ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Розробка інформаційного сайту для візуалізації даних про забруднення навколишнього природного середовища України»

на здобуття освітнього ступеня бакалавра
зі спеціальності 121 Інженерія програмного забезпечення
освітньо-професійної програми «Інженерія програмного забезпечення»

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Ярослав РОЖКО
(підпис)

Виконав: здобувач вищої освіти групи ПД-43

Ярослав РОЖКО

Керівник: Віталій ЗАЛИВА
доктор філософії (PhD)

Рецензент: _____

Київ 2025

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти Бакалавр

Спеціальність 121 Інженерія програмного забезпечення

Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного забезпечення
_____ Ірина ЗАМРІЙ
« ____ » _____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Рожко Ярославу Юрійовичу

1. Тема кваліфікаційної роботи: «Розробка інформаційного сайту для візуалізації даних про забруднення навколишнього природного середовища України» керівник кваліфікаційної роботи доктор філософії (PhD) Віталій ЗАЛИВА, затверджені наказом Державного університету інформаційно-комунікаційних технологій від «24» лютого 2025 р. № 56.

2. Строк подання кваліфікаційної роботи «02» червня 2025 р.

3. Вихідні дані до кваліфікаційної роботи: теоретичні відомості про методи розробки веб-сайтів, опис роботи веб-сайтів, документація бібліотекм Chart.js.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Огляд існуючих веб-ресурсів екологічного моніторингу та аналіз існуючих аналогів, визначення

вимог до програмного забезпечення.
2. Огляд та аналіз засобів реалізації застосунку засобів реалізації веб-сайту екологічного моніторингу.
3. Проєктування архітектури веб-сайту екологічного моніторингу.
4. Програмна реалізація та тестування веб-сайту для екологічного моніторингу.

5. Перелік графічного матеріалу: *презентація*

1. Аналіз аналогів.
2. Вимоги до програмного забезпечення.
3. Програмні засоби реалізації.
4. Діаграма архітектури системних компонентів
5. Діаграма модульна структура клієнтської частини веб-сайту
6. Алгоритм роботи створення графіку
7. Екранні форми.
8. Демонстрація роботи застосунку
9. Апробація результатів дослідження

6. Дата видачі завдання «25» лютого 2025

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Підбір та аналіз науково-технічної літератури	25.02.-04.03.2025	
2	Аналіз та дослідження існуючих аналогів	05.03-13.03.2025	
3	Дослідження впливу забруднення повітря на здоров'я	14.03-15.03.2025	
4	Аналіз існуючих веб-ресурсів екологічного моніторингу	16.03-21.03.2025	
5	Огляд та аналіз засобів реалізації веб-сайту екологічного моніторингу	22.03-03.04.2025	
6	Проєктування архітектури веб-сайту екологічного моніторингу	04.04-06.04.2024	
7	Програмна реалізація та тестування веб-сайту екологічного моніторингу	04.07-28.04.2024	
8	Оформлення роботи: вступ, висновки, реферат	29.04-11.05.2025	
9	Розробка демонстраційних матеріалів	12.05-18.05.2025	
10	Попередній захист роботи	19.05-30.05.2025	

Здобувач вищої освіти _____ Ярослав РОЖКО
(підпис)

Керівник
кваліфікаційної роботи _____ Віталій ЗАЛИВА
(підпис)

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 46 стор., 1 табл., 10 рис., 20 джерел.

Мета роботи – спрощення візуалізації якості зміни повітря в Україні.

Об'єкт дослідження – процес візуалізації змін якості повітря в Україні.

Предмет дослідження – розробка адаптивного веб-сайту на основі JavaScript, HTML, CSS та візуалізаційних бібліотек який дозволяє відстежувати зміни якості повітря в Україні можливості фільтрації за регіонами та рекомендації для громадян.

Короткий зміст роботи: В роботі досліджено інструменти та методи для візуалізації змін якості повітря в Україні з урахуванням регіональної специфіки. Проаналізовано можливості провідних екосистем моніторингу, таких як SaveEcoBot, AQICN, OpenAQ та Breezometer. Визначено функціональні та нефункціональні вимоги до веб-застосунку для відображення даних екологічного моніторингу.

Розроблено клієнтський веб-додаток з використанням HTML5, CSS3, JavaScript і бібліотеки Chart.js, що дозволяє будувати графіки динаміки індексу якості повітря (AQI) на основі заздалегідь підготовлених JSON-файлів. Реалізовано інтерфейс з фільтрами вибору місяця, області, міста та дати, а також блок рекомендацій, що динамічно оновлюється залежно від рівня AQI. Структура даних structuredData дозволяє ефективно організовувати, обробляти та відображати дані про стан повітря за різними регіонами й часовими інтервалами.

Проведено тестування продуктивності та адаптивності інтерфейсу, що підтвердило стабільну роботу веб-додатку в основних браузерах. Для публікації проєкту використано платформу GitLab Pages, що забезпечує його відкритий доступ та можливість подальшого масштабування.

Сфера застосування розробленого веб-додатку — інформування населення та екологічна просвіта з метою підвищення рівня екологічної обізнаності громадян.

КЛЮЧОВІ СЛОВА: ЯКІСТЬ ПОВІТРЯ, AQI, ВІЗУАЛІЗАЦІЯ ДАНИХ, CHART.JS, STRUCTUREDDATA, ФІЛЬТРАЦІЯ, ЕКОЛОГІЧНІ ПОРАДИ, GITLAB PAGES

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

-

ПЗ – Програмне забезпечення

AQI – Air Quality Index (Індекс якості повітря)

HTML – HyperText Markup Language

CSS – Cascading Style Sheets

JS (JavaScript) – Мова програмування, що використовується для створення інтерактивної поведінки вебсторінок

SPA – Single Page Application (односторінковий додаток)

DOM – Document Object Model (модель об'єктів документа)

JSON – JavaScript Object Notation (формат обміну структурованими даними)

structuredData – Ієрархічна структура даних, побудована у скрипті на основі JSON-файлів

Canvas – HTML-елемент для малювання графіки

Chart.js – Бібліотека JavaScript для візуалізації даних у вигляді графіків

API – Application Programming Interface (інтерфейс прикладного програмування)

Git – Система контролю версій

GitHub Pages – Сервіс для публікації статичних веб-сайтів безпосередньо з GitHub репозиторію

UI – User Interface (інтерфейс користувача)

UX – User Experience (досвід користувача)

Live Server – Локальний сервер для запуску вебпроєктів у реальному часі

ВСТУП

Актуальність: моніторинг якості атмосферного повітря є одним із найбільш важливих аспектів екологічної безпеки. Забруднення повітря має прямий вплив на здоров'я населення, спричиняючи хронічні респіраторні захворювання, алергії, проблеми серцево-судинної системи, а в окремих випадках — навіть передчасну смертність. В умовах кліматичних змін, зростання кількості автомобільного транспорту, індустріалізації та енергетичних криз, які супроводжуються спалюванням викопного палива, якість повітря стрімко погіршується, особливо в густонаселених регіонах України. Відсутність доступу до візуалізованих екологічних даних для широкого загалу громадян знижує ефективність екологічного інформування та профілактичних заходів.

Впровадження цифрових інформаційних технологій дає змогу створити зручний і доступний інструмент для швидкого отримання актуальних даних про стан атмосферного повітря. Адаптивні веб-сайти є ефективним засобом передачі інформації, оскільки можуть бути використані як на персональних комп'ютерах, так і на мобільних пристроях. Такий сайт дозволяє користувачам не лише переглядати загальну картину по країні, а й аналізувати ситуацію по конкретних регіонах, населених пунктах, а також отримувати рекомендації щодо безпечної поведінки у випадках перевищення допустимих норм якості повітря.

Об'єктом дослідження є процес візуалізації змін якості повітря в Україні.

Предметом дослідження є розробка адаптивного веб-сайту на основі JavaScript, HTML, CSS та візуалізаційних бібліотек який дозволяє відстежувати зміни якості повітря в Україні можливості фільтрації за регіонами та рекомендації для громадян

Метою роботи є спрощення візуалізації якості зміни повітря в Україні.

Методи дослідження: на початковому етапі було здійснено порівняльний аналіз функціональності існуючих веб-платформ, таких як SaveEcoBot, AQICN, OpenAQ, які збирають, агрегують та демонструють показники стану повітря. Проведено вивчення технічних рішень, застосованих у цих проєктах. З урахуванням результатів аналізу було визначено функціональні вимоги до веб-додатку, що розробляється. Для реалізації клієнтської частини використано HTML5 та CSS3 із сучасними технологіями адаптивної верстки (Flexbox, Grid). JavaScript забезпечує динамічну взаємодію з інтерфейсом, фільтрацію даних та побудову графіків. Бібліотека Chart.js реалізує графічне відображення змін індексу якості повітря в часі. Посилання на SaveEcoBot забезпечує інтеграцію інтерактивної карти. Для структурування та зберігання даних застосовано формат JSON..

Наукова новизна роботи визначається наступними функціональними складовими: полягає у створенні веб-інструменту для багаторівневої фільтрації та візуалізації екологічних даних у зручному форматі, який поєднує картографічне відображення, часові графіки, а також інтерактивні підказки та поради для користувачів. У розробленому додатку передбачено гнучке оновлення інтерфейсу залежно від вибраного регіону, міста, місяця чи конкретної дати.

Практична значущість результатів полягає в можливості використання реалізованого застосунку для інформування населення про рівень забруднення повітря у реальному часі, включення екологічної інформації до інформаційних панелей громадського транспорту, шкільних та муніципальних сайтів. Також рішення може бути інтегроване у мобільні додатки для розширення можливостей персонального моніторингу екологічних умов проживання.

1 АНАЛІЗ ЗАСТОСУНКУ ДЛЯ ВІЗУАЛІЗАЦІЇ ДАНИХ ПРО ЗАБРУДНЕННЯ ПОВІТРЯ

1.1 Інформаційний сайт як засіб екологічного моніторингу

Інформаційний сайт для візуалізації даних про забруднення повітря — це інтерактивна платформа, що надає користувачам можливість оперативно отримувати екологічну інформацію. Такий ресурс дозволяє переглядати картографічні зображення рівня забруднення, аналізувати зміну індексу AQI (Air Quality Index) по регіонах, а також отримувати рекомендації відповідно до рівня небезпеки для здоров’я.

Основні функціональні компоненти сайту включають:

–інтерактивна карта України з посиланням на сервіс SaveEcoBot;

–система фільтрації за регіоном, населеним пунктом, місяцем, датою;

–побудова графіків середнього рівня AQI з використанням бібліотеки Chart.js;

–кольорова класифікація рівнів AQI із відповідними порадами для населення;

–інформаційні блоки щодо заходів для покращення якості повітря;

Основні цілі вебсайту екологічного моніторингу:

– підвищення екологічної обізнаності населення;

–надання візуально зрозумілої інформації у зручному форматі;

–зниження ризиків для здоров’я завдяки превентивним рекомендаціям;

Оцінимо переваги та недоліки інформаційний сайт як засіб екологічного моніторингу.

Переваги:

– доступність із будь-якого пристрою та місця, де є інтернет-з'єднання;

-адаптивний дизайн, що забезпечує зручність користування;

-простота оновлення даних завдяки роботі з JSON-файлами;

-можливість масштабування для інших типів екологічних показників (PM10,

CO2 тощо) ;

Недоліки:

–необхідність регулярного оновлення даних (автоматичне або ручне завантаження JSON);

-залежність точності результатів від достовірності первинних джерел (екологічні датчики);

-обмежена інтерактивність у поточній версії (відсутність персонального акаунту, сповіщень тощо) ;

1.2 Специфіка розробки екологічних веб-додатків.

Особливістю створення подібних інформаційних систем є потреба у точній роботі з великим обсягом даних, що надходять із численних джерел (сенсори, відкриті державні API). Дані повинні бути швидко оброблені та структуровані за ознаками часу, місця та типу показника. Це вимагає впровадження ефективної структури збереження та індексації (у цьому випадку обрана структура об'єкта structuredData з вкладеністю: місяць → область → населений пункт → дата).

1.3 Цільова аудиторія

Застосунок для екологічного моніторингу орієнтований на широке коло користувачів, які відрізняються за своїми навчальними потребами та цілями. До основних категорій користувачів належать:

– широка громадськість — для перевірки стану повітря у своєму регіоні;

-освітні заклади — як інструмент для викладання екології;

-представників місцевого самоврядування — для планування заходів реагування;

-екологічних активістів та волонтерів — для поширення інформації у спільнотах;

1.4 Дослідження існуючих аналогів

На сьогоднішній день одними з найпопулярніших вебсайтів для екологічного моніторингу є SaveEcoBot, AQICN, OpenAQ, Breezometer. Розглянемо їх детальніше.

1.4.1 SaveEcoBot

SaveEcoBot — це український екологічний проєкт, який відображаєінтерактивну карту забруднення повітря, використовуючи дані з державних та громадських станцій моніторингу якості повітря. Платформа дозволяє бачити поточні та історичні значення індексу AQI, а також рівні небезпечних речовин у повітрі (PM2.5, PM10, NO2 тощо).

