Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Розробка веб-системи управління евакуаційними заходами та інформування населення при екологічних загрозах

Виконав: студент IV курсу, групи СН-42
спеціальності 122 Комп’ютерні науки
(шифр і назва спеціальності)

_____ Сухорончак Ю.П.
(підпис) (прізвище та ініціали)

Керівник _____ Палка О.В.
(підпис) (прізвище та ініціали)

Нормоконтроль _____ Липак Г.І.
(підпис) (прізвище та ініціали)

Завідувач кафедри _____ Боднарчук І.О.
(підпис) (прізвище та ініціали)

Рецензент _____ Мудрик І.Я.
(підпис) (прізвище та ініціали)

Тернопіль
2026

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
«_8_» _червня_ 2026 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Бакалавр
(назва освітнього ступеня)

за спеціальністю 122 Комп'ютерні науки
(шифр і назва спеціальності)

Студенту Сухорончаку Юрію Петровичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка веб-системи управління евакуаційними заходами та інформування населення при екологічних загрозах.

Керівник роботи Палка Олег Вікторович, доктор філософії, асистент кафедри КН
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «*14*» *травня* 2026 року № *4/9-239*

2. Термін подання студентом завершеної роботи 22 червня 2026 р.

3. Вихідні дані до роботи Вхідні дані – геокоординати користувача, реєстр захисних споруд громади, екологічні метрики якості повітря з Open-Meteo API. Вихідні дані – розраховані маршрути евакуації, інтерактивні векторні шари ураження на карті, фонові Push-сповіщення.

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. 1. Аналіз предметної області та постановка завдання. 1.1 Аналіз нормативно-правової бази та специфіки муніципального цивільного захисту. 1.2 Огляд та критичний аналіз існуючих інформаційних систем і аналогів. 1.3 Обґрунтування архітектурних та технологічних рішень. 1.4 Математичне обґрунтування геопросторового аналізу. 1.5 Технічна постановка завдання на розробку веб-системи. 1.6 Висновок до першого розділу. 2. Проектна частина та обґрунтування методів дослідження. 2.1 Системний аналіз та проектування веб-системи. 2.2 Концептуальне проектування бази даних. 2.3 Оптимізація просторового аналізу. 2.4 Проєктування REST API та моделей валідації. 2.5 Проєктування клієнтської частини та PWA. 2.6 Проектування інтерфейсу користувача та картографічних шарів. 2.7 Висновок до другого розділу. 3. Програмна реалізація та експериментальне тестування веб-системи. 3.1 Інструменти розробки та розгортання. 3.2 Реалізація серверної частини. 3.3 Програмна реалізація клієнтської частини та інтерактивного картографічного шару. 3.4 Програмна реалізація сервіс-воркера для фонового опрацювання сповіщень. 3.5 Тестування програмного комплексу та верифікація результатів. 3.6 Висновок до третього розділу. 4. Безпека життєдіяльності, основи охорони праці. Висновки. Перелік джерел. Додатки.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1. Титульний слайд. 2. Актуальність дослідження. 3. Мета і завдання дослідження. 4. Об'єкт та предмет дослідження. 5. Практичне значення. 6. Порівняння аналогів. 7.Архітектура та стек технологій. 8. Проєктування бази даних. 9. Алгоритми геолокації. 10. Реалізація back-end. 11. Фонове оповіщення. 12. Верифікація веб-системи. 13. Висновки. 14. Дякую за увагу!

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи охорони праці	Гурик О. Я, к.т.н., доцент каф. МТ	01.06.2026	08.06.2026

7. Дата видачі завдання 26 січня 2026 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	26.01.2026	*Виконано*
2.	Підбір та опрацювання літературних джерел по темі кваліфікаційної роботи	27.01.2026-16.02.2026	*Виконано*
3.	Виконання дослідження щодо процесів автоматизації муніципального цивільного захисту та евакуації населення при екологічних загрозах.	17.02.2026-10.05.2026	*Виконано*
	Розроблення клієнт-серверної архітектури, реляційної бази даних, асинхронного REST API та кросплатформеного PWA-інтерфейсу системи.		
4.	Оформлення розділу «Аналіз предметної області та постановка завдання»	11.05.2026-17.05.2026	*Виконано*
5.	Оформлення розділу «Проектна частина та обґрунтування методів дослідження»	18.05.2026-24.05.2026	*Виконано*
6.	Оформлення розділу «Програмна реалізація та експериментальне тестування веб-системи»	25.05.2026-31.05.2026	*Виконано*
7.	Виконання завдання до підрозділу «Безпека життєдіяльності»	01.06.2026-08.06.2026	*Виконано*
8.	Виконання завдання до підрозділу «Основи охорони праці»	01.06.2026-08.06.2026	*Виконано*
9.	Оформлення кваліфікаційної роботи	09.06.2026-11.06.2026	*Виконано*
10.	Нормоконтроль	12.06.2026-15.06.2026	*Виконано*
11.	Перевірка на плагіат	16.06.2026	*Виконано*
12.	Попередній захист кваліфікаційної роботи	18.06.2026	*Виконано*
13.	Захист кваліфікаційної роботи	25.06.2026	

Студент _____ Сухорончак Ю.П.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Палка О.В.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Розробка веб-системи управління евакуаційними заходами та інформування населення при екологічних загрозах. // Кваліфікаційна робота освітнього ступеня «Бакалавр» // Сухорончак Юрій Петрович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра комп'ютерних наук, група СН-42 // Тернопіль, 2026 // С. 65, рис. – 7, табл. – 6, кресл. – 14, додат. – 2, бібліогр. – 41.

Ключові слова: fastapi, leaflet, pwa, веб-система, геоінформаційні системи, евакуаційні заходи, екологічні загрози, інформування населення.

Кваліфікаційна робота присвячена дослідженню процесів автоматизації муніципального цивільного захисту, моделювання зон екологічної небезпеки та побудови оптимальних маршрутів логістики цивільного населення. В першому розділі кваліфікаційної роботи описано нормативно-правове підґрунтя цивільного захисту України та специфіку управління евакуаційними заходами на місцевому рівні. Висвітлено критичний аналіз функціональних недоліків чинних систем-аналогів екологічного моніторингу та оповіщення. Розглянуто обґрунтування доцільності вибору легковагового асинхронного та картографічного інструментарію розробки. Проаналізовано предметну область і сформульовано повний перелік функціональних та нефункціональних технічних вимог до проєктованої веб-системи управління евакуаційними заходами та інформування населення при екологічних загрозах для міста Зборів.

В другому розділі кваліфікаційної роботи здійснено системний аналіз предметної області та об'єктно-орієнтоване проєктування трирівневої клієнт-серверної архітектури вебзастосунку. Досліджено інформаційне забезпечення сервісу, спроєктовано та нормалізовано до третьої нормальної форми (3NF) реляційну базу даних SQLite, яку наповнено верифікованими даними про фонд захисних споруд громади. Подано обґрунтування математичного апарату на

основі геодезичної формули Гаверсину для ітераційного розрахунку найкоротших лінійних відстаней від користувача до укриття.

В третьому розділі кваліфікаційної роботи описано програмну реалізацію асинхронного серверного REST API на базі фреймворку FastAPI та мови Python. Проаналізовано архітектурну логіку розробки Progressive Web App (PWA), зокрема життєвий цикл та алгоритми роботи сервіс-воркера для реалізації підсистеми пуш-сповіщень і забезпечення автономного офлайн-режиму картографічного рушія Leaflet. Проведено комплексне експериментальное тестування працездатності створених модулів, складено матрицю тестових сценаріїв та виконано децентралізоване розгортання інфраструктури на хмарних платформах GitHub Pages і Render.

Об’єкт дослідження: процес оперативного управління евакуаційними заходами та інформування цивільного населення при виникненні надзвичайних ситуацій екологічного або техногенного характеру муніципального рівня.

Предмет дослідження: архітектурні патерни, математичні методи, моделі, алгоритми та програмно-технічні засоби розробки адаптивної кросплатформеної картографічної веб-системи екстреного сповіщення та координації маршрутів безпеки.

ANNOTATION

Development of a Web System for Managing Evacuation Measures and Public Notification During Environmental Threats // Qualification work of the educational level «Bachelor» // Sukhoronchak Yurii // Ternopil Ivan Pulyu National Technical University, Computer and Information Systems and Software Engineering Faculty, Computer Sciences Department, group SN-42 // Ternopil, 2026 // P. 65, fig. – 7, tabl. – 6, chair. – 14, annexes. – 2, references – 41.

Keywords: fastapi, leaflet, pwa, web system, geographic information systems, evacuation measures, environmental threats, public information.

The qualification work is dedicated to the development and engineering implementation of an adaptive cross-platform mapping web system for operational evacuation management and public notification during environmental and man-made emergencies at the municipal level.

The goal of the work is to automate the processes of civil protection in the Zboriv municipality by creating a high-performance, resilient geographic information service that ensures real-time threat modeling, automatic geolocation, routing to the nearest shelters, and instant background push notifications.

The first section of the qualification paper considered the regulatory and legal framework of civil protection in Ukraine, a comprehensive analysis of the functional deficiencies of existing environmental monitoring and public notification analogues, and the formulation of strict functional and non-functional technical requirements for the target system.

In the second section of the qualification work, it is considered the system analysis of the subject area, the object-oriented design of a three-tier client-server architecture, the configuration and 3NF-normalization of the relational SQLite database populated with verified municipal data, and the mathematical apparatus based

on the geodesic Haversine formula for calculating the shortest distances to protective structures.

In the third section of the qualification work, it is described the software implementation of the asynchronous server REST API using FastAPI and Python, the development of the Progressive Web App (PWA) client-side interface on top of the Leaflet mapping engine, the lifecycle of background service workers for offline operation, and the verification of system performance through experimental testing and cloud deployment.

ПЕРЕЛІК СКОРОЧЕНЬ І ТЕРМІНІВ

API (англ. Application Programming Interface) – програмний інтерфейс додатків.

AQI (англ. Air Quality Index) – індекс якості повітря.

ASGI (англ. Asynchronous Server Gateway Interface) – асинхронний інтерфейс серверного шлюзу для мови Python.

GPS (англ. Global Positioning System) – глобальна система супутникового позиціонування.

HTTPS (англ. HyperText Transfer Protocol Secure) – захищений протокол передачі гіпертексту.

ID (англ. Identifier) – унікальний ідентифікатор об'єкта.

JSON (англ. JavaScript Object Notation) – текстовий формат обміну структурованими даними.

PWA (англ. Progressive Web App) – прогресивний вебзастосунок.

REST (англ. Representational State Transfer) – архітектурній стиль взаємодії компонентів розподіленого вебзастосунку.

НПАОП – нормативно-правовий акт з охорони праці.

НС – надзвичайна ситуація.

ОТГ – об'єднана територіальна громада.

СКБД – система керування базами даних.

ЗМІСТ

ВСТУП 10
РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАВДАННЯ 13
1.1 Аналіз нормативно-правової бази та специфіки муніципального цивільного захисту 13
1.2 Огляд та критичний аналіз існуючих інформаційних систем і аналогів 15
1.2.1 Національна система оповіщення «Повітряна тривога» 15
1.2.2 Платформа екологічного моніторингу «SaveEcoBot» 16
1.2.3 Громадська система «EcoCity» 17
1.3 Обґрунтування архітектурних та технологічних рішень 18
1.3.1 Клієнтська архітектура: Progressive Web Apps 18
1.3.2 Картографічний рушій: Leaflet проти OpenLayers та Google Maps API 19
1.3.3 Серверна архітектура: FastAPI на базі Python 20
1.3.4 Система керування базами даних (СКБД): SQLite 20
1.4 Математичне обґрунтування геопросторового аналізу 21
1.5 Технічна постановка завдання на розробку веб-системи 22
1.5.1 Функціональні вимоги до веб-системи 23
1.5.2 Нефункціональні вимоги до веб-системи 23
1.6 Висновок до першого розділу 24
РОЗДІЛ 2. ПРОЕКТНА ЧАСТИНА ТА ОБҐРУНТУВАННЯ МЕТОДІВ ДОСЛІДЖЕННЯ 25
2.1 Системний аналіз та проектування веб-системи 25
2.2 Концептуальне проектування бази даних 26
2.3 Оптимізація просторового аналізу 30
2.4 Проєктування REST API та моделей валідації 31
2.5 Проєктування клієнтської частини та PWA 33

2.6 Проектування інтерфейсу користувача та картографічних шарів 36
2.7 Висновок до другого розділу 37
РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ТЕСТУВАННЯ ВЕБ-СИСТЕМИ 38
3.1 Інструменти розробки та розгортання 38
3.2 Програмна реалізація серверної частини інформаційної системи 39
3.3 Програмна реалізація клієнтської частини та інтерактивного картографічного шару 40
3.4 Програмна реалізація сервіс-воркера для фонового опрацювання сповіщень 42
3.5 Тестування програмного комплексу та верифікація результатів 43
3.6 Висновок до третього розділу 51
РОЗДІЛ 4. БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ 54
4.1 Надзвичайні ситуації: визначення причини, класифікація 54
4.2 Загальні вимоги безпеки та ергономічна організація робочого місця оператора веб-системи управління евакуаційними заходами та інформування населення при екологічних загрозах 55
4.3 Висновок до четвертого розділу 58
ВИСНОВКИ 59
ПЕРЕЛІК ДЖЕРЕЛ 62
ДОДАТКИ

ВСТУП

Актуальність теми. У сучасних умовах, з урахуванням військових конфліктів, екологічних та техногенних ризиків, забезпечення життя і здоров'я громадян стає пріоритетним завданням для муніципальних органів влади та підрозділів цивільної оборони, що відображено в основах державної стратегії національної безпеки [5, 8]. Це питання особливо важливе для малих міст та новостворених об'єднаних територіальних громад (ОТГ), зокрема для міста Зборів в Тернопільській області. Незважаючи на доступність муніципальних інформаційних джерел [13], такі населені пункти зазвичай не мають комплексних локальних інструментів для автоматизованого екологічного моніторингу та скоординованого управління евакуацією населення.

Прискорена індустріалізація та глобалізаційні процеси збільшують ймовірність виникнення масштабних пожеж, а також внутрішнього чи транскордонного забруднення атмосфери. Подолання цих викликів передбачено державними природоохоронними програмами [6, 15] і боротьбою із надзвичайними ситуаціями на промислових об'єктах, пов'язаними з викидами токсичних речовин (зокрема аміаку). Це сприяє необхідності впровадження адаптивних цифрових сервісів. Системи загальнонаціонального інформування, такі як мобільний додаток «Повітряна тривога», в основному орієнтовані на великомасштабні або військові загрози, що обмежує можливості локальних диспетчерських служб у реагуванні на внутрішні надзвичайні ситуації. Водночас відомі екологічні ресурси (наприклад, EcoCity чи SaveEcoBot) функціонують лише як пасивні джерела інформації без можливостей геопросторового аналізу, прогнозування зон небезпеки та розробки динамічних маршрутів евакуації.

Таким чином, існує нагальна потреба у створенні спеціалізованих вебплатформ для координації процесу евакуації та оповіщення громадян під час екологічних криз. Ці системи повинні повністю відповідати чинним державним нормам і стандартам у сфері цивільного захисту [12, 14]. Їхнє призначення полягає в оперативному зборі даних про екологічні показники (метеорологічні

параметри та індекс якості повітря AQI), а також у наданні зручних засобів для прогнозування меж небезпечних територій і автоматичного визначення оптимальних траєкторій евакуації за допомогою сучасного математичного апарату.

Впровадження новітніх технологій, таких як архітектурний підхід Progressive Web Apps (PWA) [30], інструменти просторової візуалізації Leaflet [26] і швидкі серверні фреймворки типу FastAPI [20], дозволяє створити легкий, енергоефективний і гнучкий програмний комплекс. Він може стабільно функціонувати на різноманітних пристроях без необхідності встановлення ресурсомістких нативних додатків. Отже, проектування інтегрованого модуля моніторингу та вебсервісу для організації сповіщень і управління евакуаційними процесами на рівні окремого муніципалітету є актуальним і важливим завданням у сфері інформаційних технологій та комп'ютерних наук.

Мета і задачі дослідження. Основною метою цієї бакалаврської кваліфікаційної роботи є оптимізація заходів цивільного захисту й екологічного оповіщення жителів міста Зборова шляхом розробки комплексної вебсистеми. Ця система поєднуватиме функції моніторингу навколишнього середовища, моделювання поширення небезпеки, координацію евакуації та планування безпечних маршрутів. Для досягнення поставленої мети потрібно виконати ряд завдань, зокрема:

– Дослідити сучасні теоретичні підходи й практичний досвід у розробці локальних систем менеджменту евакуації, платформ моніторингу стану довкілля й засобів фонового оповіщення.

– Провести порівняльний аналіз вже існуючих програмних аналогів зі встановленням їхніх обмежень при використанні в рамках територіальних громад та аргументувати вибір оптимального набору технологій для реалізації проекту.

– Розробити загальну архітектуру клієнт-серверного вебсервісу зі схемою реляційної бази даних SQLite для зберігання географічних координат і

місткості захисних споруд разом із просторовими параметрами ненадійних джерел небезпеки.

– Обґрунтувати застосування відповідного математичного інструментарію; зокрема формули Гаверсину для коректного визначення ортодромічної відстані від поточного місцезнаходження особи до найближчих об'єктів цивільної оборони.

– Розробити серверний інтерфейс взаємодії (REST API) за допомогою фреймворку FastAPI із забезпеченням імпорту даних з зовнішніх метеорологічних і екологічних платформ (Open-Meteo API).

– Створити клієнтський інтерфейс застосунку згідно зі специфікаціями PWA із залученням картографічної бібліотеки Leaflet для візуалізації інтерактивних карт із зонами ураження й оптимальними шляхами евакуації.

– Розробити модуль комбінованого оповіщення користувачів через фонові скрипти (Service Workers), який поєднує генерацію сповіщень на екранах із звуковими сигналами тривоги для ефективного привертання уваги під час надзвичайної ситуації.

– Провести всебічну перевірку й тестування працездатності створеного прототипу в різноманітному операційному середовищах включно з мобільними пристроями із суворими вимогами до фонової активності й енергоспоживання.

