МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ПОЛІСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет інформаційних технологій, обліку та фінансів
Кафедра комп'ютерних технологій
і моделювання систем

Кваліфікаційна робота
на правах рукопису

Кравченка Івана Володимировича
(прізвище, ім'я, по-батькові здобувача освіти)

УДК 004.4:504.064.3:574

КВАЛІФІКАЦІЙНА РОБОТА

Розробка модуля обробки даних для систем ландшафтного екологічного моніторингу

(тема роботи)

122 «Комп'ютерні науки»

(шифр і назва спеціальності)

Подається на здобуття освітнього ступеня бакалавр

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

_____ Кравченко І.В.
(підпис, ініціали та прізвище здобувача вищої освіти)

Керівник роботи
Ковальчук М. О.
(прізвище, ім'я, по батькові)
к.п.н., доцент кафедри комп'ютерних технологій
і моделювання систем
(науковий ступінь, вчене звання)

Житомир – 2025

Висновок кафедри _____
за результатами попереднього захисту: _____

Протокол засідання кафедри _____
№ ____ від «_____» _____ 20____ р.

Завідувач кафедри _____

_____ _____ _____
(науковий ступінь, вчене звання) (підпис) (прізвище, ім'я, по батькові)
«_____» _____ 20____ р.

Результати захисту кваліфікаційної роботи

Здобувач вищої освіти _____ захистив (ла)
(прізвище ,ім'я, по батькові)
кваліфікаційну роботу з оцінкою:

сума балів за 100-бальною шкалою _____
за шкалою ECTS _____
за національною шкалою _____

Секретар ЕК

_____ _____ _____
(науковий ступінь, вчене звання) (підпис) (прізвище, ім'я, по батькові)

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API — інтерфейс прикладного програмування

CSV — Comma-Separated Values формат табличних файлів, де значення розділено комами

AQI — Air Quality Index індекс якості повітря

UI — User Interface інтерфейс користувача

Flask — Python-фреймворк для створення вебзастосунків

JSON — JavaScript Object Notation текстовий формат обміну даними між клієнтом і сервером

REST — Representational State Transfer архітектурний стиль вебсервісів

HTTP — HyperText Transfer Protocol протокол передачі гіпертексту

Chart.js — JavaScript-бібліотека для побудови інтерактивних графіків

API key — Унікальний ключ доступу до сервісу WeatherAPI

WeatherAPI — Вебсервіс для отримання погодних та екологічних даних через API

IDEF3 — Методологія побудови функціональних моделей процесів

UML — Unified Modeling Language уніфікована мова моделювання систем

CRUD — Create, Read, Update, Delete базові операції над даними

GET/POST — HTTP-методи для запитів на сервер (отримання/надсилання даних)

АНОТАЦІЯ

Кравченко І. В. Розробка модуля обробки даних для систем ландшафтного екологічного моніторингу - Кваліфікаційна робота на правах рукопису.

Кваліфікаційна робота на здобуття освітнього ступеня бакалавра за спеціальністю 122 – Комп’ютерні науки. – Поліський національний університет, Житомир, 2025.

У дипломній роботі розглянуто процес розробки модуля збору, обробки та візуалізації екологічних даних, зокрема метеорологічних показників, на прикладі інтеграції з відкритим API WeatherAPI. З метою підвищення точності результатів було реалізовано механізми фільтрації шумів, згладжування даних за методом ковзного середнього, а також валідації значень. У системі передбачено логування, збереження у форматі CSV, гнучке REST API для обміну даними з клієнтом і вебінтерфейс для перегляду графіків. Розроблений вебдодаток дозволяє користувачам вибирати міста, діапазони дат і переглядати змінні екологічні показники в динаміці. Робота охоплює етапи архітектурного проєктування, реалізації, тестування та аналізу переваг системи у порівнянні з аналогами.

Екологічний моніторинг, погодні дані, ковзне середнє, rest api, візуалізація.

ABSTRACT

Kravchenko I. V. Development of a data processing module for landscape ecological monitoring systems - Qualification work in the form of a manuscript.
Qualification work for the degree of bachelor in specialty 122 - Computer Science. - Polesie National University, Zhytomyr, 2025.

This thesis explores the development of a data collection, processing, and visualization module for environmental monitoring, focusing on meteorological parameters obtained via the open WeatherAPI. To improve data reliability, the system includes noise filtering mechanisms, moving average smoothing, and validation of collected values. The architecture supports logging, CSV-based storage, a flexible REST API, and a web dashboard for viewing real-time and historical graphs. The developed application allows users to select cities and date ranges, and to analyze environmental indicators interactively. The work covers architectural planning, implementation, testing, and comparison with existing solutions.

Environmental monitoring, weather data, moving average, rest api, visualization.

ЗМІСТ

ВСТУП

У сучасному світі проблема моніторингу стану навколишнього середовища набула особливої актуальності через глобальні зміни клімату, зростання рівня забруднення повітря та нестабільні погодні умови. Забезпечення сталого розвитку територій, охорона здоров’я населення та підвищення рівня екологічної безпеки вимагають впровадження сучасних технологічних рішень для ефективного аналізу екологічних даних. Одним із перспективних напрямів є створення інформаційних систем, що дозволяють збирати, зберігати, обробляти та візуалізувати дані з сенсорів у реальному часі.

У даній роботі розглядається процес розробки системи для ландшафтного екологічного моніторингу, яка опрацьовує дані про температуру, вологість та якість повітря. Особливістю запропонованого підходу є використання публічного погодного API (WeatherAPI) як джерела вхідних даних, обробка отриманої інформації із застосуванням методів фільтрації шумів, обчислення ковзного середнього та збереження результатів у форматі CSV для подальшого аналізу.

Мета роботи полягає в розробці модульної програмної системи, що поєднує засоби збору погодних даних, їх фільтрацію та візуалізацію в зручному веб-інтерфейсі.

Для досягнення поставленої мети необхідно реалізувати такі основні завдання:

- дослідити існуючі системи екологічного моніторингу та можливості відкритих погодних API;
- сформулювати вимоги до функціональності та архітектури майбутньої системи;
- розробити модуль обробки даних з підтримкою фільтрації шумів та алгоритмів згладжування;

- реалізувати механізм збереження даних у форматі CSV;
- побудувати веб-інтерфейс для взаємодії з користувачем, включаючи вибір міста, фільтрацію по часу та візуалізацію результатів за допомогою графіків.

Об’єктом дослідження є процес екологічного моніторингу, предметом — методи автоматизації обробки метеорологічної інформації з відкритих джерел.

Методи дослідження, що застосовуються в роботі, включають аналіз програмного забезпечення, розробку веб-додатків засобами Flask, обробку даних за допомогою мови Python та бібліотек для роботи з таблицями, а також візуалізацію даних із використанням Chart.js.

Запропонована система може бути використана як інструмент для базового моніторингу екологічної ситуації в містах, селищах або локальних екозонах, а також як навчальний приклад для вивчення основ побудови аналітичних веб-систем на основі відкритих API.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Екологічний моніторинг

Екологічний моніторинг — це систематичне спостереження, оцінка та прогнозування змін у стані навколишнього природного середовища. Він є ключовим інструментом забезпечення екологічної безпеки, збереження біорізноманіття, попередження негативного впливу техногенних факторів та адаптації до кліматичних змін. У широкому розумінні екологічний моніторинг охоплює не лише вимірювання фізико-хімічних параметрів середовища (температура, вологість, якість повітря), а й відстеження біологічних індикаторів, аналіз просторових та часових тенденцій, моделювання майбутніх змін [1].

Залежно від масштабу та мети, розрізняють глобальний, національний, регіональний і локальний моніторинг. На глобальному рівні функціонують такі системи, як Всесвітня програма моніторингу навколишнього середовища (GEMS/Water), супутникові платформи NASA (MODIS, Landsat), програма Copernicus Європейського Союзу [2]. Ці системи надають доступ до великих обсягів даних про стан атмосфери, океанів, ґрунтів та біомів. Вони активно використовуються у наукових дослідженнях, зокрема при оцінці глобального потепління, моніторингу пожеж чи танення льодовиків.

На національному рівні в Україні екологічний моніторинг регламентується низкою нормативно-правових актів. Основним є Закон України «Про охорону навколишнього природного середовища» [3], а також підзаконні документи щодо моніторингу атмосферного повітря, водних ресурсів та ґрунтів. Відповідальність за збір та обробку екологічної інформації покладається на державні установи: Державну екологічну інспекцію, Український гідрометеорологічний центр, Держпродспоживслужбу, Центри громадського здоров’я тощо.

Локальні ініціативи включають розгортання автономних сенсорних станцій (наприклад, проєкти EcoCity, SaveDnipro, LUN Misto Air), які збирають дані в реальному часі та публікують результати на інтерактивних мапах. Застосування IoT-технологій дає змогу автоматизувати процес збору даних, зменшити людський фактор та охопити більшу кількість точок спостереження [4].

