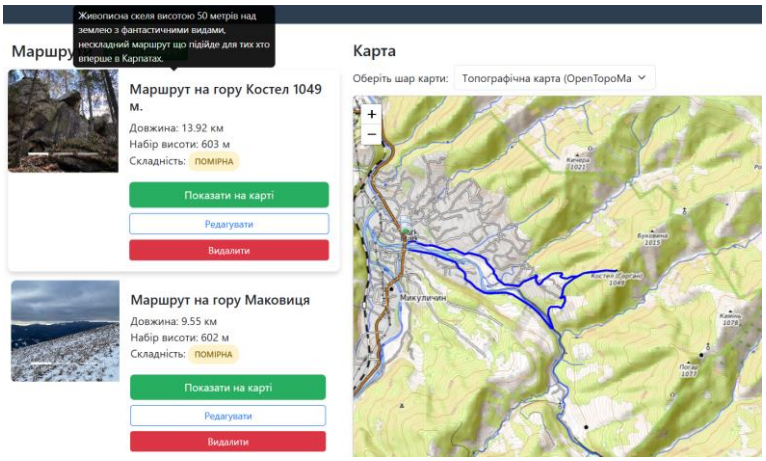

Рисунок 5.12 – Маршрут до редагування

Для прикладу необхідно додати нове фото, при цьому зберігши старі, але змінити трек, назву та опис маршруту. Для зміни даних про назву та опис маршруту необхідно натиснути на відповідні поля, видалити попередній текст та додати новий. На рисунку 5.13 зображено процес редагування даних про назву та опис маршруту.

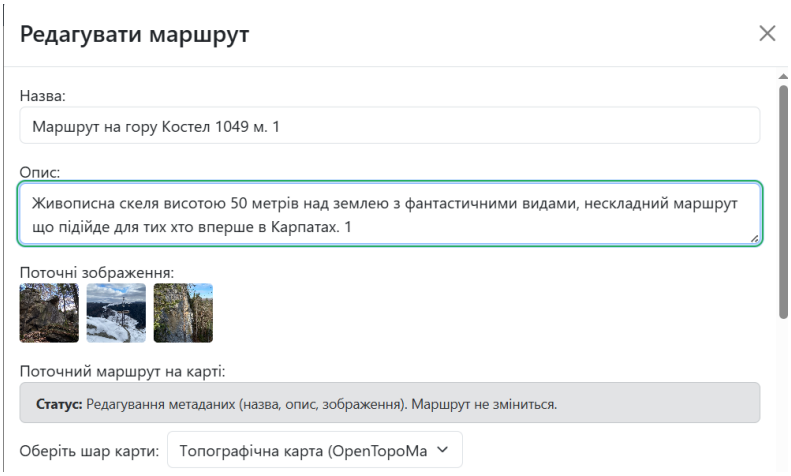

Рисунок 5.13 – Внесення змін до даних про маршрут

Для додавання нового фото необхідно натиснути кнопку “Вибрати файли” в області “Додати нові зображення”та завантажити необхідні файли. Для того щоб зберегти існуючі зображення необхідно поставити галочку у відповідному чекбоксі. Для оновлення даних про трек маршруту необхідно натиснути на кнопку “Вибрати файл” в області “Завантажити новий GPX-файл(опціонально)” та імпортувати потрібний GPX-файл. На рисунку 5.14 зображено процес додавання нового фото та GPX-файлу.

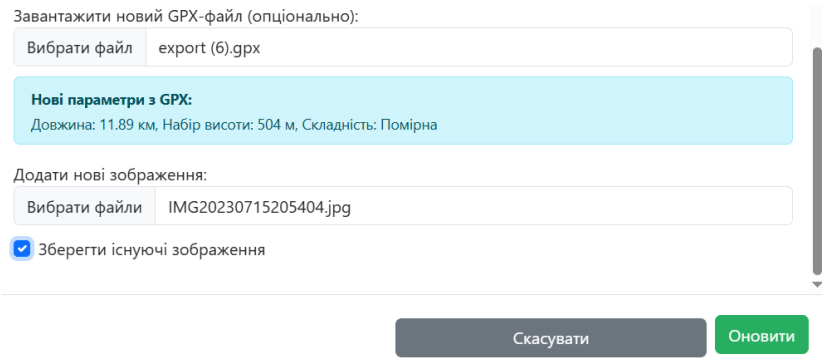

Рисунок 5.14 – Додавання нового треку маршруту та зображення

Після внесення відповідних змін необхідно натиснути кнопку оновити. Система автоматично обчислить нові значення відстані, загального набору висоти, здійснить оцінку складності зміненого маршруту та збереже в базу даних новий трек маршруту. На рисунку 5.15 продемонстровано результат внесення змін.