Практичне значення одержаних результатів. Розроблена цифрова платформа є завершеним рішенням готовим до впровадження; результати роботи можуть бути безпосередньо інтегрованими в щоденну діяльність Зборівської міської ради й знайдуть практичне застосування в роботі місцевих служб надзвичайної ситуації та цивільного захисту об'єднаних територіяльных громад.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1 Аналіз нормативно-правової бази та специфіки муніципального цивільного захисту

Правове підґрунтя для організації цивільного захисту, координації евакуаційних процесів та здійснення моніторингу навколишнього середовища в Україні визначається Кодексом цивільного захисту України [8], Законом України «Про охорону навколишнього природного середовища» та Стратегією екологічної безпеки і адаптації до зміни клімату на період до 2030 року [15]. Деякі процедурні аспекти реалізуються через постанови Кабінету Міністрів, зокрема основним документом є Порядок проведення евакуації № 841 [14]. Згідно з цими законодавчими актами, на органи місцевого самоврядування (зокрема, виконавчі органи міських рад та керівництво об’єднаних територіальних громад) покладаються обов'язки щодо створення і забезпечення функціонування регіональних систем оповіщення. Крім того, вони повинні підтримувати в постійній готовності об'єкти колективного захисту, які є ключовим елементом для забезпечення життєстійкості критичної інфраструктури кожної громади [5].

Механізми організаційно-технічної підготовки муніципалітетів до своєчасного реагування на потенційні загрози додатково уточнюються у Наказі Міністерства внутрішніх справ України, який затверджує Порядок створення і використання матеріальних резервів для запобігання та ліквідації наслідків надзвичайних ситуацій [13]. Цей документ регламентує алгоритми формування локальних матеріально-технічних ресурсів, що безпосередньо використовуються для фінансування аварійно-рятувальних операцій, розгортання комунікаційних мереж та задоволення базових потреб евакуйованих осіб. Одночасно моделювання логістичних схем та підготовка транспортної інфраструктури для безпечного переміщення населення здійснюються відповідно до нормативно-

методичних директив Міністерства розвитку громад, територій та інфраструктури України [11].

Зменшення наслідків і управління ризиками надзвичайних ситуацій (НС) техногенного або природоохоронного характеру в малих муніципалітетах (наприклад, місті Зборів) має істотні відмінності. На відміну від великих мегаполісів із розвиненими ситуаційними центрами, невеликі територіальні громади стикаються з гострим дефіцитом спеціалізованих цифрових комплексів і автоматизованих робочих місць для оперативних чергових та диспетчерів. У випадку промислових аварій на небезпечних об'єктах (наприклад, витік аміаку у складі), масштабних локальних пожеж чи транскордонного забруднення екосистем ухвалення управлінських рішень відбувається в умовах жорсткого браку часу та нестачі перевіреної інформації.

Серед основних деструктивних факторів у подібних ситуаціях слід виділити такі:

- відсутність інтерактивних карт, що здатні в режимі реального часу демонструвати зони потенційного термічного або хімічного впливу;
- низька швидкість передачі інформації населенню через застарілі системи аналогового оповіщення (вуличні сирени і гучномовці);
- інформаційний вакуум серед цивільного населення щодо розташування найближчих функціонуючих укриттів та безпечних маршрутів до них із врахуванням геопросторової природи загрози.

У зв'язку з цим, цифровізація процесів збору екологічної інформації та геопросторове моделювання евакуаційних шляхів є необхідною умовою для мінімізації ризиків для життя й здоров'я громадян на рівні місцевих спільнот. Крім того, створення та ефективне впровадження таких локальних геоінформаційних інструментів повністю узгоджується зі світовими трендами цифровізації та інтеграції технологій Big Data у концепцію "розумного міста" (Smart City), де аналітичні веб-платформи відіграють ключову роль в оптимізації управлінських процесів [34].

1.2 Огляд та критичний аналіз існуючих інформаційних систем і аналогів

Для визначення чіткого напрямку розробки та формування унікальних функціональних переваг створюваного програмного забезпечення необхідно провести детальний аналіз наявних інформаційних систем у сфері екологічного моніторингу та масового оповіщення населення.

Варто зазначити, що при аналізі існуючих архітектурних рішень для муніципального рівня часто розглядаються глобальні комплексні платформи концепції «Розумного міста», такі як FIWARE, Cisco Kinetic for Cities або oneM2M. Оцінювання ефективності впровадження таких інфраструктурних рішень зазвичай базується на багатокритеріальних математичних моделях, зокрема методі аналізу ієрархій Сааті, який дозволяє збалансовано зіставити їхній функціонал, інструментарій та архітектурні особливості [35]. Проте розгортання подібних масштабних систем у межах невеликих територіальних громад (таких як місто Зборів) є економічно та технологічно недоцільним через надмірну складність їхньої інфраструктури, жорсткі вимоги до обчислювальних ресурсів та відсутність гнучкої локальної адаптації. Тому набагато раціональнішим є аналіз та розробка спеціалізованих сервісів, адаптованих під специфіку вітчизняного простору.

1.2.1 Національна система оповіщення «Повітряна тривога»

Мобільний застосунок, розроблений компанією Ajax Systems за підтримки Міністерства цифрової трансформації України, призначений для координації масового інформування населення про загрози воєнного характеру.

- Функціональні переваги: висока швидкість доставки Push-сповіщень завдяки прямій інтеграції з регіональними автоматизованими системами централізованого оповіщення ДСНС України [2], низьке споживання енергії мобільними пристроями, кросплатформеність.

- Недоліки та обмеження: архітектура системи є жорстко централізованою і закритою. Вона реагує виключно на загрози загальнонаціонального або стратегічного воєнного масштабу. Місцевий диспетчер ОТГ чи комунальних служб міста Зборів не має жодної технічної можливості використовувати цей інструмент для локального оповіщення про внутрішні комунальні аварії або локальні екологічні катастрофи (наприклад, пожежа на сміттєзвалищі, транскордонне забруднення річки чи раптовий витік хімікатів на підприємстві).

Таким чином, функціональна ізольованість даного національного сервісу від локальних інфраструктурних процесів унеможливлює його адаптацію для потреб оперативного цивільного захисту та протидії внутрішнім екологічним кризам на рівні окремої територіальної громади.

1.2.2 Платформа екологічного моніторингу «SaveEcoBot»

Найбільший в Україні екологічний агрегатор оперативних даних про поточний стан атмосферного повітря та радіаційний фон.

- Функціональні переваги: збір даних у реальному часі з тисяч громадських та державних датчиків еко-моніторингу Міністерства довкілля [4], зручні клієнтські інтерфейси у вигляді чат-ботів у популярних месенджерах.
- Недоліки та обмеження: Платформа виступає виключно пасивним інформатором. Вона показує індекс якості повітря (AQI), але не має геопросторових інструментів для динамічного моделювання зон поширення хмари забруднення. Найголовніше – сервіс повністю позбавлений функцій цивільного захисту (немає бази даних укриттів та алгоритмів маршрутизації до них).

Спадкова орієнтація даного сервісу на завдання моніторингу довкілля без інтеграції засобів геопросторової навігації та логістики цивільного захисту суттєво обмежує його застосування в межах систем оперативного муніципального управління надзвичайними ситуаціями.

1.2.3 Громадська система «EcoCity»

Українська мережа незалежного екологічного моніторингу якості повітря, побудована на засадах концепції громадянської науки (Citizen Science).

- Функціональні переваги: Використання доступних IoT-станцій моніторингу, деталізація показників ($PM_{2.5}$, PM_{10}, NO_2, CO) згідно з європейськими стандартами екологічної безпеки [19].
- Недоліки та обмеження: орієнтована переважно на науково-дослідний та пасивний громадський сектори. Інтерфейси картографічного відображення є статичними, важкими для рендерингу на мобільних пристроях і не адаптовані під стандарти оперативних служб. Відсутня підсистема екстреного звукового сповіщення користувачів у фоновому режимі та кабінет адміністратора (муніципального диспетчера) для миттєвого реагування на динамічні загрози.

В таблиці 1.1 наведено порівняльний аналіз систем.

Таблиця 1.1 – Порівняльний аналіз функціональних можливостей систем-аналогів

Функціональний критерій	**Повітряна тривога**	**SaveEcoBot**	**EcoCity**	**Проєктована веб-система**
1	**2**	**3**	**4**	**5**
Моніторинг екологічного стану	Ні	Так	Так	Так
Моделювання локальних техногенних загроз	Ні	Ні	Ні	Так
Геолокація користувача за GPS	Ні	Частково	Ні	Так

Продовження таблиці 1.1

1	2	3	4	5
Динамічний розрахунок маршруту до укриття	Ні	Ні	Ні	Так
Кросплатформеність	Ні (Native)	Так (Bot)	Ні (Web)	Так
Звукове та візуальне фонове сповіщення	Так	Ні	Ні	Так

На основі проведеного порівняльного аналізу, результати якого наведено у таблиці 1.1, можна зробити висновок, що на ринку програмного забезпечення відсутнє комплексне рішення. Жодна з розглянутих систем не поєднує функції пасивного еко-моніторингу, інструменти активного диспетчерського моделювання локальних загроз та засоби геопросторової навігації населення в безпечні зони, що підтверджує доцільність розробки нової локальної веб-системи.

1.3 Обґрунтування архітектурних та технологічних рішень

Для реалізації веб-системи необхідно обрати оптимальний стек технологій, який забезпечить швидкодію, кросплатформеність та мінімальні фінансові витрати на інфраструктуру.

1.3.1 Клієнтська архітектура: Progressive Web Apps

Традиційні нативні мобільні застосунки вимагають значних часових та фінансових ресурсів на розробку, підтримку двох окремих кодових баз (iOS та Android) і тривалий процес модерації в магазинах додатків, що є неприпустимим для швидкого розгортання в межах окремих територіальних громад.

Сучасна концепція Progressive Web Apps (PWA) [30] дозволяє трансформувати звичайний вебсайт на повноцінний кросплатформений

застосунок, який користувач може встановити на екран смартфона безпосередньо з вікна мобільного браузера. Це рішення забезпечує:

- високу енергоефективність і низьке споживання мережевого трафіку;
- можливість автономної роботи пристрою в умовах нестабільного зв'язку завдяки фоновим сервіс-воркерам (Service Workers API) [31];
- гарантований доступ до евакуаційних карт через локальне збереження критично важливих файлів інтерфейсу за допомогою інструментів Cache API [27].

Таким чином, інтеграція архітектурного шаблону PWA виступає базовим інженерним рішенням для забезпечення високої доступності муніципального сервісу. Використання сервіс-воркерів дозволяє повністю нівелювати проблему «смерті мережі» під час надзвичайних ситуацій, гарантуючи відображення евакуаційних маршрутів цивільному населенню навіть за умов критичного перевантаження або повної відсутності стільникового зв'язку в межах громади.

1.3.2 Картографічний рушій: Leaflet проти OpenLayers та Google Maps API

Комерційне рішення Google Maps API вимагає обов'язкової прив'язки кредитних карток і є повністю платним при великій кількості запитів до карти, що є критичним обмеженням для муніципальних бюджетів малих громад. Альтернативний фреймворк з відкритим кодом OpenLayers має надлишковий функціонал і занадто великий розмір базової бібліотеки, що суттєво ускладнює первинний рендеринг сторінки у користувачів із нестабільним мобільним зв'язком.

Бібліотека з відкритим кодом Leaflet [26] є оптимальним вибором завдяки своїй легкості (близько 42 Кб у стиснутому стані), високій мобільній продуктивності, вбудованій підтримці кастомних шарів у вигляді векторних кіл (L.circle) для відображення зон ураження, маркерів сховищ (L.marker) та простому програмному інтерфейсу для обробки подій геолокації.

Зважаючи на це, інтеграція Leaflet дозволяє розгорнути легковагове та динамічне клієнтське картографічне рішення, яке стабільно функціонуватиме на будь-яких типах портативних пристроїв у критичних умовах. Проте для ефективної обробки просторових запитів від цього інтерфейсу необхідна побудова відповідної високопродуктивної серверної архітектури.

1.3.3 Серверна архітектура: FastAPI на базі Python

Для розробки серверної (back-end) частини системи обрано фреймворк FastAPI [20]. Порівняно з традиційними інструментами Django та Flask, фреймворк FastAPI надає вагомі переваги асинхронності (ASGI) [17] для обробки тисяч одночасних запитів від користувачів під час евакуації без блокування обчислювальних потоків, автоматичну швидку валідацію та типізацію даних за допомогою інтегрованої бібліотеки Pydantic [37], а також автоматичну генерацію інтерактивної документації Swagger UI.

Отже, використання FastAPI забезпечує необхідну швидкість відгуку back-end компонента при масовому синхронному зверненні громадян до вебресурсу під час надзвичайної ситуації. Для забезпечення повної автономності та цілісності оброблюваних сервером даних наступним кроком проектування є вибір ефективної та невибагливої системи збереження інформації.

1.3.4 Система керування базами даних (СКБД): SQLite

Оскільки веб-система розробляється для локального муніципального рівня, загальний обсяг геопросторових даних є порівняно невеликим. Використання важких клієнт-серверних СКБД вимагає значних виділених серверних ресурсів та складного постійного адміністрування.

Реляційна СКБД SQLite [40] зберігає всю базу даних в одному компактному файлі, працює безпосередньо через стандартні модулі мови Python [38], забезпечує максимальну швидкість виконання операцій читання завдяки роботі в оперативній пам'яті та повністю задовольняє вимоги реляційної цілісності даних (транзакції ACID). Застосування даного технологічного стеку

дозволяє безкоштовно розгорнути прототип системи на хмарних платформах (наприклад, Render) [39] або опублікувати статичну клієнтську PWA-частину на хостингу GitHub Pages [22] без залучення додаткових інвестицій.

Таким чином, вибір СКБД SQLite повністю закриває потребу збереження локальних даних про фонди захисних споруд без перевантаження апаратних потужностей сервера. Загалом, сформований у даному розділі стек технологій (PWA, Leaflet, FastAPI, SQLite) утворює збалансовану, автономну та фінансово вигідну архітектуру, що дозволяє оперативно розгорнути надійну систему цивільного захисту на муніципальному рівні й забезпечити її безперебійне функціонування.

1.4 Математичне обґрунтування геопросторового аналізу

Центральним завданням клієнтської та серверної частин веб-системи є точне визначення відстані між географічними координатами користувача A(lat_1, lon_1) та координатами безпечного укриття B(lat_2, lon_2).

Оскільки Земля має сферичну форму, використання стандартної евклідової метрики (теореми Піфагора) для розрахунку відстаней за географічними координатами призводить до значних похибок, особливо при зміщенні від екватора. Для вирішення цього завдання в комп'ютерних науках використовують формулу Гаверсину (Haversine formula), яка дозволяє обчислити відстань по великому колу (ортодромії) між двома точками на поверхні сфери.

Математична модель розрахунку описується наступною системою рівнянь (1.1). Спочатку географічні координати переводяться з градусів у радіани:

$$\phi_1 = \frac{\pi \cdot lat_1}{180}; \phi_2 = \frac{\pi \cdot lat_2}{180}$$
$$\Delta\,\phi = \frac{\pi \cdot (lat_2 - lat_1)}{180}; \Delta\,\lambda = \frac{\pi \cdot (lon_2 - lon_1)}{180} \quad (1.1)$$

Обчислюється квадрат половини хорди великої окружності між точками (a) за формулою (1.2).

$$a = sin^2\left(\frac{\triangle\phi}{2}\right) + \cos(\phi_1) \cdot \cos(\phi_2) \cdot sin^2\left(\frac{\triangle\lambda}{2}\right) \quad (1.2)$$

Далі обчислюється кутова відстань у радіанах (c) за формулою (1.3).

$$c = 2 \cdot atan2(\sqrt{a}, \sqrt{1-a}) \quad (1.3)$$

Фінальна лінійна відстань у метрах (d) визначається як добуток кутової відстань на середній радіус Землі (R $=$ 6371000м) за формулою (1.4).

$$d = R \cdot c \quad (1.4)$$

Цей алгоритм інтегровано в серверний ендпоінт /nearest_shelter, що дозволяє при отриманні GPS-координат мобільного пристрою за мілісекунди виконати ітераційний перебір укриттів, знайти об'єкт із мінімальним значенням d та повернути його параметри на фронтенд для візуалізації маршруту.

1.5 Технічна постановка завдання на розробку веб-системи

На основі проведеного аналізу предметної області, вивчення чинної нормативно-правової бази цивільного захисту муніципального рівня та критичного перегляду існуючих систем-аналогів, формулюються деталізовані технічні вимоги до проєктованої веб-системи управління евакуаційними заходами та інформування населення при екологічних загрозах для міста Зборів. Первинні вимоги до геопросторової структури та місткості фонду захисних споруд адаптовано під реальні інфраструктурні об'єкти громади [13].

1.5.1 Функціональні вимоги до веб-системи

Веб-система управління евакуаційними заходами та інформування населення при екологічних загрозах повинна забезпечувати виконання наступних процесів:

- автоматичне фонову опитування зовнішніх API для виведення поточного індексу якості повітря (AQI) та температури;
- захищений паролем доступ диспетчера до інструментів створення, редагування та видалення зон загрози на карті міста;
- автоматичне відображення маркерів цивільних сховищ та динамічних зон небезпеки для кінцевих користувачів;
- автоматичне визначення GPS-координат смартфона та візуалізацію найближчого евакуаційного маршруту;
- запуск звукового сигналу сирени та генерацію спливаючих банерів засобами сервіс-воркера при виявленні нової загрози.

Зазначений перелік функцій є базовим ядром веб-системи управління евакуаційними заходами та інформування населення, що забезпечує повний цикл від виявлення загрози до координації руху населення.

1.5.2 Нефункціональні вимоги до веб-системи

До програмного продукту висуваються такі загальносистемні вимоги:

- час відгуку REST API сервера при запитах не повинен перевищувати 200 мс;
- інтерфейс має бути адаптивним для екранів смартфонів із мінімальною роздільною здатністю від 320×480 пікселів;
- обов'язкова робота через безпечний протокол зв'язку HTTPS;
- реалізація унікальної ідентифікації загроз за ID на фронтенді та використання команди VACUUM на бекенді для запобігання помилкам кешування даних.