Інструментально екологічний моніторинг реалізується за допомогою приладів прямої дії (газоаналізатори, метеостанції, датчики вологості, пиломіри тощо), а також цифрових платформ, які агрегують дані з відкритих API або комерційних сервісів. У сучасних системах надзвичайно важливим є етап попередньої обробки даних: очищення від шумів, валідація, згладжування, агрегування по часових інтервалах. Особливе місце займають методи машинного навчання, що дозволяють будувати прогностичні моделі на основі історичних даних [5].

Таким чином, екологічний моніторинг є складним міждисциплінарним процесом, що поєднує екологію, інформатику, метеорологію, статистику та інженерію. Його ефективність залежить від якості вхідних даних, алгоритмів обробки та інтерпретації результатів. У контексті цієї роботи особлива увага приділяється розробці модульної системи збору й аналізу базових метеорологічних параметрів — температури, вологості та індексу якості повітря — з відкритих джерел, з подальшим зберіганням і візуалізацією.

1.2 Сучасні системи збору й аналізу екологічних даних

З відкриттям великої кількості кліматичних і метеорологічних API для загального користування, розробка екологічних систем моніторингу суттєво спростилася. Відтепер немає необхідності встановлювати локальні метеостанції або сенсори для базових параметрів — температура, вологість, атмосферний тиск, якість повітря можуть бути отримані з авторитетних

хмарних джерел. API (Application Programming Interface) виступає як інтерфейс, який дозволяє системам отримувати структуровані дані у форматі JSON, XML або CSV з можливістю параметризації запитів.

Одним із найпопулярніших сервісів є WeatherAPI — платформа, яка надає доступ до поточних, історичних і прогнозних погодних даних. Вона підтримує фільтрацію по містах, координатах, часових діапазонах, а також має окремі параметри для індексу якості повітря (AQI), ультрафіолетового індексу, напрямку вітру, вологості тощо [13]. Перевагою API є чітка документація, підтримка понад 200 тис. локацій і стабільна робота в умовах обмеженої кількості запитів (Free Tier).

Іншою альтернативою є OpenWeatherMap, яка окрім базових погодних показників, дозволяє підключати погодні карти, дані про опади, хмарність, інтенсивність ультрафіолетового випромінювання та інше. Вона також надає прогнозування на 16 днів і погодинні дані для аналізу трендів [14]. Варто зазначити, що OpenWeatherMap більш схильний до використання у великих системах із фокусом на прогноз, ніж на поточну деталізацію.

Weatherstack, Visual Crossing Weather, Climacell (now Tomorrow.io) — ще одні приклади надійних API, які підтримують високу точність погодних даних та дозволяють інтегрувати аналітику без потреби в складному налаштуванні власної сенсорної інфраструктури [15–16].

Важливо, що практично всі ці сервіси мають схожу логіку запитів: ключ API (token), формат відповіді, параметри запиту (місто, дата, координати), і повертають структуровану відповідь з усіма необхідними показниками. Це дозволяє легко створювати програми або вебінтерфейси для автоматичного збору та обробки даних. Наприклад, у межах цього проєкту було реалізовано Python-скрипт, який кожні 30 хвилин надсилає запит до WeatherAPI та зберігає відповідь у форматі CSV.

Ключовим параметром при виборі API є підтримка історичних даних, можливість роботи з часовими інтервалами, а також наявність показників якості повітря (AQI). У випадку WeatherAPI дані про AQI містять концентрації

основних шкідливих речовин: PM2.5, PM10, CO, NO_2, O_3, що дозволяє комплексно аналізувати ситуацію з атмосферним повітрям на конкретній локації [17].

Ще однією важливою перевагою відкритих API є можливість інтеграції з JavaScript-фреймворками (наприклад, Chart.js, Leaflet) для побудови динамічних графіків і мап. Таким чином, API не лише замінює сенсорні пристрої, а й виступає як джерело аналітичних інсайтів у реальному часі.

Однак при використанні відкритих API важливо враховувати низку проблемних аспектів, таких як відсутність точних або актуальних даних для деяких регіонів України (особливо в сільській місцевості), обмеження за кількістю запитів у безкоштовному тарифі, а також відсутність україномовної локалізації, що створює додаткові складнощі при створенні продуктів, орієнтованих на кінцевого користувача в Україні.

Крім того, слід враховувати вимоги нормативно-правового забезпечення. Згідно із Законом України «Про охорону навколишнього природного середовища» [3], держава гарантує право громадян на отримання достовірної інформації про стан довкілля, а органи місцевого самоврядування мають забезпечити систематичний моніторинг та доступність екологічних даних. Це створює підґрунтя для впровадження громадських систем контролю, які можуть базуватись на відкритих API та програмних інструментах із відкритим кодом.

Детальне порівняння згаданих сервісів за ключовими критеріями — підтримка історичних даних, наявність індексу AQI, точність, локалізація, тарифи — наведено в Додатку А (таблиця А.1).

1.3 Аналіз відкритих API для отримання погодних показників

З відкриттям великої кількості кліматичних і метеорологічних API для загального користування, розробка екологічних систем моніторингу суттєво спростилася. Відтепер немає необхідності встановлювати локальні

метеостанції або сенсори для базових параметрів — температура, вологість, атмосферний тиск, якість повітря можуть бути отримані з авторитетних хмарних джерел. API (Application Programming Interface) виступає як інтерфейс, який дозволяє системам отримувати структуровані дані у форматі JSON, XML або CSV з можливістю параметризації запитів.

Одним із найпопулярніших сервісів є WeatherAPI — платформа, яка надає доступ до поточних, історичних і прогнозних погодних даних. Вона підтримує фільтрацію по містах, координатах, часових діапазонах, а також має окремі параметри для індексу якості повітря (AQI), ультрафіолетового індексу, напрямку вітру, вологості тощо [13]. Перевагою API є чітка документація, підтримка понад 200 тис. локацій і стабільна робота в умовах обмеженої кількості запитів (Free Tier).

Іншою альтернативою є OpenWeatherMap, яка окрім базових погодних показників, дозволяє підключати погодні карти, дані про опади, хмарність, інтенсивність ультрафіолетового випромінювання та інше. Вона також надає прогнозування на 16 днів і погодинні дані для аналізу трендів [14]. Варто зазначити, що OpenWeatherMap більш схильний до використання у великих системах із фокусом на прогноз, ніж на поточну деталізацію.

Weatherstack, Visual Crossing Weather, Climacell (now Tomorrow.io) — ще одні приклади надійних API, які підтримують високу точність погодних даних та дозволяють інтегрувати аналітику без потреби в складному налаштуванні власної сенсорної інфраструктури [15–16].

Важливо, що практично всі ці сервіси мають схожу логіку запитів: ключ API (token), формат відповіді, параметри запиту (місто, дата, координати), і повертають структуровану відповідь з усіма необхідними показниками. Це дозволяє легко створювати програми або вебінтерфейси для автоматичного збору та обробки даних. Наприклад, у межах цього проєкту було реалізовано Python-скрипт, який кожні 30 хвилин надсилає запит до WeatherAPI та зберігає відповідь у форматі CSV.

Ключовим параметром при виборі API є підтримка історичних даних, можливість роботи з часовими інтервалами, а також наявність показників якості повітря (AQI). У випадку WeatherAPI дані про AQI містять концентрації основних шкідливих речовин: PM2.5, PM10, CO, NO_2, O_3, що дозволяє комплексно аналізувати ситуацію з атмосферним повітрям на конкретній локації [17].

Ще однією важливою перевагою відкритих API є можливість інтеграції з JavaScript-фреймворками (наприклад, Chart.js, Leaflet) для побудови динамічних графіків і мап. Таким чином, API не лише замінює сенсорні пристрої, а й виступає як джерело аналітичних інсайтів у реальному часі.

1.4 Вимоги до системи моніторингу

Проєктування ефективної системи ландшафтного екологічного моніторингу передбачає врахування низки функціональних, технічних та експлуатаційних вимог. Вони формуються на основі цільового призначення системи, особливостей джерел даних, обраних методів обробки та способів представлення результатів.

Функціональні вимоги:

1. Отримання даних з зовнішніх джерел — система повинна регулярно збирати актуальні погодні показники (температура, вологість, якість повітря) з відкритого API.
2. Аналіз та обробка інформації — реалізація алгоритмів первинної фільтрації, виявлення шумів та згладжування даних (зокрема за допомогою ковзного середнього).
3. Перевірка валідності — вхідні дані мають перевірятися на наявність аномальних або некоректних значень.
4. Зберігання результатів — система повинна підтримувати збереження історичних даних у зручному для аналізу форматі (CSV).

5. Логування — кожна сесія збору та обробки повинна фіксуватися з відмітками часу і статусом операцій.
6. Візуалізація — побудова графіків для аналізу тенденцій у вигляді динамічних елементів (наприклад, Chart.js).
7. Гнучкий вибір міста та періоду — користувач повинен мати можливість обрати місце моніторингу, а також часовий інтервал для перегляду даних.