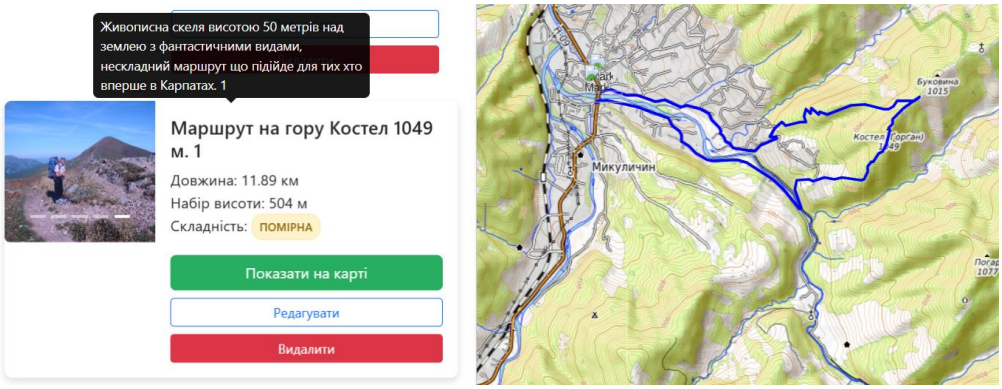

Рисунок 5.15 – Демонстрація змін після редагуванні маршруту

В списку відображається оновлений маршрут зі зміненою назвою, при наведені на маршрут в списку з’являється змінений опис, в карусель зображень додане нове фото маршруту, при натисканні кнопки “Показати на карті” відображається змінений трек маршруту.

Висновки до розділу 5

Під час роботи над розділом було детально описано процес розроблення рекомендаційної системи туристичних маршрутів в українських Карпатах, було проведено детальний аналіз особливостей її архітектури, структури компонентів, обробки даних та їх зберігання. Також було описано специфікацію усіх основних функцій системи, архітектуру клієнт-серверної взаємодії та зв'язки між компонентами системи.

При розробленні системи особливу увагу було надано архітектурі програмного забезпечення, яка базується на сучасних вебтехнологіях та забезпечує ефективну роботу з географічними даними. Було описано створення клієнтської частини розробленої з застосуванням вебфреймворку React та бібліотеки Leaflet для відображення карти. Аналогічно описано процес розроблення серверної частини створеної на основі Node.js з вебфреймворком Express та базою даних MongoDB.

Реалізована архітектура забезпечує масштабованість системи та можливість подальшого розвитку функціоналу. Модульна структура коду, використання REST API та стандартних форматів даних створюють основу для інтеграції з додатковими сервісами, наприклад прогнозами погоди.

Було складено детальне керівництво користувача, яке пояснює особливості взаємодії з системою та надає вичерпну інформацію щодо використання повного функціоналу системи. Інтерфейс спроєктовано з урахуванням принципів зручності використання та зрозумілості, що робить систему доступною як для досвідчених туристів, так і для початківців у плануванні гірських походів.

6 МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

6.1 Змістовна постановка задачі

Розроблена система призначена для обчислення відстані туристичного маршруту в Карпатах, заданого послідовністю точок у двовимірному просторі, де кожна точка представлена географічними координатами (широта та довгота). Маршрут складається з деякої кількості послідовних точок, і завдання полягає в тому, щоб знайти загальну відстань маршруту, враховуючи сферичну форму Землі. Кожна точка маршруту є вершиною, сполученою з наступною, а відстань між сусідніми точками обчислюється з урахуванням кривини земної поверхні.

6.2 Математична постановка задачі

Розглянемо маршрут, заданий послідовністю з n точок у географічному просторі. Кожна точка P_i (i = 1, 2, ..., n) має координати (φ_i, λ_i), де φ_i – широта (в радіанах), λ_i – довгота (в радіанах). Загальна відстань маршруту D є сумою відстаней між послідовними точками P_i та P_{i+1} для i = 1, 2, ..., n − 1.

Змінні:

Простір маршруту позначимо як множину точок R, де кожна точка $P_i \in R$ має координати (φ_i, λ_i).

Відстань між двома сусідніми точками P_i та P_{i+1} позначимо як d_i.

Загальна відстань маршруту позначимо як $D = \sum_{i=1}^{n-1} d_i$.

Радіус Землі R_e приймаємо як константу, приблизно R_e = 6371 км.

Обмеження:

Кожна точка P_i належить географічному простору, тобто $\varphi_i \in [-\pi/2, \pi/2]$, $\lambda_i \in [-\pi, \pi]$.