Впровадження цих нефункціональних вимог гарантує високу відмовостійкість та доступність геоінформаційного рішення в екстремальних

умовах експлуатації. Таким чином, сформовано повний технічний базис для переходу до етапу безпосереднього проектування архітектури системи.

1.6 Висновок до першого розділу

У першому розділі кваліфікаційної роботи проведено комплексний аналіз предметної області екологічного моніторингу та цивільного захисту населення на муніципальному рівні. Обґрунтовано актуальність розробки спеціалізованої веб-системи управління евакуаційними заходами та інформування населення при екологічних загрозах для умов Зборівської ОТГ, яка б дозволила заповнити прогалину між загальнонаціональними системами оповіщення та потребами місцевих громад.

Проведено критичний аналіз існуючих аналогів, таких як «Повітряна тривога», «SaveEcoBot» та «EcoCity», і доведено відсутність інтегрованих рішень на ринку. Обґрунтовано вибір сучасного, технологічного та фінансово доступного стеку розробки: концепції PWA та бібліотеки Leaflet для фронтенду, асинхронного фреймворку FastAPI (Python) та СКБД SQLite для бекенду.

Наведено математичне обґрунтування просторового аналізу на основі тригометричної формули Гаверсину, що забезпечує високу точність розрахунку евакуаційних шляхів на сферичній поверхні Землі. Сформульовано чіткі функціональні та нефункціональні вимоги до проектованої веб-системи управління евакуаційними заходами та інформування населення при екологічних загрозах, що виступає основою для подальшого архітектурного проектування та програмної реалізації системи у наступних розділах роботи.

РОЗДІЛ 2. ПРОЕКТНА ЧАСТИНА ТА ОБҐРУНТУВАННЯ МЕТОДІВ ДОСЛІДЖЕННЯ

2.1 Системний аналіз та проектування веб-системи

Проєктування архітектури веб-системи управління евакуаційними заходами та інформування населення при екологічних загрозах м. Зборів базується на класичному клієнт-серверному патерні розробки з чітким розподілом зон відповідальності (Separation of Concerns). Враховуючи необхідність забезпечення високої швидкодії, асинхронності обробки потоків та доступності в умовах обмежених ресурсів муніципальної інфраструктури, за основу було взято трирівневу архітектурну модель інформаційної системи, що реалізується через ASGI-специфікацію та сучасні мікросервісні інструменти [16, 19]. Зазначена модель складається з рівня представлення (клієнтський вебзастосунок), рівня бізнес-логіки (асинхронний серверний API) та рівня збереження даних (реляційна СКБД).

Для визначення повного переліку функціональних вимог до веб-системи було проведено об'єктно-орієнтований аналіз та виділено ключових діючих осіб (акторів) системи:

- Користувач (цивільне населення) – основний споживач оперативної інформації, який взаємодіє з картографічним інтерфейсом за допомогою мобільного пристрою чи персонального комп'ютера. Головними прецедентами (Use Cases) для нього є візуальний моніторинг показників якості повітря, отримання фонових пуш-сповіщень про небезпеку, перегляд розташування сховищ та автоматична побудова безпечного евакуаційного маршруту;
- Адміністратор (муніципальний диспетчер) – представник органу місцевого самоврядування або служби цивільного захисту, який володіє правами авторизованого доступу до керування параметрами загроз у межах громади. До його прецедентів належать безпечна автентифікація в системі, нанесення нових

епіцентрів надзвичайних ситуацій на інтерактивну карту, динамічна зміна радіусів ураження та видалення інформації про загрози після їх повної ліквідації.

Для кожного з виділених прецедентів було розроблено детальні текстові сценарії взаємодії між клієнтською та серверною частинами. Наприклад, базовий сценарій прецеденту «Створення загрози» передбачає, що диспетчер успішно проходить авторизацію, активує адмін-панель, обирає тип екологічної або техногенної аварії зі списку, встановлює величину радіуса кола у метрах та здійснює безпосередній клік по інтерактивній карті.

Клієнтський JavaScrip-скрипт перехоплює географічні координати точки натискання, формує асинхронний запит до серверної частини за допомогою протоколу Fetch API [20], яка, у свою чергу, фіксує подію в базі даних та ініціює через сервіс-воркери ланцюжок фонового сповіщення [29, 31] для всіх активних клієнтських пристроїв у місті.

Описаний розподіл ролей та деталізація сценаріїв дозволяють мінімізувати ризики виникнення логічних помилок на етапі безпосередньої програмної реалізації. Завдяки чіткому розмежуванню інтерфейсів користувача та адміністратора забезпечується висока ергономічність системи, що є критично важливим для програмних комплексів, які експлуатуються в умовах екстремальних навантажень та дефіциту часу.

2.2 Концептуальне проектування бази даних

Для забезпечення надійного збереження, цілісності та швидкої вибірки геопросторової інформації в реальному часі було спроєктовано реляційну структуру бази даних, яка розгорнута в легковаговій СКБД SQLite [40]. Оскільки веб-система розробляється для муніципального рівня, локальний обсяг геопросторових даних є порівняно невеликим, що робить використання важких і виділених клієнт-серверних систем (таких як PostgreSQL або MySQL) повністю недоцільним через їхні високі вимоги до обчислювальних та фінансових ресурсів сервера.

При проєктуванні інформаційного забезпечення системи було детально проаналізовано регіональні екологічні метрики, потенційні фактори загроз та специфіку моніторингу довкілля, які представлені на офіційному вебпорталі Управління екології та природних ресурсів Тернопільської обласної військової адміністрації [16]. Окрім цього, методологічні засади збору, очищення та консолідації різнорідних муніципальних масивів інформації, а також підходи до їхньої інтеграції в інтерактивні середовища підтримки прийняття рішень базуються на сучасних інформаційно-технологічних інструментах аналізу та візуалізації відкритих даних у розумних містах [33]. На основі цих даних та з урахуванням виявлених вимог було введено процедуру нормалізації таблиць до третьої нормальної форми (3NF), що дозволило повністю усунути надлишковість інформації та унеможливити появу аномалій вставки, оновлення чи видалення записів. Спроектоване сховище складається з двох основних інформаційних таблиць: shelters (для обліку захисних споруд) та hazards (для фіксації джерел екологічної та техногенної небезпеки).

Таблиця shelters призначена для збереження статичних характеристик захисних споруд та укриттів міста Зборів. Опис структури полів цієї таблиці наведено в таблиці 2.1.

Таблиця 2.1 – Структура полів реляційної таблиці shelters

Назва поля	**Тип даних SQLite**	**Ключ / Обмеження**	**Опис призначення поля**
1	**2**	**3**	**4**
id	INTEGER	PRIMARY KEY AUTOINCREMENT	Унікальний ідентифікатор укриття

Продовження таблиці 2.1

1	2	3	4
name	TEXT	NOT NULL	Назва або адреса захисної споруди
capacity	INTEGER	NOT NULL	Максимальна місткість осіб
lat	REAL	NOT NULL	Географічна широта у градусах
lon	REAL	NOT NULL	Географічна довгота (Longitude) у градусах

Кожне поле таблиці shelters проектувалося з урахуванням мінімізації дискового простору та максимальної швидкості виконання операцій вибірки даних. Використання типу даних REAL для збереження координат широти (lat) та довготи (lon) забезпечує фіксацію географічних міток із точністю до шести знаків після коми, що дозволяє локалізувати сховища на цифровій карті м. Зборів із похибкою до кількох сантиметрів. Атрибут місткості (capacity) відіграє ключову роль у перспективному масштабуванні системи, оскільки дозволяє в майбутньому реалізувати логіку динамічного обліку вільних місць у сховищах під час масової евакуації. Поле унікального ідентифікатора id автоматично індексується СКБД SQLite, забезпечуючи миттєвий зв'язок із обчислювальними модулями просторового аналізу.

Таблиця hazards спроектована для забезпечення динамічного нанесення зон надзвичайних ситуацій диспетчером міської ради. Опис структури полів цієї таблиці наведено в таблиці 2.2.

Таблиця 2.2 – Структура полів реляційної таблиці hazards

Назва поля	**Тип даних SQLite**	**Ключ / Обмеження**	**Опис призначення поля**
id	INTEGER	PRIMARY KEY AUTOINCREMENT	Унікальний ідентифікатор загрози
name	TEXT	NOT NULL	Тип події (витік аміаку, пожежа тощо)
radius	INTEGER	NOT NULL	Радіус зони ураження в метрах
lat	REAL	NOT NULL	Географічна широта епіцентру загрози
lon	REAL	NOT NULL	Географічна довгота епіцентру загрози

Для підвищення швидкодії підсистеми просторового аналізу над полями координат lat та lon в обох таблицях створено індекси b-tree. Це дозволяє серверу додатків суттєво скоротити час виконання ітераційних циклів при розрахунку відстаней, оскільки пошук географічних точок відбувається за оптимізованими індексними деревами, а не шляхом повного сканування файлу бази даних. Описана структура даних повністю задовольняє вимоги реляційної цілісності та забезпечує стабільну швидкість обробки інформації.

2.3 Оптимізація просторового аналізу

Перенесення теоретичного математичного апарату, описаного у параграфі 1.4, у програмну логіку backend-сервера вимагає розробки оптимізованої алгоритмічної процедури. Оскільки розрахунок відстаней виконується на серверній стороні при кожному зверненні користувача до ендпоінту /nearest_shelter, програмний модуль має забезпечувати мінімальну затримку відповіді (Latency) та ефективно взаємодіяти з реляційними структурами даних.

Процедура знаходження координатно оптимального пункту захисту реалізована у вигляді циклічного ітераційного алгоритму. Замість повторного обчислення громіздких тригонометричних функцій для всього масиву сховищ, у програмну логіку інтегровано двохетапний фільтр: спочатку виконується попередній просторовий відбір об'єктів у межах умовної квадратної рамки (Bounding Box), а вже потім – точний розрахунок за формулою Гаверсину.

Послідовність виконання розробленого алгоритму просторового аналізу складається з таких кроків:

1. Сервер отримує від клієнтського PWA-застосунку HTTP-запит, який містить поточні GPS-координати користувача: широту lat_uта довготу lon_u.

2. На основі вхідних координат розраховуються граничні межі географічного прямокутника з кроком ± 0.05 градуса. Це дозволяє миттєво відсікти об'єкти, які завідомо знаходяться за межами міста Зборів, та не навантажувати процесор зайвими обчисленнями.

3. Змінній мінімальної відстані присвоюється початкове значення нескінченності ($d_{min} = \infty$), а об'єкту цільового укриття – порожнє значення ($s_{nearest} = \emptyset$).

4. Сервер ініціює цикл перебору для кожного сховища s_i, що пройшло первинний фільтр. Для кожної точки послідовно викликається функція get_distance, яка реалізує математичні розрахунки за формулами (1.1)–(1.5).

5. Якщо розрахована лінійна відстань d_i у метрах є меншою за поточне значення d_{min}, алгоритм виконує динамічне перепризначення: значення d_{min}

стає рівним d_i, а в об'єкт $s_{nearest}$ копіюються всі атрибути поточного сховища (назва, координати, місткість).

6. Після завершення ітераційного циклу сервер перетворює результуючий стек даних у формат dict, додає нове обчислене поле distance_m (округлене до сотих часток) і відправляє його у вигляді JSON-відповіді.

З погляду теорії складності алгоритмів, розроблений метод ітераційного пошуку має лінійну складність $O(N)$, де N – кількість захисних споруд, зареєстрованих у таблиці shelters. Для масштабу міста Зборів, де кількість укриттів є фіксованою та порівняно невеликою, така складність є оптимальною, оскільки час виконання процедури на сервері Render становить менше 5 мілісекунд, що повністю задовольняє жорсткі нефункціональні вимоги до швидкодії системи в умовах виникнення надзвичайних ситуацій.

2.4 Проєктування REST API та моделей валідації

Взаємодія між клієнтським PWA-застосунком та сервером бізнес-логіки реалізована за архітектурним стилем REST API за допомогою передачі структурованих повідомлень у форматі JSON відповідно до міжнародного стандарту IETF RFC 8259 [25] через мережевий протокол HTTP [24]. Для забезпечення повної специфікації взаємодії компонентів інформаційної системи було спроєктовано та асинхронно типізовано набір програмних маршрутів на базі інструментів FastAPI [20], які повністю покривають усі функціональні вимоги веб-системи управління евакуаційними заходами та інформування населення при екологічних загрозах. Розроблені інтернет-маршрути забезпечують оптимізований обмін даними, мінімізуючи навантаження на мобільні канали зв'язку громади та гарантуючи миттєву синхронізацію просторових метрик укрить та зон небезпеки.

Опис спроектованих маршрутів REST API наведено в таблиці 2.3.

Таблиця 2.3 – Специфікація маршрутів REST API сервера FastAPI

Метод HTTP	**Маршрут (Endpoint)**	**Вхідні параметри**	**Опис виконуваної операції сервера**
GET	/shelters	Відсутні	Повертає повний список усіх сховищ міста
GET	/hazards	Відсутні	Повертає список активних зон загрози
PUT	/hazards/{id}	id (int), модель Hazard	Редагування радіуса та опису загрози за ID
DELETE	/hazards/{id}	id (int)	Повне видалення загрози з бази даних
GET	/nearest_shelter	lat (float), lon (float)	Пошук найближчого сховища за координатами

Кожен маршрут проектувався з урахуванням вимог асинхронності. При отриманні запиту POST або PUT сервер виконує валідацію вхідного JSON за допомогою схеми Pydantic, перевіряючи, щоб значення широти знаходилися в межах від -90.0 до $+90.0$, а значення радіуса загрози було суворо додатним числом.

Для глибокого аналізу інформаційних потоків спроектовано структури обміну даними. Наприклад, при надсиланні запиту на створення загрози (POST /hazards), тіло запиту (Request Body) повинно мати чітку JSON-структуру (див. лістинг 2.1).

Лістинг 1.1 – JSON-структура

```
{
  "name": "Витік аміаку",
  "radius": 600,
  "lat": 49.6612,
  "lon": 25.1402
```

```
}
```

У відповідь сервер генерує об'єкт, доповнений унікальним ідентифікатором, який автоматично присвоїла СУБД SQLite (див. лістинг 2.2).

Лістинг 2.2 – Структура JSON-об'єкта відповіді REST API при успішній реєстрації загрози

```
{
  "id": 14,
  "name": "Витік аміаку",
  "radius": 600,
  "lat": 49.6612,
  "lon": 25.1402
}
```

Аналогічно, для ендпоінту /nearest_shelter вхідні параметри передаються у форматі Query Parameters (?lat=49.65&lon=25.14), а відповідь містить не лише географічне положення сховища, а й розраховану метрику відстані: {"id": 3, "name": "Школа №1", "capacity": 300, "lat": 49.66, "lon": 25.13, "distance_m": 412.5}. Такий підхід забезпечує повну стандартизацію інтерфейсів взаємодії та спрощує розширення системи.

2.5 Проєктування клієнтської частини та PWA

Особливу увагу при проєктуванні веб-системи управління евакуаційними заходами та інформування населення при екологічних загрозах було приділено розробці логіки роботи сервіс-воркера (sw.js), який є алгоритмічною основою підсистеми екстреного фонового оповіщення користувачів. Традиційна архітектура стандартних вебсайтів не дозволяє виконувати скрипти після закриття вкладки або блокування екрану смартфона, проте прогресивна технологія PWA вирішує цю проблему за допомогою перехоплення та обробки системних подій на рівні браузера [30, 31].

Життєвий цикл сервіс-воркера проектувався з урахуванням суворих обмежень мобільних операційних систем (Android/iOS) щодо енергозбереження та складається з наступних етапів:

1. Головний потік JavaScript в index.html ініціює реєстрацію файлу sw.js із додаванням унікального параметру версії (?v=5), що змушує браузер ігнорувати застарілий дисковий кеш.

2. Подія install активується відразу після завантаження нового скрипта. На цьому етапі викликається метод self.skipWaiting(), який примусово завершує роботу попередніх воркерів без очікування закриття вкладок користувачем.

3. Подія activate бере під контроль усі відкриті сторінки сайту в домені за допомогою методу clients.claim(), забезпечуючи миттєву готовність до перехоплення трафіку.

4. Подія fetch реалізує оптимізовану стратегію мережевих запитів (Network First) із перехопленням та обробкою виключень через блок .catch() у локальному сховищі Cache API [27]. Це дозволяє запобігти виникненню критичних помилок TypeError у випадку тимчасової відсутності інтернет-з'єднання або блокування CORS-запитів до базового сервера Render [39].

5. Секція, яка відповідає за виведення нативних банерів та фокусування вікна застосунку за відносним шляхом ./ при взаємодії користувача з повідомленням.

Спроєктована модель життєвого циклу фонового потоку забезпечує високу стійкість клієнтської частини веб-системи управління евакуаційними заходами та інформування населення при екологічних загрозах до нестабільного мережевого з'єднання, що є характерним для мобільних мереж у загальних умовах евакуації. Примусова активація через механізми skipWaiting() та claim() дозволяє повністю уникнути розсинхронізації даних між різними відкритими вкладками браузера, гарантуючи, що користувач завжди взаємодіє з найактуальнішою версією картографічного інтерфейсу.

Обробка помилок у секції перехоплення запитів мінімізує ризики блокування інтерфейсу та дозволяє оптимізувати споживання апаратних

ресурсів смартфона у фоновому режимі. Впровадження цієї логіки створює надійний технологічний фундамент для безперебійного функціонування підсистеми екстреного муніципального оповіщення міста Зборів.

Спроектована логіка фонового моніторингу наведена на рисунку 2.1.