Основні функції SaveEcoBot включають:

– карта якості повітря з інтерактивними маркерами;

– пошук по населеному пункту;

– графіки з динамікою показників;

– інтеграція з Telegram-ботом для сповіщень;

– API для розробників;

Переваги:

– деталізація по багатьох містах України, що дозволяє локальну аналітику;

– інтуїтивно зрозумілий інтерфейс, що підходить для широкого кола користувачів;

– можливість отримувати миттєві сповіщення про погіршення стану повітря через месенджери;

Недоліки:

– обмежений вибір періодів для аналізу — переважно лише поточні або добові дані;

– неможливість побудови графіків по місяцях або детального фільтрування;

– відсутність персоналізації даних та рекомендацій за результатами аналізу;

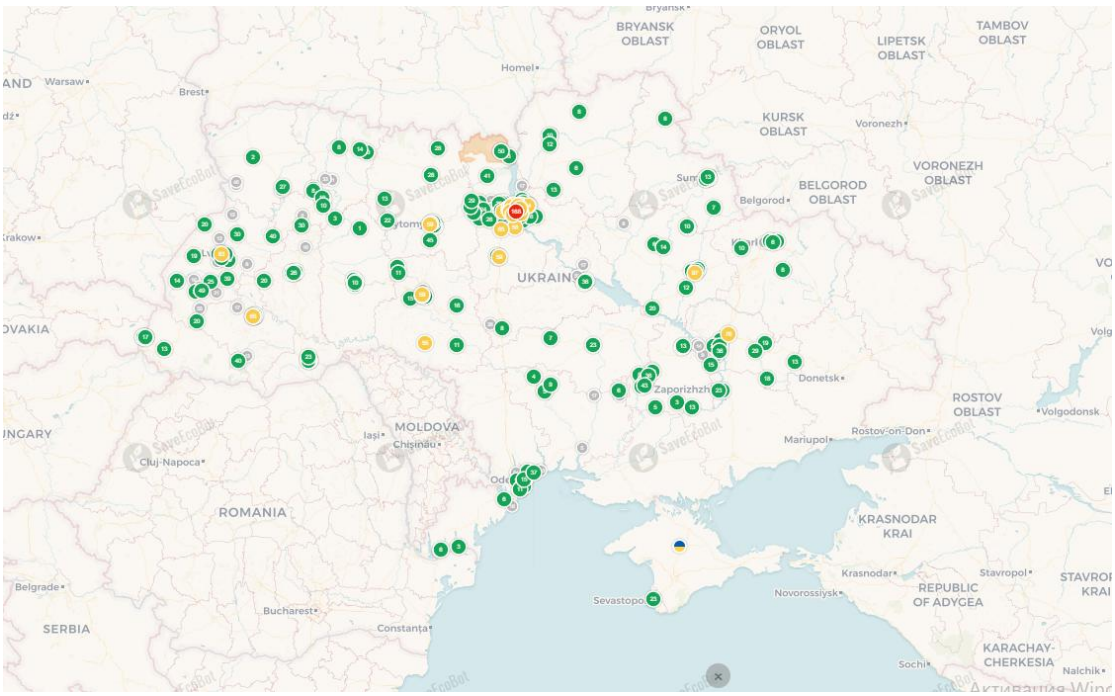


Рис. 1.1 Приклад карти показнигків повітря SaveEcoBot

1.4.2 AQICN

AQICN (The World Air Quality Index) — міжнародний сервіс, який об’єднує дані з різних країн і створює глобальну карту якості повітря. Він базується на відкритих API та агрегує інформацію в реальному часі з тисяч станцій моніторингу.

Основні функції AQICN включають:

– глобальна карта з кольоровим позначенням рівнів AQI;

– детальні графіки показників по містах;

– дані у реальному часі;

– API для інтеграції у сторонні додатки;

Переваги:

– величезне охоплення — від мегаполісів до невеликих населених пунктів;

– висока точність та надійність джерел даних;

– наявність історичних графіків забруднення та можливість порівняння регіонів;

Недоліки:

– відсутність користувацького вебінтерфейсу — потрібні технічні знання

для роботи;

– переважна орієнтація на англомовне середовище і користувачів з IT-компетенціями;

– відсутність візуалізації та рекомендацій;

– лише часткове покриття території України;

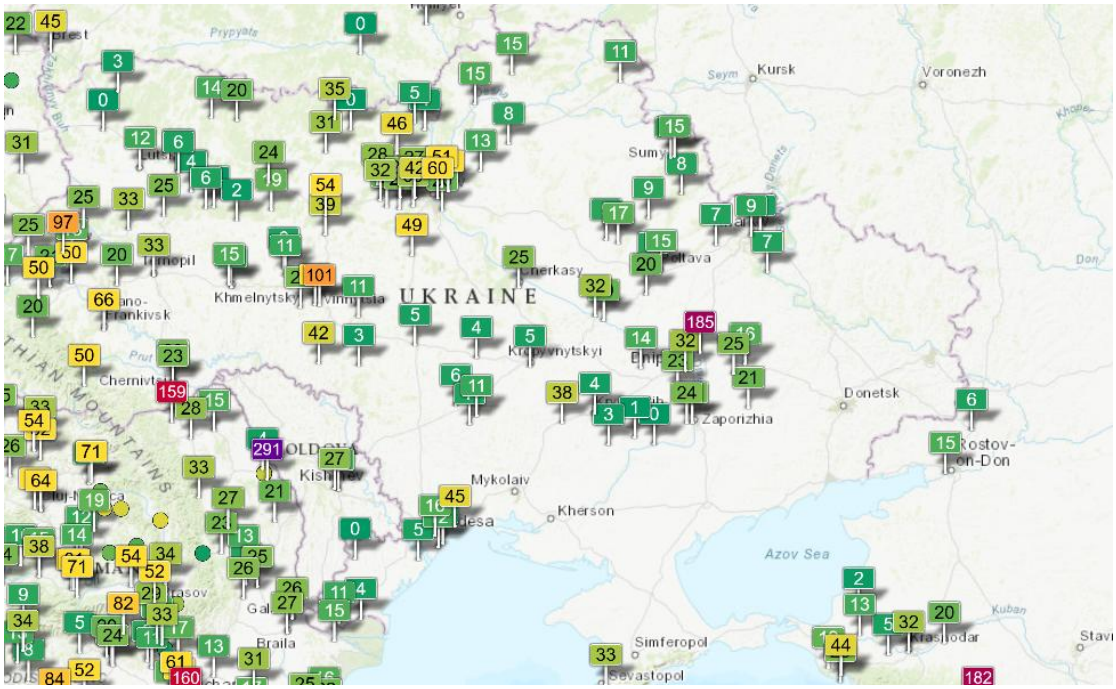


Рис. 1.2 Приклад карти показників повітря AQICN

1.4.3 Breezometer

Breezometer — комерційна платформа, яка надає прогнозовані дані про якість повітря на базі алгоритмів машинного навчання, метеоданих та супутникового моніторингу.

Основні функції Breezometer включають:

– прогнозування якості повітря на 5–7 днів наперед;

– інтеграція з розумними пристроями та мобільними додатками;

– підказки для вразливих категорій населення (діти, алергіки тощо);

Переваги:

– надточне прогнозування на основі багатьох джерел;

– персоналізовані рекомендації для користувачів залежно від їхнього місцезнаходження;

– широке застосування в охороні здоров'я, страхуванні та міському управлінні;

Недоліки:

– майже всі сервіси доступні лише за підпискою або ліцензією;

– відсутність відкритого коду та можливостей модифікації;

– лише часткове покриття території України;

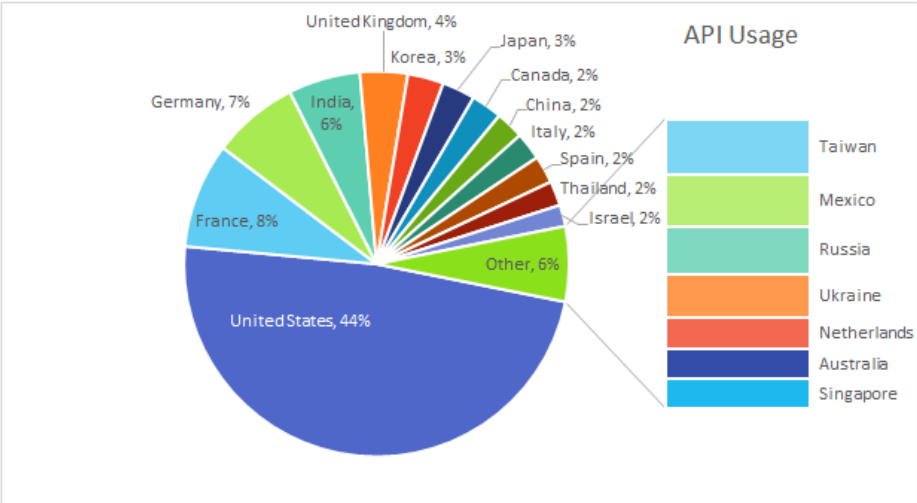


Рис. 1.3 Приклад графіка показника повітря Breezometer

1.4.4 OpenAQ

OpenAQ — некомерційна ініціатива, що забезпечує відкритий доступ до глобальної екологічної інформації. Сервіс орієнтований на дослідників, аналітиків та розробників екосистем даних.

Основні функції OpenAQ включають:

– API для зчитування даних у форматі JSON;

– імпорт та експорт даних у відкритих форматах;

– велика база даних зі світових джерел;

Переваги:

– повна відкритість даних для будь-якого використання;

– підтримка інтеграції в наукові, муніципальні або комерційні проєкти;

– публікація джерел і документоване API.

Недоліки:

– Відсутність користувацького вебінтерфейсу — потрібні технічні знання для роботи.

– Переважна орієнтація на англомовне середовище і користувачів з IT-компетенціями.

– Лише часткове покриття території України.

- Відсутність візуалізації та рекомендацій.

В таблиці 1.1 зведено результати аналізу існуючих застосунків для вивчення японської мови та їх порівняння.

Таблиця 1.1

Зведена таблиця аналізу існуючих застосунків для екологічного моніторингу та їх порівняння

Показник	**SaveEcoBot**	**AQICN**	**OpenAQ**	**Breezometer**
Карта забруднення	Так	Так	Ні	Так
Дані у реальному часі	Так	Так	Так	Так
Графіки	Добові	Годинні, добові	Потрібна інтеграція	Прогноз на 5–7 днів
Рівень персоналізації	Низький	Середній	Відсутній	Високий
Поради користувачам	Обмежені	Обмежені	Відсутні	Деталізовані
Відкритість даних	Часткова	Часткова	Повна	Обмежена
Інтерфейс	Простий	Ускладнений	Відсутній	Сучасний
Орієнтація	Громадяни України	Міжнародна	Розробники	Комерційні користувачі
Українська локалізація	Так	Ні	Ні	Ні

1.5 Визначення вимог до застосунку

Проаналізувавши сутність застосунку, його специфіку, цільову аудиторію та аналогів, було визначено наступні вимоги до застосунку для візуалізації змін якості повітря в Україні.

– Відображення інтерактивної карти забруднення повітря (посиланням на SaveEcoBot).

– Можливість вибору регіону, міста, місяця та конкретної дати для фільтрації екологічних даних.

– Побудова графіка середнього індексу якості повітря (AQI) за обраними параметрами.

– Динамічне оновлення фільтрів у залежності від попереднього вибору користувача: даны часто бувають не стабільні, додаються нові міста з показниками.

– Інформаційний блок з порадами щодо покращення стану повітря: можлтвість навчитися що робити в тих чи інших ситуація.

– Адаптивність інтерфейсу — підтримка екранних роздільностей від 1400 до 2500 пікселів, що забезпечує зручність перегляду на різних пристроях.

– Підтримка популярних веббраузерів, зокрема Google Chrome, Mozilla Firefox, Safari, Microsoft Edge та Opera.

– Продуктивність — час генерації графіка на основі введених параметрів не повинен перевищувати 2 секунд, що є критично важливим для користувацького досвіду

– Простота навігації — користувач має змогу інтуїтивно орієнтуватися у функціоналі сайту без додаткового навчання чи інструкцій

Сукупність цих вимог забезпечує створення ефективного, доступного та соціально корисного онлайн-ресурсу, який допомагає громадянам України краще орієнтуватися у стані довкілля та приймати обґрунтовані рішення щодо поведінки у разі підвищення рівня забруднення повітря.

2 ЗАСОБИ РЕАЛІЗАЦІЇ

2.1 Середовище розробки

Інтегроване середовище розробки (IDE) – це комплексне програмне забезпечення, що надає програмістам різноманітні інструменти для розробки програмного коду. Інтегровані середовища розробки зазвичай включають текстовий редактор, інструменти для автоматизації збірки коду, дебагер та, часто, інтерфейс для системи контролю версій. Це допомагає розробникам ефективно писати, тестувати і відлагоджувати свої програми в одному середовищі.

2.1.1 Visual Studio Code

Visual Studio Code (VS Code) — це передовий текстовий редактор від Microsoft, який широко використовується програмістами для розробки програмного забезпечення. Хоча VS Code сам по собі не є повноцінним інтегрованим середовищем розробки (IDE), його можна ефективно використовувати як IDE завдяки гнучкості та численним розширенням, які можуть значно розширити його функціональність.

VS Code підтримує багато мов програмування "з коробки", включаючи JavaScript, TypeScript, Python, PHP, C++, та інші. Це досягається через використання розширень, які можна легко встановити та налаштувати, адаптуючи редактор до специфічних вимог розробки.

Однією з ключових переваг VS Code є його система плагінів. Користувачі можуть встановлювати розширення для майже всього — від підтримки додаткових мов програмування і фреймворків до інтеграції з іншими інструментами та сервісами, такими як контейнери Docker, системи управління версіями (наприклад, Git) і бази даних.

Хоча VS Code не є IDE в класичному розумінні цього слова, його можливості та широка адаптованість роблять його відмінним вибором для

розробників, які шукають легкий, але потужний інструмент для кодування.

2.2 Мови програмування

Мова програмування — це формалізована мова, призначена для комунікації інструкцій до машини, зокрема комп'ютера. Вона використовується для створення програм, які виконують алгоритми. Більшість мов програмування складається з інструкцій для комп'ютерів. Вони дозволяють розробникам ефективно створювати програми, виражаючи розрахунки та управління даними.

2.2.1 HTML

HTML — це стандартизована мова розмітки, яка використовується для створення структури та вмісту веб-сторінок. Створена у 1991 році Вітом Бушем, HTML дозволяє розробникам визначати різні елементи сайту, такі як заголовки, параграфи, посилання, зображення та багато іншого. Він є основою всіх веб-технологій і працює у парі з CSS і JavaScript для створення інтуїтивних та динамічних сайтів.