Маршрут складається щонайменше з двох точок: $n \geq 2$.

Відстань між точками обчислюється з урахуванням сферичної геометрії Землі.

Цільова функція

Мета – знайти загальну відстань маршруту *D*, яка визначається як:

$$\mathrm{D} = \sum_{i=1}^{n-1} d_i \qquad (6.1)$$

де d_i – відстань між точками P_i та P_{i+1}, обчислена за формулою гаверсинусів.

6.3 Обґрунтування методу розв'язання

Для обчислення відстані між двома точками на поверхні Землі існує кілька підходів, зокрема формула гаверсинусів, формула косинусів для сферичної тригонометрії та спрощена евклідова формула для плоскої поверхні [22]. Оскільки Карпати є гірською місцевістю, де відстані між точками можуть бути відносно невеликими, але точність обчислень критично важлива через нерівності рельєфу, необхідно обрати метод, який враховує сферичну форму Землі.

Формула гаверсинусів є стандартним методом для обчислення відстані між двома точками на сфері за їхніми географічними координатами. Вона враховує кривизну Землі та є точною для малих і середніх відстаней, що ідеально підходить для туристичних маршрутів у Карпатах[23]. Формула проста в реалізації та обчислювально-ефективна.

Формула косинусів також враховує сферичну геометрію, але є менш стійким до числових похибок при малих відстанях (менше 1 км), оскільки включає обчислення арккосинуса, що може втрачати точність.

Евклідова формула це прощена формула для плоскої поверхні не враховує кривизну Землі, що може призвести до значних похибок навіть на відстанях у десятки кілометрів, особливо в гірській місцевості.

Формула гаверсинусів була обрана через її точність, простоту реалізації та обчислювальну ефективність. Вона дозволяє коректно обчислювати відстані між точками маршруту, враховуючи сферичну геометрію, і є стандартним

інструментом у географічних інформаційних системах та навігаційних застосунках.

6.4 Опис методу розв’язання

Формула гаверсинусів обчислює відстань між двома точками на сфері за їхніми координатами широти та довготи. Для двох точок $P_i(\varphi_i, \lambda_i)$ та P_{i+1} $(\varphi_{i+1}, \lambda_{i+1})$ відстань d_i визначається так:

$$a = sin^2\left(\frac{\varphi_{i+1} - \varphi_i}{2}\right) + \cos(\varphi_i)\cos(\varphi_{i+1})\, sin^2\left(\frac{\lambda_{i+1} - \lambda_i}{2}\right) \quad (6.2)$$

$$c = 2\tan^{-1} 2\left(\sqrt{a}, \sqrt{1-a}\right) \quad (6.3)$$

$$d_i = R_e c \quad (6.4)$$

φ_i, φ_{i+1} – широти точок P_i та P_{i+1} (у радіанах),

λ_i, λ_{i+1} – довготи точок P_i та P_{i+1} (у радіанах),

R_e – середній радіус Землі, приблизно 6371 км,

a – квадрат гаверсинуса центрального кута між точками,

c – центральний кут у радіанах,

d_i – відстань між точками P_i та P_{i+1} у кілометрах.

Загальна відстань маршруту обчислюється як сума відстаней між усіма послідовними парами точок за формулою 6.1.

Алгоритм є ітеративним і послідовно обчислює відстань для кожної пари точок, додаючи її до загальної суми. Блок-схему алгоритму наведено в кресленику ІС12.060БАК.005 Д5. Нижче наведено псевдокод алгоритму для обчислення загальної відстані маршруту:

```
function CalculateRouteDistance(points):
R_e = 6371 //Радіус Землі в кілометрах
total_distance = 0
```

```
for i from 1 to length(points) - 1:
    lat1 = points[i].latitude * π / 180 // Конвертація широти в радіани
    lat2 = points[i+1].latitude * π / 180
    lon1 = points[i].longitude * π / 180 // Конвертація довготи в радіани
    lon2 = points[i+1].longitude * π / 180
    delta_lat = lat2 - lat1
    delta_lon = lon2 - lon1
    a = sin(delta_lat/2)^2 + cos(lat1) * cos(lat2) * sin(delta_lon/2)^2 c = 2 *
    atan2(sqrt(a), sqrt(1-a))
    distance = R_e * c total_distance
    += distance
return total_distance
```

6.5 Приклади роботи алгоритму

Для демонстрації роботи алгоритму було розглянуто тестовий маршрут, що складається з трьох точок A(48.15288889, 24.50430556), B(48.15025, 24.51105556) і C(48.154187, 24.517811).