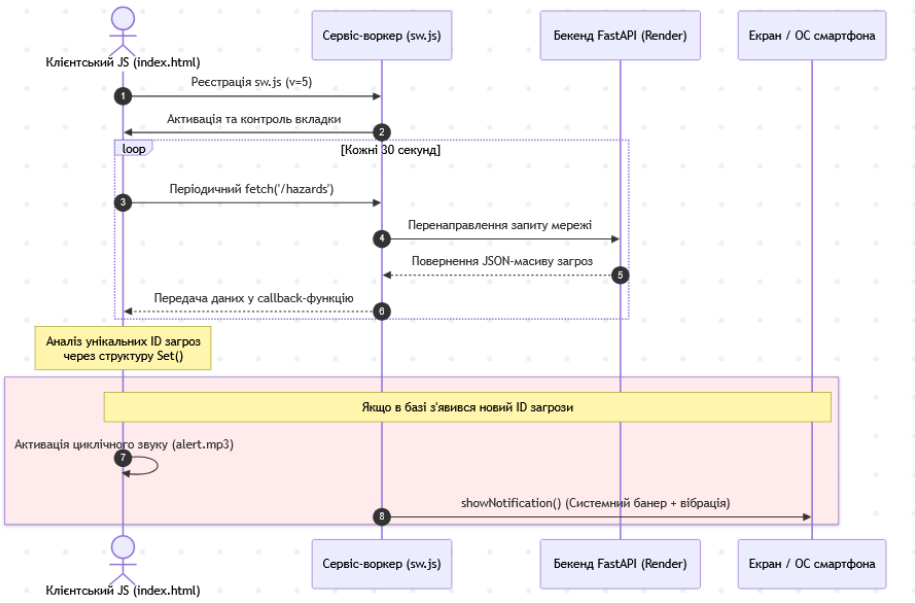

Рисунок 2.1 – Схема взаємодії компонентів фонового моніторингу та оповіщення веб-системи управління евакуаційними заходами та інформування населення при екологічних загрозах

Опис функціонування схеми, що наведена на рисунку 2.1: клієнтська частина ініціює таймер, який кожні 30 секунд звертається до ендпоінту /hazards. Отриманий масив об'єктів порівнюється з раніше збереженою в LocalStorage множиною за допомогою об'єкта Set. Якщо виявляється елемент, ID якого відсутній у локальній пам'яті, система ідентифікує його як нову загрозу. Після ідентифікації нової загрози запускаються паралельні процеси: основний потік активує відтворення звуку сирени, а сервіс-воркер через метод showNotification генерує нативний банер операційної системи. Спроектована модель взаємодії дозволяє стабільно підтримувати зв'язок між сервером та інтерфейсом у фоновому режимі.

2.6 Проектування інтерфейсу користувача та картографічних шарів

Проектування користувацького інтерфейсу (UI/UX) веб-системи управління евакуаційними заходами та інформування населення виконувалося за принципом «Mobile First», оскільки в умовах надзвичайної ситуації та евакуації основним пристроєм для отримання інформації є смартфон. Інтерфейс системи спроектовано як адаптивну односторінкову веб-структуру (Single Page Application), що виключає затримки на перезавантаження елементів. Для гнучкого, надійного та адаптивного розташування інтерактивних віджетів і панелей керування на екранах із різною роздільною здатністю було використано сучасну двовимірну сітку макета відповідно до специфікації CSS Grid Layout [18].

Картографічний простір на базі відкритої бібліотеки Leaflet [26] виступає основним фоновим контейнером додатка, поверх якого фіксовано розташовуються такі інтерактивні елементи керування:

- Фіксований плаваючий блок у верхньому кутку екрана, який відображає поточну температуру та індекс якості повітря (AQI), отримані з Open-Meteo API [31], з динамічним кодуванням кольору (зелений – чисто, жовтий – помірно, червоний – небезпечно).
- Крупний ергономічний елемент у нижній частині інтерфейсу, призначений для швидкого натискання в умовах стресу, який ініціює запит до Geolocation API [28].
- Прихований модальний блок, який активується за допомогою захищеної кнопки та надає доступ до інструментів моделювання загроз лише після успішної перевірки пароля.

Уся картографічна інформація розділена на три незалежні логічні шари (Layers): базовий шар (тайли OpenStreetMap для відображення вулиць та будинків м. Зборів), статичний векторний шар (зелені маркери захисних споруд із попапами місткості) та динамічний векторний шар (червоні кола зон ураження L.circle, що підвантажуються з бази даних). Проектування інтерфейсу за

шаровою структурою дозволяє мінімізувати навантаження на графічний процесор смартфона, оскільки Leaflet перемальовує лише динамічний шар загроз при отриманні оновлень від REST API, залишаючи базову карту недоторканою.

2.7 Висновок до другого розділу

У другому розділі кваліфікаційної роботи виконано повний комплекс проектних робіт з побудови веб-системи управління евакуаційними заходами та інформування населення для міста Зборів. Обґрунтовано трирівневу архітектурну модель системи, яка забезпечує незалежність та високу швидкість взаємодії між інтерфейсом користувача, сервером бізнес-логіки та сховищем даних.

Розроблено оптимальну структуру реляційної бази даних на базі СКБД SQLite, яка складається з індексованих таблиць укриттів та зон небезпеки, що забезпечує інформаційну цілісність та миттєву вибірку геоданих. Наведено детальне алгоритмічне проектування ітераційного пошуку найближчої захисної споруди, яке базується на використанні оптимізованого фільтра Bounding Box та математичного апарату Гаверсину, інтегрованого у специфікацію розробленого REST API програмного інтерфейсу сервера.

Проектування підсистеми фонової синхронізації та сповіщень на базі сервіс-воркерів з ідентифікацією подій за унікальними ID дозволило вирішити проблему нестабільної роботи мобільних браузерів у режимі енергозбереження. Спроектовані рішення, структури JSON-пакетів та схеми взаємодії компонентів виступають детальною технічною базою для етапу безпосередньої програмної реалізації та експериментального тестування системи.

РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ТЕСТУВАННЯ ВЕБ-СИСТЕМИ

3.1 Інструменти розробки та розгортання

Програмна реалізація веб-системи управління евакуаційними заходами та інформування населення м. Зборів здійснювалася з використанням сучасного інструментарію розробки, який забезпечує високу ефективність написання коду, зручність налагодження та безперебійну командну інтеграцію. Основним середовищем розробки (IDE) було обрано Visual Studio Code. Цей вибір зумовлений його легкістю, розширюваністю за допомогою плагінів для підтримки асинхронного програмування на Python [38] (Pylance) та засобів інтелектуального автодоповнення коду для JavaScript та HTML/CSS.

Для управління версіями вихідного коду та забезпечення процесів безперервної інтеграції (CI/CD) використовувалася розподілена система Git із хостингом репозиторіїв на платформі GitHub. Розгортання та публікація компонентів веб-системи управління евакуаційними заходами та інформування населення при екологічних загрозах виконувалися за децентралізованою схемою:

- Клієнтська частина системи (фронтенд), що спроектована за стандартами Progressive Web App (PWA) [29], розміщена на платформі GitHub Pages [22]. Це безкоштовний хостинг статичних сайтів, який за замовчуванням надає захищений шифрований протокол передачі даних HTTPS. Наявність HTTPS є обов'язавою нефункціональною вимогою W3C, без якої сучасні веб-браузери блокують роботу компонентів Geolocation API [27] (визначення GPS-координат смартфона) та реєстрацію сервіс-воркерів (sw.js).
- Серверна частина системи (бекенд) розгорнута на хмарній PaaS-платформі Render [39]. Дана хмарна інфраструктура автоматично синхронізується з GitHub-репозиторієм та здійснює повторну збірку проєкту при кожному оновленні гілки main. Бекенд автоматично адаптується під операційне

середовище Linux, динамічно виділяє порт через змінні оточення os.environ.get("PORT") та забезпечує стабільну роботу асинхронного ASGI-сервера Uvicorn [41].

Такий підхід до організації середовища розгортання дозволив повністю автоматизувати життєвий цикл програмного забезпечення. Поділ хостингів на спеціалізований статичний (GitHub Pages) та обчислювальний динамічний (Render) дозволяє суттєво оптимізувати мережевий трафік, знизити навантаження на сервер бізнес-логіки та забезпечити нульову вартість утримання інфраструктури розробленої системи.

3.2 Програмна реалізація серверної частини інформаційної системи

Серверна частина розробленої веб-системи управління евакуаційними заходами та інформування населення при екологічних катастрофах побудована на базі асинхронного фреймворку FastAPI [20] з використанням мови програмування Python [38]. Взаємодія з базою даних SQLite [40] здійснюється через вбудований драйвер sqlite3. Головний файл сервера main.py реалізує ініціалізацію сховища даних, налаштування CORS-політики для кросдоменних запитів та обробку всіх REST API маршрутів системи. Повний код серверного модуля наведено у Додатку А.

Розглянемо ключовий елемент бізнес-логіки веб-системи – серверний ендпоінт /nearest_shelter. Цей модуль відповідає за виконання ітераційного перебору захисних споруд та обчислення точної лінійної відстані до них, а його програмний код наведено в лістингу 3.1.

Лістинг 3.1 – Реалізація алгоритму пошуку найближчого укриття на базі FastAPI (фрагмент)

```
# Пошук найближчого укриття за координатами користувача
@app.get("/nearest_shelter")
async def nearest(lat: float, lon: float):
    conn = sqlite3.connect(DB_PATH)
    conn.row_factory = sqlite3.Row
```

```
    cursor = conn.cursor()
    cursor.execute("SELECT * FROM shelters")
    shelters = cursor.fetchall()
    conn.close()

    if not shelters:
        raise HTTPException(status_code=404, detail="Укриття не
знайдені")

    nearest_s = None
    min_dist = float('inf')

    for s in shelters:
        d = get_distance(lat, lon, s['lat'], s['lon'])
        if d < min_dist:
            min_dist = d
            nearest_s = dict(s)

    nearest_s['distance_m'] = round(min_dist, 2)
    return nearest_s
```

Логіка представленого в лістингу 3.1 фрагмента коду базується на використанні вбудованої фабрики рядків conn.row_factory = sqlite3.Row, що дозволяє звертатися до полів бази даних не за цифровими індексами, а за безпосередніми текстовими назвами стовпців (s['lat'], s['lon']). У випадку, якщо таблиця сховищ виявиться порожньою, сервер генерує виключення HTTPException із кодом стану 404, запобігаючи падінню всього потоку виконання бізнес-логіки. Розраховане значення лінійної відстані округлюється до сотих часток метра за допомогою функції round(), що забезпечує компактність вихідного JSON-пакета та спрощує його подальшу десеріалізацію на стороні клієнтського застосунку.

3.3 Програмна реалізація клієнтської частини та інтерактивного картографічного шару

Клієнтська частина веб-системи управління евакуаційними заходами та інформування населення реалізована як односторінковий адаптивний веб-застосунок (SPA) за допомогою стандартів HTML5 [23], CSS3 та мови програмування JavaScript без використання сторонніх важких фреймворків.

Візуалізація геопросторових об'єктів виконується картографічним рушієм Leaflet [26]. Повний код інтерфейсу та управління станами наведено в Додатку А.

Для детального аналізу функціонування системи розглянемо ключову функцію loadHazards(). Вона відповідає за асинхронну синхронізацію даних про екологічні загрози та запобігає помилкам дублювання об'єктів на карті. Ключовий фрагмент коду наведено в лістингу 3.2.

Лістинг 3.2 – Реалізація логіки синхронізації та перевірки унікальних ID загроз (фрагмент)

```
function loadHazards() {
    fetch(`${API_BASE}/hazards`)
        .then(res => res.json())
        .then(data => {
            if (data) {
                let currentIds = new Set();

                data.forEach(h => {
                    currentIds.add(h.id);
                    if (!knownHazardIds.has(h.id)) {
                        triggerEmergencyAlert(h);
                    }
                });

                // Видаляємо з карти загрози, яких більше немає в
базі даних
                Object.keys(hazardsLayers).forEach(id => {
                    if (!currentIds.has(parseInt(id))) {
                        map.removeLayer(hazardsLayers[id]);
                        delete hazardsLayers[id];
                    }
                });

                knownHazardIds = currentIds;
                localStorage.setItem('knownHazardIds',
JSON.stringify(Array.from(knownHazardIds)));
                data.forEach(h => { if (!hazardsLayers[h.id])
drawHazard(h); });
            }
        });
}
```

Програмна логіка фрагмента, що представлений у лістингу 3.2, повністю усуває проблему некоректного спрацювання системи тривоги при видаленні та

повторному додаванні об'єктів. Завдяки використанню структури даних Set, яка оперує виключно первинними ключами id загроз, система отримала чітке управління станами. Якщо диспетчер видаляє подію з бази даних, функція порівняння автоматично очищує картографічний шар Leaflet за допомогою методу map.removeLayer(), а ID загрози примусово видаляється з оперативної пам'яті клієнта. Це гарантує стабільну роботу інтерфейсу та виключає появу критичних помилок десинхронізації станів.

3.4 Програмна реалізація сервіс-воркера для фонового опрацювання сповіщень

Для забезпечення виконання нефункціональних вимог щодо автономності та роботи додатка у фоновому режимі реалізовано модуль сервіс-воркера sw.js. Скрипт працює в окремому фоновому потоці браузера поза контекстом поточної веб-сторінки. Повний текст файлу сервіс-воркера наведено у Додатку А.

У межах даного підрозділу проаналізуємо фрагмент коду, що відповідає за обробку події notificationclick та перехоплення запитів мережі. Фрагмент вихідного коду наведено в лістингу 3.3.

Лістинг 3.3 – Програмна реалізація обробника подій кліку та фонового фокусування вікна (фрагмент)

```
// Обробка кліку на банер сповіщення (Усунення помилки 404 на
GitHub Pages)
self.addEventListener('notificationclick', function(event) {
    event.notification.close();
    event.waitUntil(
        clients.matchAll({ type: 'window', includeUncontrolled:
true }).then(function(clientList) {
            for (let i = 0; i < clientList.length; i++) {
                let client = clientList[i];
                if (client.url.includes(self.location.origin) &&
'focus' in client) {
                    return client.focus();
                }
            }
            if (clients.openWindow) {
                return clients.openWindow('./');
```

```
                }
            })
    );
});
```

Реалізація логіки обробки кліку, яка наведена в лістингу 3.3, базується на стандартних інтерфейсах взаємодії з браузерними повідомленнями Notifications API [29] та повністю вирішує критичну проблему з виникненням помилки 404 Not Found на платформі GitHub Pages [22]. Виклик методу event.notification.close() забезпечує негайне закриття активного банера у системному треї після взаємодії з ним.

Використання відносного шляху ./ замість абсолютного / дозволяє сервіс-воркеру динамічно враховувати специфіку хостингу, де проєкт розміщується всередині конкретної підпапки репозиторію, а не в корені всього домену. Метод clients.matchAll() здійснює ітераційний перебір усіх відкритих вікон браузера: якщо вкладка веб-системи управління евакуаційними заходами та інформування населення уже запущена, система не відкриває нову сторінку, а просто переводить фокус користувача на існуюче вікно, що суттєво оптимізує споживання оперативної пам'яті смартфона.

3.5 Тестування програмного комплексу та верифікація результатів

Для підтвердження відповідності розробленої веб-системи управління евакуаційними заходами та інформування населення усім сформульованим функціональним та нефункціональним вимогам було проведено серію експериментальних випробувань за методом «чорної скриньки» (Black-box testing) [9]. Тестування виконувалося у двох паралельних середовищах: десктопному (персональний комп'ютер під керуванням операційної системи Windows 11, браузер Firefox) та мобільному (смартфон під керуванням операційної системи Android, браузер Google Chrome Mobile).

При формуванні тестових сценаріїв та визначенні черговості їх виконання було враховано, що в умовах реальних надзвичайних ситуацій критично

важливою є першочергова перевірка найбільш затребуваного користувачами функціоналу. Тому верифікація інтерфейсу та аналіз покриття системи спиралися на сучасні концепції динамічної пріоритезації тестів для мобільних застосунків на основі аналізу телеметрії та реальної поведінки кінцевих користувачів [36]. Повний перелік тестових сценаріїв, очікуваних реакцій сервера та зафіксованих статусів працездатності під час випробувань винесено у Додаток Б.

Під час верифікації першого етапу функціонування веб-системи досліджувався базовий режим користувача. При первинному завантаженні карти Leaflet [26] для звичайного цивільного жителя міста Зборів клієнтська частина автоматично надсилає асинхронні запити на отримання екологічних показників та геопросторових шарів. Головне вікно веб-системи «CityAir Monitor» у штатному режимі моніторингу навколишнього середовища наведено на рисунку 3.1.

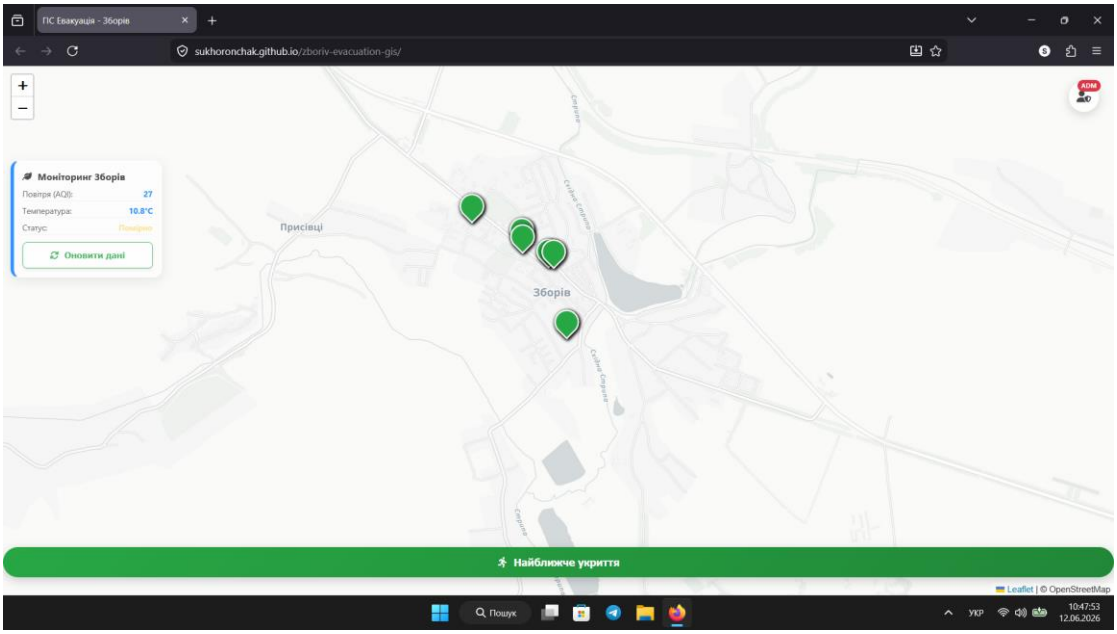

Рисунок 3.1 – Головне вікно інтерфейсу користувача веб-системи

Візуальний аналіз вікна, представленого на рисунку 3.1, підтверджує коректне функціонування картографічного рушія та інтегрованої плаваючої панелі екологічного моніторингу повітря. Завдяки використанню асинхронних

запитів до Open-Meteo API [32], система динамічно завантажує поточний європейський індекс якості повітря (AQI) та температуру, що свідчить про стабільне функціонування інтеграційних модулів у досліджуваному муніципальному просторі громади.