HTML підтримує використання тегів, які структуровано описують вміст сторінки, забезпечуючи правильну семантику та доступність. Це робить його простим для освоєння як для початківців, так і для досвідчених веб-розробників. Поширені стандарти і регулярні оновлення, зокрема введення HTML5, розширюють можливості мови, додаючи підтримку мультимедіа, графіки та більш складної взаємодії з користувачем [3].

HTML застосовується у створенні будь-якого веб-контенту — від простих односторінкових сайтів до складних веб-додатків. Широкий спектр інструментів і розширень, що працюють із HTML, дозволяє розробникам створювати сучасні й зручні для користувачів інтерфейси, забезпечуючи високу сумісність та адаптивність.-ресурсу, який допома

2.2.2 CSS

CSS — це мова каскадних таблиць стилів, яка використовується для

оформлення та візуального оформлення веб-сторінок. Створена у 1996 році, CSS дозволяє веб-розробникам задавати вигляд елементів сторінки, такі як кольори, шрифти, відступи, розміри та розташування. Вона робить можливим створення привабливих та узгоджених інтерфейсів, відокремлюючи дизайн від структури, закладеної у HTML.

CSS підтримує каскадність правил, щоб застосовувати стилі до різних елементів у різний час та в різних контекстах, що робить її гнучкою та потужною. Останні версії CSS, зокрема CSS3, додають можливості для анімацій, трансформацій, медіа-запитів і створення адаптивних дизайнів, що автоматично підлаштовуються під різні розміри екранів [3].

CSS застосовується для формування зовнішнього вигляду сайтів, зроблення їх привабливими та зручними для користувачів. Завдяки великій кількості фреймворків і препроцесорів, таких як Bootstrap, Sass або Less, розробники можуть створювати складні і сучасні дизайн-системи швидко та ефективно.ся у стані довкілля та приймати обґрунтовані рішення щодо поведінки у разі підвищення рівня забруднення повітря.

2.2.3 JavaScript

JavaScript — це високорівнева, динамічна мова програмування, яка використовується для створення інтерактивних і динамічних веб-сторінок. Створена у 1995 році Бренданом Ейхом і нині є ключовою технологією веб-розробки разом із HTML і CSS. JavaScript дозволяє додавати на сторінку функціональність, таку як обробка подій, динамічне оновлення вмісту, створення ігор, анімації і багато іншого.

Мова підтримує об’єктно-орієнтоване, імперативне та функційне програмування, дозволяючи писати гнучкий і масштабований код. Розробники використовують різноманітні бібліотеки та фреймворки, наприклад React, Angular та Vue.js, для створення сучасних веб-додатків. Крім того, JavaScript активно застосовується для розробки серверних додатків через платформу Node.js, що розширює його можливості за межі браузера [4].

Завдяки своїй універсальності і широкій спільноті, JavaScript став одним із найпопулярніших інструментів для створення цілих екосистем веб-розробки, забезпечуючи високий рівень інтерактивності та користувацького досвіду.

2.3 Бібліотеки

Бібліотеки — це колекції попередньо написаних функцій, класів або модулів, які використовуються для спрощення процесу програмування та швидкого розроблення програмного забезпечення. Вони містять готові рішення для виконання поширених задач, таких як обробка даних, робота з мережею, графіка, безпека, математичні обчислення та багато інших.

Використання бібліотек дозволяє розробникам не писати код з нуля, а зосередитися на логіці конкретного проекту, зменшуючи час розробки і підвищуючи якість продукту. Наприклад, у веб-розробці популярні бібліотеки — React, Lodash, jQuery, Angular, які надають зручні інструменти для роботи з DOM, обробки подій, стану додатка та інших функцій.

Кодові бібліотеки можуть бути написані на різних мовах програмування і часто поширюються у вигляді пакетів через менеджери пакетів, такі як npm для JavaScript, pip для Python чи Maven для Java. Вони є ключовим інструментом у сучасній розробці, дозволяючи створювати більш складні, ефективні й надійні програми швидше і з меншими зусиллями [11].

2.3.1 Chart.js

Chart.js — це популярна відкриту джерело JavaScript-бібліотека, яка дозволяє легко створювати інтерактивні та красиві графіки й діаграми на веб-сторінках. Вона базується на HTML5 Canvas і пропонує широкий спектр типів графіків, таких як лінійні, стовпчикові, кругові, полярні та інші.

Chart.js був розроблений для того, щоб спростити процес візуалізації даних і зробити його доступним для розробників будь-якого рівня. Зовнішній вигляд та поведінка графіків легко налаштовуються за допомогою конфігураційних опцій та стилів, що дозволяє створювати адаптивні й інтерактивні діаграми.

Ця бібліотека широко використовується у веб-додатках для візуалізації статистичних даних, аналітики та дашбордів. Вона підтримує анімацію, масштабування, інтеркативність та інтеграцію з іншими фреймворками, що робить її одним із найпопулярніших інструментів для роботи з графікою та даними

2.4 Система управління базою даних

JSON — це легкий формат обміну даними, який зорієнтований на просту та людську читабельність. Створений у 2001 році, JSON використовує синтаксис, схожий на об’єктну нотацію в JavaScript, але є незалежним від мови і широко підтримується різноманітними мовами програмування. Його основне призначення — структуровано зберігати та передавати дані між клієнтськими і серверними системами, а також між компонентами веб-додатків.

JSON підтримує основні типи даних, такі як рядки, числа, логічні значення, масиви та об’єкти, що робить його дуже гнучким та універсальним для різних задач. Його структура — це поєднання пар "ключ-значення", що дозволяє легко представляти складні дані у вигляді ієрархічної моделі.

Завдяки високій простоті і легкості читання, JSON швидко став стандартним форматом для обміну даними у веб-розробці, мобільних додатках іAPI. Його підтримка майже у всіх сучасних мовах програмування та фреймворках робить його незамінним інструментом для сучасних розробників

2.5 Система контролю версій

GitHub — це веб-платформа для хостингу коду, яка базується на системі контролю версій Git. GitHub швидко став однією з найбільш впливових платформ у світі розробки програмного забезпечення, забезпечуючи інструменти для спільної роботи, контролю версій та коду розподіленого доступу.

Центральний елемент GitHub — репозиторії, де зберігається код. Користувачі можуть створювати публічні чи приватні репозиторії для своїх проектів, куди входить управління версіями за допомогою Git [7].

GitHub дозволяє користувачам створювати копії чужих репозиторіїв,

вносити зміни та пропонувати ці зміни авторам оригінальних репозиторіїв через пул-реквести. Це ключова можливість для співпраці над відкритими проектами.

2.6 Інструменти проєктування інтерфейсу користувача

Figma є веб-базованим інструментом для інтерфейсного дизайну, який дозволяє командам створювати, співпрацювати та ділитися дизайном інтерфейсів та прототипами. Заснований у 2012 році Діланом Філдом та Еваном Воллесом, Figma стала популярною серед дизайнерів завдяки своїй доступності, гнучкості та багатофункціональності.

Однією з ключових особливостей Figma є її повна веб-інтеграція, яка дозволяє користувачам працювати з будь-якого комп'ютера без необхідності завантаження додатків. Це забезпечує легкий доступ та співпрацю між учасниками проекту з різних географічних точок.

Figma включає інструменти для створення інтерактивних прототипів без необхідності стороннього програмного забезпечення. Це дозволяє дизайнерам швидко створювати прототипи з анімацією переходів та мікровзаємодіями, і тестувати їх прямо в середовищі Figma.

Figma підтримує створення та управління дизайн-системами, дозволяючи дизайнерам зберігати стилі, компоненти та інші ресурси в легкодоступних бібліотеках, що спрощує узгодженість і повторне використання в багатьох проектах.

3 ПРОЄКТУВАННЯ

3.1 Проєктування архітектури застосунку

Веб-додаток реалізовано за класичною моделлю клієнтської архітектури, у якій основне навантаження з візуалізації даних та обробки взаємодії з користувачем виконується безпосередньо у браузері. Вся логіка обробки, фільтрації та відображення екологічних показників відбувається на клієнтській стороні, що мінімізує затримки та зменшує залежність від серверної інфраструктури.

Клієнтська частина створена із застосуванням HTML5, CSS3 та JavaScript. Для побудови графіків використовується бібліотека Chart.js, яка дозволяє реалізовувати адаптивні лінійні графіки для візуалізації зміни показника AQI (Air Quality Index) у часовому розрізі. Дані для побудови графіків завантажуються з локальних JSON-файлів (cities.json та main.json) під час ініціалізації сторінки. Після завантаження відбувається попередня агрегація та структурування даних у вигляді вкладеного об’єкта structuredData, що оптимізує подальші фільтрації.

Фільтрація даних забезпечується динамічними селекторами, які оновлюються при зміні попередніх значень. Користувач може послідовно обрати місяць, область, місто та дату. Після вибору параметрів та натискання кнопки «Показати» відбувається генерація графіка за обраними даними. Реалізовано перевірку повноти вибору перед побудовою графіка, що підвищує стабільність взаємодії з додатком.

Додаткова інтеграція із зовнішнім сервісом SaveEcoBot здійснюється через гіперпосилання на інтерактивну карту, яка відкривається у новому вікні. Це дозволяє користувачам отримувати актуальні географічні дані про забруднення повітря по регіонах України без потреби локального рендерингу картографії, зменшуючи навантаження на систему.

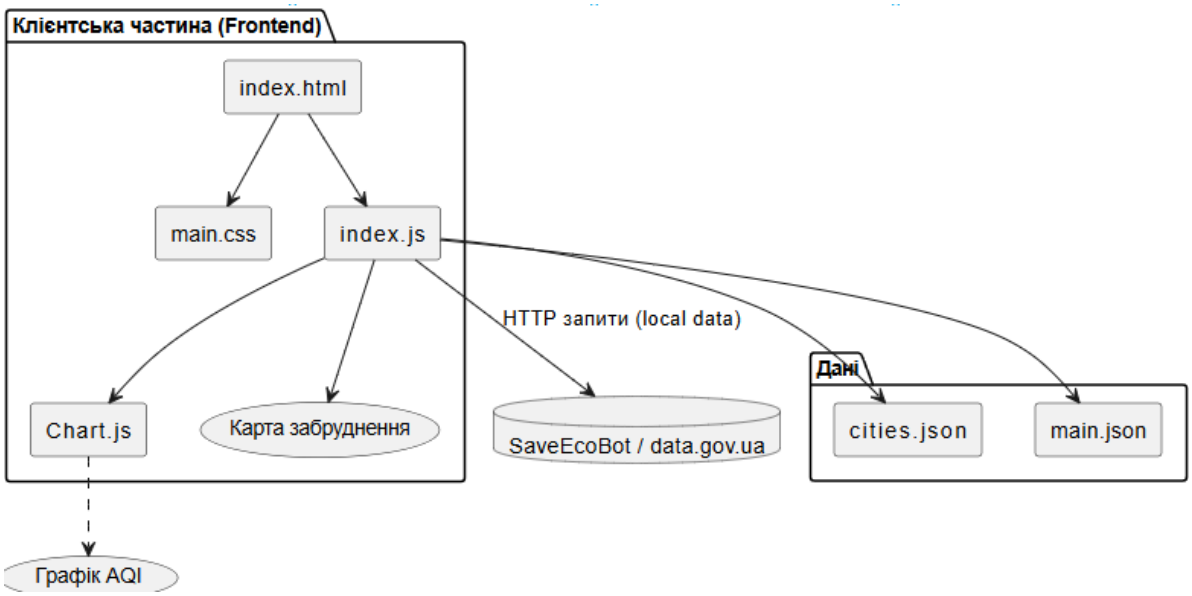


Рис. 3.1 Діаграма архітектури системних комплнентів

3.2 Архітектура клієнтської частини

Фронтенд має логічно структуровану організацію, що полегшує підтримку та масштабування веб-додатку. Основні компоненти клієнтської архітектури включають:

HTML-шаблон (index.html) — відповідає за структурування інтерфейсу, містить елементи фільтрації (селектори місяця, регіону, міста, дати), секції опису, порад, блок візуалізації (canvas для графіка) та інші візуальні елементи.

CSS-стилі (main.css, normalize.css) — забезпечують адаптивне оформлення сторінки, підтримку широкого діапазону роздільностей (від 1400px до 2500px), а також естетичну привабливість та доступність користувацького інтерфейсу.

JavaScript-логіка (index.js) — обробляє події, завантажує дані з JSON-файлів, структурує їх, динамічно оновлює фільтри, здійснює обчислення середнього AQI та формує графік на основі вибраних параметрів користувача.

У JavaScript реалізовано алгоритм, який враховує різні рівні деталізації: у випадку вибору конкретної дати графік будується за годинами, інакше — за днями місяця. Побудова графіків здійснюється з урахуванням продуктивності (до 2 секунд), а також коректного масштабування осей та підписів для зручного аналізу.

Поради для користувачів змінюються динамічно в залежності від рівня AQI,

що забезпечує зворотний зв’язок та інформує про безпечність перебування на вулиці. Додатковий блок екологічних порад пропонує загальні рекомендації для покращення якості повітря, підвищуючи соціальну обізнаність.

Завдяки описаній архітектурі веб-додаток демонструє високу швидкодію, стійкість, легкість підтримки та можливість масштабування для розширення функціональності в майбутньому. Вибір клієнтської архітектури виявився оптимальним для відкритого екологічного моніторингу в режимі реального часу на рисунках 3.2-3.3.

3.3 Архітектура системної частини

У реалізованому рішенні серверна частина не включає складної логіки обробки даних чи взаємодії з базами даних. Проте як джерело даних виступають статичні файли main.json та cities.json, що зберігаються локально та завантажуються клієнтським скриптом під час запуску додатку. Такий підхід дозволяє обійтись без повноцінного бекенду, мінімізуючи затрати на інфраструктуру та покращуючи швидкодію.