Спочатку необхідно перевести градуси в радіани:

Для точки A:

$\text{lat}_\text{a} = 48.15288889 \frac{\pi}{180} = 0.84034653$ рад

$\text{lon}_\text{a} = 24.50430556 \frac{\pi}{180} = 0.42757445$ рад

Для точки B:

$\text{lat}_\text{b} = 48.15025 \frac{\pi}{180} = 0.84029864$ рад

$\text{lon}_\text{b} = 24.51105556 \frac{\pi}{180} = 0.42769310$ рад

Для точки C:

$\text{lat}_\text{c} = 48.154187 \frac{\pi}{180} = 0.840346977$ рад

$\text{lon}_\text{c} = 24.517811 \frac{\pi}{180} = 0.042781174$ рад

$\Delta lat = lat_b - lat_a = -0.00004789$рад

$\Delta lon = lon_b - lon_a = 0.00011865$ рад

Обчислити a – квадрат гаверсинуса центрального кута між точками за формулою 6.2:

$$a = sin^2\left(\frac{\Delta lat}{2}\right) + \cos(\text{lat}_\text{a})\cos(\text{lat}_b)\, sin^2\left(\frac{\Delta lon}{2}\right) = 2.138 * 10^{-9}$$

Обчислення c – центрального кута за формулою 6.3:

$$c = 2\tan^{-1} 2\left(\sqrt{a}, \sqrt{1-a}\right) = 9.248 \cdot 10^{-5}\text{рад}$$

Далі необхідно розрахувати відстань між точками А та В за формулою 6.4:

$$d_{ab} = R_e c = 6371 \cdot 9.248 \cdot 10^{-5} = 0.589\text{км}$$

Аналогічно відстань ВС

$\Delta lat = lat_c - lat_b = 0.00007113$рад

$\Delta lon = lon_c - lon_b = 0.00011864$ рад

$$a = sin^2\left(\frac{\Delta lat}{2}\right) + \cos(\text{lat}_b)\cos(\text{lat}_c)\, sin^2\left(\frac{\Delta lon}{2}\right) = 2.830 \cdot 10^{-9}$$

$$c = 2\tan^{-1} 2\left(\sqrt{a}, \sqrt{1-a}\right) = 1.064 \cdot 10^{-4}\text{рад}$$

$$d_{bc} = R_e c = 6371 \cdot 1.064 \cdot 10^{-4} = 0.678\text{км}$$

Для отримання відстані ABC необхідно додати відстані АВ та ВС:

$$d_{abc} = d_{ab} + d_{bc} = 1.267\text{км}$$

Для перевірки коректності алгоритму необхідно порівняти отримані дані з вимірами здійсненими іншим програмним забезпеченням. Для цього використано інструмент лінійки в додатку Google Maps. На рисунку 6.1 зображено дані виміри.

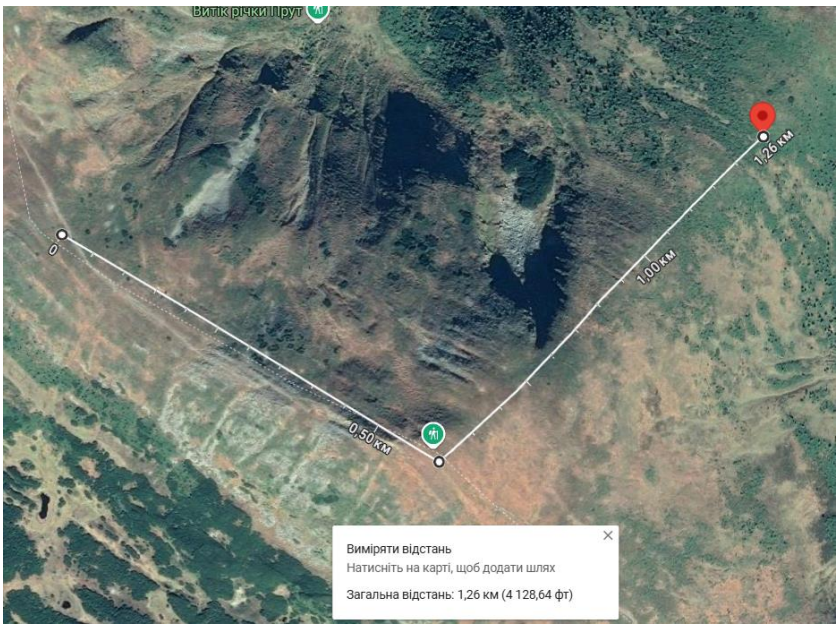

Рисунок 6.1 – перевірка точності алгоритму

Висновки до розділу 6

У цьому розділі було сформульовано змістовну та математичну постановки задачі обчислення відстані деякого маршруту. Проведено аналіз можливих методів розв’язання, серед яких розглянуто формулу гаверсинусів, формулу косинусів та евклідову формулу. Було обрано метод обчислення відстані через формулу гаверсинусів через її точність, простоту реалізації та обчислювальну ефективність.