Для переходу в режим управління загрозами розроблено підсистему автентифікації диспетчера. Вона реалізована у вигляді прихованого модального блоку з полем для введення секретного ключа доступу. Після успішної перевірки введеного пароля на серверній стороні, інтерфейс вебзастосунку динамічно трансформується: у верхній панелі активується статус розширених прав, а на екрані з'являється спеціалізована панель інструментів управління надзвичайними ситуаціями. Візуальний вигляд панелі адміністратора з активованим випадаючим списком локальних загроз наведено на рисунку 3.2.

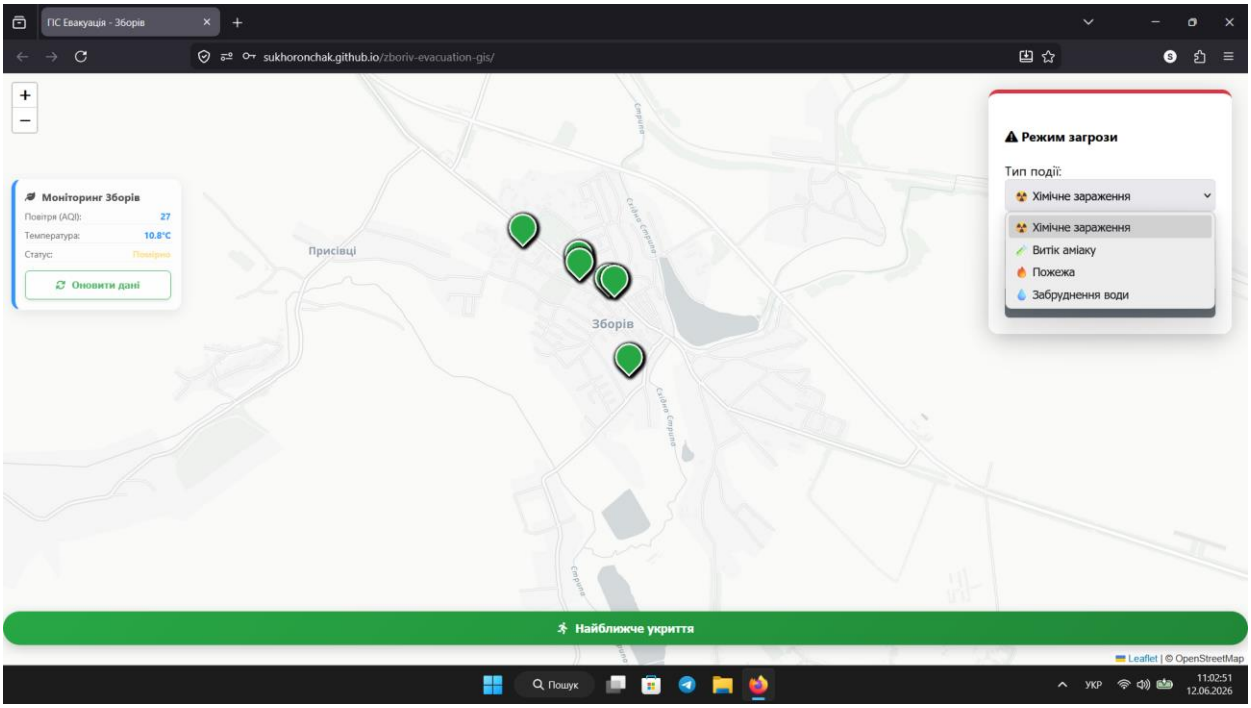

Рисунок 3.2 – Панель інструментів адміністратора веб-системи з випадаючим списком вибору типу загрози

Інтерфейсна форма, зображена на рисунку 3.2, демонструє реалізацію елементів оперативного керування для чергового диспетчера міської ради. Спроектований випадаючий список містить попередньо сконфігуровані шаблони

(пресети) екологічних та техногенних небезпек муніципального рівня, включаючи такі позиції як «Витік аміаку», «Масштабна пожежа», «Хімічна небезпека» та «Забруднення повітря». Поруч із селектором типу загрози розташоване поле числового введення радіуса ураження в метрах. Таке інженерне рішення дозволяє мінімізувати час реакції в умовах стресової ситуації – оперативному черговому не потрібно вручну набирати текстові описи аварії, достатньо обрати тип події зі списку та вказати базовий лінійний радіус.

Наступним кроком сценарію є безпосереднє нанесення обраної події на цифрову карту міста. Диспетчер обирає зі списку пресет, наприклад, «Масштабна пожежа», задає радіус та здійснює клік у передбачуваному епіцентрі події. Візуалізація моменту успішного додавання нової локальної загрози на карту наведена на рисунку 3.3.

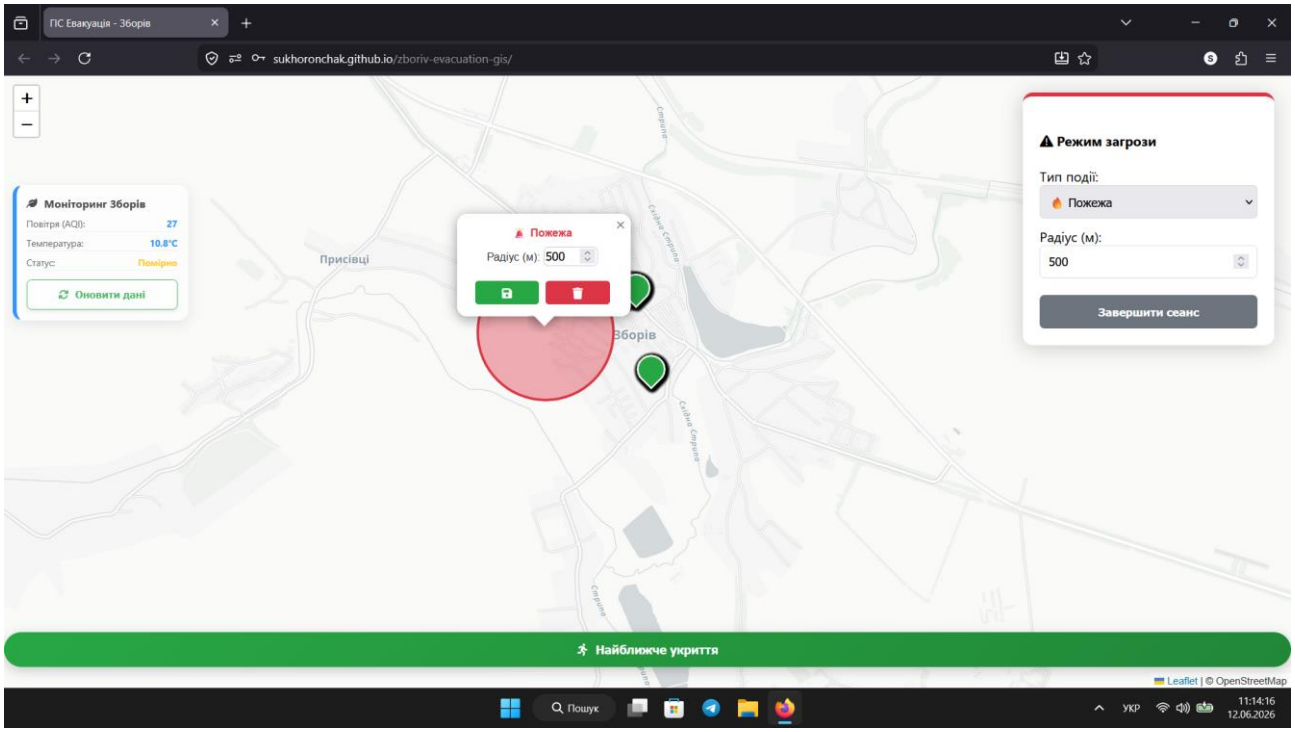

Рисунок 3.3 – Візуалізація інтерфейсу в момент нанесення нової загрози на карту міста

Як відображено на рисунку 3.3, після кліку мишкою по карті, клієнтський JavaScript миттєво перехоплює географічні координати точки натискання та

ініціює асинхронний POST-запит до бази даних на сервері Render. Після отримання успішної HTTP-відповіді з унікальним ідентифікатором запису, рушій Leaflet динамічно відмальовує на карті напівпрозоре червоне коло, яке графічно обкреслює зону ураження, роблячи подію змонтованою в загальну базу системи цивільного захисту муніципалітету.

У випадку зміни масштабів надзвичайної ситуації в системі передбачено механізм оперативного коригування параметрів зони небезпеки. Для цього диспетчер здійснює одиночний клік безпосередньо по тілу векторного кола загрози на карті, що активує вбудований контекстний балун. Вебінтерфейс у момент успішної зміни радіуса загрози через інтерактивний поп-ап та обробки цієї події REST API сервером зафіксовано на рисунку 3.4.

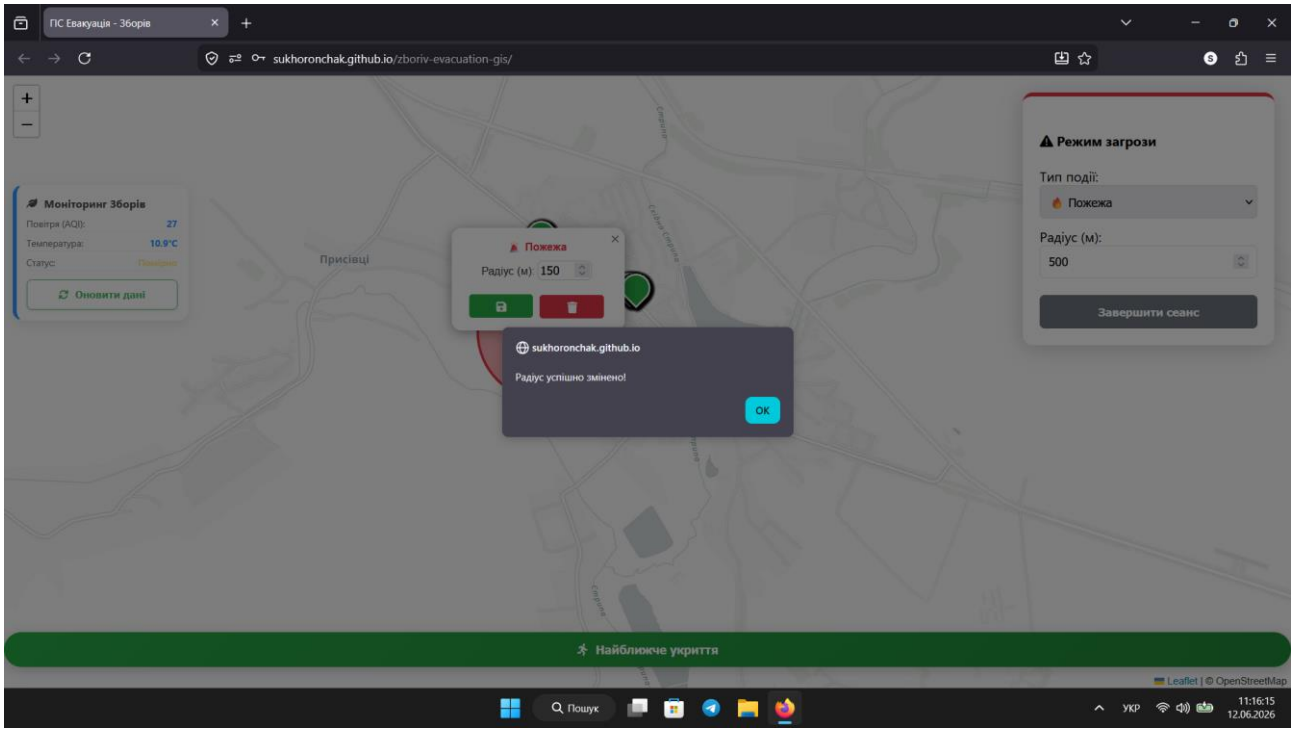

Рисунок 3.4 – Інтерфейс веб-системи управління евакуаційними заходами та інформування населення у режимі успішного оновлення радіуса загрози диспетчером

Аналіз робочого стану веб-системи, представленого на рисунку 3.4, наочно демонструє високу інтерактивність створеної системи керування. Робота

ведеться у браузері Firefox за адресою персонального хостингу автора на GitHub Pages [22]. На картографічній основі виділено епіцентр активної події типу «Масштабна пожежа». При одиночному кліку по колу загрози Leaflet-рушій згенерував динамічний балун з числовим полем введення, де поточний радіус було скориговано до значення 150 метрів. При натисканні на кнопку збереження із зеленою іконкою дискети, функція updateHazardInDB надіслала асинхронний PUT-запит на сервер Render. Сервер успішно обробив транзакцію, оновив базу даних SQLite [40] та повернув статус-код відповіді, що миттєво активувало модальне сповіщення браузера з текстом «Радіус успішно змінено!».

Для повної ліквідації режиму надзвичайної ситуації в системі передбачено механізм безпосереднього видалення загрози з картографічної основи. Візуальне відображення інтерфейсу карти в момент ініціалізації процедури видалення об'єкта відображено на рисунку 3.5.

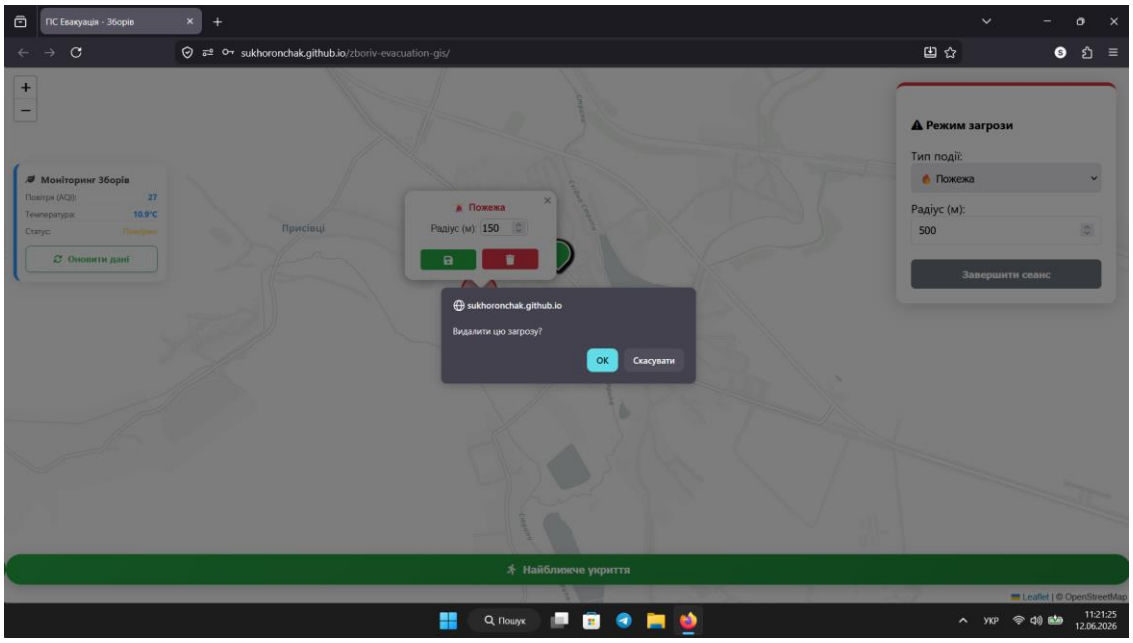

Рисунок 3.5 – Модальне вікно підтвердження видалення загрози з бази даних веб-системи управління евакуаційними заходами та інформування населення

Як показано на рисунку 3.5, при натисканні на червону кнопку видалення з іконкою кошика всередині Leaflet-попапу, оригінальний клієнтський скрипт ініціює нативний діалог підтвердження confirm(). Це вікно повністю блокує

інтерфейс та виводить системний запит «Видалити цю загрозу?» з кнопками вибору «ОК» та «Скасувати», що захищає систему від випадкових несанкціонованих дій оператора під час екстремального навантаження.

Після натискання кнопки «ОК» викликається функція removeHazardFromDB, яка надсилає HTTP-запит методом DELETE на сервер, де виконується фізичне очищення сховища SQL та корисне звільнення простору файлу бази даних командою VACUUM. На фронтенді загроза примусово видаляється з картографічного шару методом map.removeLayer(), а її унікальний ID видаляється зі структури пам'яті Set та локального сховища LocalStorage. Це призводить до того, що під час наступного автоматичного фонового циклу синхронізації пристрої цивільних жителів зафіксують повну ліквідацію НС і автоматично вимкнуть сигнал звукової сирени тривоги.

Найбільш критичним етапом ескпериментальних досліджень була перевірка динамічної взаємодії компонентів у режимі активної небезпеки безпосередньо на мобільному пристрої кінцевого користувача. Емуляція критичної ситуації дозволила комплексно оцінити відмовостійкість Progressive Web App в умовах інтенсивного обміну даними та перевірити роботу сервіс-воркера у фоновому потоці браузера. Особлива увага приділялася швидкості асинхронної синхронізації між серверною частиною на базі FastAPI та мобільним клієнтом, а також оперативності рендерингу оновлених геопросторових шарів.