У перспективі система може бути доповнена серверною архітектурою на базі Node.js або Python Flask, яка дозволить:

централізовано оновлювати екологічні дані з відкритих API (наприклад, OpenAQ);

зберігати історію запитів користувачів або персоналізовані фільтри;

реалізовувати авторизацію, сповіщення та інші інтерактивні сервіси;

розгортати додаток у вигляді PWA (прогресивного веб-додатку) з кешуванням даних.

Системна архітектура у базовій реалізації включає лише веб-сервер (наприклад, Nginx або локальний сервер розробника), який обслуговує HTML-сторінку, стилі, скрипти та JSON-файли. Усі обчислення виконуються на боці користувача, що відповідає концепції «тонкого» сервера та «товстого» клієнта.

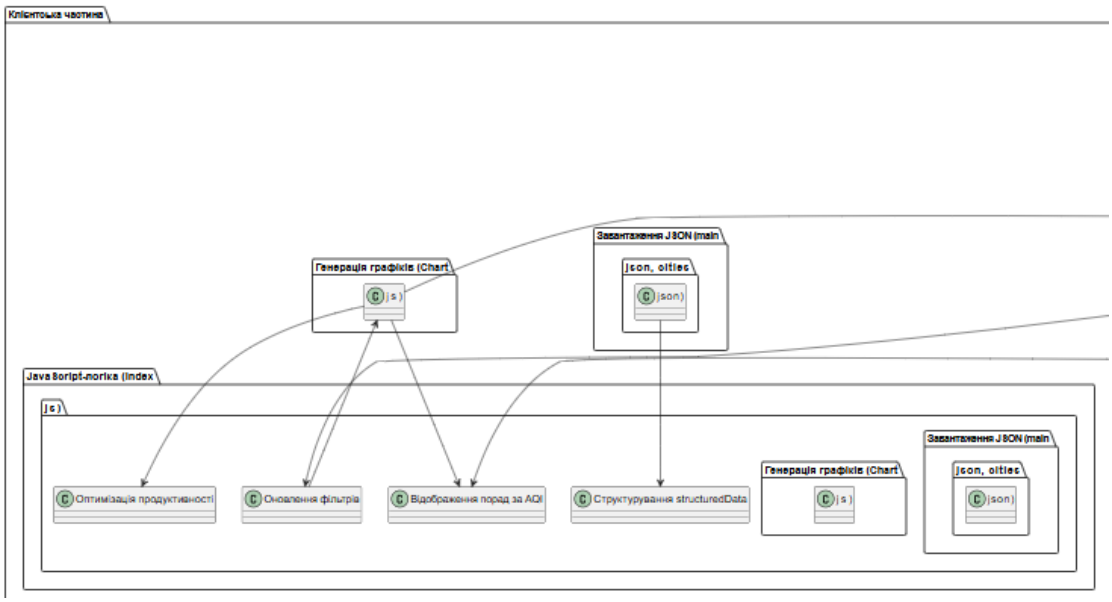


Рис. 3.2 Модульна структура клієнтської частини веб-сайту

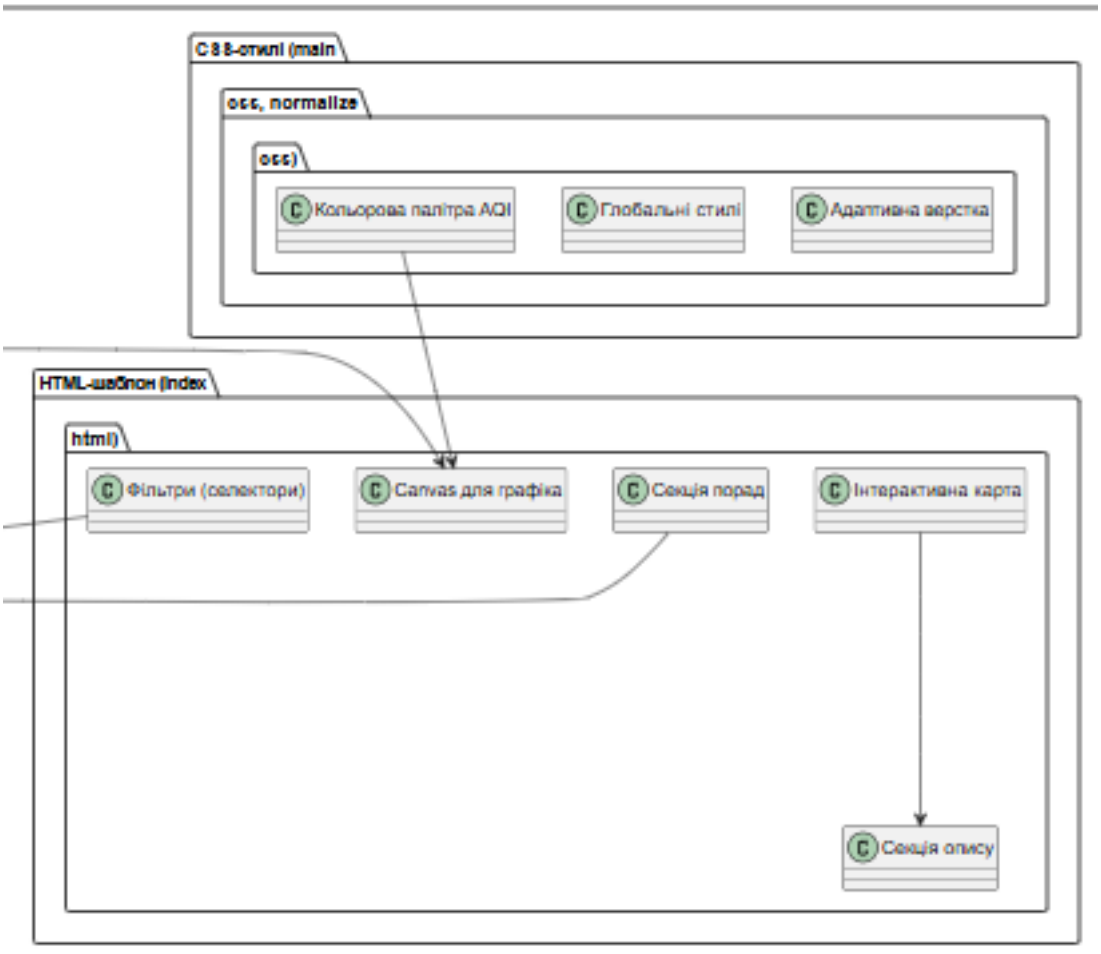


Рис. 3.3 Модульна структура клієнтської частини веб-сайту

3.4 Архітектура бази даних

У поточній реалізації веб-додатку не використовується традиційна реляційна база даних. Замість цього екологічні дані представлені у вигляді двох JSON-файлів — cities.json (перелік населених пунктів із регіонами) та main.json (історичні значення індексу якості повітря AQI). Це дозволяє працювати з даними на боці клієнта без залучення серверної частини та суттєво спрощує структуру системи.

JSON-структура main.json включає поля:

city_id: ідентифікатор населеного пункту;

aqi: значення індексу якості повітря;

pm25: концентрація часток PM2.5;

logged_at: дата та час фіксації показника.

Файл cities.json містить ідентифікатори населених пунктів, їх назви та назви регіонів.

Під час ініціалізації сторінки дані з обох файлів завантажуються за допомогою fetch-запитів. На їх основі формується об'єкт structuredData, у якому інформація агрегується за місяцями, регіонами, містами та датами.

Це дозволяє ефективно будувати графіки без потреби повторного звернення до джерела.

Таким чином, JSON-файли у цій системі виконують роль псевдо-бази даних, яка повністю функціонує у браузері користувача. У майбутньому структура даних може бути перенесена до повноцінної бази даних (наприклад, PostgreSQL), якщо буде реалізовано серверну частину із збиранням, аналізом та зберіганням показників якості повітря у режимі реального часу.

4 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ

4.1 Ініціалізація та структура проєкту

Розробка клієнтської частини веб-додатку здійснювалась без використання фреймворків (наприклад, Angular чи React), а на базі класичного підходу з HTML5, CSS3 та JavaScript. Це дозволило забезпечити максимальну гнучкість та мінімальну залежність від сторонніх бібліотек при реалізації кастомної логіки.

Проєктна структура веб-додатку включає такі основні елементи:

index.html — головна HTML-сторінка з вбудованою структурою елементів: заголовків, селекторів фільтрації, секцій порад, блоку графіка (canvas) тощо;

style/ — директорія стилів, що містить:

normalize.css — файл для стандартизації стилів у різних браузерах;

main.css — основні стилі, які забезпечують адаптивне оформлення інтерфейсу, візуальну ієрархію та підтримку кольорових схем залежно від рівнів AQI;

script/ — директорія зі скриптами, де index.js відповідає за всю бізнес-логіку додатку;

data/ — директорія з JSON-файлами (cities.json, main.json), що містять дані для візуалізації та фільтрації.

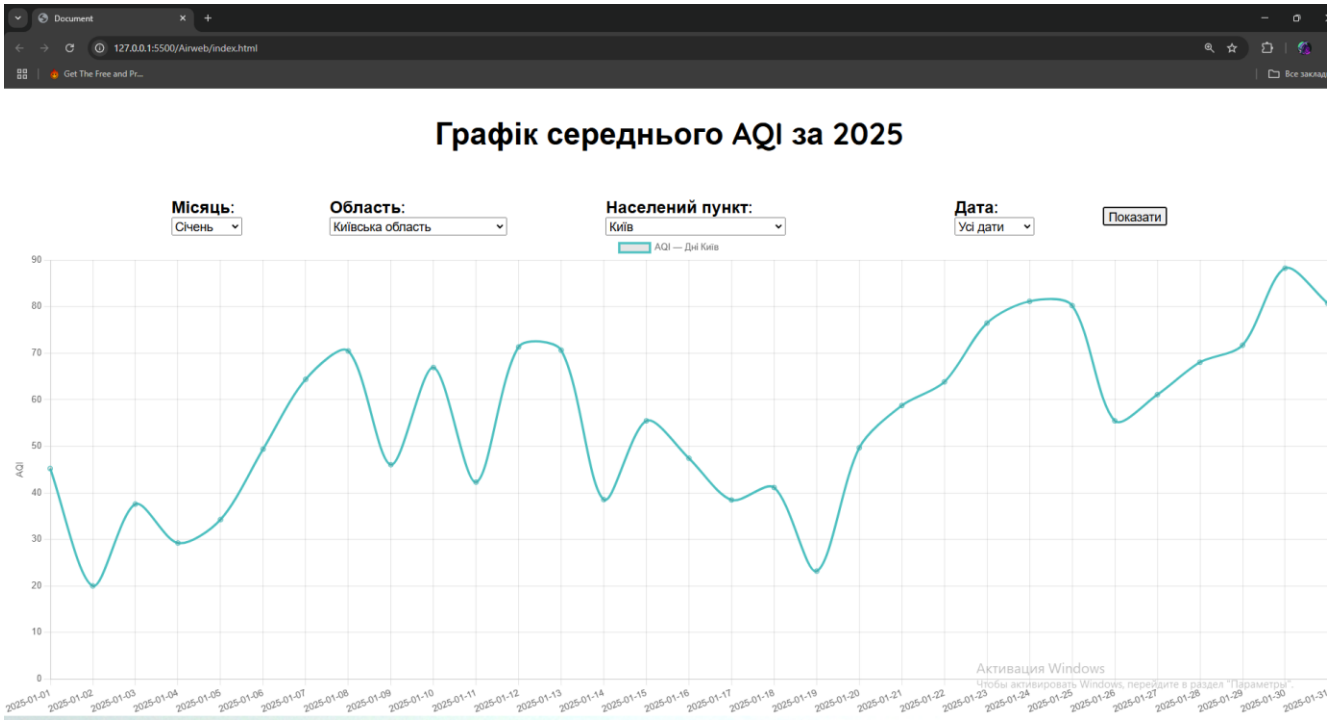


Рис. 4.1 Екранна форма Візуалізація взаємодії з фільтрами та графіком

4.2 Основні компоненти реалізації

Ключовим скриптом у реалізації є файл index.js, який виконує такі функції:

Завантаження даних — на початку роботи скрипт асинхронно завантажує JSON-файли з інформацією про міста та показники забруднення. Після отримання файлів вони з’єднуються у спільну структуру для подальшої агрегації.

Побудова структури structuredData — дані агрегуються у вкладений об’єкт, який структуровано за ієрархією: місяць → область → місто → дата. Це дозволяє максимально швидко відфільтровувати необхідну підмножину даних під час побудови графіка.

Оновлення фільтрів — при виборі значення в одному із селекторів (наприклад, місяць), автоматично оновлюються залежні селектори (область, місто, дата). Це реалізовано через DOM-маніпуляції та обробники подій.

Побудова графіків — після натискання кнопки «Показати» відбувається обчислення середнього значення AQI (або побудова погодинного графіка, якщо обрано дату) та формування лінійного графіка за допомогою бібліотеки Chart.js. Графік адаптується до розміру екрана та масштабується згідно з кількістю даних.

Інтерактивні поради — після побудови графіка система динамічно

відображає поради залежно від рівня AQI. Для цього реалізовано окрему логіку, яка враховує граничні значення індексу та відображає відповідні повідомлення.

Блок екологічних порад — фіксований блок, який відображається незалежно від вибраного рівня забруднення і містить універсальні рекомендації щодо покращення якості повітря: зменшення використання транспорту, підтримка озеленення тощо.

4.3 Реалізація серверної частини

У межах цього проєкту серверна частина як окремий бекенд не створювалася, оскільки додаток є повністю статичним SPA (Single Page Application), що працює лише з локальними JSON-файлами. Проте важливу роль у логіці програми виконує попередньо сформована структура даних у файлах main.json і cities.json, які емулюють API-відповіді сервера.

Ці файли розміщено у директорії /data/, а структура даних у них дозволяє гнучко використовувати інформацію для побудови графіків. Таким чином, реалізована логіка на JavaScript працює як «вбудований бекенд», що обробляє дані без запиту до справжнього сервера.