Описано математичну модель задачі, наведено детальний опис формули гаверсинусів та псевдокод алгоритму. Проведено експеримент з тестовим маршрутом, який підтвердив надійність та точність обраного методу. Розроблений алгоритм є універсальним і може бути використаний у туристичних застосунках для планування маршрутів у будь-якій місцевості в тому числі гірській.

7 ТЕСТУВАННЯ СИСТЕМИ

7.1 Мета випробувань

Метою випробувань рекомендаційної системи туристичних маршрутів в українських Карпатах є перевірка коректності її роботи та відповідність усім поставленим вимогам.

Для досягнення цієї мети потрібно визначити та описати набір сценаріїв тестування, які будуть охоплювати весь реалізований функціонал. Відповідно до визначених сценаріїв потрібно провести тестування розробленої системи та впевнитись, що при цьому не виникає непередбачуваних помилок, система працює без затримок та збоїв.

7.2 Загальні положення

Для отримання релевантних результатів, необхідно провести такі види тестів: функціональні, тести продуктивності та стрес-тести.

Функціональне тестування дозволить перевірити, чи реалізовані усі потрібні функції та чи вони працюють відповідно до визначених у розділі 3 вимог. Необхідно перевірити коректність роботи інтерфейсу користувача, обчислення географічних параметрів маршрутів, системи фільтрації і сортування, а також збереження даних.

Тестування продуктивності дозволить перевірити, як система працює під навантаженням при обробці великої кількості маршрутів та складних геоданих. Це дозволить зрозуміти, наскільки вона підходить для використання з великими масивами даних про туристичні маршрути.

Стрес-тестування призначене для демонстрації реакції системи на дуже високі навантаження та нестандартні ситуації. Такі тести проводяться для виявлення потенційних збоїв у роботі системи при обробці некоректних даних або при збоях зовнішніх API.

7.3 Результати випробувань

Відповідно до обраної стратегії та визначених раніше вимог були створені сценарії тестування, що покриватимуть увесь функціонал розробленої системи. Результати усіх випробувань наведені у таблицях 7.1–7.12.

7.3.1 Тестування основного функціоналу системи

Спочатку було випробувано основний функціонал системи: створення треків маршрутів, збереження інформації про маршрут в системі, відображення треку створеного маршруту на карті. Створення треків маршрутів випробувано з допомогою малювання треку маршруту вручну на карті та імпорту GPX-файлу маршруту. У таблицях 7.1–7.4 наведено результати цих випробувань.

Таблиця 7.1 – Тестування створення маршруту методом малювання на карті

Елемент тестування	Опис
Ціль тестування	Перевірка коректності створення маршруту через інтерфейс малювання на карті
Передумови	Система завантажена, модальне вікно створення маршруту відкрите
Вхідні дані	Послідовність натискань курсором на карті для створення треку маршруту
Послідовність кроків	Натиснути кнопку Draw на карті, встановити 7 точок маршруту, натиснути кнопку Finish
Очікуваний результат	Маршрут відображається на карті, автоматично обчислюються відстань, набір висоти та складність
Фактичний результат	Система успішно створила маршрут, заповнила поля відстань, набір висоти та складність, відобразила трек на карті

Таблиця 7.2 – Тестування створення маршруту через імпорт GPX-файлу

Елемент тестування	Опис
Ціль тестування	Перевірка коректності створення треку маршруту через імпорт GPX-файлу
Передумови	Система завантажена, модальне вікно створення маршруту відкрите
Вхідні дані	Коректний GPX-файл маршруту
Послідовність кроків	Обрати GPX-файл через “Вибрати файл”, дочекатися обробки файлу
Очікуваний результат	Маршрут відображається на карті, автоматично обчислюються відстань, набір висоти та складність
Фактичний результат	Система успішно створила маршрут, заповнила поля відстань, набір висоти та складність, відобразила трек на карті

Таблиця 7.3 – Тестування збереження маршруту

Елемент тестування	Опис
Ціль тестування	Перевірка коректності збереження повної інформації про маршрут в системі
Передумови	Створений трек маршруту з обчисленими параметрами відстані, набору висоти та складності
Вхідні дані	Назва, опис та зображення маршруту
Послідовність кроків	Заповнити назву та опис маршруту, додати зображення, натиснути кнопку “Додати”
Очікуваний результат	Маршрут збережено, він відображається в списку маршрутів з повною коректною інформацією
Фактичний результат	Новий маршрут успішно додано до списку, всі дані збережені правильно