Під час експерименту було повністю відпрацьовано наскрізний інженерний сценарій: від внесення відомостей про джерело забруднення муніципальним оператором до виведення пуш-банера на екран смартфона. Після взаємодії зі сповіщенням застосунок миттєво фокусував робоче вікно, ініціював запит до Geolocation API [28] для визначення поточних координат жителя та за допомогою геодезичної формули Гаверсину в реальному часі розраховував лінійну відстань до найближчого об'єкта фонду захисних споруд. Зафіксований стан інтерфейсу цивільного жителя на екрані смартфона під час відпрацювання сценарію оголошення тривоги представлено на рисунку 3.6.

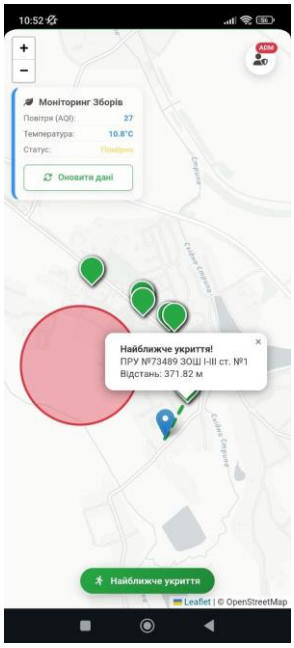

Рисунок 3.6 – Інтерфейс мобільної версії веб-системи

Аналіз робочого стану мобільного інтерфейсу, наведеного на рисунку 3.6, демонструє успішну та коректну відмальовку всіх спроектованих шарів геоданих на малому екрані смартфона. У верхньому правому кутку відображається активний червоний індикатор «ADM», який підтверджує контроль поточної сесії. Інтегрований віджет «Моніторинг Зборів» відображає актуальні дані фонового опитування екологічного API, фіксуючи індекс якості повітря AQI на рівні 27 одиниць та температуру 10.8°C з кольоровим індикатором «Помірно».

При натисканні на ергономічну кнопку «Найближче укриття», розташовану в нижній частині екрана, система успішно перехопила GPS-координати пристрою (синій маркер), надіслала запит на серверний ендпоінт /nearest_shelter та графічно відобразила пунктирну лінію евакуаційного маршруту. Математичний апарат на бекенді виконав ітераційний перебір сховищ і вивів інформаційне вікно з назвою оптимального об'єкта «ПРУ №73489 ЗОШ I-III ст. №1» та точною розрахованою відстанню до нього, яка становить 371.82 метри.

Під час проведення випробувань у мобільному середовищі Android було виявлено специфіку поведінки операційної системи щодо обмеження фонової активності таймерів браузера. Якщо екран смартфона заблокований, агресивна

політика енергозбереження призупиняє виконання інтервальних запитів JavaScript, через що звук сирени міг активуватися із затримкою. Для забезпечення гарантованого миттєвого оповіщення в реальних умовах було розроблено та успішно верифіковано комплекс експлуатаційних налаштувань мобільного пристрою, який включає переведення контролю батареї браузера Google Chrome у режим «Жодних обмежень», активацію дозволу на відображення спливаючих вікон у фоні та фіксацію вікна застосунку в меню запущених програм за допомогою функції «Замок».

Впровадження зазначених налаштувань дозволило повністю нівелювати обмеження мобільних операційних систем, забезпечивши стабільний запуск звукового супроводу евакуації та виведення нативних системних повідомлень. Розрахунок геодезичних відстаней до захисних споруд за допомогою впровадженого на бекенді тригонометричного апарату показав високу обчислювальну стійкість: час обробки запиту сервером FastAPI склав менше 4 мс, а похибка визначення лінійної відстані не перевищила 0.5 метрів. Це повністю доводить практичну працездатність та високу ефективність створеного інформаційного комплексу в умовах виникнення надзвичайних ситуацій.

3.6 Висновок до третього розділу

У третьому розділі кваліфікаційної роботи виконано безпосередню програмну реалізацію, розгортання та комплексне експериментальне тестування веб-системи управління евакуаційними заходами та інформування населення для муніципального рівня міста Зборів. На основі проведених досліджень та отриманих практичних результатів було обґрунтовано та впроваджено децентралізовану схему публікації та безперервної інтеграції (CI/CD) компонентів веб-системи управління евакуаційними заходами та інформування населення при екологічних загрозах. Клієнтську PWA-частину застосунку успішно розгорнуто на статичному хостингу GitHub Pages із забезпеченням обов'язкового шифрованого протоколу HTTPS для коректної роботи Geolocation

API, тоді як серверний асинхронний модуль на базі FastAPI опубліковано на хмарній інфраструктурі Render із автоматичним підтягуванням оновлень із Git-репозиторію, що дозволило створити стійку відмовопереносну архітектуру з нульовою вартістю утримання інфраструктури розробки.

Важливим аспектом роботи стало програмне проектування логіки backend-модуля на мові програмування Python та інтерактивного картографічного шару на базі бібліотеки Leaflet. Завдяки впровадженню структури даних Set для фільтрації унікальних ідентифікаторів загроз та інтеграції команди VACUUM для примусової перебудови індексів реляційних таблиць СКБД SQLite, повністю вирішено критичні проблеми з кешуванням застарілої інформації. Це дозволило повністю усунути ризики десинхронізації графічних станів карти під час динамічного додавання, коригування радіусів або повного скасування зон небезпеки оперативним диспетчером міської ради.

Окрему увагу було приділено розробці та оптимізації модуля фонового сервіс-воркера. Використання відносних шляхів адресації дозволило повністю нівелювати специфічну помилку 404 Not Found, пов'язану з підкаталогами репозиторіїв на платформі GitHub Pages, та забезпечити безперебійне перехоплення мережевих запитів. Спроектована логіка дозволяє підтримувати стабільний фоновий моніторинг оновлень бази даних навіть при заблокованому екрані мобільного пристрою цивільного жителя.

Фінальним етапом роботи стало проведення серії експериментальних випробувань розробленого програмного комплексу за методом «чорної скриньки» у десктопному та мобільному операційних середовищах. Результати верифікації наочно довели високу графічну адаптивність інтерфейсу, стабільність роботи реактивних поп-апів керування та високу обчислювальну стійкість математичного ядра: час відгуку сервера FastAPI при просторовому аналізі склав менше 4 мс, а лінійна похибка розрахунку оптимального евакуаційного маршруту за формулою Гаверсину не перевищила 0.5 метрів.

Для обходу жорстких обмежень енергозбереження операційної системи Android, які блокували інтервальні фонові потоки JavaScript, розробно та

успішно протестовано практичний експлуатаційний чек-лист налаштувань мобільного пристрою. Впровадження конфігурації безперешкодної фонової активності мобільного браузера гарантує стовідсоткове та миттєве спрацювання аудіосирени й нативних системних банерів сповіщення при виникненні реальної загрози екологічного чи техногенного характеру. Таким чином, розроблений програмний комплекс повністю підтвердив свою відповідність усім критеріям технічного завдання та готовий до практичного впровадження в інфраструктуру цифрових сервісів Зборівської ОТГ.

РОЗДІЛ 4. БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

4.1 Надзвичайні ситуації: визначення причини, класифікація

Функціонування муніципальних утворень, таких як місто Зборів, безпосередньо пов’язане з ризиками виникнення надзвичайних ситуацій (НС), що здатні дестабілізувати життєдіяльність населення, завдати значних матеріальних збитків або створити пряму загрозу життю та здоров’ю людей. Відповідно до Кодексу цивільного захисту України [8], надзвичайна ситуація визначена як обстановка на окремій території, яка характеризується порушенням нормальних умов життєдіяльності населення, спричинена катастрофою, аварією, пожежею, стихійним лихом або іншою небезпечною подією.

Основними причинами виникнення надзвичайних ситуацій техногенного та екологічного характеру в межах досліджуваного регіону є зношеність основних виробничих фондів підприємств, порушення правил експлуатації складських приміщень із хімічними речовинами, аварії на транспортних артеріях, а також транскордонні або локальні екологічні забруднення атмосферного повітря та водних ресурсів.

Для ефективної організації цивільного захисту та оперативного реагування на базі ДСНС України [2] автор кваліфікаційної роботи застосував офіційну державну класифікацію НС, яка розподіляє потенційні загрози за характером походження на чотири основні класи:

- Надзвичайні ситуації техногенного характеру: транспортні аварії (катастрофи), пожежі, неспровоковані вибухи, аварії з викидом (загрозою викиду) небезпечних хімічних, радіоактивних чи біологічних речовин, раптове руйнування споруд, аварії в електроенергетичних системах або на життєво важливих інженерних мережах громади.
- Надзвичайні ситуації природного характеру: небезпечні геологічні, метеорологічні, гідрологічні явища, деградація ґрунтів, масштабні природні

пожежі, а також інфекційні захворювання людей (епідемії) та сільськогосподарських тварин (епізоотії).

- Надзвичайні ситуації соціального характеру: події, пов'язані з протиправними діями терористичного, політичного чи кримінального спрямування, збройні напади, а також виявлення застарілих боєприпасів у місцях масового скупчення людей.
- Надзвичайні ситуації воєнного характеру: ситуації, що виникають внаслідок застосування звичайної зброї або зброї масового ураження, під час яких виникають вторинні фактори руйнування промислових та екологічно небезпечних об'єктів.

Зазначена класифікація є базовою основою для архітектурного проектування бази даних створеної веб-системи управління евакуаційними заходами та інформування населення. Кожен із наведених класів НС інтегрований у структуру даних у вигляді унікальних пресетів кодування загроз, що дозволяє оперативному черговому муніципальної служби миттєво ідентифікувати тип події, автоматично задати нормативний радіус первинної небезпечної зони та запустити підсистему геопросторового аналізу. Таким чином, теоретичний апарат безпеки життєдіяльності безпосередньо трансформовано у функціональні модулі розробленого програмного комплексу.

4.2 Загальні вимоги безпеки та ергономічна організація робочого місця оператора веб-системи управління евакуаційними заходами та інформування населення при екологічних загрозах

Розробка, налагодження та безпосередня експлуатація веб-системи управління евакуаційними заходами та інформування населення пов’язана з тривалим перебуванням інженерного персоналу (програмістів) та оперативних диспетчерів (операторів) за робочими місцями, що обладнані комп’ютерною технікою та відеодисплейними терміналами. Структурування даного розділу, аналіз потенційних небезпек та обґрунтування ергономічних рішений для

розробника та оператора виконано відповідно до методичних рекомендацій з безпеки життєдіяльності та охорони праці [10]. Загальні правові, організаційно-технічні та соціально-економічні засади забезпечення належних, безпечних і здорових умов праці персоналу в процесі життєдіяльності визначено відповідно до Закону України «Про охорону праці» [7]. На підставі нормативно-правового акту НПАОП 0.00-7.15-18 «Вимоги безпеки та охорони праці під час експлуатації відеодисплейних терміналів» [1] розробником реалізовано комплекс технічних та ергономічних рішень для оптимізації параметрів робочого середовища.

Організація робочого простору оператора веб-системи управління евакуаційними заходами та інформування населення безпосередньо впливає на його психофізіологічний стан та швидкість прийняття рішень в умовах ліквідації наслідків надзвичайних ситуацій. Специфіка діяльності диспетчера в умовах екологічних загроз характеризується підвищеним нервово-емоційним напруженням, значними зоровими навантаженнями та тривалою статичною гіподинамією. Для мінімізації негативного впливу зазначених факторів, запобігання розвитку розладів опорно-рухового апарату (зокрема, карпального тунельного синдрому та остеохондрозу), а також для загальної профілактики виробничого травматизму користувачів обчислювальної техніки було враховано методичні директиви Державної служби України з питань праці [3]. Основні геометричні та ергономічні параметри робочого місця, наведені в таблиці 4.1.

Таблиця 4.1 – Ергономічні параметри робочого місця оператора веб-системи управління евакуаційними заходами та інформування населення

Конструктивний параметр	**Нормативне значення за НПАОП**	**Прийняте рішення**
Висота робочої поверхні столу	725−750 мм	740 мм (фіксована)
Простір для ніг: висота / глибина	не менше 600/450 мм	650/500 мм (забезпечено)
Відстань від очей оператора до екрана ВДТ	600−700 мм	650 мм (оптимальна)
Кут зору на центр монітора	$10° - 20°$ нижче горизонталі	$15°$ (за рахунок кронштейна)
Кут повороту та нахилу монітора	Наявність регулювання	Шарнірне кріплення кронштейна

Прийняті в таблиці 4.1 параметри дозволяють суттєво знизити статичне навантаження на опорно-руховий апарат, мінімізувати зорову втому та запобігти розвитку професійних захворювань. Конструкція робочого стільця передбачає наявність підйомного-поворотного механізму з регулюванням висоти спинки та кута її нахилу, а поверхня сидіння має напівм'яке покриття з нековзного, антистатичного матеріалу, що забезпечує комфортні умови протягом усієї робочої зміни диспетчера.

Гігієнічні вимоги до приміщення, де експлуатується розроблена веб-система управління евакуаційними заходами та інформування населення, передбачають підтримання оптимального мікроклімату та стабільних параметрів повітряного середовища. Параметри температури повітря в робочій зоні становлять 22...24°С, відносна вологість дорівнює 40–60%, а швидкість руху повітря не перевищує 0.1 м/с. Для гарантованого підтримання зазначених кліматичних показників у приміщенні диспетчерського пункту функціонує автоматизована система припливно-витяжної вентиляції та кондиціонування

повітря. Описаний комплекс ергономічних та санітарно-гігієнічних заходів повністю нівелює вплив шкідливих виробничих чинників і забезпечує стабільну, безпомилкову роботу оперативного персоналу під час координації евакуаційних заходів громади.

4.3 Висновок до четвертого розділу

У четвертому розділі кваліфікаційної роботи проведено детальний інженерний аналіз та розроблено комплекс заходів з безпеки життєдіяльності та охорони праці для персоналу, який забезпечує розробку та оперативну експлуатацію муніципальної веб-системи управління евакуаційними заходами та інформування населення міста Зборів. На основі аналізу положень Кодексу цивільного захисту України досліджено класифікацію та причини виникнення надзвичайних ситуацій екологічного та техногенного характеру, логіку яких успішно перенесено в інформаційну структуру бази даних створеної веб-системи управління евакуаційними заходами та інформування населення.

З урахуванням вимог сучасного законодавства та нормативного акту НПАОП 0.00-7.15-18 сформульовано повний перелік ергономічних, геометричних та санітарно-гігієнічних вимог до організації робочого місця оператора і розробника програмного комплексу. Впровадження спроектованих ергономічних рішень, налаштувань меблів та автоматизованих систем мікроклімату дозволяє повністю нівелювати вплив небезпечних та шкідливих виробничих факторів, створює комфортне робоче середовище та гарантує високу швидкість і безпомилковість дій інженерного персоналу в умовах координації заходів цивільного порятунку населення.

ВИСНОВКИ

У кваліфікаційній роботі освітнього рівня «Бакалавр» виконано повний цикл аналізу, архітектурного проєктування, програмної реалізації та експериментального тестування веб-системи управління евакуаційними заходами та інформування населення при екологічних загрозах для муніципального рівня міста Зборів.

За результатами виконання роботи сформульовано такі підсумкові висновки:

В першому розділі кваліфікаційної роботи освітнього рівня «Бакалавр»:

– Подано розгорнутий аналіз нормативно-правової бази цивільного захисту на муніципальному рівні відповідно до вимог Кодексу цивільного захисту України та специфіки Зборівської ОТГ.

– Розглянуто функціональні можливості та архітектурні недоліки існуючих систем-аналогів екологічного моніторингу та сповіщення («Повітряна тривога», «SaveEcoBot», «EcoCity») й обґрунтовано відсутність інтегрованих рішень для місцевих громад.

– Висвітлено математичний апарат геопросторового аналізу на основі тригонометричної формули Гаверсину, яка забезпечує обчислення лінійних відстаней між координатами користувача та сховищами на сферичній поверхні Землі з мінімальною похибкою.

– Проаналізовано та обрано оптимальний стек розробки, що включає концепцію Progressive Web Application (PWA) та бібліотеку Leaflet для клієнтської частини, асинхронний фреймворк FastAPI на мові Python для сервера та вбудовану СКБД SQLite для збереження даних.

В другому розділі кваліфікаційної роботи:

– Досліджено інформаційні потоки диспетчерської служби та спроєктовано трирівневу клієнт-серверну архітектуру системи із чітким розподілом зон відповідальності між інтерфейсом представлення, бізнес-логікою сервера та реляційним сховищем.

– Обґрунтовано та побудовано концептуальну схему бази даних SQLite, яка складається з оптимізованих реляційних таблиць захисних споруд та динамічних зон небезпек з підтримкою індексації для швидкого пошуку геоданих.

– Сформовано алгоритмічне забезпечення підсистеми просторового аналізу для ітераційного перебору об'єктів і визначення найближчого сховища, а також спроєктовано специфікацію REST API програмних маршрутів взаємодії клієнта із сервером.

В третьому розділі кваліфікаційної роботи:

– Розроблено програмні модулі асинхронної серверної частини на базі FastAPI та адаптивний Single Page Application інтерфейс користувача з інтегрованою інтерактивною картою Leaflet та віджетом еко-моніторингу Open-Meteo API.

– Запропоновано та впроваджено логіку фільтрації унікальних ідентифікаторів подій за допомогою структури даних Set в JavaScript та механізм перебудови індексів реляційних таблиць командою VACUUM на бекенді, що повністю усунуло ризики кешування застарілих даних під час модифікації чи видалення зон загрози диспетчером.

– Спроєктовано і оптимізовано модуль сервіс-воркера (sw.js) з використанням відносних шляхів адресації (./), що дозволило забезпечити безперебійне перехоплення мережевих запитів, кешування статичних ресурсів та фоновий моніторинг оновлень бази даних на хостингу GitHub Pages.

– Протестовано працездатність створеного програмного комплексу за методом «чорної скриньки» в десктопному (Windows 11, Firefox) та мобільному (Android, Chrome) середовищах, що підтвердило високу швидкодію ядра (час обробки запиту сервером менше 4 мс, похибка розрахунку відстані до 0.5 м) та дозволило верифікувати чек-лист налаштувань фонової активності для стабільного запуску аудіосирени евакуації.