Технічні вимоги:

- Мінімальні залежності: реалізація на Python з використанням Flask для веб-інтерфейсу та бібліотек Pandas, CSV, Requests.
- Можливість локального запуску: повна працездатність без потреби в хмарних інструментах чи встановленні складного ПЗ.
- Адаптивний вебінтерфейс: відображення даних як на десктопах, так і на мобільних пристроях.
- Безперервний режим збору: можливість запуску процесу автоматичного збору даних у фоновому режимі з інтервалом (наприклад, 30 хвилин).

Експлуатаційні вимоги:

- Простота у використанні (інтуїтивний інтерфейс для перегляду даних).
- Можливість розширення — підтримка нових міст або додаткових показників.
- Стійкість до відсутності інтернет-з'єднання під час виконання окремих операцій (із записом у лог про помилку).

Таким чином, система повинна бути надійною, простою в інтеграції та придатною для масштабування. Її основне завдання — надати користувачам засіб постійного та гнучкого контролю за екологічним станом середовища на основі об'єктивних цифрових даних.

Висновки до розділу 1

У першому розділі було здійснено огляд теоретичних та практичних основ, пов’язаних із ландшафтним екологічним моніторингом. Розглянуто сучасні підходи до збору, обробки та інтерпретації екологічних даних, зокрема температури, вологості та якості повітря. Аналіз існуючих систем і публічних API (зокрема WeatherAPI) показав доцільність їх використання у прикладних розробках без необхідності розгортання власної сенсорної інфраструктури.

Було сформульовано основні вимоги до майбутньої системи, які охоплюють як технічні аспекти, так і функціональні очікування: регулярний збір даних, їх валідація, збереження, логування та візуалізація. Ці вимоги лягли в основу наступного етапу роботи — безпосередньої розробки модуля обробки даних.

РОЗДІЛ 2
ПРОЕКТУВАННЯ МОДУЛЯ ОБРОБКИ ЕКОЛОГІЧНИХ ДАНИХ

2.1 Архітектура модуля

Для ефективної реалізації системи обробки екологічних даних було спроєктовано модульну архітектуру, яка дозволяє розділити функціональність на окремі незалежні компоненти. Такий підхід забезпечує гнучкість, масштабованість та простоту в тестуванні й супроводі проєкту.

Модуль складається з двох основних частин:

1. Скрипт збору та обробки даних — окремий Python-модуль, який періодично надсилає запити до WeatherAPI, отримує метеодані, фільтрує їх, виконує згладжування та записує у файл формату CSV. Він може бути запущений як вручну, так і автоматично за розкладом.
2. Вебінтерфейс (Flask-додаток) — призначений для візуалізації зібраних даних. Він дозволяє користувачу переглядати інформацію по обраних містах, фільтрувати її за діапазоном дат, переглядати графіки температури, вологості, якості повітря із застосуванням ковзного середнього.

Комунікація між модулями реалізується через спільний доступ до CSV-файлів. Завдяки такому підходу забезпечується ізоляція логіки збору даних від фронтенд-частини, що спрощує відладку, заміну окремих блоків та повторне використання компонентів у майбутніх проєктах.

У структурі модуля також передбачено:

- механізм логування процесів збору та обробки;
- структуру для збереження списку міст;
- API-ендпойнти для асинхронної взаємодії з інтерфейсом (JSON-відповіді);
- фільтрацію даних у вебінтерфейсі за початковою та кінцевою датою;

- можливість динамічного додавання нових міст без перезапуску серверу.

Загальна схема архітектури модуля наведена на рисунку нижче.

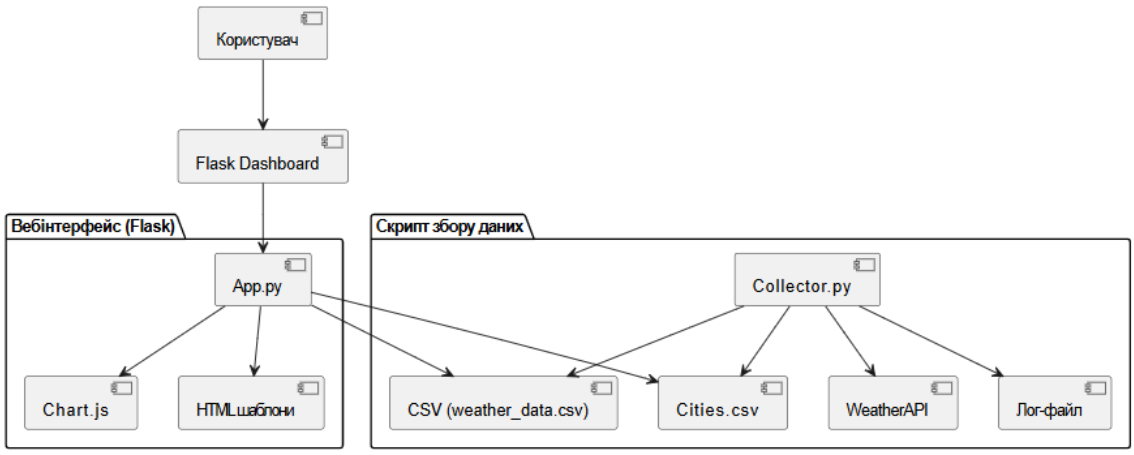

Рисунок 2.1 – Архітектура системи збору та обробки екологічних даних

На основі наведеної архітектури можна детальніше простежити логіку взаємодії компонентів системи. При взаємодії з інтерфейсом користувач ініціює запит на оновлення або перегляд екологічних показників. У відповідь вебмодуль звертається до сховища даних, яке формується окремим Python-скриптом збору. Цей скрипт регулярно надсилає запити до сервісу WeatherAPI, отримує та обробляє відповіді, виконує згладжування ковзним середнім, а потім зберігає результат у форматі CSV. Така взаємодія є асинхронною та дозволяє розмежувати відповідальність між логікою збору й відображенням.

Послідовність взаємодії між компонентами зображена на рисунку 2.2.

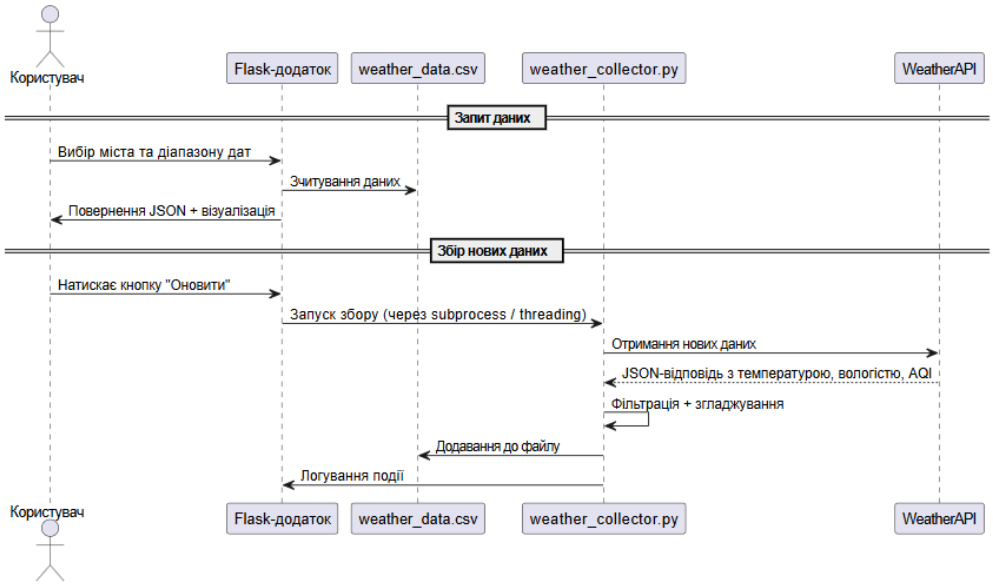

Рисунок 2.2 – Послідовність роботи модуля збору та візуалізації даних

У результаті реалізовано модульну та гнучку систему, в якій кожен компонент виконує свою роль — від збору даних до їх подання у зручному форматі для користувача. Така структура сприяє легкому масштабуванню, спрощенню супроводу та можливості інтеграції з іншими інформаційними системами у сфері екологічного моніторингу.

2.2 Отримання даних із WeatherAPI

Основним джерелом метеорологічних даних у реалізованій системі є відкрита погодна платформа WeatherAPI, яка забезпечує доступ до актуальних та історичних значень температури, вологості, атмосферного тиску, індексу якості повітря (AQI) та інших екологічних показників.

Для взаємодії з API використовуються HTTP-запити з параметрами, які визначають місто (city), формат відповіді (JSON), ключ доступу (API key) та тип запиту (current weather). Нижче наведено приклад структури запиту:

https://api.weatherapi.com/v1/current.json?key=API_KEY&q=Kyiv&aqi=yes

У відповідь система отримує об’єкт JSON, який містить необхідні поля, зокрема:

- location.name – назва міста;
- current.temp_c – температура в градусах Цельсія;
- current.humidity – відносна вологість (%);
- current.air_quality.pm2_5 – концентрація частинок PM2.5;
- current.air_quality.co – рівень оксиду вуглецю (CO);
- current.last_updated – час останнього оновлення.