Таблиця 7.4 – Тестування відображення маршруту на карті

Елемент тестування	Опис
Ціль тестування	Перевірка коректності відображення існуючого маршруту на карті
Передумови	В списку маршрутів є декілька маршрутів
Вхідні дані	Довільний маршрут зі списку
Послідовність кроків	Обрати маршрут зі списку та натиснути кнопку “Показати на карті”
Очікуваний результат	Трек маршруту, відображається на карті в правій частині екрану
Фактичний результат	Карта автоматично масштабувалась до меж маршруту, трек маршруту відображений синьою лінією

7.3.2 Тестування системи фільтрації та сортування

Наступним етапом є тестування коректності сортування та фільтрації маршрутів за різними параметрами. Результати тестування наведені у таблицях 7.5–7.8.

Таблиця 7.5 – Тестування фільтрації за довжиною маршруту

Елемент тестування	Опис
Ціль тестування	Перевірка коректності фільтрації списку маршрутів за їх довжиною
Передумови	Список з декількох маршрутів різної довжини
Вхідні дані	Обмеження довжини маршруту 15 кілометрів
Послідовність кроків	Встановити повзунок довжини на 15 км та дочекатися оновлення списку

Елемент тестування	Опис
Очікуваний результат	В списку маршрутів відображено лише ті маршрути, чия довжина менше 15 кілометрів
Фактичний результат	Система правильно відфільтрувала маршрути, показавши лише ті, що відповідають критерію

Таблиця 7.6 – Тестування фільтрації за набором висоти

Елемент тестування	Опис
Ціль тестування	Перевірка коректності фільтрації списку маршрутів за набором висоти
Передумови	Список з декількох маршрутів з різним набором висоти
Вхідні дані	Обмеження набору висоти на маршруті 800 метрів
Послідовність кроків	Встановити повзунок максимального набору висоти на 800 м та дочекатися оновлення списку
Очікуваний результат	В списку маршрутів відображено лише ті маршрути, чий набір висоти складає менше 800 метрів
Фактичний результат	Система правильно відфільтрувала маршрути, показавши лише ті, що відповідають критерію

Таблиця 7.7 – Тестування фільтрації за складністю

Елемент тестування	Опис
Ціль тестування	Перевірка коректності фільтрації маршрутів за параметром складності
Передумови	Список з декількох маршрутів з різним рівнями складності
Вхідні дані	“Помірна” складність маршруту

Елемент тестування	Опис
Послідовність кроків	Обрати “Помірна” в списку фільтру складності, перевірити оновлений список маршрутів
Очікуваний результат	В списку маршрутів відображаються лише маршрути “Помірної” складності
Фактичний результат	Фільтр успішно застосований, показані маршрути відповідної складності

Таблиця 7.8 – Тестування сортування маршрутів

Елемент тестування	Опис
Ціль тестування	Перевірка функціональності сортування
Передумови	Список маршрутів з різними параметрами
Вхідні дані	Критерій сортування “За довжиною”
Послідовність кроків	Обрати “За довжиною” в списку сортування, дочекатися оновлення списку, перевірити порядок відображення маршрутів в оновленому списку
Очікуваний результат	Маршрути в списку відображені в порядку збільшення їх довжини
Фактичний результат	Список маршрутів успішно відсортований за зростанням довжини

7.3.3 Тестування редагування та видалення маршрутів

Наступним етапом є тестування коректності роботи функціоналу редагування інформації про маршрут та видалення маршрутів з системи. Результати випробувань наведені у таблицях 7.9–7.10.

Таблиця 7.9 – Тестування редагування даних існуючого маршруту

Елемент тестування	Опис
Ціль тестування	Перевірка коректності редагування даних про маршрут
Передумови	Збережений маршрут в системі
Вхідні дані	Нові назва, опис, GPX-файл треку маршруту, нове зображення
Послідовність кроків	Натиснути “Редагувати” в картці маршруту, змінити назву та опис, додати нове зображення, додати новий GPX-файл маршруту, натиснути “Оновити”
Очікуваний результат	Дані про маршрут оновлено, він відображається в списку з зміненою інформацією, на карті відображається змінений треком маршруту
Фактичний результат	Зміни успішно збережені, маршрут відображається з оновленою інформацією

Таблиця 7.10 – Тестування видалення маршруту

Елемент тестування	Опис
Ціль тестування	Перевірка коректності видалення маршруту
Передумови	Існуючий збережений маршрут
Вхідні дані	–
Послідовність кроків	Натиснути кнопку “Видалити” на картці маршруту, підтвердити видалення
Очікуваний результат	Маршрут видалений зі списку та бази даних
Фактичний результат	Маршрут успішно видалений, файли зображень також очищені

7.3.4 Тестування продуктивності та стрес-тести

Заключним етапом тестувань є перевірка роботи системи під навантаженням. Результати випробувань наведені у таблицях 7.11–7.12.