Якщо буде потреба у майбутньому масштабуванні або реальному API, логіку взаємодії з structuredData можна адаптувати під серверну платформу на Node.js, Python (Flask/Django), або іншу технологію.

4.4 Тестування та запуск

Тестування клієнтської частини проводилось вручну через локальний запуск HTML-файлу у середовищі розробки (VS Code із Live Server). Оскільки додаток працює повністю на стороні браузера, основна увага приділялась перевірці інтерактивних функцій, продуктивності та адаптивності інтерфейсу.

Основні тести:

– побудова графіка для кожного регіону при виборі різних місяців і міст.

– оновлення фільтрів відповідно до попереднього вибору.

– відображення екологічних порад у відповідності до рівня AQI.

– перевірка роботи графіка з великими обсягами даних (затримка не перевищує 2 секунд).

– адаптивність елементів на різних розширеннях екрана (від 1400px до 2500px).

– відкриття зовнішніх посилань (наприклад, SaveEcoBot) у новій вкладці.

– перевірка коректності обробки структури structuredData під час побудови графіка.

Тестування охоплювало такі браузери:

– Google Chrome

– Mozilla Firefox

– Microsoft Edge

– Safari (macOS)

– Opera

Результати:

Жодних критичних помилок виявлено не було. Функціонал працює стабільно в усіх протестованих браузерах, що відповідає нефункціональним вимогам, зазначеним у технічному завданні.

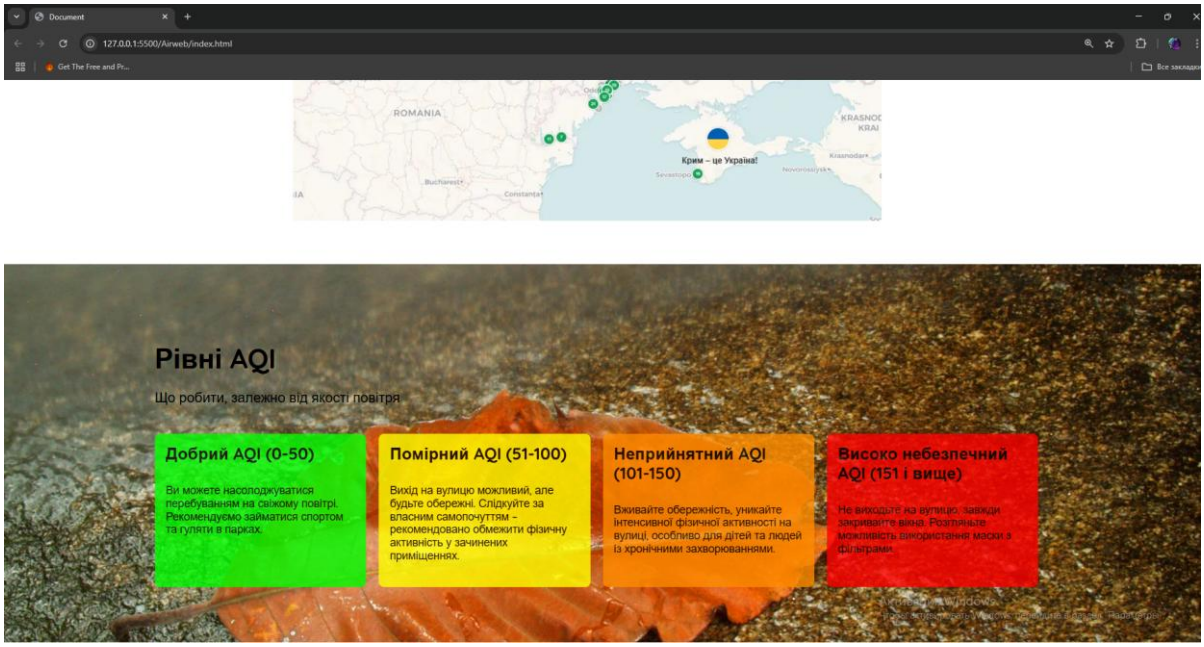


Рис. 4.2 Екранна форма реалізованої сторінки веб-додатку

4.5 Завантаження проєкту на GitHub

Для того, щоб додати проект до існуючого репозиторію на GitHub, необхідно виконати кілька кроків. Вони включають ініціалізацію Git у

проекті, додавання файлів до репозиторію та завантаження змін до GitHub. Для початку треба відкрити термінал у кореневому каталозі проєкту. Далі виконати команду для ініціалізації Git-репозиторію: – git init

Далі додаємо URL існуючого репозиторію GitHub як віддалений репозиторій:

– https://github.com/yaroslavrozhko/Airweb

Далі треба додати всі файли проєкту до індексу Git:

– git add .

Наступним кроком створюємо коміт з повідомленням, яке описує зміни, внесені у проєкт:

– git commit -m "Ad system"

Останнім кроком є завантаження змін до віддаленого репозиторію на GitHub:

– git push -u origin master

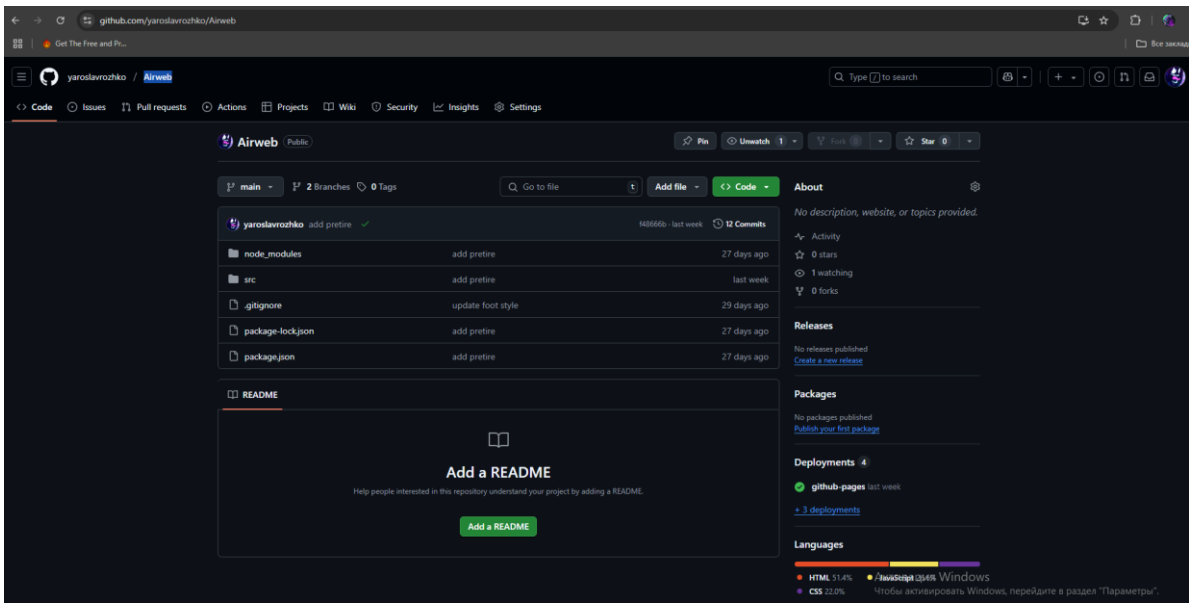


Рис. 4.3 GitHub репозиторій

4.6 Завантаження на GitHub Pages

GitHub Pages — це безкоштовний сервіс хостингу статичних вебсайтів без потреби у налаштуванні власного сервера. Оскільки клієнтська частина розробленого веб-додатку реалізована за допомогою HTML, CSS, JavaScript та бібліотеки Chart.js, вона повністю підходить для розміщення на GitHub Pages.

Кроки публікації:

Створення публічного репозиторію на GitHub

Спершу створюється новий репозиторій з назвою Airweb, після чого додаємо початковий код:

–git init

–git remote add origin https://github.com/yaroslavrozhko/Airweb

–git add .

–git commit -m "Initial upload for GitHub Pages"

–git push -u origin master

Для правильного відображення проєкту на GitHub Pages, всі основні файли мають бути у кореневій директорії або в гілці gh-pages. В каталозі повинні бути:

–index.html — головна сторінка;

–./style/ — каталог із CSS-файлами;

–./script/ — каталог із JavaScript-файлами;

–./img/ — зображення;

–./data/ — JSON-дані.

Перехід до налаштувань репозиторію

У розділі Settings → Pages вибрати:

Source: Deploy from a branch

Branch: master (або main)

Folder: /root (або /docs, якщо файли в папці docs/)

Очікування розгортання

Після збереження налаштувань, GitHub автоматично згенерує публічну адресу для сайту.

–.https://yaroslavrozhko.github.io/Airweb/

Тестування доступності:

–перевіряється коректність завантаження сайту, включно з:

–роботою фільтрів;

–побудовою графіків;

–відображенням карти та зображень;

–доступністю порад та екологічних блоків.

GitHub Pages надає простий спосіб для розміщення готового веб-додатку, що дозволяє користувачам без додаткових інсталяцій миттєво оцінити функціональність рішення, переглянувши сайт за прямим посиланням.

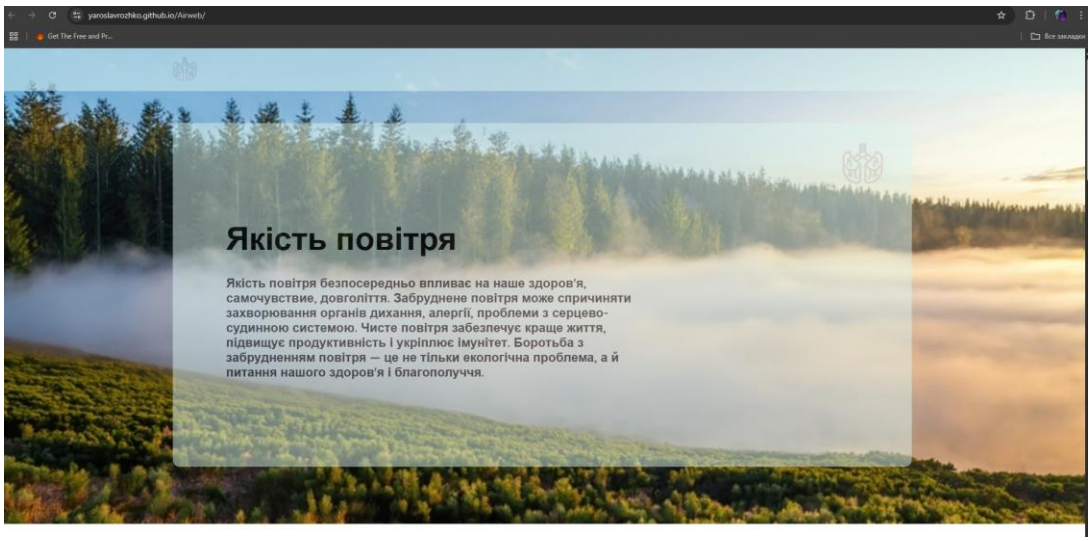


Рис. 4.4 Завантажений веб-сайт за допомогою GitHub Pages

ВИСНОВКИ

Результатом виконаної кваліфікаційної роботи стало створення адаптивного веб-додатку для візуалізації динаміки змін якості повітря в Україні. Система дозволяє здійснювати фільтрацію за часовими та географічними параметрами, виводити графіки середніх значень AQI (Air Quality Index), а також надавати рекомендації для громадян відповідно до поточного рівня забруднення.

У процесі виконання роботи було досягнуто наступні цілі:

– проаналізовано предметну область: досліджено сучасні сервіси з моніторингу якості повітря (зокрема SaveEcoBot, AQICN, OpenAQ, Breezometer), їхні можливості, переваги та недоліки. Проведено аналіз функціональних вимог користувачів, цільової аудиторії та типових сценаріїв використання таких систем.

– визначено вимоги до програмного забезпечення: сформовано перелік функціональних (інтерактивна карта, побудова графіків, фільтрація даних, екологічні поради) та нефункціональних (адаптивність, продуктивність, підтримка браузерів, легка навігація) вимог до веб-додатку.

– розроблено архітектуру клієнтської частини: створено модульну структуру додатку з використанням HTML5, CSS3, JavaScript та бібліотеки Chart.js. Реалізовано систему вкладених фільтрів та структуру structuredData, яка забезпечує швидкий доступ до даних у форматі: місяць → область → місто → дата.

– створено функціональний прототип: реалізовано систему завантаження локальних JSON-файлів (main.json, cities.json), динамічну побудову графіків, реактивне оновлення фільтрів, виведення порад залежно від рівня AQI та екологічних рекомендацій. Забезпечено продуктивність побудови графіка (менше 2 секунд).

– проведено тестування: веб-додаток перевірено на відповідність заявленим вимогам у популярних браузерах (Chrome, Firefox, Safari, Edge, Opera) з різними роздільностями екранів (від 1400px до 2500px). Тестування підтвердило стабільну роботу інтерфейсу, коректність виводу графіків та логіку взаємодії з фільтрами.

– здійснено публікацію проєкту: веб-додаток було завантажено на GitHub та розгорнуто за допомогою GitHub Pages, що дозволяє кожному користувачу

отримати доступ до додатку без потреби встановлення додаткового ПЗ або запуску локального сервера.

Підсумок: розроблений веб-додаток є прикладом ефективного використання сучасних вебтехнологій для візуалізації екологічних даних. Система дозволяє не лише спостерігати за змінами якості повітря, а й підвищує екологічну обізнаність громадян завдяки інтеграції порад та простому інтерфейсу. У подальшому проєкт може бути масштабований за рахунок підключення до реального API та розширення аналітичної частини. Робота пройшла апробацію. За її результатами було опубліковано наступні тези доповідей:

1. Рожко Я.Ю., Залива В.В. Використання цифрових технологій для моніторингу стану довкілля України. Матеріали VI Всеукраїнської науково технічної конференції «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях», 24.04.2025, ДУІКТ, Київ, Україна. С.160-161.
2. Рожко Я.Ю., Залива В.В. Інформаційний вебсайт як інструмент підвищення екологічної обізнаності населення України. Матеріали VI Всеукраїнської науково-технічної конференції «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях», 24.04.2025, ДУІКТ, Київ, Україна. С. 162-163.