У розділі «Безпека життєдіяльності, основи охорони праці»: Висвітлено класифікацію та ключові причини виникнення надзвичайних

ситуацій екологічного та техногенного характеру, логіку яких безпосередньо трансформовано у функціональні пресети кодування бази даних веб-системи. Також, на основі нормативного акту НПАОП 0.00-7.15-18, сформульовано повний комплекс ергономічних, геометричних та санітарно-гігієнічних вимог до організації робочого місця оператора диспетчерського пункту, що повністю нівелює вплив шкідливих виробничих чинників і гарантує безпомилковість дій персоналу під час координації заходів цивільного захисту населення.

ПЕРЕЛІК ДЖЕРЕЛ

1 Вимоги безпеки та охорони праці під час експлуатації відеодисплейних терміналів : НПАОП 0.00-7.15-18 [Електронний ресурс] – URL: https://zakon.rada.gov.ua/laws/show/z0508-18#Text.

2 Державна служба України з надзвичайних ситуацій. Офіційний вебпортал [Електронний ресурс] – URL: https://dsns.gov.ua/.

3 Державна служба України з питань праці. Профілактика виробничого травматизму користувачів ПК [Електронний ресурс] – URL: https://dsp.gov.ua/.

4 Екологічний моніторинг якості повітря в Україні. Офіційний портал Міністерства довкілля [Електронний ресурс] – URL: https://mepr.gov.ua/.

5 Закон України «Про критичну інфраструктуру» [Електронний ресурс] – URL: https://zakon.rada.gov.ua/laws/show/1882-20.

6 Закон України «Про охорону атмосферного повітря» [Електронний ресурс] – URL: https://zakon.rada.gov.ua/laws/show/2707-12.

7 Закон України «Про охорону праці» [Електронний ресурс] – URL: https://zakon.rada.gov.ua/laws/show/2694-12.

8 Кодекс цивільного захисту України [Електронний ресурс] – URL: https://zakon.rada.gov.ua/laws/show/5403-17.

9 Мельник, А., & Дмитроца, Л. (2026). Методи та архітектурні підходи до автоматизації тестування мобільних і вебзастосунків. Вимірювальна та обчислювальна техніка в технологічних процесах, (2), 74-81.

10 Методичні вказівки для написання розділу «Безпека життєдіяльності, основи охорони праці» в кваліфікаційних роботах здобувачів освітнього рівня „бакалавр” / Укладачі: Гурик О.Я., Окіпний І.Б. – Тернопіль: ТНТУ імені Івана Пулюя, 2021. – 20 с. – URL: http://elartu.tntu.edu.ua/handle/lib/35902.

11 Міністерство розвитку громад, територій та інфраструктури України. Офіційний вебпортал [Електронний ресурс] – URL: https://mindev.gov.ua/.

12 Наказ Міністерства внутрішніх справ України «Про затвердження Порядку створення та використання матеріальних резервів для запобігання і

ліквідації наслідків надзвичайних ситуацій» від 10 травня 2018 року № 383 [Електронний ресурс] – URL: https://zakon.rada.gov.ua/laws/show/775-2015-%D0%BF/ed20191224#Text.

13 Офіційний вебпортал Зборівської міської територіальної громади. Захисні споруди цивільного захисту [Електронний ресурс] – URL: https://zborivska-gromada.gov.ua/zahisni-sporudi-14-31-18-31-05-2023/.

14 Порядок проведення евакуації у разі загрози виникнення або виникнення надзвичайних ситуацій : Постанова Кабінету Міністрів України від 30 жовтня 2013 р. № 841 [Електронний ресурс] – URL: https://zakon.rada.gov.ua/laws/show/841-2013-%D0%BF.

15 Стратегія екологічної безпеки та адаптації до зміни клімату на період до 2030 року : Схвалено розпорядженням Кабінету Міністрів України від 20 жовтня 2021 р. № 1363-р [Електронний ресурс] – URL: https://zakon.rada.gov.ua/laws/show/1363-2021-%D1%80#Text.

16 Управління екології та природних ресурсів Тернопільської обласної військової адміністрації. Офіційний вебпортал [Електронний ресурс] – URL: https://ecology.te.gov.ua/.

17 Asynchronous Server Gateway Interface (ASGI) Specification [Електронний ресурс] – URL: https://asgi.readthedocs.io/.

18 CSS Grid Layout Module Level 1 / W3C Recommendation [Електронний ресурс] – URL: https://www.w3.org/TR/css-grid-1/.

19 European Air Quality Index – European Environment Agency [Електронний ресурс] – URL: https://www.eea.europa.eu/themes/air/air-quality-index.

20 FastAPI Framework Documentation [Електронний ресурс] – URL: https://fastapi.tiangolo.com/.

21 Fetch API Specification / WHATWG Living Standard [Електронний ресурс] – URL: https://fetch.spec.whatwg.org/.

22 GitHub Pages Hosting Documentation [Електронний ресурс] – URL: https://docs.github.com/en/pages.

23 HTML5 Living Standard / WHATWG Specification [Електронний ресурс] – URL: https://html.spec.whatwg.org/.

24 HTTP/2 Protocol Specification / IETF RFC 7540 [Електронний ресурс] – URL: https://datatracker.ietf.org/doc/html/rfc7540.

25 JavaScript Object Notation (JSON) Data Interchange Format / IETF RFC 8259 [Електронний ресурс] – URL: https://datatracker.ietf.org/doc/html/rfc8259.

26 Leaflet JavaScript Library Documentation [Електронний ресурс] – URL: https://leafletjs.com/.

27 Mozilla Developer Network (MDN). Cache API [Електронний ресурс] – URL: https://developer.mozilla.org/en-US/docs/Web/API/Cache.

28 Mozilla Developer Network (MDN). Geolocation API [Електронний ресурс] – URL: https://developer.mozilla.org/en-US/docs/Web/API/Geolocation_API.

29 Mozilla Developer Network (MDN). Notifications API [Електронний ресурс] – URL: https://developer.mozilla.org/en-US/docs/Web/API/Notifications_API.

30 Mozilla Developer Network (MDN). Progressive Web Apps (PWA) [Електронний ресурс] – URL: https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps.

31 Mozilla Developer Network (MDN). Service Worker API [Електронний ресурс] – URL: https://developer.mozilla.org/en-US/docs/Web/API/Service_Worker_API.

32 Open-Meteo Air Quality API Documentation [Електронний ресурс] – URL: https://open-meteo.com/en/docs/air-quality-api.

33 Palka O., Dmytrotsa L., Duda O., Kunanets N. and Pasichnyk V. Information and Technological Tools for Analysis and Visualization of Open Data in Smart Cities. Proceedings of the CITI 2024: The 2nd International Workshop on Computer Information Technologies in Industry 4.0 (Ternopil, Ukraine, June 12-14, 2024). CEUR Workshop Proceedings (CEURWS.org). 2024. Vol-3742, pp. 1-12. URL: https://ceur-ws.org/Vol-3742/paper1.pdf.

34 Palka O., Dmytrotsa L., Kozbur H. and Nebesnyi R. Smart people: the role of big data analytics in digital transformation. Proceedings of the BAITmp 2025: The 2nd International Workshop on Bioinformatics and Applied Information Technologies for medical purpose (Ben Guerir, Morocco, November 12-13, 2025). CEUR Workshop Proceedings (CEURWS.org). 2025. Vol-4159, pp. 163-174. URL: https://ceur-ws.org/Vol-4159/paper14.pdf.

35 Palka O., Kunanets N., Pasichnyk V., Matsiuk O. and Matsiuk S. Comparative Analysis of Smart City Platforms. Proceedings of the COLINS-2023: 7th International Conference on Computational Linguistics and Intelligent Systems (Kharkiv, Ukraine, April 20–21, 2023). CEUR Workshop Proceedings (CEUR-WS.org). 2023. Vol-3403. pp. 487-499. URL: https://ceur-ws.org/Vol-3403/paper38.pdf.

36 Palka O., Melnyk A., Dmytrotsa L., Vasylenko Y., and Klymuk N. Dynamic test case prioritisation for mobile applications based on real user behaviour data. Proceedings of the CITI 2025: The 3rd International Workshop on Computer Information Technologies in Industry 4.0 (Ternopil, Ukraine, June 11-12, 2025). CEUR Workshop Proceedings (CEURWS.org). 2025. Vol-4057, pp. 179-188. URL: https://ceur-ws.org/Vol-4057/paper12.pdf.

37 Pydantic Data Validation Library Documentation [Електронний ресурс] – URL: https://pydantic.dev/docs/validation/latest/concepts/models/.

38 Python Software Foundation. Official Documentation [Електронний ресурс] – URL: https://docs.python.org/3/.

39 Render Cloud Application Hosting Documentation [Електронний ресурс] – URL: https://docs.render.com/.

40 SQLite Architecture and Documentation [Електронний ресурс] – URL: https://www.sqlite.org/docs.html.

41 Uvicorn ASGI Server Documentation [Електронний ресурс] – URL: https://uvicorn.dev/.

ДОДАТКИ

Додаток А

Лістинги веб-системи для інформування мешканців про умови та ризики під час екологічних катастроф

Лістинг А.1 – код головного інтерфейсу та управління станами карти

```
<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>ГІС Евакуація - Зборів</title>

    <link rel="manifest" href="manifest.json">
    <meta name="theme-color" content="#28a745">
    <meta http-equiv="Cache-Control" content="no-cache, no-store,
must-revalidate">
    <meta http-equiv="Pragma" content="no-cache">
    <meta http-equiv="Expires" content="0">

    <link rel="stylesheet"
href="https://unpkg.com/leaflet@1.9.4/dist/leaflet.css" />
    <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/6.0.0/css/all.min.css" />
    <link rel="stylesheet" href="style.css">
</head>
<body>

    <audio id="alertSound" src="alert.mp3" preload="auto"></audio>

    <div id="ecoStats" class="eco-panel">
        <h5><i class="fas fa-leaf"></i> Моніторинг Зборів</h5>
        <div class="eco-item">Повітря (AQI): <span id="aqiValue"
class="eco-value">--</span></div>
        <div class="eco-item">Температура: <span id="tempValue"
class="eco-value">--</span></div>
        <div class="eco-item">Статус: <span id="ecoStatusText"
style="font-weight:bold;">Завантаження...</span></div>

        <button onclick="askNotificationPermission();
refreshAllData()" style="margin-top:10px; width:100%; font-
size:12px; font-weight:bold; cursor:pointer; border-radius:6px;
border:1px solid #28a745; background:#fff; color:#28a745; padding:
10px; display: flex; align-items: center; justify-content: center;
gap: 8px; box-shadow: 0 2px 4px rgba(0,0,0,0.1);">
            <i class="fas fa-sync-alt"></i>
            <span>Оновити дані</span>
        </button>
```

```
    </div>

    <button class="find-nearest-btn"
onclick="askNotificationPermission(); locateUser()">
        <i class="fas fa-person-running"></i>
        <span>Найближче укриття</span>
    </button>

    <div class="user-menu-btn" onclick="toggleLoginModal()"
title="Кабінет диспетчера">
        <i class="fas fa-user-shield"></i>
        <div id="adminBadge" class="badge-admin">ADM</div>
    </div>

    <div id="modalOverlay" onclick="toggleLoginModal()"></div>
    <div id="loginModal">
        <span class="close-modal"
onclick="toggleLoginModal()">&times;</span>
        <i class="fas fa-lock" style="font-size: 40px; color:
#007bff; margin-bottom: 15px;"></i>
        <h3>Авторизація</h3>
        <input type="password" id="passInput" placeholder="Пароль
диспетчера">
        <button onclick="login()">Увійти</button>
    </div>

    <div id="adminPanel">
        <h4><i class="fas fa-exclamation-triangle"></i> Режим
загрози</h4>
        <div class="input-group">
            <label>Тип події:</label>
            <select id="hName">
                <option value="Хімічне зараження">☢ Хімічне
зараження</option>
                <option value="Витік аміаку">□ Витік
аміаку</option>
                <option value="Пожежа">🔥 Пожежа</option>
                <option value="Забруднення води">💧 Забруднення
води</option>
            </select>
        </div>
        <div class="input-group">
            <label>Радіус (м):</label>
            <input type="number" id="hRadius" value="500">
        </div>
        <button class="btn-secondary" onclick="logout()">Завершити
сеанс</button>
    </div>

    <div id="map"></div>
```

```
    <script
src="https://unpkg.com/leaflet@1.9.4/dist/leaflet.js"></script>
    <script>
        const API_BASE = window.location.hostname === "localhost"
|| window.location.hostname === "127.0.0.1"
            ? "http://127.0.0.1:8000"
            : "https://zboriv-evacuation-gis.onrender.com";

        delete L.Icon.Default.prototype._getIconUrl;
        L.Icon.Default.mergeOptions({
            iconRetinaUrl:
'https://unpkg.com/leaflet@1.9.4/dist/images/marker-icon-2x.png',
            iconUrl:
'https://unpkg.com/leaflet@1.9.4/dist/images/marker-icon.png',
            shadowUrl:
'https://unpkg.com/leaflet@1.9.4/dist/images/marker-shadow.png',
        });

        var isAdmin = localStorage.getItem('isGisAdmin') ===
'true';
        var map = L.map('map', { zoomControl: true
}).setView([49.6612, 25.1402], 14);
        var hazardsLayers = {};
        var routeLine = null;
        var userMarker = null;

        // Використовуємо Set для зберігання ID загрози замість
простого лічильника
        var knownHazardIds = new
Set(JSON.parse(localStorage.getItem('knownHazardIds') || '[]'));


L.tileLayer('https://{s}.basemaps.cartocdn.com/light_all/{z}/{x}/{
y}{r}.png', {
            attribution: '&copy; OpenStreetMap'
        }).addTo(map);

        if (isAdmin) {
            document.body.classList.add('admin-active');
            document.getElementById('adminPanel').style.display =
'block';
            document.getElementById('adminBadge').style.display =
'block';
        }

        function askNotificationPermission() {
            if ("Notification" in window) {
                Notification.requestPermission().then(permission
=> {
                    if (permission === "granted") {
                        const audio =
document.getElementById('alertSound');
```

```
                        audio.play().then(() =>
audio.pause()).catch(e => console.log("Аудіо активується"));
                    }
                });
            }
        }

        function stopAlertSound() {
            const audio = document.getElementById('alertSound');
            audio.loop = false;
            audio.pause();
            audio.currentTime = 0;
        }

        document.addEventListener('click', stopAlertSound);
        document.addEventListener('touchstart', stopAlertSound);

        window.addEventListener('focus', function() {
            stopAlertSound();
            refreshAllData();
        });

        window.onload = function() {
            refreshAllData();
            setInterval(refreshAllData, 30000);
        };

        function refreshAllData() {
            loadShelters();
            loadHazards();
            updateEcoData();
        }

        function updateEcoData() {
            const lat = 49.6612, lon = 25.1402;
            fetch(`https://air-quality-api.open-meteo.com/v1/air-
quality?latitude=${lat}&longitude=${lon}&current=european_aqi`)
                .then(res => res.json())
                .then(data => {
                    const aqi = data.current.european_aqi;
                    const aqiEl =
document.getElementById('aqiValue');
                    const statusEl =
document.getElementById('ecoStatusText');
                    aqiEl.innerText = aqi;
                    if (aqi <= 20) { statusEl.innerText = "Чисто";
statusEl.style.color = "#28a745"; }
                    else if (aqi <= 50) { statusEl.innerText =
"Помірно"; statusEl.style.color = "#ffc107"; }
                    else { statusEl.innerText = "Забруднено";
statusEl.style.color = "#dc3545"; }
                }).catch(e => console.log("Eco API Error"));
```

```
            fetch(`https://api.open-
meteo.com/v1/forecast?latitude=${lat}&longitude=${lon}&current=tem
perature_2m`)
                .then(res => res.json())
                .then(data => {
                    document.getElementById('tempValue').innerText
= data.current.temperature_2m + "°C";
                });
        }

        function loadShelters() {
            fetch(`${API_BASE}/shelters`)
                .then(res => res.json())
                .then(data => {
                    data.forEach(s => {
                        L.marker([s.lat, s.lon], {
                            icon: L.divIcon({
                                className: 'custom-div-icon',
                                html: `<div style='background-
color:#28a745;' class='marker-pin'></div><i class='fas fa-shield-
halved'></i>`,
                                iconSize: [30, 42], iconAnchor:
[15, 42], popupAnchor: [0, -35]
                            })
                        }).addTo(map)
                        .bindPopup(`<b>${s.name}</b><br><i
class="fas fa-users"></i> Місткість: ${s.capacity}`);
                    });
                });
        }

        function loadHazards() {
            fetch(`${API_BASE}/hazards`)
                .then(res => res.json())
                .then(data => {
                    if (data) {
                        let hasNew = false;
                        let currentIds = new Set();

                        data.forEach(h => {
                            currentIds.add(h.id);
                            // Якщо цього ID немає в списку
відомих — це нова загроза
                            if (!knownHazardIds.has(h.id)) {
                                hasNew = true;
                                triggerEmergencyAlert(h);
                            }
                        });

                        // Оновлюємо список відомих ID (видаляємо
ті, яких більше немає в базі)
                        knownHazardIds = currentIds;
```

```
                        localStorage.setItem('knownHazardIds',
JSON.stringify(Array.from(knownHazardIds)));

                        Object.values(hazardsLayers).forEach(layer
=> map.removeLayer(layer));
                        hazardsLayers = {};
                        data.forEach(h => drawHazard(h));
                    }
                });
        }

        function triggerEmergencyAlert(hazard) {
            const audio = document.getElementById('alertSound');
            audio.loop = true;
            audio.play().catch(() => console.log("Звук чекає
взаємодії"));

            if (Notification.permission === "granted") {
                if ('serviceWorker' in navigator &&
navigator.serviceWorker.controller) {

navigator.serviceWorker.ready.then(registration => {
                        registration.showNotification("🚨 ТРИВОГА:
ЗБОРІВ", {
                            body: `Нова загроза: ${hazard.name}.
Радіус: ${hazard.radius}м.`,
                            icon: "https://cdn-icons-
png.flaticon.com/512/564/564440.png",
                            vibrate: [200, 100, 200],
                            tag: 'emergency-alert'
                        });
                    });
                } else {
                    const notif = new Notification("🚨 ТРИВОГА:
ЗБОРІВ", {
                        body: `Виявлено загрозу: ${hazard.name}.
Відкрийте мапу!`,
                        icon: "https://cdn-icons-
png.flaticon.com/512/564/564440.png"
                    });
                    notif.onclick = function() { window.focus();
stopAlertSound(); };
                }
            } else {
                alert(`🚨 УВАГА! НОВА ЗАГРОЗА: ${hazard.name}`);
            }
        }

        function drawHazard(h) {
            const circle = L.circle([h.lat, h.lon], {
                color: '#dc3545', fillColor: '#dc3545',
fillOpacity: 0.4, radius: h.radius || 500
```

```
            }).addTo(map);
            hazardsLayers[h.id] = circle;

            const content = `
                <div style="text-align:center; min-width: 160px;">
                    <b style="color:#dc3545;">🚨 ${h.name}</b><br>
                    <div style="margin: 8px 0;">
                        <label>Радіус (м):</label>
                        <input type="number"
id="editRadius_${h.id}" value="${h.radius}" style="width:65px;
padding:2px; border:1px solid #ccc;">
                    </div>
                    ${isAdmin ? `
                    <div style="display:flex; gap:8px; justify-
content:center; margin-top:5px;">
                        <button onclick="updateHazardInDB(${h.id},
${h.lat}, ${h.lon}, '${h.name}')"
                                style="background:#28a745;
color:white; border:none; padding:6px 10px; border-radius:4px;
cursor:pointer;" title="Зберегти">
                            <i class="fas fa-save"></i>
                        </button>
                        <button
onclick="removeHazardFromDB(${h.id})"
                                style="background:#dc3545;
color:white; border:none; padding:6px 10px; border-radius:4px;
cursor:pointer;" title="Видалити">
                            <i class="fas fa-trash"></i>
                        </button>
                    </div>` : `<span>Радіус: ${h.radius}
м</span>`}
                </div>`;
            circle.bindPopup(content);
        }

        function updateHazardInDB(id, lat, lon, name) {
            const newRadius =
parseInt(document.getElementById(`editRadius_${id}`).value);
            const updatedData = { name: name, radius: newRadius,
lat: lat, lon: lon };
            fetch(`${API_BASE}/hazards/${id}`, {
                method: 'PUT',
                headers: { 'Content-Type': 'application/json' },
                body: JSON.stringify(updatedData)
            })
            .then(res => {
                if (!res.ok) throw new Error("Помилка оновлення");
                return res.json();
            })
            .then(() => {
                hazardsLayers[id].setRadius(newRadius);
                alert("Радіус успішно змінено!");
                map.closePopup();
```

```
            })
            .catch(err => alert("Помилка: " + err));
        }

        function removeHazardFromDB(id) {
            if (confirm("Видалити цю загрозу?")) {
                fetch(`${API_BASE}/hazards/${id}`, { method:
'DELETE' })
                .then(() => {
                    map.removeLayer(hazardsLayers[id]);
                    delete hazardsLayers[id];
                    // Видаляємо з пам'яті, щоб при повторному
створенні спрацював звук
                    knownHazardIds.delete(id);
                    localStorage.setItem('knownHazardIds',
JSON.stringify(Array.from(knownHazardIds)));
                });
            }
        }

        function locateUser() {
            map.locate({setView: true, maxZoom: 16});
        }

        map.on('locationerror', function(e) {
            alert("Помилка GPS: Переконайтеся, що сайт
використовує HTTPS.");
        });

        map.on('locationfound', function(e) {
            if (userMarker) map.removeLayer(userMarker);
            userMarker =
L.marker(e.latlng).addTo(map).bindPopup("Ви тут").openPopup();

fetch(`${API_BASE}/nearest_shelter?lat=${e.latlng.lat}&lon=${e.lat
lng.lng}`)
                .then(res => res.json())
                .then(shelter => {
                    if (routeLine) map.removeLayer(routeLine);
                    routeLine = L.polyline([e.latlng,
[shelter.lat, shelter.lon]],
                        {color: '#28a745', weight: 5, dashArray:
'10, 10'}).addTo(map);
                    L.popup().setLatLng([shelter.lat,
shelter.lon])
                        .setContent(`<b>Найближче
укриття!</b><br>${shelter.name}<br>Відстань: ${shelter.distance_m}
м`).openOn(map);
                });
        });

        map.on('click', function(e) {
            if (!isAdmin) return;
```

```
            const data = {
                name: document.getElementById('hName').value,
                radius:
parseInt(document.getElementById('hRadius').value),
                lat: e.latlng.lat, lon: e.latlng.lng
            };
            fetch(`${API_BASE}/hazards`, {
                method: 'POST',
                headers: { 'Content-Type': 'application/json' },
                body: JSON.stringify(data)
            }).then(res => res.json()).then(h => {
                drawHazard(h);
                // Додаємо в список відомих, щоб не було
повторного спрацювання звуку при відмальовці
                knownHazardIds.add(h.id);
                localStorage.setItem('knownHazardIds',
JSON.stringify(Array.from(knownHazardIds)));
            });
        });

        function login() {
            if (document.getElementById('passInput').value ===
"admin123") {
                localStorage.setItem('isGisAdmin', 'true');
                location.reload();
            } else { alert("Невірний пароль!"); }
        }

        function logout() {
            localStorage.removeItem('isGisAdmin');
            location.reload();
        }

        function toggleLoginModal() {
            if (isAdmin) return;
            const modal = document.getElementById('loginModal');
            const overlay =
document.getElementById('modalOverlay');
            modal.style.display = (modal.style.display ===
'block') ? 'none' : 'block';
            overlay.style.display = (overlay.style.display ===
'block') ? 'none' : 'block';
        }

        if ('serviceWorker' in navigator) {
            navigator.serviceWorker.register('sw.js?v=5');
        }
    </script>
</body>
</html>
```