Таблиця 7.11 – Тестування продуктивності з великою кількістю маршрутів

Елемент тестування	Опис
Ціль тестування	Перевірка продуктивності при роботі з великим обсягом даних
Передумови	База даних з 100 маршрутами
Вхідні дані	Запити на фільтрацію та сортування
Послідовність кроків	Завантажити сторінку, застосувати різні фільтри, змінити критерії сортування
Очікуваний результат	Швидка обробка запитів менше 2 секунд
Фактичний результат	Система обробляє запити вкладаючись у встановлений час, продуктивність задовільна

Таблиця 7.12 – Стрес-тестування обробки некоректних GPX-файлів

Елемент тестування	Опис
Ціль тестування	Перевірка стійкості системи до внесення некоректних вхідних даних
Передумови	Підготовлені некоректні файли, відкрите вікно створення маршруту
Вхідні дані	Пошкоджений GPX, файл іншого формату, порожній файл

Елемент тестування	Опис
Послідовність кроків	Спробувати завантажити пошкоджений GPX-файл, завантажити файл неправильного формату та порожній файл
Очікуваний результат	Система правильно обробляє помилки без збоїв
Фактичний результат	Система показала відповідні повідомлення про помилки, збоїв не виявлено

Висновки до розділу 7

Під час роботи над розділом було визначено мету випробувань та обрано методики тестування, що відповідають специфіці структури розробленої системи. Відповідно до обраних підходів, було проведено детальне тестування всіх компонентів системи.

Результати випробувань показали, що система відповідає усім визначеним у розділі 3 вимогам, є стабільною та продуктивною в різних умовах використання. Весь необхідний функціонал реалізований у повному обсязі:

1) коректно працює створення маршрутів як малюванням на карті, так і імпортом GPX-файлів;
2) система фільтрації та сортування працює коректно;
3) функції редагування та видалення маршрутів працюють коректно;
4) система демонструє достатню продуктивність при великій кількості маршрутів.

Виявлено мінімальні недоліки: незначну затримку при обробці дуже великих GPX-файлів та залежність від зовнішнього API для отримання висот при малюванні вручну. Ці обмеження не впливають на основну функціональність системи.

Загалом, розроблена система готова до практичного використання та може бути застосована туристам для планування безпечних походів в українських Карпатах.

ВИСНОВКИ

Основною мотивацією розробки рекомендаційної системи туристичних маршрутів в українських Карпатах стала критична потреба у підвищенні безпеки гірського туризму через усунення суб'єктивності в оцінці складності маршрутів. Аналіз існуючих рішень виявив суттєвий недолік – залежність від людських оцінок, що часто призводить до отримання непідготовленим туристами некоректної інформації про реальну складність маршрутів.

Головною перевагою розробленої системи є автоматизована оцінка складності маршрутів на основі математичних розрахунків географічних параметрів відстані та загального набору висоти на маршруті. Це дозволяє туристам отримувати достовірну інформацію про складність маршруту незалежно від досвіду попередніх користувачів.

Вибір технологічного стеку React, Node.js, MongoDB забезпечує високу масштабованість та продуктивність системи. React надає можливість створення інтуїтивного інтерактивного інтерфейсу з картами, що особливо важливо для геоінформаційних систем. Node.js з Express забезпечує ефективну обробку асинхронних запитів до зовнішніх API, а MongoDB з індексацією полів значно пришвидшує фільтрацію та сортування маршрутів навіть при великих обсягах даних. Модульна архітектура системи дозволяє легко розширювати функціонал без перебудови основної структури.

Серед недоліків системи слід відзначити залежність від доступності зовнішнього Open-Elevation API, що може призводити до тимчасової недоступності функції розрахунку висот. Обробка дуже великих GPX-файлів може призводити до затримок, що впливає на користувацький досвід. Алгоритм оцінки складності базується на відстані та наборі висоти, проте не враховує фактори типу поверхні чи стану стежок, що також може впливати на складність маршруту.

Використання системи є доцільним завдяки її здатності підвищити безпеку гірського туризму через надання об'єктивної інформації про туристичні маршрути в гірській місцевості. Автоматизація процесу оцінки складності усуває людський

фактор та забезпечує релевантність результатів. Реалізація у вигляді вебдодатка забезпечує широку доступність системи та сумісність з різними операційними системами і пристроями.

