Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра програмного забезпечення

Бакалаврська кваліфікаційна робота

на тему: «Розробка методу і засобів реалізації програмної системи моніторингу й аналізу екологічної ситуації мікрорайону»

Виконав: студент IV курсу
групи 4ПІ-21б
спеціальності
121 – Інженерія програмного забезпечення
(шифр і назва напряму підготовки, спеціальності)
Пислар О. І.
(прізвище та ініціали)

Керівник: к.т.н., доц. каф. ПЗ Войтко В. В.
(прізвище та ініціали)

Рецензент: к.т.н., доц. каф. ОТ Городецька О. С.
(прізвище та ініціали)

Допущено до захисту

Зав. кафедри ПЗ Романюк О.Н.
«02» 06 2025 р.

ВНТУ – 2025

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення
Рівень вищої освіти перший (бакалаврський)
Галузь знань 12 «Інформаційні технології»
Спеціальність 121 «Інженерія програмного забезпечення»
Освітньо-професійна програма «Інженерія програмного забезпечення

УЗГОДЖЕНО
НАЧАЛЬНИК ДЕРЖАВНОЇ
ЕКОЛОГІЧНОЇ ІНСПЕКЦІЇ
ВІННИЦЬКІЙ ОБЛАСТІ
Дубовий Ю.В.
«21» березня 2025 р.

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
Романюк О. Н.
«21» березня 2025 р.

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Пислару Олександру Івановичу

1. Тема роботи – розробка методу і засобів реалізації програмної системи моніторингу й аналізу екологічної ситуації мікрорайону.

Керівник роботи: Войтко Вікторія Володимирівна, к.т.н., доцент кафедри ПЗ, затверджені наказом вищого навчального закладу від 20 березня 2025 р. №97.

2. Строк подання студентом роботи: 2 червня 2025 р.

3. Вихідні дані до роботи: середовище розробки – Visual Studio Code, мови розробки – HTML, CSS, JavaScript, Node.js; фреймворки – Express.js, Chart.js; база даних – MongoDB; операційна система – Windows 10/11; середовище хостингу – GitHub Pages, Railway, Firebase Hosting.

4. Зміст текстової частини: вступ, аналіз стану розвитку систем для аналізу й моніторингу еколгічної ситуації мікрорайону та постановка задач розробки, розробка методу, моделі та алгоритмів роботи програми, розробка програмного забезпечення, тестування програми, висновки, список використаних джерел, додатки, ілюстративна частина.

5. Перелік ілюстративного матеріалу: титульний слайд – автор, науковий керівник бакалаврської кваліфікаційної роботи, тема; актуальність розробки; мета, об'єкт та предмет дослідження; постановка задачі; наукова новизна; практична цінність; огляд та аналіз аналогів; модель роботи системи, загальний алгоритм роботи системи; діаграми станів; схеми інтерфейсу; використані технології;

функціонал розробленої системи; апробація та публікація результатів роботи; впровадження; висновки; фінальний слайд.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада Консультанта	Підпис, дата	
		Завдання Видав	Завдання прийняв
1-4	Войтко В. В., к.т.н., доцент кафедри ПЗ	24.03.25	01.06.25

7. Дата видачі завдання 24 березня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз розвитку систем для аналізу й моніторингу екологічного стану та постановка задач дослідження	25.03.25 – 29.03.25	виконано
2	Розробка методів та алгоритмів роботи системи	30.03.25 – 10.04.25	виконано
3	Розробка програмного забезпечення системи	11.04.25 – 10.05.25	виконано
4	Тестування програми	11.05.25 – 14.05.25	виконано
5	Оформлення матеріалів до захисту БКР	15.05.25 – 30.05.25	виконано

Студент _____ **Пислар О. І.**
(підпис) (ініціали та прізвище)

Керівник бакалаврської кваліфікаційної роботи _____ **Войтко В. В.**
(підпис) (ініціали та прізвище)

АНОТАЦІЯ

Бакалаврська кваліфікаційна робота складається з 62 сторінок формату А4, що містять 30 рисунків і 2 таблиці, список використаних джерел налічує 28 найменувань.

У бакалаврській кваліфікаційній роботі проаналізовано сучасні підходи до моніторингу та аналізу екологічної ситуації мікрорайону. Розглянуто наявні мобільні та вебрішення, що дозволяють відстежувати якість повітря та ґрунту, виявлено їх ключові обмеження — вузька спеціалізація, відсутність інтеграції з мапами та недостатня взаємодія з користувачем у реальному часі. На основі цього обґрунтовано доцільність створення єдиної універсальної системи моніторингу.