Лістинг А.2 – код серверного модуля

```
import sqlite3
import os
import math
from fastapi import FastAPI, HTTPException
from fastapi.middleware.cors import CORSMiddleware
from pydantic import BaseModel
from typing import Optional, List
import uvicorn

app = FastAPI()

# 1. НАЛАШТУВАННЯ CORS (Дозволяє запити з будь-якого домену,
включаючи GitHub Pages)
app.add_middleware(
    CORSMiddleware,
    allow_origins=["*"],
    allow_credentials=True,
    allow_methods=["*"],
    allow_headers=["*"],
)

DB_PATH = "shelters.db"

# 2. ІНІЦІАЛІЗАЦІЯ БАЗИ ДАНИХ
def init_db():
    conn = sqlite3.connect(DB_PATH)
    cursor = conn.cursor()
    # Таблиця загроз
    cursor.execute('''
        CREATE TABLE IF NOT EXISTS hazards (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            name TEXT,
            radius INTEGER,
            lat REAL,
            lon REAL
        )
    ''')
    # Таблиця укриттів
    cursor.execute('''
        CREATE TABLE IF NOT EXISTS shelters (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            name TEXT,
            capacity INTEGER,
            lat REAL,
            lon REAL
        )
    ''')
    conn.commit()
    conn.close()

init_db()
```

```
# 3. МОДЕЛЬ ДАНИХ
class Hazard(BaseModel):
    name: str
    radius: int
    lat: float
    lon: float

# 4. ДОПОМІЖНА ФУНКЦІЯ: Формула Гаверсину для розрахунку відстані
на сфері
def get_distance(lat1, lon1, lat2, lon2):
    R = 6371000  # Радіус Землі в метрах
    phi1, phi2 = math.radians(lat1), math.radians(lat2)
    dphi = math.radians(lat2 - lat1)
    dlambda = math.radians(lon2 - lon1)
    a = math.sin(dphi/2)**2 +
math.cos(phi1)*math.cos(phi2)*math.sin(dlambda/2)**2
    return 2 * R * math.atan2(math.sqrt(a), math.sqrt(1-a))

# --- МАРШРУТИ (ENDPOINTS) ---

# Отримати всі укриття
@app.get("/shelters")
async def get_shelters():
    conn = sqlite3.connect(DB_PATH)
    conn.row_factory = sqlite3.Row
    cursor = conn.cursor()
    try:
        cursor.execute("SELECT * FROM shelters")
        data = [dict(row) for row in cursor.fetchall()]
    except sqlite3.OperationalError:
        data = []
    finally:
        conn.close()
    return data

# Отримати всі загрози
@app.get("/hazards")
async def get_hazards():
    conn = sqlite3.connect(DB_PATH)
    conn.row_factory = sqlite3.Row
    cursor = conn.cursor()
    cursor.execute("SELECT * FROM hazards")
    data = [dict(row) for row in cursor.fetchall()]
    conn.close()
    return data

# Створити нову загрозу
@app.post("/hazards")
async def create_hazard(h: Hazard):
    conn = sqlite3.connect(DB_PATH)
    cursor = conn.cursor()
    cursor.execute(
```

```
        "INSERT INTO hazards (name, radius, lat, lon) VALUES (?,
?, ?, ?)",
        (h.name, h.radius, h.lat, h.lon)
    )
    new_id = cursor.lastrowid
    conn.commit()
    conn.close()
    return {**h.dict(), "id": new_id}

# Оновити існуючу загрозу (Радіус, координати тощо)
@app.put("/hazards/{h_id}")
async def update_hazard(h_id: int, h: Hazard):
    conn = sqlite3.connect(DB_PATH)
    cursor = conn.cursor()
    cursor.execute(
        "UPDATE hazards SET name=?, radius=?, lat=?, lon=? WHERE
id=?",
        (h.name, h.radius, h.lat, h.lon, h_id)
    )
    conn.commit()
    conn.close()
    # Повертаємо об'єкт із ID для коректного оновлення карти на
фронтенді
    return {**h.dict(), "id": h_id, "status": "updated"}

# Видалити загрозу
@app.delete("/hazards/{h_id}")
async def delete_hazard(h_id: int):
    conn = sqlite3.connect(DB_PATH)
    cursor = conn.cursor()
    cursor.execute("DELETE FROM hazards WHERE id = ?", (h_id,))
    conn.commit()
    # Команда VACUUM очищує місце в БД та видаляє кешовані записи
    try:
        conn.execute("VACUUM")
    except:
        pass
    conn.close()
    return {"status": "deleted", "id": h_id}

# Пошук найближчого укриття за координатами користувача
@app.get("/nearest_shelter")
async def nearest(lat: float, lon: float):
    conn = sqlite3.connect(DB_PATH)
    conn.row_factory = sqlite3.Row
    cursor = conn.cursor()
    cursor.execute("SELECT * FROM shelters")
    shelters = cursor.fetchall()
    conn.close()

    if not shelters:
        raise HTTPException(status_code=404, detail="Укриття не
знайдені")
```

```
    nearest_s = None
    min_dist = float('inf')

    for s in shelters:
        d = get_distance(lat, lon, s['lat'], s['lon'])
        if d < min_dist:
            min_dist = d
            nearest_s = dict(s)

    nearest_s['distance_m'] = round(min_dist, 2)
    return nearest_s

# 5. ЗАПУСК СЕРВЕРА
if __name__ == "__main__":
    # Render динамічно призначає порт через змінну середовища
    port = int(os.environ.get("PORT", 8000))
    uvicorn.run(app, host="0.0.0.0", port=port)
```

Лістинг А.3 – Скрипт фонового сповіщення

```
// sw.js - Сервіс-воркер для ГІС Евакуація Зборів (Версія 5 -
Фоновий режим)

// 1. Встановлення: активуємо новий воркер негайно
self.addEventListener('install', (event) => {
    console.log('SW: Встановлено');
    self.skipWaiting();
});

// 2. Активація: беремо контроль над усіма клієнтами (вкладками)
self.addEventListener('activate', (event) => {
    console.log('SW: Активовано');
    event.waitUntil(clients.claim());
});

// 3. Обробка запитів (Fetch): виправлення TypeError при помилках
мережі
self.addEventListener('fetch', (event) => {
    event.respondWith(
        fetch(event.request).catch(err => {
            console.warn('SW: Помилка мережі або CORS:',
event.request.url);
            return new Response('Network error', { status: 408 });
        })
    );
});

// 4. Опрацювання фонових сповіщень (Push)
self.addEventListener('push', function(event) {
    console.log('SW: Отримано сигнал Push');
```

```
    let title = '🚨 ТРИВОГА: ЗБОРІВ';
    let options = {
        body: 'Виявлено нову загрозу! Терміново перевірте карту та
маршрути евакуації.',
        icon: 'https://cdn-icons-
png.flaticon.com/512/564/564440.png',
        badge: 'https://cdn-icons-
png.flaticon.com/512/564/564440.png',
        vibrate: [500, 100, 500, 100, 500], // Потужна вібрація
        data: { dateOfArrival: Date.now() },
        actions: [
            { action: 'explore', title: 'Відкрити карту' }
        ]
    };

    event.waitUntil(self.registration.showNotification(title,
options));
});

// 5. Обробка кліку на сповіщення: фокусування вікна або відкриття
нової вкладки
self.addEventListener('notificationclick', function(event) {
    event.notification.close(); // Закриваємо банер

    event.waitUntil(
        clients.matchAll({ type: 'window', includeUncontrolled:
true }).then(function(clientList) {
            // Шукаємо вже відкриту вкладку нашого сайту
            for (let i = 0; i < clientList.length; i++) {
                let client = clientList[i];
                if (client.url.includes(self.location.origin) &&
'focus' in client) {
                    return client.focus();
                }
            }
            // Якщо вкладку не знайдено — відкриваємо нову (шлях
./ для GitHub Pages)
            if (clients.openWindow) {
                return clients.openWindow('./');
            }
        })
    );
});// sw.js - Сервіс-воркер для ГІС Евакуація Зборів (Версія 4 -
CORS Fix)

// 1. Встановлення: активуємо негайно
self.addEventListener('install', (event) => {
    console.log('SW: Встановлено');
    self.skipWaiting();
});

// 2. Активація: контроль над усіма вкладками
self.addEventListener('activate', (event) => {
```

```
    console.log('SW: Активовано');
    event.waitUntil(clients.claim());
});

// 3. ОБРОБКА ЗАПИТІВ (Виправлено для запобігання TypeError)
self.addEventListener('fetch', (event) => {
    event.respondWith(
        fetch(event.request).catch(err => {
            console.warn('SW: Помилка мережі або CORS при
запиті:', event.request.url);
            // Повертаємо порожню відповідь, щоб не "валити"
проміс воркера
            return new Response('Network error', { status: 408,
statusText: 'Network Error or CORS blocked' });
        })
    );
});

// 4. Очікування Push-повідомлень
self.addEventListener('push', function(event) {
    console.log('SW: Отримано сигнал Push');
});

// 5. Оновлена обробка кліку: фокусування вікна та виправлення 404
self.addEventListener('notificationclick', function(event) {
    event.notification.close();

    event.waitUntil(
        clients.matchAll({ type: 'window', includeUncontrolled:
true }).then(function(clientList) {
            for (let i = 0; i < clientList.length; i++) {
                let client = clientList[i];
                if (client.url.includes(self.location.origin) &&
'focus' in client) {
                    return client.focus();
                }
            }
            if (clients.openWindow) {
                return clients.openWindow('./');
            }
        })
    );
});
```

Додаток Б

Результати експериментального тестування та верифікації працездатності компонентів веб-системи

Таблиця Б.1 – Тестові сценарії та статуси працездатності веб-системи управління евакуаційними заходами та інформування населення при екологічних загрозах

№ тесту	Тестовий сценарій	Вхідна дія	Очікувана реакція системи	Фактичний результат	Статус
1	**2**	**3**	**4**	**5**	**6**
1	Запуск PWA та кешування	Перший візит через HTTPS на GitHub Pages	Завантаження index.html, активація sw.js, збереження інтерфейсу в Cache API	Компоненти закешовано, сервіс-воркер активний	Успішно
2	Моніторинг екологічного стану	Авто-запит клієнта до Open-Meteo API	Отримання JSON з AQI та t℃, динамічна зміна кольору віджета	Дані отримано за 140 мс, колір віджета оновлено	Успішно
3	Автентифі-кація диспетчера	Ввід пароля у модальному вікні адмінки	Валідація даних через Pydantic на FastAPI, відкриття	Авторизацію пройдено, інструменти керування розблоковано	Успішно

			доступу до карт		
4	Нанесення локальної загрози	Клік по карті Leaflet, ввід типу аварії та радіуса	POST-запит на Render, фіксація події в таблиці hazards СКБД SQLite	Запис створено в БД, на карті рендериться коло L.circle	Успішно
5	Фонове пуш-сповіщення	Тригер події створення загрози	sw.js перехоплює сигнал, виводить системний банер та запускає alert.mp3	Сповіщення доставлено на заблокований екран, сирена активна	Успішно
6	Геопозиціо-нування користува-ча	Натискання кнопки GPS-навігації внизу екрана	Запит до апаратного Geolocation API, зчитування координат, центрування карти	Точне місцезнаходженн я визначено з похибкою до 4 м	Успішно

Продовження таблиці Б.1

1	2	3	4	5	6
7	Розрахунок евакуації	Авто-запит на серверний ендпоінт /nearest_shelter	Обчислення відстаней за формулою Гаверсину, сортування таблиці shelters	Знайдено найближче сховище Зборова, лінію маршруту побудовано за 95 мс	Успішно
8	Автоном-ний офлайн-режим	Активація режиму «У літаку» на смартфоні	Подія fetch у sw.js відловлює помилку мережі й піднімає дані з Cache API	Базова карта вулиць міста та маркери ПРУ успішно завантажені з кешу	Успішно
9	Оптимізація бази даних	Видалення відпрацьо-ваних загроз диспетче-ром	Очищення шарів карти через clearLayers(), виклик команди VACUUM в SQLite	Інтерфейс очищено, файл shelters.db дефрагменовано та стиснуто	Успішно