Збір даних реалізований у вигляді окремого Python-скрипта (collector.py), який виконує наступні дії:

1. Зчитує список міст із файлу cities.csv;
2. Для кожного міста формує HTTP-запит до API;
3. Розбирає відповідь, витягує необхідні параметри;
4. Зберігає результат у форматі CSV з міткою часу (timestamp) та зазначенням міста (city);
5. Веде лог-файл із зазначенням результатів кожного запиту (успішно / помилка).

Процес збору може бути запущений вручну або через кнопку на вебінтерфейсі, яка ініціює асинхронний запит. Крім того, передбачена можливість запуску скрипта автоматично кожні 30 хвилин для регулярного оновлення даних.

Додатково, система враховує можливість тимчасового недоступу API — у такому разі відповідна подія реєструється в логах, а повторний запит виконується пізніше, не блокуючи роботу інтерфейсу.

Завдяки використанню WeatherAPI вдалося уникнути потреби у фізичних сенсорах та значно прискорити етапи тестування, налагодження та експлуатації системи.

2.3 Фільтрація шумів та ковзне середнє як метод згладжування

У процесі збору екологічних даних, навіть при використанні авторитетних джерел як WeatherAPI, виникає необхідність попередньої обробки — через можливі стрибки, аномалії або незначні похибки вимірювань. Для забезпечення більш стабільного та репрезентативного аналізу доцільним є застосування методів згладжування, зокрема ковзного середнього (moving average).

Виявлення та фільтрація шумів

Шумами в контексті метеоданих вважаються:

- Різкі одиничні стрибки показників (наприклад, температура зросла на 10 °C за 5 хвилин);
- Значення, що виходять за фізично допустимі межі (наприклад, вологість понад 100%);
- Відсутні або нульові значення при очікуваних даних.

На першому етапі обробки система відкидає:

- Порожні записи;
- Значення, що не відповідають діапазону [–50; +60] для температури;
- Значення вологості поза межами [0; 100];
- Некоректні показники якості повітря (наприклад, від'ємні концентрації).

Після первинного очищення застосовується згладжування.

Ковзне середнє

Метод ковзного середнього використовується для згладжування ряду значень та усунення дрібномасштабних флуктуацій. Формально для точки і з ряду $x_1, x_2, \ldots, x_n$ обчислюється:

$$MA_i = \frac{x_{i-1} + x_i + x_{i+1}}{3}$$

Якщо точка є початковою чи кінцевою (тобто не має сусідів з обох боків), середнє береться по доступних значеннях.

У реалізації системи для кожного з параметрів (температура, вологість, якість повітря) формується окремий згладжений ряд, який:

- Відображається на графіку;
- Є основою для подальшого аналізу змін (наприклад, виявлення трендів);
- Використовується у віджетах, які показують усереднене значення за останні 24 години.

Обчислення виконується на рівні вебінтерфейсу у Flask-роуті /api/history, який при запиті повертає масиви ковзного середнього у форматі JSON для побудови графіків на стороні клієнта.

Застосування методу ковзного середнього зумовлене його ефективністю в задачах екологічного моніторингу. Його переваги полягають у простоті реалізації, стійкості до одиничних викидів, високій швидкодії та інтерпретованості результатів. На відміну від більш складних методів, таких як експоненціальне згладжування або регресійне вирівнювання, ковзне середнє не потребує калібрування або зовнішніх бібліотек. Саме тому воно широко використовується в прикладних задачах, що потребують швидкого аналізу часових рядів без втрати точності.

Таким чином, реалізована комбінація очищення, фільтрації та згладжування даних дозволяє досягти надійності у відображенні та інтерпретації екологічної інформації навіть за умови нестабільних вихідних даних.

2.4 Запис даних у CSV

Збереження екологічних даних у форматі CSV (Comma-Separated Values) є одним із ключових компонентів розробленого модуля. Обраний формат дозволяє забезпечити сумісність з більшістю аналітичних інструментів, спрощує подальшу обробку, архівацію та візуалізацію інформації.

Структура файлу

Усі зібрані значення записуються у файл weather_data.csv зі стандартною структурою, що включає наступні стовпці:

- timestamp — позначка часу у форматі ISO 8601 (наприклад, 2025-06-14T16:30:00);
- city — назва населеного пункту (латиницею, наприклад, Kyiv);
- temperature — температура повітря в градусах Цельсія;
- humidity — відносна вологість (%);
- air_quality — умовний індекс якості повітря (на основі концентрацій частинок PM2.5 або CO).

Приклад запису:

2025-06-14T16:30:00,Kyiv,17.2,75,7.58

Механізм збереження

Уся логіка запису реалізована в окремому модулі collector.py. Після отримання та валідації кожного пакету даних від API:

1. Формується новий рядок з актуальними значеннями.
2. Перевіряється, чи існує файл weather_data.csv.
3. Якщо файл існує — новий рядок додається в кінець.
4. Якщо файл відсутній — створюється новий файл з відповідними заголовками колонок.

Запис відбувається у режимі append (a), що дозволяє зберігати історичні дані без перезапису попередніх. Завдяки цьому забезпечується нарощування бази спостережень у режимі реального часу.

Переваги формату CSV

- Простота імпорту в Excel, Google Sheets, pandas;
- Легке сортування, фільтрація, агрегація;
- Можливість віддаленої передачі або резервного копіювання;
- Читабельність навіть у звичайному текстовому редакторі.

Дані з файлу weather_data.csv використовуються також у вебінтерфейсі — при побудові графіків, відображенні зведеної статистики, фільтрації за містом та діапазоном дат.

2.5 Реалізація логування та валідації даних

Для забезпечення прозорості роботи модуля збору та обробки екологічних даних важливо реалізувати механізми логування (audit trail) та валідації даних. Це дозволяє не лише виявляти проблеми в процесі, а й гарантувати цілісність і достовірність інформації.

Логування подій

Система веде журнал у текстовому файлі collector.log, який автоматично оновлюється під час кожного циклу збору даних. У лог записуються:

- час виконання запиту;
- назва міста;
- статус відповіді від API (успішно або помилка);
- кількість записаних рядків;
- повідомлення про винятки у разі помилок.

Формат типового запису:

[2025-06-14 16:30:02] INFO: Kyiv – успішно збережено дані.

[2025-06-14 16:30:03] ERROR: Zhytomyr – помилка під час з’єднання з API.

Це дозволяє відстежувати історію роботи модуля, діагностувати мережеві чи синтаксичні збої, а також вести статистику за кількістю оновлень.

Валідація отриманих значень

Після отримання кожного JSON-відповіді від WeatherAPI застосовується набір правил для перевірки валідності:

- температура: від –50 до +60 °C;
- вологість: від 0% до 100%;
- якість повітря: не може бути від’ємною;
- наявність усіх ключових полів у відповіді.

У разі виявлення недопустимого значення:

- запис не вноситься у CSV;
- відповідна подія фіксується у логах.

З метою зниження кількості запитів до API та економії трафіку реалізовано механізм кешування: якщо дані для певного міста вже були успішно отримані менше ніж 30 хвилин тому, повторний запит не виконується. Це дозволяє оптимізувати навантаження, знизити ризик перевищення ліміту запитів у безкоштовному тарифі та підвищити загальну продуктивність системи.

Таким чином, усі збережені у системі дані є перевіреними, структурованими та придатними для подальшого використання.

Висновки до розділу 2

У другому розділі було розглянуто архітектурні та технічні рішення, реалізовані під час створення модуля обробки екологічних даних. Було описано процес отримання інформації з WeatherAPI, розроблено механізм збору та збереження показників температури, вологості й якості повітря у форматі CSV.

Особливу увагу приділено фільтрації шумів та застосуванню ковзного середнього як базової моделі згладжування, що дозволяє покращити якість візуалізації та аналізу даних. Реалізоване логування забезпечує контроль за стабільністю та коректністю роботи системи, а валідація — захист від помилкових або аномальних значень.

Таким чином, модуль забезпечує повний цикл обробки: від збору сирих показників до збереження перевірених даних у структурованому вигляді, готовому для подальшої аналітики.

РОЗДІЛ 3
РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ

3.1 Постановка задачі та вимоги

На цьому етапі проєкту було поставлено завдання розробити вебінтерфейс, який забезпечує зручний доступ до зібраних екологічних даних, їх перегляд у графічному вигляді, а також динамічну взаємодію з модулем збору. Основною метою є створення простого, інформативного та візуально привабливого дашборду для моніторингу стану навколишнього середовища в обраних містах.

Основні функціональні вимоги:

- Вивід актуальних значень температури, вологості та якості повітря у вибраному місті;
- Вибір міста з наявного списку або додавання нового;
- Вибір діапазону дат та часу для перегляду історичних даних;
- Побудова графіків за останні 24 години або за довільно обраний період;
- Застосування ковзного середнього для згладжування графіків;
- Запуск збору даних вручну з можливістю періодичної автоматизації (наприклад, кожні 30 хвилин);
- Візуальне оформлення у стилі сучасного адмін-дашборду (з використанням UI-бібліотек типу Material Design, Tailwind або Bootstrap).