Проведено аналіз аналогів розроблюваної системи, визначено їх основні переваги та недоліки та доведено доцільність розробки власного програмного продукту.

Розроблена система включає мобільну та вебверсії, що дозволяють здійснювати аналіз екологічної ситуації в мікрорайоні за допомогою інтерактивної карти, графіків, рекомендацій та сповіщень. Реалізовано модулі геолокації, інтеграції з відкритими API, персоналізованого профілю користувача та збереження екосповіщень. Додаток підтримує адаптивний інтерфейс і PWA-функціональність, що підвищує зручність користування.

Систему розроблено з використанням HTML, CSS, JavaScript, бібліотеки Chart.js, а також Node.js та MongoDB на стороні сервера. Для реалізації й встановлення як мобільного застосунку використано технології Progressive Web App. В якості середовища розробки обрано Visual Studio Code. Фронтенд-система розгорнута на GitHub Pages, а серверна частина – на Railway з підключенням до хмарної бази даних MongoDB.

В результаті виконання бакалаврської кваліфікаційної роботи було розроблено систему моніторингу й аналізу екологічного стану мікрорайону, що дасть можливість швидко отримувати дані про стан навколишнього середовища, зберігати історію, аналізувати зміни і приймати обґрунтовані рішення щодо здоров’я і безпеки.

Ключові слова: вебсистема, мобільна система, геолокація, JavaScript, Node.js, MongoDB, PWA, екосповіщення, інтерактивна карта.

ABSTRACT

The bachelor's qualification work consists of 62 pages of A4 format, containing 30 figures and 2 tables, the list of used sources includes 28 items.

The bachelor's thesis analyzes current approaches to environmental monitoring and assessment in urban areas. Existing mobile and web-based solutions for tracking air and soil quality have been reviewed, revealing key limitations such as narrow specialization, lack of integration with maps, and limited user interaction in real time.

The developed system includes both mobile and web versions that allow users to analyze the environmental situation in a specific neighborhood using an interactive map, visual graphs, personalized recommendations, and real-time alerts. The system incorporates modules for geolocation, integration with open APIs, personalized user profiles, and saving environmental reports. The application features an adaptive interface, d and offline access through Progressive Web App functionality to improve user experience.

The web system was implemented using HTML, CSS, and JavaScript, including the Chart.js library, while the backend was developed using Node.js and MongoDB. The development environment chosen was Visual Studio Code. The frontend was deployed via GitHub Pages, and the backend was hosted on Railway with integration to a cloud MongoDB database.

As a result of the thesis work, a universal environmental monitoring system was developed, enabling users to receive timely data on the environmental state, track historical data, analyze trends, and make informed decisions regarding personal health and safety.

Keywords: environmental monitoring, web system, mobile system, geolocation, JavaScript, Node.js, MongoDB, PWA, ecological alerts, interactive map.

ЗМІСТ

ВСТУП

Обґрунтування вибору теми дослідження. На тлі стрімкого погіршення екологічної ситуації у містах України все більшої актуальності набувають засоби моніторингу стану довкілля на локальному рівні — зокрема, у межах окремих мікрорайонів [1-3]. В умовах зростання чисельності населення, швидкої урбанізації, транспортного навантаження і діяльності промислових підприємств мешканці стикаються з низкою загроз — погіршення якості повітря, зменшення родючості ґрунтів. Відсутність зручних інструментів для швидкої оцінки стану навколишнього середовища створює бар'єри для громадського контролю за подібними проблемами, екологічної обізнаності мешканців та своєчасного реагування на катастрофи екологічного характеру [4].

Традиційні підходи до екологічного моніторингу, що базуються на громіздких приладах і звітності з великим часовим проміжком, не задовольняють потреб сучасного суспільства. Їм бракує інтерактивності, доступності і гнучкості. З іншого боку, розвиток мобільних технологій, відкритих екологічних API [1], геолокаційних сервісів і аналітичних засобів надає нові можливості для побудови системи реального часу з інтерактивним інтерфейсом, що дозволяє користувачам швидко отримувати інформацію про стан довкілля саме у їхньому районі.

У зв'язку з поширенням смарт-систем та екологічної цифровізації, з'являється потреба у мобільних рішеннях, що здатні забезпечити збір, аналіз, візуалізацію і подання екологічних даних у зручному для мешканців мікрорайонів форматі [5]. Розробка мобільної системи моніторингу якості повітря та ґрунту з аналітикою, картами та порадами для збереження здоров'я є важливим кроком у напрямі формування екологічно свідомого суспільства. Такі системи можуть застосовуватися не лише у побуті, а й у міському управлінні, при плануванні заходів щодо покращення екології, в освітньому процесі і навіть у надзвичайних ситуаціях.