ПЕРЕЛІК ПОСИЛАНЬ

1. Міністерство захисту довкілля та природних ресурсів України. Законодавче підґрунтя для організації та здійснення моніторингу навколишнього природного середовища в Україні. URL: https://mepr.gov.ua/wp-content/uploads/2023/02/Monitoring-Green-Paper_15_02_2022.pdf.

2. Investory.news. Сервіси для стеження за екологічними проблемами планети. URL: https://investory.news/servisi-dlya-stezhennya-za-ekologichnimi-problemami-planeti/ (дата звернення: 01.06.2025).

3. ResearchGate. Air Quality Forecasting with the Support of Machine Learning. URL: https://www.researchgate.net/publication/374748191_AIR_QUALITY_FORECASTING_WITH_THE_SUPPORT_OF_MACHINE_.

4. Команда підтримки реформ Міндовкілля. Моніторинг довкілля: аналітична записка щодо стану та перспектив розвитку державної системи моніторингу довкілля. Київ, 2022. 112 с. URL: https://mepr.gov.ua/wp-content/uploads/2023/02/Monitoring-Green-Paper_15_02_2022.pdf (дата звернення: 01.06.2025).

5. Chart.js. Documentation. URL: https://www.chartjs.org/.

6. Mozilla Developer Network. HTML Reference. URL: https://developer.mozilla.org/en-US/docs/Web/HTML.

7. Mozilla Developer Network. CSS Reference. URL: https://developer.mozilla.org/en-US/docs/Web/CSS.

8. JavaScript.info. Document Object Model. URL: https://uk.javascript.info/document.

9. Mozilla Developer Network. Introduction to JSON. URL: https://developer.mozilla.org/en-US/docs/Learn_web_development/Core/Scripting/

10. GitHub. Code Hosting Platform. URL: https://github.com/

11. SVG Repo. Air Anatomy Icon. URL: https://www.svgrepo.com/svg/418326/air-anatomy-lungs

12. SaveEcoBot. Інтерактивна карта забруднення. URL: https://www.saveecobot.com/

13. AQICN. Real-time Air Quality Index. URL: https://aqicn.org/here/

14. BreezoMeter. Air Quality API Documentation. URL: https://docs.breezometer.com/

15. OpenAQ. Open Air Quality Data. URL: https://openaq.org/

16. Figma. Help Center. URL: https://help.figma.com/hc/en-us

17. Лазоренко-Гевель Н., Галіус І., Зацерковний В. та ін. Особливості створення геоінформаційного забезпечення для моніторингу стану довкілля // Вісник Київського національного університету імені Тараса Шевченка. Геологія. 2020. №93. С. 100–105. URL: https://geology.bulletin.knu.ua/article/download/972/777/2833.

18. Міністерство захисту довкілля та природних ресурсів України. Екологічна свідомість українців та довкілля: аналітичне дослідження. Київ, 2020. URL: https://epl.org.ua/wp-content/uploads/2020/12/ekosvidomist.pdf

19. Державне агентство водних ресурсів України. Розробка еколого-інформаційного забезпечення морської екології Чорного моря: звіт про науково-дослідну роботу. Київ, 2020. URL: https://sea.gov.ua/img/reports/2020/report2020_theme8.pdf

20. Parhomenko Y.M., Kyslun O.V. Використання інформаційних технологій в екології та процесах охорони навколишнього середовища // Науковий вісник: Інформатика та енергетика. 2024. №9(40). С. 15–20. URL: https://mapiea.kntu.kr.ua/pdf/9%2840%29_I/3.pdf (дата звернення: 01.06.2025).

ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Розробка інформаційного сайту для візуалізації даних про забруднення навколишнього природного середовища Україниим

Виконав: студент групи ПД -43 Рожко Ярослав Юрійович

Керівник: Залива Віталій Вікторович,
старший викладач кафедри, доктор філософії (PhD)

Київ – 2025

МЕТА, ОБ’ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

- **Мета роботи** : спрощення візуалізації якості зміни повітря в Україні
- **Об’єкт дослідження:** процес візуалізації змін якості повітря в Україні
- **Предмет дослідження** : розробка адаптивного веб -сайту на основі JavaScript, HTML, CSS та візуалізаційних бібліотек який дозволяє відстежувати зміни якості повітря в Україні можливості фільтрації за регіонами та рекомендації для громадян .

ЗАДАЧІ КВАЛІФІКАЦІЙНОЇ РОБОТИ

1. Провести аналіз вплив забруднення повітря на здоров’я
2. Провести аналіз існуючих веб-ресурсів екологічного моніторингу
3. Визначення функціональних і не функціональних вимоги
4. Вибрати технічні інструменти для реалізації
5. Спроєктувати архітектуру веб-сайту
6. Реалізувати верстку адаптивного інтерфейсу
7. Підключити інструменти візуалізації даних (графік)
8. Інтегрувати екологічні дані через API
9. Протестувати на відповідність вимогам

3

АНАЛІЗ АНАЛОГІВ

Платформа	Візуалізація	API	Фільтрація за регіонами	Рекомендації для громадян	Адаптивність	Графіки зміну стану повітря
SaveEcoBot	+	+	Частково	-	+	-
AQICN	+	+	Обмежено	Частково	+	-
Breezometer	+	+	+	+	+	-
OpenAQ	-	+	-	-	-	-
AirInfo	+	+	+	+	+	+

4

ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Функціональні:

- Інтерактивна карта забруднення (посилання наSaveEcoBot)
- Побудова графіка AQI по регіонах
- Вибір місяця, області, міста та дати для створення графіків
- Динамічне оновлення фільтрів
- Поради залежно від рівня AQI
- Блок з екологічними порадами

Нефункціональні:

- Адаптивність інтерфейсу (для різних розширень монітору від 1400px до 2500 px)
- Підтримка популярних браузерів(Chrome, Firefox, Safari, Edge, Opera)
- Продуктивність побудови графіка(не більше 2 секунд)
- Легка навігація

Нефункціональні:

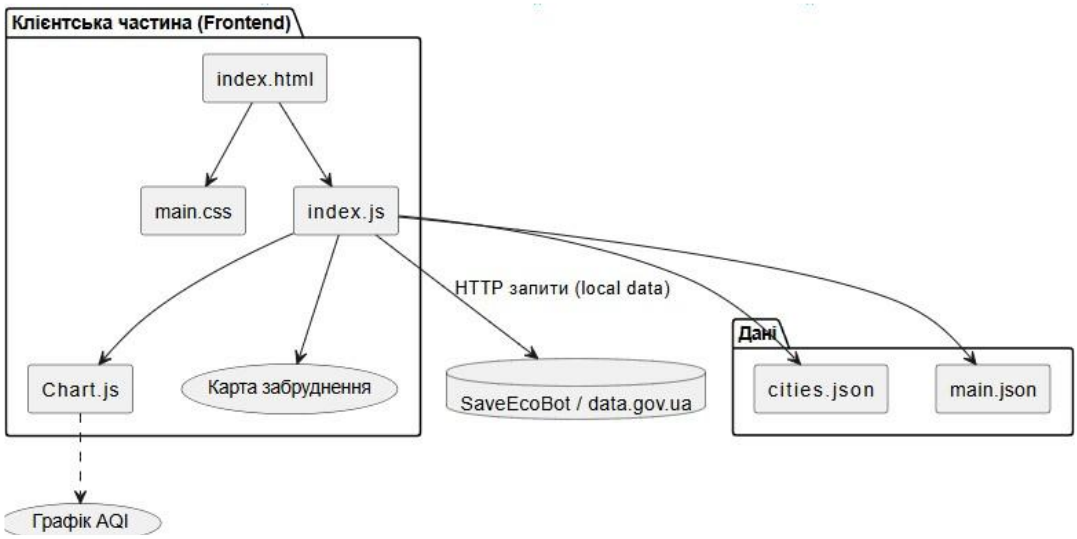
- Адаптивність інтерфейсу(для різних розширень монітору від 1400px до 2500 px)
- Підтримка популярних браузерів(Chrome, Firefox, Safari, Edge, Opera)
- Продуктивність побудови графіка(не більше 2 секунд)
- Легка навігація

5

ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ

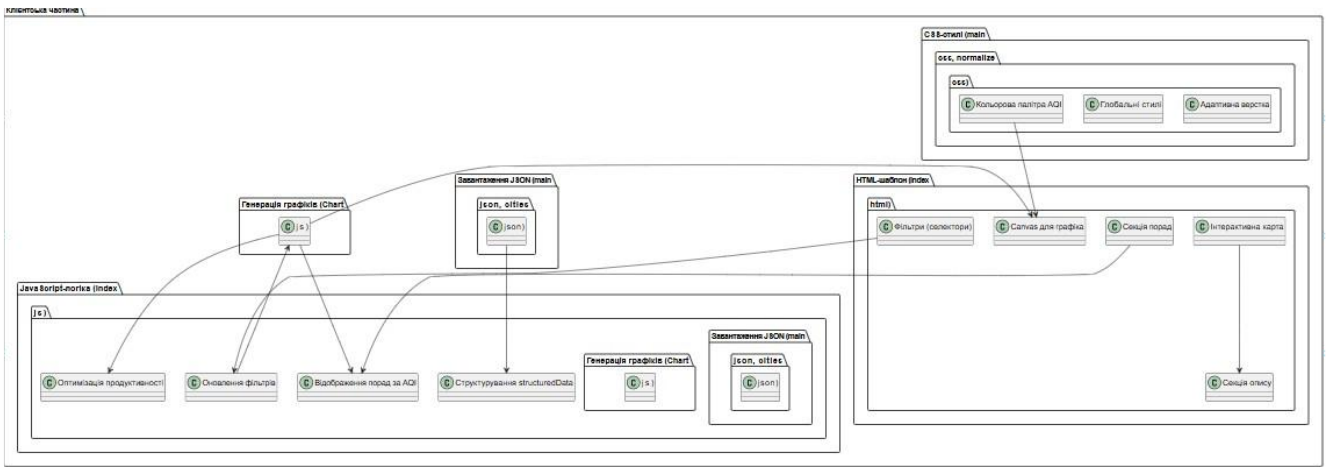
6

ДІАГРАМА АРХІТЕКТУРИ СИСТЕМНИХ КОМПОНЕНТІВ



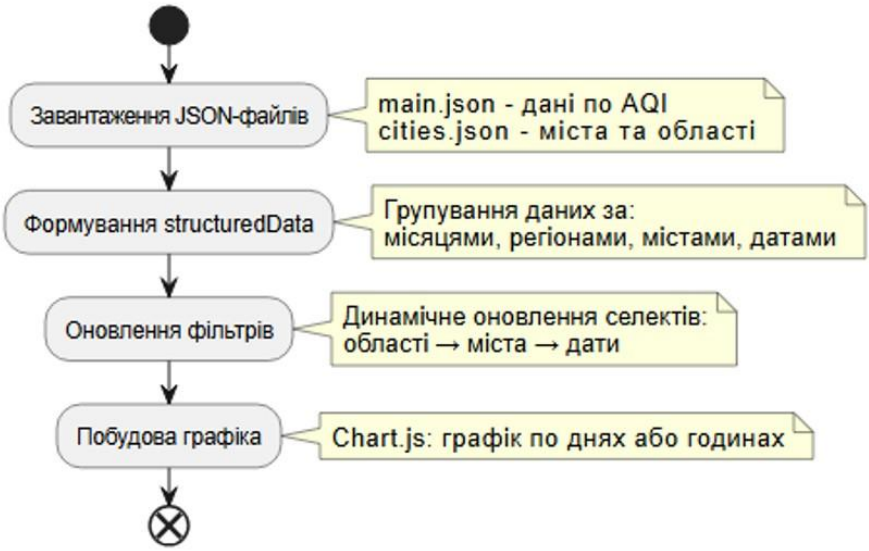
7

МОДУЛЬНА СТРУКТУРА КЛІЄНТСЬКОЇ ЧАСТИНИ ВЕБ-САЙТУ



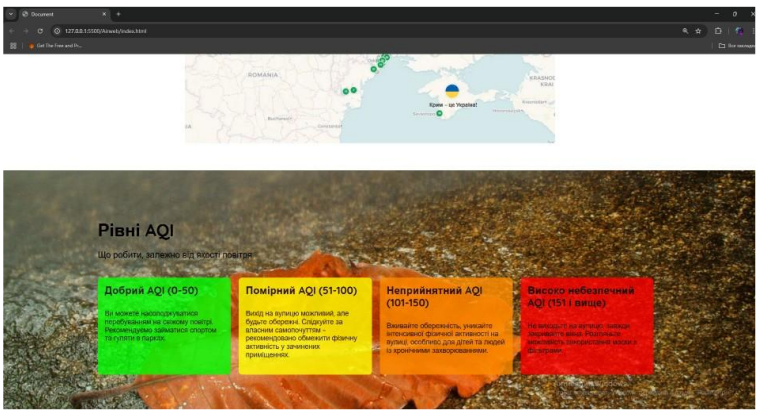
8

АЛГОРИТМ РОБОТИ СТВОРЕННЯ ГРАФІКУ



9

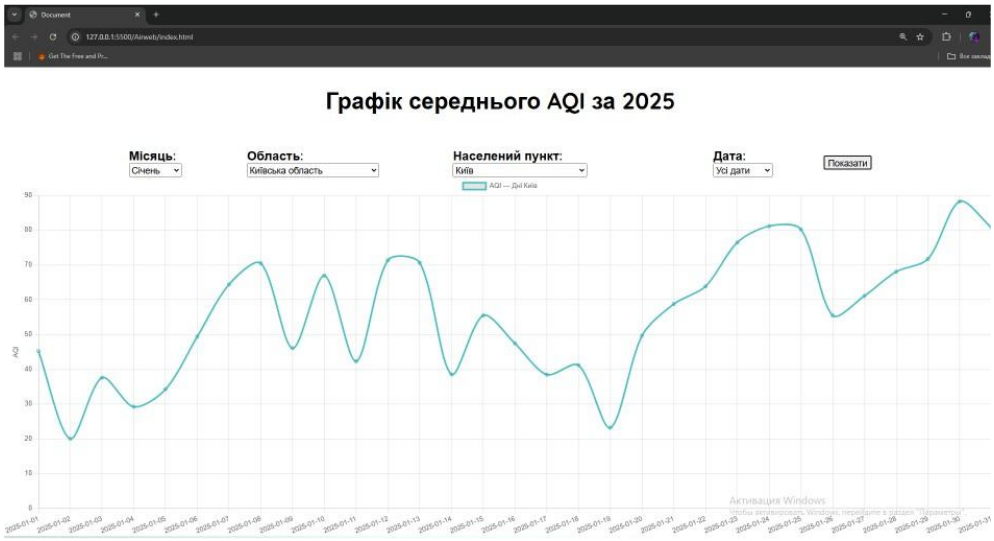
ЕКРАННІ ФОРМИ



Рівні AQI

10

ЕКРАННІ ФОРМИ



Графік AQI

11

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

1. Рожко Я.Ю., Залива В.В. Використання цифрових технологій для моніторингу стану довкілля України. Матеріали VI Всеукраїнської науково технічної конференції «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях», 24.04.2025, ДУІКТ, Київ, Україна. С.160-161.
2. Рожко Я.Ю., Залива В.В. Інформаційний вебсайт як інструмент підвищення екологічної обізнаності населення України. Матеріали VI Всеукраїнської науково-технічної конференції «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях», 24.04.2025, ДУІКТ, Київ, Україна. С. 162-163.