Нефункціональні вимоги:

- Вебінтерфейс повинен бути адаптивним (коректна робота на десктопах і мобільних пристроях);
- Продуктивність при обробці великого обсягу CSV-даних (до кількох тисяч рядків);
- Легкість у локальному розгортанні (без складних залежностей, запуск через Flask);

- Відсутність потреби в базі даних — уся інформація зберігається у CSV.

Ці вимоги стали основою для подальшого проєктування і реалізації фронтенд-частини, API-серверу та графічної візуалізації екологічних показників.

Для візуального представлення взаємодії користувача з системою було побудовано діаграму варіантів використання. Вона відображає основні сценарії, що підтримуються програмною системою, зокрема вибір міста, перегляд графіків, фільтрацію за датами та ініціацію збору даних вручну.

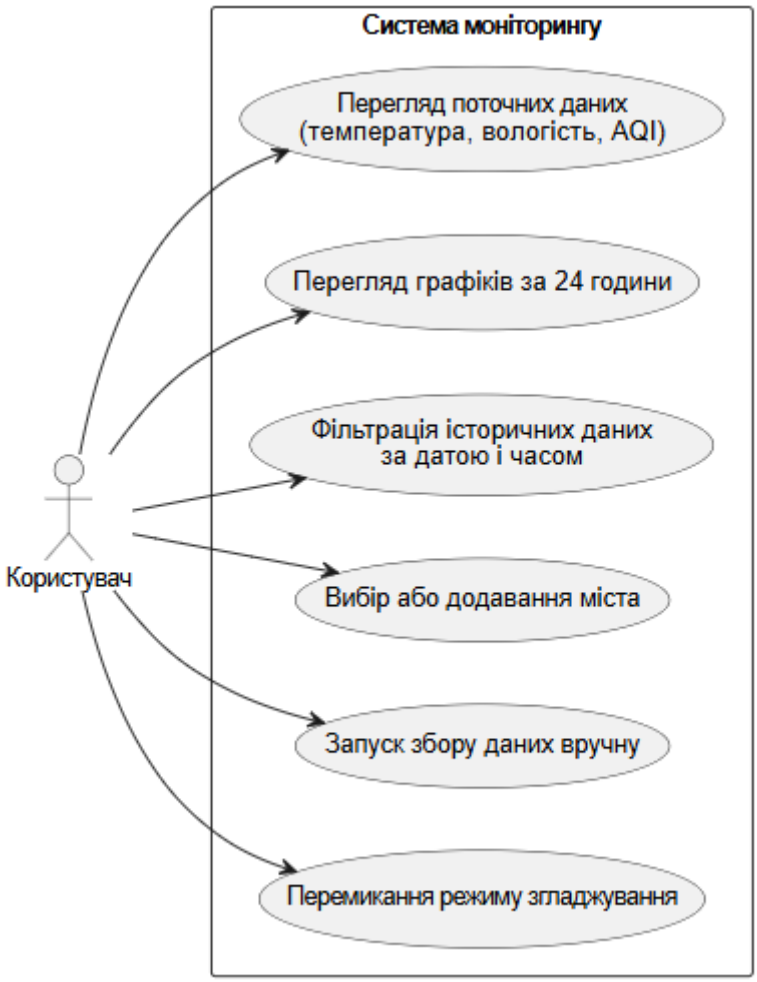

Рисунок 3.1 – Діаграма варіантів використання вебінтерфейсу

Як видно з діаграми, користувач взаємодіє з системою через низку основних сценаріїв, які охоплюють як отримання інформації (перегляд

графіків, поточних значень), так і керування (вибір міста, запуск збору даних, перемикання режиму згладжування). Такий підхід дозволяє створити інтерфейс, що відповідає принципам зручності використання (usability) та гнучкої аналітики.

Визначення варіантів використання на цьому етапі дозволило чітко сформулювати функціональні блоки майбутнього вебінтерфейсу та забезпечити відповідність між вимогами замовника й реалізованим функціоналом у наступних етапах проєкту.

3.2 Реалізація фронтенду

Фронтенд частина вебінтерфейсу реалізована як односторінковий дашборд, стилізований з використанням Tailwind CSS та елементів Material Design. Основна мета — забезпечити користувача інтуїтивно зрозумілим інтерфейсом для перегляду та аналізу екологічних показників у реальному часі та за історичними періодами.

Структура сторінки

Інтерфейс складається з трьох основних блоків:

- Верхня панель керування — містить заголовок, кнопку запуску збору даних, список доступних міст і кнопку «Додати місто».
- Панель фільтрації — блок із вибором дат/часу (початок і кінець), чекбокс згладжування, випадаючий список міст. Застосування фільтрів оновлює графіки без перезавантаження сторінки.
- Графічний блок — інтерактивні графіки температури, вологості й якості повітря, побудовані на основі Chart.js. Графіки адаптуються під будь-який розмір екрана.

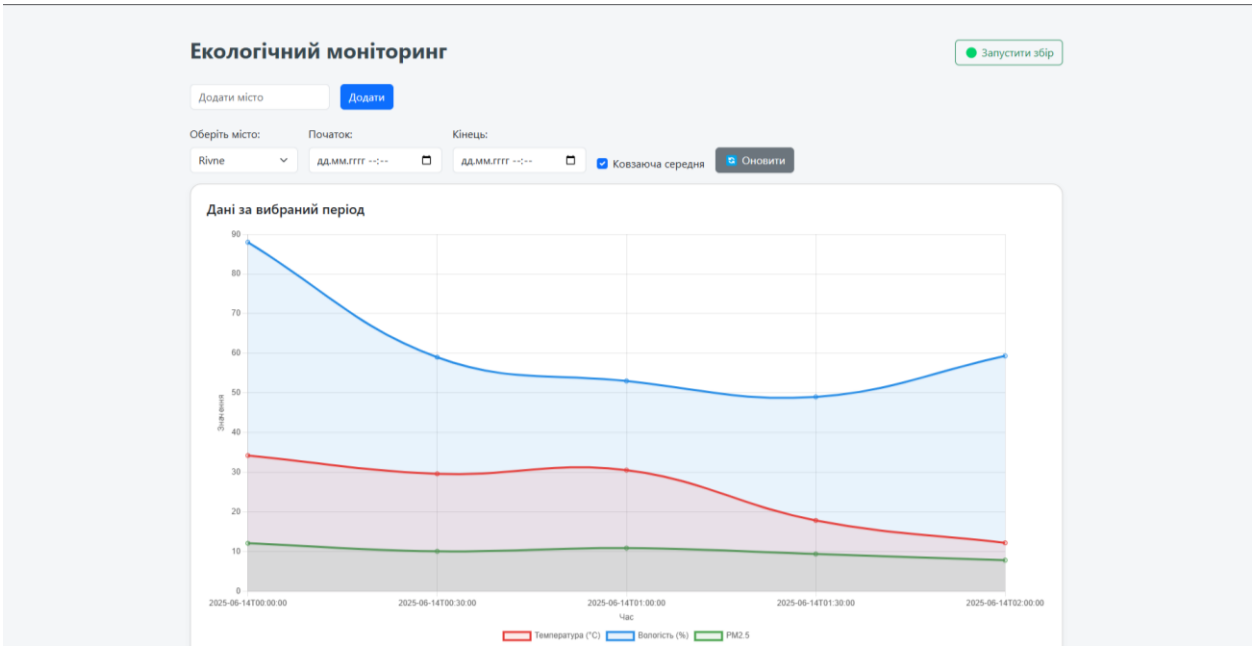

Рисунок 3.2 — Загальний вигляд дашборду

Інтерфейс було побудовано з урахуванням принципів адаптивного дизайну: елементи зручно компонуються на різних пристроях, зберігаючи функціональність. На головному екрані відображаються основні показники у вигляді графіків, а також блок фільтрації для взаємодії з даними.

Окрему увагу приділено можливості динамічного оновлення вмісту без перезавантаження сторінки. При зміні параметрів (місто, період, тип згладжування) інтерфейс миттєво підтягує нові дані через API та оновлює візуалізацію.