Система може бути покращена шляхом інтеграції з додатковими джерелами даних, такими як метеорологічні сервіси, бази даних про стан стежок чи різноманітні об’єкти на маршрутах. Впровадження технологій машинного навчання дозволить удосконалити алгоритм оцінки складності на основі аналізу статистики проходження маршрутів реальними користувачами.

Розроблена система повністю відповідає поставленій меті підвищення безпеки гірського туризму в українських Карпатах. Всі функціональні вимоги реалізовані, а тестування підтвердило працездатність системи в різних сценаріях використання. Система має високий потенціал для впровадження в туристичну індустрію України, може використовуватися звичайними туристами, туристичними організаціями, рятувальними службами та державними установами для моніторингу туристичного навантаження на природні території.

Таким чином, розроблена рекомендаційна система туристичних маршрутів представляє собою ефективне технологічне рішення для підвищення безпеки гірського туризму, що успішно поєднує сучасні вебтехнології з математичними методами аналізу геопросторових даних та має значний потенціал для практичного впровадження і подальшого розвитку.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Головне управління ДСНС в Івано-Франківській області. URL: https://if.dsns.gov.ua/news/ostanni-novini/u-golovnomu-upravlinni-dsns-ukrayini-v-oblasti-vidbuvsia-brifing-shhodo-nebezpek-litnyogo-periodu (дата звернення 8.06.2025)
2. Бейдик О. О. Рекреаційно-туристські ресурси України: методологія та методика аналізу, термінологія, районування. Київ : Видавничо-поліграфічний центр «Київський університет», 2019. 395 с.
3. Gavalas D., Konstantopoulos C., Mastakas K., Pantziou G. A survey on algorithmic approaches for solving tourist trip design problems. Journal of Heuristics. 2014. Vol. 20, № 3. P. 291–328.
4. Bao J., Zheng Y., Mokbel M. F. Location-based and preference-aware recommendation of routes for tourists. IEEE Transactions on Knowledge and Data Engineering. 2012. Vol. 24, № 11. P. 1990–2003.
5. Kozak J., Ostapowicz K., Bytnerowicz A., Wyżga B. The Carpathian Mountains: Challenges for the Central and Eastern European Landmark. GeoJournal. 2002. Vol. 57, № 1-2. P. 139–152.
6. Komoot. URL: https://www.komoot.com/ (дата звернення 10.05.2025)
7. AllTrails. URL: https://www.alltrails.com/ (дата звернення 10.05.2025)
8. Wikiloc. URL: https://uk.wikiloc.com/ (дата звернення 10.05.2025)
9. React Documentation. URL: https://react.dev/learn (дата звернення: 15.05.2025).
10. Vue.js Guide. URL: https://vuejs.org/guide/ (дата звернення: 15.05.2025).
11. Node.js Documentation. OpenJS Foundation. URL: https://nodejs.org/en/docs/ (дата звернення: 15.05.2025).
12. Express.js Documentation. URL: https://expressjs.com/en/ (дата звернення: 15.05.2025).
13. MongoDB Documentation. MongoDB Inc. URL: https://www.mongodb.com/docs/ (дата звернення: 15.05.2025).

14. Mongoose Documentation. URL: https://mongoosejs.com/docs/ (дата звернення: 15.05.2025).
15. Multer Documentation. URL: https://github.com/expressjs/multer#readme (дата звернення: 15.05.2025).
16. Django Documentation. Django Software Foundation. URL: https://docs.djangoproject.com/en/stable/ (дата звернення: 15.05.2025).
17. PostgreSQL Documentation. PostgreSQL Global Development Group. URL: https://www.postgresql.org/docs/current/ (дата звернення: 15.05.2025).
18. Leaflet Documentation. URL: https://leafletjs.com/reference.html (дата звернення: 15.05.2025).
19. Mapbox GL JS API Documentation. Mapbox Inc. URL: https://docs.mapbox.com/mapbox-gl-js/api/ (дата звернення: 15.05.2025).
20. Turf.js Documentation. URL: https://turfjs.org/docs/ (дата звернення: 15.05.2025).
21. Open-Elevation API Documentation. URL: https://open-elevation.com/ (дата звернення: 15.05.2025).
22. Vincenty T. Direct and inverse solutions of geodesics on the ellipsoid with application of nested equations. Survey Review. 1975. Vol. 23, № 176. P. 88–93.
23. Chopde N. R., Nichat M. K. Landmark based shortest path detection by using A* and Haversine formula. International Journal of Innovative Research in Computer and Communication Engineering. 2013. Vol. 1, № 2. P. 298–302.