ВИСНОВКИ

1. Проаналізовано предметну область: досліджено сучасні сервіси з моніторингу якості повітря (зокрема SaveEcoBot, AQICN, OpenAQ, Breezometer), їхні можливості, переваги та недоліки. Проведено аналіз функціональних вимог користувачів, цільової аудиторії та типових сценаріїв використання таких систем.
2. Визначено вимоги до програмного забезпечення: сформовано перелік функціональних (інтерактивна карта, побудова графіків, фільтрація даних, екологічні поради) та нефункціональних (адаптивність, продуктивність, підтримка браузерів, легка навігація) вимог до веб-додатку.
3. Розроблено архітектуру клієнтської частини: створено модульну структуру додатку з використанням HTML5, CSS3, JavaScript та бібліотеки Chart.js. Реалізовано систему вкладених фільтрів та структуру structuredData, яка забезпечує швидкий доступ до даних у форматі місяць → область → місто → дата.
4. Створено функціональний прототип: реалізовано систему завантаження локальних JSON-файлів (main.json, cities.json), динамічну побудову графіків, реактивне оновлення фільтрів, виведення порад залежно від рівня AQI та екологічних рекомендацій. Забезпечено продуктивність побудови графіка (менше 2 секунд).
5. Проведено тестування: веб-додаток перевірено на відповідність заявленим вимогам у популярних браузерах (Chrome, Firefox, Safari, Edge, Opera) з різними роздільностями екранів (від 1400px до 2500px). Тестування підтвердило стабільну роботу інтерфейсу, коректність виводу графіків та логіку взаємодії з фільтрами.
6. Здійснено публікацію проєкту: веб-додаток було завантажено на GitHub та розгорнуто за допомогою GitHub Pages, що дозволяє кожному користувачу отримати доступ до додатку без потреби встановлення додаткового ПЗ або запуску локального сервера.

ДОДАТОК Б. ЛІСТИНГИ ПРОГРАМНИХ МОДУЛІВ

Вихідний код файлу index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Document</title>
<link rel="stylesheet" href="./style/normalize.css">
<link rel="stylesheet" href="./style/main.css">
<link rel="preconnect" href="https://fonts.googleapis.com">
<link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
<link href="https://fonts.googleapis.com/css2?family=Quicksand:wght@300..700&display=swap" rel="stylesheet">
<link rel="preconnect" href="https://fonts.googleapis.com">
<link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
<link href="https://fonts.googleapis.com/css2?family=Cabin:ital,wght@0,400..700;1,400..700&family=Quicksand:wght@300..700&display=swap" rel="stylesheet">
<script src="https://cdn.jsdelivr.net/npm/chart.js"></script>


</head>
<body>
<header class="head">
<div class="container">
<a href="#">
<img class="head__logo" src="./img/logo.svg" alt="">
</a>
</div>


</header>
<main>
<section class="hero">
<div class="container">
<div class="hero__wraper">
<img class="logo" src="./img/logo.svg" alt="">
<div class="hero__content">
<h2 class="hero__title">Якість повітря</h2>
<p class="hero__text">Якість повітря безпосередньо впливає на наше здоров'я, самочувствие, довголіття. Забруднене  повітря може спричиняти захворювання органів дихання, алергії, проблеми з серцево-судинною системою. Чисте повітря забезпечує краще життя, підвищує продуктивність і укріплює імунітет. Боротьба з забрудненням повітря — це не тільки екологічна проблема, а й питання нашого здоров'я і благополуччя.</p>
</div>
</div>
</div>
</section>

<section class="map container">
<h2 class="map__title"> Карта забруднення повітря україни</h2>
<a target="_blank" href="https://www.saveecobot.com/maps#6/48.727/30.828/aqi">

<img class="map__img" src="./img/map.webp" alt="">
</a>


</section>

</section>
<section class="info ">
<div class="container">
<h2 class="info__title">Рівні AQI</h2>
<p class="info__subtitle">Що робити, залежно від якості повітря</p>
<div class="info__content">
<div class="info__content-card">
<h3 class="info__content-card-title  container__card-info">Добрий AQI (0-50)</h3>
<p class="info__content-card-subtitle container__card-info"> Ви можете насолоджуватися перебуванням на
свіжому повітрі. Рекомендуємо займатися спортом та гуляти в парках.</p>
</div>
<div class="info__content-card">
<h3 class="info__content-card-title  container__card-info">Помірний AQI (51-100)</h3>
<p class="info__content-card-subtitle container__card-info">Вихід на вулицю можливий, але будьте
обережні. Слідкуйте за власним самопочуттям – рекомендовано обмежити фізичну активність у зачинених
приміщеннях.</p>
</div>
<div class="info__content-card">
<h3 class="info__content-card-title  container__card-info">Неприйнятний AQI (101-150)</h3>
<p class="info__content-card-subtitle container__card-info"> Вживайте обережність, уникайте інтенсивної
фізичної активності на вулиці, особливо для дітей та людей із хронічними захворюваннями.
</p>
</div>
<div class="info__content-card">
<h3 class="info__content-card-title  container__card-info">Високо небезпечний AQI (151 і вище)</h3>
<p class="info__content-card-subtitle container__card-info">Не виходьте на вулицю, завжди
```

```
                закривайте
                вікна. Розгляньте можливість
            використання маски з фільтрами.</p>
                        </div>
                    </div>
                </section>
            <section class="graf ">
        <h2 class="graf__title container ">Графік
            середнього AQI за 2025</h2>

        <div class="graf__wraper container">
                    <div>
                        <label for="month"
            class="text__title">Місяць:</label>
                        <select id="month">
            <option value="January">Січень</option>
            <option value="February">Лютий</option>
            <option value="March">Березень</option>
                        </select>
                    </div>
                    <div>


                        <label for="region"
            class="text__title">Область:</label>
                        <select id="region">
            <option value="">Всі області</option>
            <option value="Дніпропетровська
область">Дніпропетровська область</option>
            <option value="Івано-Франківська
область">Івано-Франківська область</option>
            <option value="Тернопільська
область">Тернопільська область</option>
            <option value="Рівненська
область">Рівненська область</option>
            <option value="Київська
область">Київська область</option>
            <option value="Одеська область">Одеська
                область</option>
            <option value="Запорізька
область">Запорізька область</option>
            <option value="Львівська
область">Львівська область</option>
            <option value="Донецька
область">Донецька область</option>
            <option value="Харківська
область">Харківська область</option>
            <option value="Хмельницька
область">Хмельницька область</option>
            <option value="Чернівецька
область">Чернівецька область</option>
            <option value="Сумська
область">Сумська область</option>
            <option value="Полтавська
область">Полтавська область</option>
            <option value="Автономна Республіка
Крим">Автономна Республіка Крим</option>
            <option value="Вінницька
область">Вінницька область</option>
            <option value="Волинська
область">Волинська область</option>
            <option value="Житомирська
область">Житомирська область</option>
            <option value="Закарпатська
область">Закарпатська область</option>
            <option value="Кіровоградська
область">Кіровоградська область</option>
            <option value="Луганська
область">Луганська область</option>
            <option value="Миколаївська
область">Миколаївська область</option>
            <option value="Херсонська
область">Херсонська область</option>
            <option value="Черкаська
область">Черкаська область</option>
            <option value="Чернігівська
область">Чернігівська область</option>
                        </select>
                    </div>

        <div id="city-container" style="display:none;">

                        <label for="city"
            class="text__title">Населений пункт:</label>
                        <select id="city">
            <option value="">Всі населені
                пункти</option>
                        </select>
                    </div>
        <div id="date-container" style="display:none;">
                        <label for="date"
            class="text__title">Дата:</label>
                        <select id="date">
            <option value="">Всі дати</option>
                        </select>
                    </div>

        <button id="show">Показати</button>
                    </div>


        <canvas id="aqiChart" width="800"
            height="300"></canvas>

        <script src="./script/index.js"></script>
                    </section>

                    <section class="design">
                        <div class="container">

        <h2 class="design__title">Для покращення
            якості повітря ви можете:</h2>
                        <div class="design__wraper">
            <div class="design__wraper-position">
                            <h2 class="info__content-card-
        title  container__card-info">Підтримувати зелені
                насадження</h2>
                        <p class="info__content-card-subtitle
        container__card-info">Використовуйте громадський
            транспорт, велосипеди або ходіть пішки.</p>

                    </div>
                <div class="design__wraper-position">
                            <h2 class="info__content-card-
        title  container__card-info">Зменшити використання
                автомобіля</h2>
                        <p class="info__content-card-subtitle
        container__card-info">Деревця і рослини очищають
                повітря від забруднень.</p>
                    </div>
                <div class="design__wraper-position">
                            <h2 class="info__content-card-
```

```
title  container__card-info">Слідкувати за споживанням</h2>
                <p class="info__content-card-subtitle container__card-info">Вибирайте екологічно чисті продукти та матеріали, щоб зменшити викиди.</p>
            </div>
        </div>
    </div>


</section>


</main>
<footer class="foot">
<a class="foot__text foot__link" target="_blank" href="https://data.gov.ua/dataset/51558857-4f00-4166-bd0b-152e8e637434">
<p>Первинні показники якості атмосферного повітря</p>
</a>
<p class="foot__text">AirQualityWeb@gmail.com</p>
</footer>
</body>
</html>
```

Вихідний код файлу main.css

```
@layer base, layout, components, utilities;
@layer base {
:root {
--color-bg-main: #DBF3ED;
--color-bg-white: #ffffff;
--color-text-primary: #000000;
--color-text-secondary: #0F8277;

--color-card-1: #00FF00;
--color-card-2: #FFFF00;
--color-card-3: #FF8C00;
--color-card-4: #FF0000;

--color-hover-link: blue;
}

body {
font-family: "Quicksand", sans-serif;
}
}

@layer layout {
.container {
width: 1296px;
margin: 0 auto;
}

.hero,
.info,
.design,
.graf,
.foot {
background-repeat: no-repeat;
background-size: cover;
background-position: center;
}

.hero__wraper,
.graf__wraper {
display: flex;
align-items: center;
column-gap: 30px;
}

.design__wraper {
display: grid;
grid-template-columns: repeat(3, 1fr);
gap: 20px;
}

.info__content {
display: grid;
grid-template-columns: repeat(4, 1fr);
}
}

@layer components {
.text__title {
font-size: 22px;
font-weight: 600;
line-height: 26px;
margin-bottom: 32px;
}

.head {
background-color: var(--color-bg-main);
opacity: 0.5;
position: absolute;
left: 0;
right: 0;
}

.head__logo {
width: 42px;
padding-block: 15px;
}

.logo {
width: 72px;
margin-top: 15px;
position: absolute;
top: 20px;
right: 50px;
}

.hero {
background-image: url(../img/foggy-forest-landscape-0410-5710001.webp);
padding-block: 100px;
}

.hero__wraper {
width: 1296px;
height: 602px;
```

```
background-color: var(--color-bg-main);
margin: 0 auto;
margin-top: 31px;
border-radius: 12px;
position: relative;
opacity: 0.5;
}

.hero__content {
width: 712px;
padding-inline: 94px 110px;
}

.hero__title {
font-size: 56px;
font-weight: 700;
line-height: 67px;
color: var(--color-text-primary);
margin-bottom: 32px;
}

.hero__text {
font-size: 22px;
font-weight: 600;
line-height: 26px;
margin-bottom: 32px;
}

.graf {
width: 100%;
height: 800px;
background-color: var(--color-bg-white);
}

.info {
padding-block: 80px;
background-image: url(../img/nature-1384554.webp);
}

.info__content-card {
width: 306px;
border-radius: 8px;
position: relative;
}

.container__card-info {
width: 274px;
margin-inline: auto;
}

.info__content-card-title {
font-size: 24px;
font-weight: 700;
line-height: 28px;
color: var(--color-text-primary);
margin-top: 16px;
}

.info__content-card-subtitle {
font-size: 16px;
font-weight: 500;
line-height: 19px;
color: var(--color-text-primary);
margin-top: 28px;
padding-bottom: 20px;
}

.info__content-card-text {
font-size: 16px;
font-weight: 500;
line-height: 19px;
color: var(--color-text-secondary);
position: absolute;
top: 93%;
left: 16px;
}