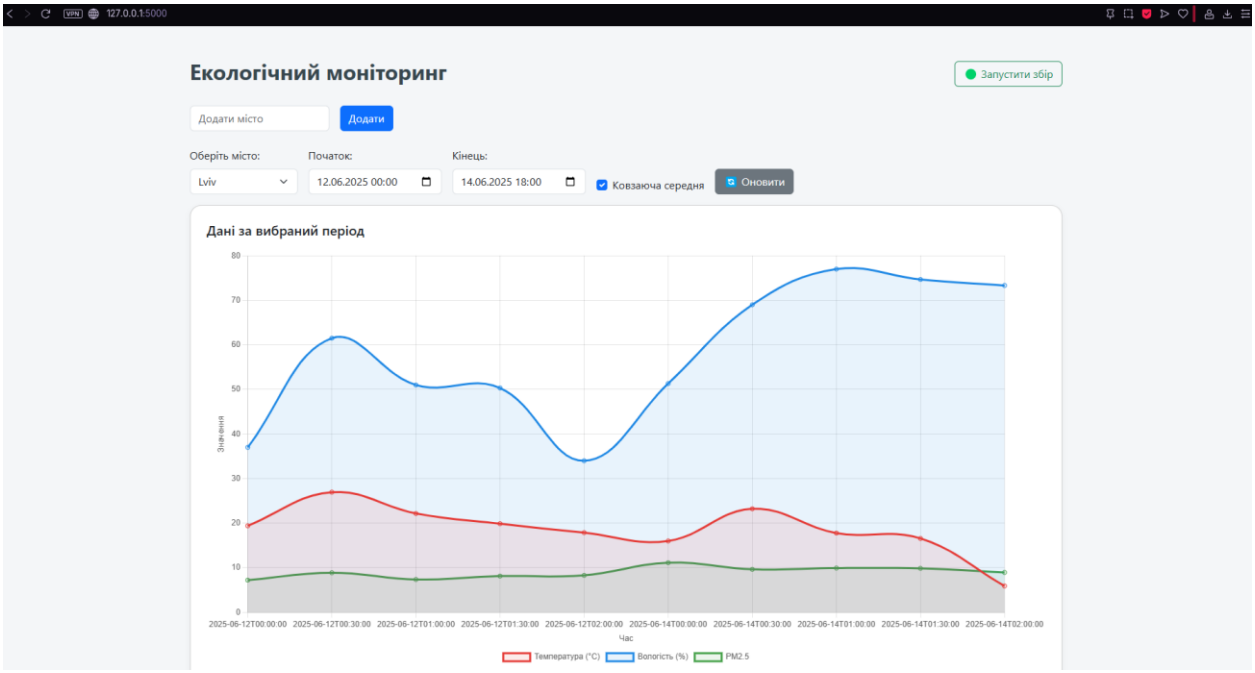

Рисунок 3.3 — Приклад фільтрації даних за діапазоном дат

Реактивність і динаміка

Користувачі можуть:

- додавати нові міста, які автоматично зберігаються у cities.csv;
- змінювати місто — сторінка автоматично завантажує відповідні графіки;
- запускати збирання даних вручну або включити періодичний збір (кожні 30 хвилин);
- перемикати режим згладжування за допомогою чекбокса («Застосувати ковзне середнє»).

Уся логіка взаємодії реалізована на Vanilla JS з асинхронними fetch-запитами до Flask API (/api/history, /api/collect, /api/cities).

3.3 Реалізація API Flask для взаємодії з модулем

Для забезпечення взаємодії між фронтендом та модулем обробки даних у системі реалізовано набір REST API-ендпойнтів на базі Flask. Серверна логіка обробляє запити з клієнта, повертає відповідні дані з CSV-файлу або виконує новий цикл збору інформації з WeatherAPI.

Основні маршрути (ендпойнти)

/api/history

Цей ендпойнт відповідає за формування історичних даних для графіків. Приймає параметри:

- city — назва міста;
- start — початок періоду (опційно);
- end — кінець періоду (опційно);
- avg — булевий параметр (1/0), чи застосовувати ковзне середнє.

Формат відповіді — JSON-об'єкт із полями labels, temperature, humidity, air_quality, які використовуються Chart.js для побудови графіків.

/api/cities

Повертає список міст, збережених у cities.csv. Також обробляє POST-запити з додаванням нових міст.

- GET — повертає масив міст;
- POST — приймає city у JSON і додає його до CSV, якщо такого ще нема.

/api/collect

Запускає ручний збір даних з API для всіх міст із cities.csv. Використовується як у фонових завданнях (планувальник), так і за запитом користувача.

Детальна структура API-ендпойнтів із зазначенням маршрутів, методів і функціонального призначення наведена в Додатку В.

Обробка логіки

Усі API-запити реалізовані через Flask @app.route, з використанням бібліотек:

- csv — для роботи з файлами;
- datetime — для фільтрації за часом;
- jsonify — для структурування відповіді;
- requests — для підключення до WeatherAPI;
- logging — для ведення логу collector.log.

API побудований з урахуванням розділення логіки: модуль збору (collector.py) працює незалежно, а Flask — лише як інтерфейс до готових даних.

Безпека та обмеження

Для простої локальної системи аутентифікація не передбачена, але структура маршрутизатора дозволяє легко додати захист доступу (наприклад, API-токени або Flask-Login) при розгортанні в продакшн.

3.4 Візуалізація даних за допомогою Chart.js

Для графічного відображення погодних показників на вебсторінці було використано бібліотеку Chart.js, яка забезпечує динамічну побудову графіків на основі JSON-даних, отриманих з Flask API.

Chart.js обрано завдяки її гнучкості, простоті інтеграції та широким можливостям налаштування стилів, підписів, масштабування та адаптації під різні пристрої.

Типи графіків

На дашборді представлено три лінійні графіки:

- Температура повітря (в °C);
- Вологість (у %);
- Індекс якості повітря (умовна шкала на основі PM2.5/CO).

Усі графіки синхронізовані за часовою шкалою — на осі X відображаються мітки часу, а значення беруться з масивів temperature, humidity, air_quality, які повертаються з /api/history.

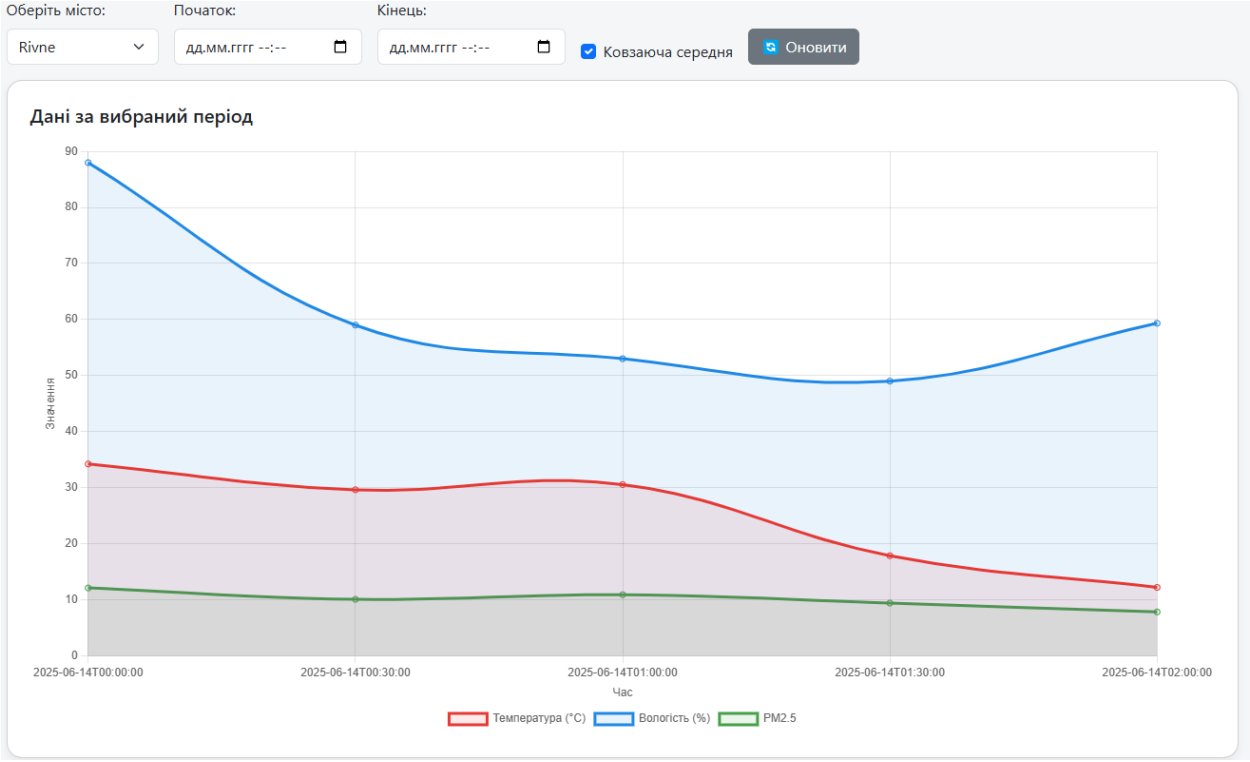

Рисунок 3.4 — Побудова графіків погодних показників за останні 24 години

Початкове відображення даних на дашборді включає погодні показники за останню добу. Це дозволяє користувачу швидко оцінити зміни екологічної ситуації та виявити можливі коливання.

Завдяки логіці, користувач має змогу змінювати параметри фільтрації у реальному часі — обрати новий діапазон дат або інше місто, після чого графіки автоматично оновлюються.

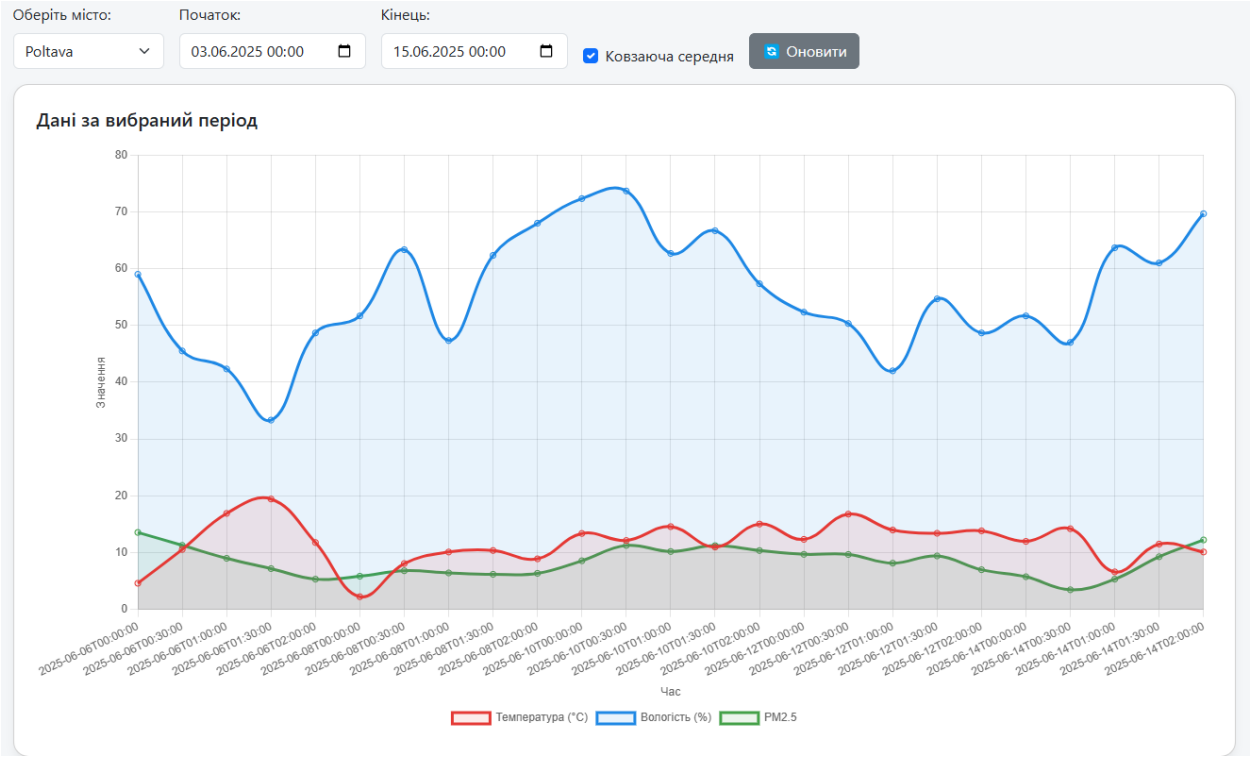

Рисунок 3.5 — Графіки після зміни міста і діапазону дат

Особливості реалізації

- Графіки створюються у тегах <canvas> з унікальними ID.
- Запит до API (/api/history) виконується через fetch, і після отримання результату графіки оновлюються без перезавантаження сторінки.
- Якщо увімкнено параметр згладжування (чекбокс «Ковзне середнє»), графіки будується за усередненими значеннями.

Адаптивність

Chart.js дозволяє динамічно змінювати розмір полотна відповідно до ширини браузера. Це забезпечує коректне відображення даних як на широких моніторах, так і на мобільних пристроях.

Кастомізація

- Підсвічування поточного значення;
- Плавна лінія (tension: 0.4) для зручнішого візуального аналізу;
- Автоматична генерація підписів та легенди.

3.5 Тестування, локальний запуск, приклади результатів

Після реалізації функціональних модулів було проведено повноцінне тестування системи в локальному середовищі. Мета — перевірити стабільність, коректність взаємодії компонентів та відповідність системи функціональним вимогам.

Умови тестування

- ОС: Windows 10 / Linux Ubuntu 22.04
- Python 3.10
- Браузери: Chrome 124+, Firefox 115+
- Старт проєкту: python app.py
- Початкові CSV-файли (weather_data.csv, cities.csv) заповнені вручну та автоматично

Таблиця 3.1 — Результати тестування функціональних сценаріїв

№	Тестовий випадок	Вхідні дані	Очікуваний результат	Статус
1	Запуск інтерфейсу	-	Сторінка відкривається, елементи видно	Успішно
2	Додавання нового міста	"city": "Lviv"	Місто з’являється у списку, додається у cities.csv	Успішно
3	Збір даних вручну	Кнопка «Зібрати дані»	Дані додаються в weather_data.csv, лог оновлюється	Успішно
4	Фільтрація за діапазоном дат	12.06.2025 – 14.06.2025	Відображаються лише записи у заданому періоді	Успішно

5	Застосування ковзного середнього	avg=1	Графік оновлюється згладженими значеннями	Успішно
6	Вибір іншого міста	"Zhytomyr"	Дані оновлюються відповідно до вибраного міста	Успішно
7	Відсутність з’єднання з API при зборі	Симуляція offline	Запис помилки у лог, дані не додаються	Успішно
8	Валідація некоректних значень (темп. > 100 °C)	Симуляція вручну у JSON	Значення ігноруються, лог містить запис про помилку	Успішно
9	Адаптивність інтерфейсу	Мобільна ширина < 600px	Графіки та елементи стискаються, навігація коректна	Успішно
10	Повторне додавання того самого міста	"city": "Kyiv" (вже є)	Повідомлення про дубль або ігнорування	Успішно

Проведене тестування підтвердило повну працездатність розробленої системи в межах поставлених функціональних вимог. Всі ключові сценарії — включно з ручним збором даних, побудовою графіків, додаванням міст та роботою з історичними даними — були успішно реалізовані й пройшли перевірку.

Система демонструє стабільну роботу в офлайн-умовах, коректно обробляє некоректні чи аномальні значення, та може бути легко масштабована або розгорнута на іншому середовищі. Верифікована структура CSV-

документів дозволяє інтегрувати дані з іншими системами моніторингу чи аналітики.

Висновки до розділу 3

У цьому розділі було реалізовано повноцінний вебінтерфейс для взаємодії з модулем екологічного моніторингу. Основні компоненти — фронтенд, API-сервер і візуалізація даних — були спроєктовані з урахуванням зручності використання, гнучкості налаштувань і стабільної роботи в локальному середовищі.

Інтерфейс надає користувачу можливість:

- вибирати місто для моніторингу;
- переглядати динамічні графіки показників за довільний період;
- запускати ручний збір нових даних;
- аналізувати тенденції за допомогою згладжування даних ковзним середнім.

Тестування підтвердило відповідність системи функціональним вимогам. Рішення є придатним для подальшого розширення, наприклад, з додаванням бази даних, телеметрії або аналітичних звітів.

ВИСНОВКИ

У межах даної дипломної роботи було розроблено повноцінну систему ландшафтного екологічного моніторингу, орієнтовану на збір, обробку та візуалізацію даних про температуру, вологість та якість повітря. Реалізація включає два основні компоненти: модуль обробки даних та інтерактивний вебінтерфейс, побудований на основі Flask і Chart.js.

На етапі аналітичного дослідження було проаналізовано існуючі системи екологічного моніторингу, визначено ключові вимоги до функціоналу та обґрунтовано вибір джерела даних — WeatherAPI.

У технічній частині роботи реалізовано:

- збір даних з API з фільтрацією шумів та логуванням;
- зберігання очищених даних у форматі CSV;
- застосування методу ковзного середнього для згладжування;
- зручний вебінтерфейс із підтримкою фільтрації, вибору міста та побудови графіків.

Тестування підтвердило стабільність та відповідність системи заявленим функціональним вимогам. Отримане рішення є придатним до практичного застосування для аналізу змін кліматичних умов, виявлення аномалій та моніторингу екологічного стану на локальному рівні.

У перспективі систему можна розширити шляхом інтеграції з базами даних, хмарними сервісами або платформами довготривалого зберігання й аналітики.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Голуб О. І., Топчій Л. І. Екологічний моніторинг: навч. посібник. – Київ: Центр учбової літератури, 2017. – 268 с.
2. European Commission. Copernicus Programme. URL: https://www.copernicus.eu/en (Дата звернення: 14.06.2025).
3. Закон України «Про охорону навколишнього природного середовища» від 25.06.1991 №1264-XII. URL: https://zakon.rada.gov.ua/laws/show/1264-12 (Дата звернення: 14.06.2025).
4. SaveDnipro. Проєкт EcoCity. URL: https://www.saveecobot.com/ (Дата звернення: 14.06.2025).
5. Kioutsioukis I., Tarantola S., Saltelli A. Uncertainty and sensitivity analysis of air quality models. – Atmospheric Environment, 2004. – Vol. 38. – P. 769–780.
6. World Health Organization. Air pollution. URL: https://www.who.int/health-topics/air-pollution (Дата звернення: 14.06.2025).
7. NOAA. National Weather Service. URL: https://www.weather.gov (Дата звернення: 14.06.2025).
8. AQICN.org. Real-time Air Quality Index. URL: https://aqicn.org/map/world (Дата звернення: 14.06.2025).
9. EcoCity (SaveDnipro). Екологічний моніторинг. URL: https://www.saveecobot.com (Дата звернення: 14.06.2025).
10. Горбенко І. Системи моніторингу на базі Arduino: посібник. – Київ: ВНТУ, 2020. – 120 с.
11. WeatherAPI Documentation. URL: https://www.weatherapi.com/docs/ (Дата звернення: 14.06.2025).
12. McKinney W. Python for Data Analysis. 3rd ed. – Sebastopol: O’Reilly Media, 2022. – 544 p.