.info__title {
font-size: 40px;
font-weight: 700;
line-height: 48px;
margin-bottom: 12px;
}

.info__subtitle {
font-size: 20px;
font-weight: 500;
line-height: 24px;
margin-bottom: 40px;
}

.design {
background-image: url(../img/fog-2-1410205.webp);
padding-top: 82px;
position: relative;
padding-bottom: 100px;
}

.design__title {
font-size: 40px;
font-weight: 700;
line-height: 48px;
padding-block: 19px;
margin-bottom: 40px;
text-align: center;
}

.design__title__wraper {
margin-top: 38px;
display: flex;
flex-wrap: wrap;
column-gap: 46px;
padding-bottom: 138px;
}

.design__wraper-position {
background-color: var(--color-bg-white);
border-radius: 10%;
margin-bottom: 70px;
margin-inline: 30px;
}

.graf__title,
.map__title {
margin-block: 20px;
font-size: 40px;
font-weight: 700;
line-height: 48px;
padding-block: 19px;
margin-bottom: 40px;
text-align: center;
}
```

```
.map__img {
margin-left: 200px;
margin-bottom: 60px;
}

.foot {
background-color: var(--color-bg-main);
padding-block: 20px;
display: flex;
justify-content: center;
column-gap: 160px;
}

.foot__text {
font-size: 20px;
font-weight: 500;
line-height: 24px;
color: var(--color-text-primary);
}

.foot__link:hover {
color: var(--color-hover-link);
}

.info__content-card:nth-child(1) {
background-color: var(--color-card-1);
opacity: 0.8;
}

.info__content-card:nth-child(2) {
background-color: var(--color-card-2);
opacity: 0.8;
}

.info__content-card:nth-child(3) {
background-color: var(--color-card-3);
opacity: 0.8;
}

.info__content-card:nth-child(4) {
background-color: var(--color-card-4);
opacity: 0.8;
}
}

@layer utilities {
.text-center {
text-align: center;
}

.mb-40 {
margin-bottom: 40px;
}

.pb-20 {
padding-bottom: 20px;
}
}
```

Вихідний код файлу normalize.css

```
/*! normalize.css v8.0.1 | MIT License |
github.com/necolas/normalize.css */

/* Document
========================================
======================================== */

/**
* 1. Correct the line height in all browsers.
* 2. Prevent adjustments of font size after orientation
changes in iOS.
*/

html {
line-height: 1.15; /* 1 */
-webkit-text-size-adjust: 100%; /* 2 */
}

/* Sections
========================================
======================================== */

/**
* Remove the margin in all browsers.
*/

body {
margin: 0;
}

/**
* Render the `main` element consistently in IE.
*/

main {
display: block;
}

/**
* Correct the font size and margin on `h1` elements
within `section` and
* `article` contexts in Chrome, Firefox, and Safari.
*/

h1 {
font-size: 2em;
margin: 0.67em 0;
}

/* Grouping content
========================================
======================================== */

/**
* 1. Add the correct box sizing in Firefox.
* 2. Show the overflow in Edge and IE.
*/

hr {
box-sizing: content-box; /* 1 */
height: 0; /* 1 */
overflow: visible; /* 2 */
}
```

```
/**
 * 1. Correct the inheritance and scaling of font size in
 *    all browsers.
 * 2. Correct the odd `em` font sizing in all browsers.
 */

pre {
  font-family: monospace, monospace; /* 1 */
  font-size: 1em; /* 2 */
}

/* Text-level semantics
   ========================================
   ======================================== */

/**
 * Remove the gray background on active links in IE 10.
 */

a {
  background-color: transparent;
}

/**
 * 1. Remove the bottom border in Chrome 57-
 * 2. Add the correct text decoration in Chrome, Edge,
 *    IE, Opera, and Safari.
 */

abbr[title] {
  border-bottom: none; /* 1 */
  text-decoration: underline; /* 2 */
  text-decoration: underline dotted; /* 2 */
}

/**
 * Add the correct font weight in Chrome, Edge, and
 *    Safari.
 */

b,
strong {
  font-weight: bolder;
}

/**
 * 1. Correct the inheritance and scaling of font size in
 *    all browsers.
 * 2. Correct the odd `em` font sizing in all browsers.
 */

code,
kbd,
samp {
  font-family: monospace, monospace; /* 1 */
  font-size: 1em; /* 2 */
}

/**
 * Add the correct font size in all browsers.
 */

small {
  font-size: 80%;
}

/**
 * Prevent `sub` and `sup` elements from affecting the
 *    line height in
 *    all browsers.
 */

sub,
sup {
  font-size: 75%;
  line-height: 0;
  position: relative;
  vertical-align: baseline;
}

sub {
  bottom: -0.25em;
}

sup {
  top: -0.5em;
}

/* Embedded content
   ========================================
   ======================================== */

/**
 * Remove the border on images inside links in IE 10.
 */

img {
  border-style: none;
}

/* Forms
   ========================================
   ======================================== */

/**
 * 1. Change the font styles in all browsers.
 * 2. Remove the margin in Firefox and Safari.
 */

button,
input,
optgroup,
select,
textarea {
  font-family: inherit; /* 1 */
  font-size: 100%; /* 1 */
  line-height: 1.15; /* 1 */
  margin: 0; /* 2 */
}

/**
 * Show the overflow in IE.
 * 1. Show the overflow in Edge.
 */

button,
input { /* 1 */
  overflow: visible;
}

/**
 * Remove the inheritance of text transform in Edge,
```

```
Firefox, and IE.
* 1. Remove the inheritance of text transform in
Firefox.
*/

button,
select { /* 1 */
text-transform: none;
}

/**
* Correct the inability to style clickable types in iOS
and Safari.
*/

button,
[type="button"],
[type="reset"],
[type="submit"] {
-webkit-appearance: button;
}

/**
* Remove the inner border and padding in Firefox.
*/

button::-moz-focus-inner,
[type="button"]::-moz-focus-inner,
[type="reset"]::-moz-focus-inner,
[type="submit"]::-moz-focus-inner {
border-style: none;
padding: 0;
}

/**
* Restore the focus styles unset by the previous rule.
*/

button:-moz-focusring,
[type="button"]:-moz-focusring,
[type="reset"]:-moz-focusring,
[type="submit"]:-moz-focusring {
outline: 1px dotted ButtonText;
}

/**
* Correct the padding in Firefox.
*/

fieldset {
padding: 0.35em 0.75em 0.625em;
}

/**
* 1. Correct the text wrapping in Edge and IE.
* 2. Correct the color inheritance from `fieldset`
elements in IE.
* 3. Remove the padding so developers are not caught
out when they zero out
*    `fieldset` elements in all browsers.
*/

legend {
box-sizing: border-box; /* 1 */
color: inherit; /* 2 */
display: table; /* 1 */
max-width: 100%; /* 1 */
padding: 0; /* 3 */
white-space: normal; /* 1 */
}

/**
* Add the correct vertical alignment in Chrome,
Firefox, and Opera.
*/

progress {
vertical-align: baseline;
}

/**
* Remove the default vertical scrollbar in IE 10+.
*/

textarea {
overflow: auto;
}

/**
* 1. Add the correct box sizing in IE 10.
* 2. Remove the padding in IE 10.
*/

[type="checkbox"],
[type="radio"] {
box-sizing: border-box; /* 1 */
padding: 0; /* 2 */
}

/**
* Correct the cursor style of increment and decrement
buttons in Chrome.
*/

[type="number"]::-webkit-inner-spin-button,
[type="number"]::-webkit-outer-spin-button {
height: auto;
}

/**
* 1. Correct the odd appearance in Chrome and Safari.
* 2. Correct the outline style in Safari.
*/

[type="search"] {
-webkit-appearance: textfield; /* 1 */
outline-offset: -2px; /* 2 */
}

/**
* Remove the inner padding in Chrome and Safari on
macOS.
*/

[type="search"]::-webkit-search-decoration {
-webkit-appearance: none;
}

/**
* 1. Correct the inability to style clickable types in iOS
and Safari.
* 2. Change font properties to `inherit` in Safari.
```

```
*/

::-webkit-file-upload-button {
-webkit-appearance: button; /* 1 */
font: inherit; /* 2 */
}

/* Interactive
========================================
======================================== */

/*
* Add the correct display in Edge, IE 10+, and Firefox.
*/

details {
display: block;
}

/*
* Add the correct display in all browsers.
*/

summary {
display: list-item;
}

/* Misc
========================================
======================================== */

/**
* Add the correct display in IE 10+.
*/

template {
display: none;
}

/**
* Add the correct display in IE 10.
*/

[hidden] {
display: none;
}
```

Вихідний код файлу index.js

```
chartInstance = null;
function renderChart(labels, values, labelTitle) {
const canvas = document.getElementById("aqiChart");

// Якщо існує графік, знищити перед створенням нового
if (chartInstance !== null && typeof chartInstance.destroy === "function") {
chartInstance.destroy();
}

const ctx = canvas.getContext("2d");

chartInstance = new Chart(ctx, {
type: "line",
data: {
labels: labels,
datasets: [
{
label: `AQI — ${labelTitle}`,
data: values,
borderColor: "rgb(75, 192, 192)",
fill: false,
tension: 0.3,
},
],
},
options: {
responsive: true,
animation: false,
scales: {
y: {
beginAtZero: true,
title: {
display: true,
text: "AQI",
},
},
x: {
title: {
display: true,
text: labelTitle.includes("Години") ? "Година" : "Дата",
},
},
},
},
});
}

let structuredData = {
January: {},
February: {},
March: {},
};

// Завантаження та побудова структури
async function loadData() {
const citiesRes = await fetch("./data/cities.json");
const mainRes = await fetch("./data/main.json");
const cities = await citiesRes.json();
const data = await mainRes.json();

const cityMap = {};
cities.forEach((city) => {
cityMap[city.id] = {
city_name: city.city_name,
region_name: city.region_name,
};
});

const months = {
"2025-01": "January",
"2025-02": "February",
"2025-03": "March",
};
```

```
data.forEach((entry) => {
const cityInfo = cityMap[entry.city_id];
if (!cityInfo) return;

const monthKey = entry.logged_at.slice(0, 7);
const month = months[monthKey];
if (!month) return;

const date = entry.logged_at.slice(0, 10);
const { region_name, city_name } = cityInfo;

if (!structuredData[month][region_name]) {
structuredData[month][region_name] = {};
}

if (!structuredData[month][region_name][city_name])
{
structuredData[month][region_name][city_name] =
{};
}

if
(!structuredData[month][region_name][city_name][date]
) {
structuredData[month][region_name][city_name][dat
e] = [];
}

structuredData[month][region_name][city_name][date]
.push({
aqi: entry.aqi,
pm25: entry.pm25,
logged_at: entry.logged_at,
});
});
}

// Оновлення select'ів
function populateSelect(select, values, defaultOption =
"Усі") {
select.innerHTML = `<option
value="">${defaultOption}</option>`;
values.forEach((val) => {
const option = document.createElement("option");
option.value = val;
option.textContent = val;
select.appendChild(option);
});
}

// Логіка фільтрації
function setupFilters() {
const monthSelect =
document.getElementById("month");
const regionSelect =
document.getElementById("region");
const citySelect = document.getElementById("city");
const dateSelect = document.getElementById("date");

const cityContainer = document.getElementById("city-
container");
const dateContainer = document.getElementById("date-
container");

monthSelect.addEventListener("change", () => {
const month = monthSelect.value;
const regions = Object.keys(structuredData[month] ||
{});
populateSelect(regionSelect, regions, "Усі області");
regionSelect.value = "";
citySelect.innerHTML = "";
dateSelect.innerHTML = "";
cityContainer.style.display = "none";
dateContainer.style.display = "none";
});

regionSelect.addEventListener("change", () => {
const month = monthSelect.value;
const region = regionSelect.value;
const cities = region
? Object.keys(structuredData[month][region] || {})
: [];
populateSelect(citySelect, cities, "Усі населені
пункти");
citySelect.value = "";
cityContainer.style.display = cities.length ? "block" :
"none";
dateContainer.style.display = "none";
dateSelect.innerHTML = "";
});

citySelect.addEventListener("change", () => {
const month = monthSelect.value;
const region = regionSelect.value;
const city = citySelect.value;
const dates = city
? Object.keys(structuredData[month][region][city] ||
{})
: [];
populateSelect(dateSelect, dates, "Усі дати");
dateSelect.value = "";
dateContainer.style.display = dates.length ? "block" :
"none";
});
document.getElementById("show").addEventListener("
click", () => {
const month =
document.getElementById("month").value;
const region =
document.getElementById("region").value;
const city = document.getElementById("city").value;
const date = document.getElementById("date").value;

if (!month || !region || !city) {
alert("Будь ласка, виберіть місяць, область і
місто");
return;
}

const entries =
structuredData[month]?.[region]?.[city];
if (!entries) {
alert("Немає даних для вибраного міста");
return;
}

let labels = [];
let values = [];

if (date) {
// по годинах
```

```
const records = entries[date] || [];
labels = records.map((e) => e.logged_at.slice(11, 16)); // HH:MM
values = records.map((e) => e.aqi);
} else {
// по днях
labels = Object.keys(entries).sort(); // YYYY-MM-DD
values = labels.map((day) => {
const dayEntries = entries[day];
const avg = dayEntries.reduce((sum, e) => sum + e.aqi, 0) / dayEntries.length;
return Math.round(avg * 100) / 100;
});
}
console.log(labels, values);
renderChart(labels, values, date ? `Години ${date}` : `Дні ${city}`);
});
}

// ініціалізація
loadData().then(setupFilters);
```


```
"id": 34681,
"url_slug": "sheptytskyi",
"type_name": "місто",
"city_name": "Шептицький",
"region_name": "Львівська область",
"koatuu": "NULL",
"katottg": "NULL"
},
{
"id": 34682,
"url_slug": "samar",
"type_name": "місто",
"city_name": "Самар",
"region_name": "Дніпропетровська область",
"koatuu": "NULL",
"katottg": "NULL"
}
]
```