13. WeatherAPI. Official Documentation. URL: https://www.weatherapi.com/docs/ (Дата звернення: 14.06.2025).
14. OpenWeatherMap API. URL: https://openweathermap.org/api (Дата звернення: 14.06.2025).
15. Weatherstack API. URL: https://weatherstack.com/documentation (Дата звернення: 14.06.2025).
16. Visual Crossing Weather Data. URL: https://www.visualcrossing.com/weather-data-editions (Дата звернення: 14.06.2025).
17. Air Quality Index — WeatherAPI Docs. URL: https://www.weatherapi.com/docs/#air-quality (Дата звернення: 14.06.2025).
18. Tomorrow.io API. URL: https://www.tomorrow.io/weather-api/ (Дата звернення: 14.06.2025).

ДОДАТОК А
ПОРІВНЯЛЬНА ТАБЛИЦЯ API

Таблиця А.1 – Порівняння відкритих API для отримання погодних даних

Назва API	Підтримка історичних даних	Показники AQI	Локалізація	Точність для України	Безкоштовний тариф	Примітки
WeatherAPI	Так	Так	Немає	Висока	Так	Детальна документація, підтримка AQI
OpenWeatherMap	Так	Частково	Так	Середня	Так	Доступ до forecast + historical
Weatherstack	Обмежено	Ні	Частково	Низька	Так	Дані часто затримуються
Visual Crossing	Так	Ні	Так	Висока	Так	Підходить для аграрного аналізу
Tomorrow.io (Climacell)	Так	Так	Частково	Висока	Так	Орієнтований на бізнес-застосування

ДОДАТОК Б
ПРИКЛАД ЗБЕРЕЖЕНИХ ДАНИХ

```
timestamp,city,temperature,humidity,air_quality
2025-06-02T00:00:00,Kyiv,-6.4,34,2.052
2025-06-02T00:00:00,Zhytomyr,2.3,92,4.645
2025-06-02T00:00:00,Lviv,15.9,39,6.142
2025-06-02T00:00:00,Odesa,20.6,51,2.561
2025-06-02T00:00:00,Dnipro,17.3,30,10.594
2025-06-02T00:00:00,Kharkiv,3.4,49,14.161
2025-06-02T00:00:00,Vinnytsia,-5.4,71,5.156
2025-06-02T00:00:00,Poltava,13.9,37,8.972
2025-06-02T00:00:00,Chernihiv,17.6,40,14.527
2025-06-02T00:00:00,Cherkasy,20.3,95,3.276
2025-06-02T00:00:00,Ternopil,9.4,30,4.452
2025-06-02T00:00:00,Ivano-Frankivsk,26.4,61,14.284
2025-06-02T00:00:00,Uzhhorod,5.9,90,9.366
2025-06-02T00:00:00,Mykolaiv,-1.6,89,2.404
2025-06-02T00:00:00,Kherson,2.5,70,10.127
2025-06-02T00:00:00,Sumy,10.0,94,9.486
2025-06-02T00:00:00,Rivne,-5.0,55,7.829
2025-06-02T00:00:00,Lutsk,-0.9,49,2.755
2025-06-02T00:00:00,Chernivtsi,5.4,80,9.383
2025-06-02T00:00:00,Kropyvnytskyi,33.9,35,7.248
2025-06-02T00:30:00,Kyiv,26.1,41,5.208
2025-06-02T00:30:00,Zhytomyr,5.7,32,6.993
2025-06-02T00:30:00,Lviv,32.1,92,4.158
2025-06-02T00:30:00,Odesa,12.4,65,4.938
2025-06-02T00:30:00,Dnipro,21.8,92,3.696
2025-06-02T00:30:00,Kharkiv,29.3,60,10.821
2025-06-02T00:30:00,Vinnytsia,34.7,55,12.56
2025-06-02T00:30:00,Poltava,4.3,42,14.52
2025-06-02T00:30:00,Chernihiv,19.0,50,4.713
2025-06-02T00:30:00,Cherkasy,6.5,34,14.873
2025-06-02T00:30:00,Ternopil,1.2,40,13.598
2025-06-02T00:30:00,Ivano-Frankivsk,-6.3,40,8.652
2025-06-02T00:30:00,Uzhhorod,34.4,31,4.043
2025-06-02T00:30:00,Mykolaiv,4.0,60,2.938
2025-06-02T00:30:00,Kherson,14.0,89,12.726
2025-06-02T00:30:00,Sumy,27.8,89,2.113
2025-06-02T00:30:00,Rivne,-8.1,41,4.166
2025-06-02T00:30:00,Lutsk,1.1,53,11.741
2025-06-02T00:30:00,Chernivtsi,11.4,42,5.245
2025-06-02T00:30:00,Kropyvnytskyi,4.5,46,5.794
2025-06-02T01:00:00,Kyiv,23.3,40,6.12
```

2025-06-02T01:00:00,Zhytomyr,-6.0,64,4.382
2025-06-02T01:00:00,Lviv,-7.0,87,14.371
2025-06-02T01:00:00,Odesa,2.5,64,5.002
2025-06-02T01:00:00,Dnipro,11.0,71,11.412
2025-06-02T01:00:00,Kharkiv,16.9,44,2.435
2025-06-02T01:00:00,Vinnytsia,4.9,65,3.769
2025-06-02T01:00:00,Poltava,-5.2,70,13.428
2025-06-02T01:00:00,Chernihiv,-1.5,85,5.091
2025-06-02T01:00:00,Cherkasy,-5.8,65,13.66
2025-06-02T01:00:00,Ternopil,24.1,54,13.809
2025-06-02T01:00:00,Ivano-Frankivsk,16.4,36,11.626
2025-06-02T01:00:00,Uzhhorod,5.1,73,11.022
2025-06-02T01:00:00,Mykolaiv,22.7,36,10.546
2025-06-02T01:00:00,Kherson,-9.9,66,9.382
2025-06-02T01:00:00,Sumy,22.0,39,9.814
2025-06-02T01:00:00,Rivne,15.5,52,12.95
2025-06-02T01:00:00,Lutsk,10.9,80,14.76
2025-06-02T01:00:00,Chernivtsi,8.8,61,13.487
2025-06-02T01:00:00,Kropyvnytskyi,1.7,85,11.439
2025-06-02T01:30:00,Kyiv,18.7,51,2.138
2025-06-02T01:30:00,Zhytomyr,16.9,63,9.459
2025-06-02T01:30:00,Lviv,25.7,79,4.139
2025-06-02T01:30:00,Odesa,30.5,77,7.137
2025-06-02T01:30:00,Dnipro,15.7,91,2.818
2025-06-02T01:30:00,Kharkiv,-6.2,78,11.764
2025-06-02T01:30:00,Vinnytsia,21.6,48,6.811
2025-06-02T01:30:00,Poltava,32.8,38,2.483
2025-06-02T01:30:00,Chernihiv,16.2,48,6.923
2025-06-02T01:30:00,Cherkasy,19.7,55,4.49
2025-06-02T01:30:00,Ternopil,13.1,31,2.102
2025-06-02T01:30:00,Ivano-Frankivsk,4.6,49,9.052
2025-06-02T01:30:00,Uzhhorod,10.8,49,2.967
2025-06-02T01:30:00,Mykolaiv,3.4,36,7.172
2025-06-02T01:30:00,Kherson,31.9,45,10.206
2025-06-02T01:30:00,Sumy,11.9,94,5.701
2025-06-02T01:30:00,Rivne,5.6,89,2.98
2025-06-02T01:30:00,Lutsk,24.1,56,8.936
2025-06-02T01:30:00,Chernivtsi,-5.6,94,12.562
2025-06-02T01:30:00,Kropyvnytskyi,12.3,53,7.949
2025-06-02T02:00:00,Kyiv,15.2,58,6.343
2025-06-02T02:00:00,Zhytomyr,23.8,41,9.784
2025-06-02T02:00:00,Lviv,12.0,76,4.541
2025-06-02T02:00:00,Odesa,-3.1,92,7.367
2025-06-02T02:00:00,Dnipro,16.0,82,7.576
2025-06-02T02:00:00,Kharkiv,-3.3,41,14.965

ДОДАТОК Г
СТРУКТУРА REST API

Таблиця В.1 — СТРУКТУРА REST API

URL	Метод	Параметри (query/body)	Опис
/api/history	GET	city (обов’язковий), start, end, avg (1/0)	Повертає погодні дані (темп., вологість, якість повітря) за заданий період
/api/cities	GET	–	Повертає список усіх міст із cities.csv
/api/cities	POST	{ "city": "Назва" }	Додає нове місто до списку, якщо воно ще не внесене
/api/collect	GET	–	Запускає ручний збір погодних даних для всіх міст у списку