Зв’язок роботи з науковими програмами, планами, темами. Робота виконувалася згідно плану виконання наукових досліджень на кафедрі програмного забезпечення.

Мета та завдання дослідження. Метою роботи є покращення можливостей моніторингу й аналізу екологічної ситуації навколишнього середовища за рахунок розробки і використання спеціалізованої програмної системи, яка дозволить в інтерактивному режимі відображати стан повітря та ґрунту, надавати рекомендації користувачам, зберігати дані в базі, забезпечувати підтримку PWA-функціоналу.

Відповідно до поставленої мети потрібно вирішити такі завдання:

– провести аналіз стану розвитку застосунків аналізу й моніторингу екологічної ситуації;

– розробити загальну модель вебсистеми;

– розробити загальний алгоритм роботи платформи;

– розробити метод збору екологічних сповіщень;

– розробити метод побудови динамічних графіків та візуалізації даних;

– розробити вебсистему аналізу й моніторингу екологічної ситуації мікрорайону;

– розробити інтуїтивний інтерфейс користувача;

– провести тестування роботи вебсистеми.

Об’єктом дослідження є процес розробки вебсистеми моніторингу й аналізу еклогічної ситуації мікрорайону.

Предметом дослідження є методи і засоби реалізації вебсистеми моніторингу й аналізу екологічної ситуації мікрорайону.

Методи дослідження. У процесі дослідження використовувалися такі методи:

- методи створення UI/UX для роозробки інтерфейсу користувача для розробки зручного, інтуїтивно зрозумілого інтерфейсу користувача на всіх сторінках вебсистеми;
- методи програмування з використанням HTML, CSS, JavaScript, PWA для створення адаптивної фронтенд-частини, реалізації логіки інтерфейсу і динамічної взаємодії з сервером;
- методи обробки даних з екологічних API для отримання та обробки реальних даних про якість повітря і ґрунту, а також їх подальшої візуалізації на картах і графіках;
- методи побудови графіків за допомогою Chart.js для виведення візуальної аналітики у вигляді інтерактивних стовпчикових діаграм;
- методи інтеграції з Google Maps з геолокацією для відображення карти з маркерами екологічної ситуації, а також автоматичного визначення місцезнаходження користувача та прив’язки до конкретного району;
- методи тестування для перевірки роботи програмної системи для виявлення помилок, перевірки поведінки інтерфейсу при різних сценаріях з метою забезпечення стабільної роботи системи.

Новизна отриманих результатів.

1. Подальшого розвитку отримав метод збору екологічного сповіщення, який, на відміну від існуючих, крім парсингу даних з відкритих джерел, реалізує ідентифікацію користувача на карті з визначенням мікрорайону, що дозволяє провести аналіз моніторингових даних для локалізованого місця.

2. Подальшого розвитку отримала модель вебсистеми моніторигу екологічного стану мікрорайону, яка, на відміну від існуючих, інтегрує інтерактивні модулі відстеження користувача на карті, отримання моніторингових даних з відкритих джерел, аналіз отриманої інформації з можливістю візуалізації

результатів моніторингового аналізу, що дозволяє покращити інтерактивну взаємодію користувача з програмною системою.

Практичне значення роботи. Розроблена система може використовуватись як мешканцями для оцінки рівня забруднення, так і органами місцевого самоврядування для контролю екологічної ситуації. Вона підтримує відображення історії входу в особистий кабінет, локалізацію та рекомендовані дії. Окрім того, модульна архітектура дозволяє легко адаптувати систему під інші території або типи екологічного моніторингу.

Особистий внесок здобувача. Усі результати, подані в бакалаврській кваліфікаційній роботі, були отримані автором особисто. У науковій праці [6], опублікованій у співавторстві, автору належать такі результати: таблиця із порівнянням функціональних можливостей власного продукту та наявних аналогів, блок-схема загального алгоритму роботи системи.

Апробація результатів роботи. Результати бакалаврської кваліфікаційної роботи доповідалися на Міжнародній науково-практичній конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2025)».

Публікації. Результати дослідження були опубліковані у матеріалах Міжнародної науково-практичної інтернет-конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2025)» [6].

Аналіз. У пояснювальній записці до бакалаврської кваліфікаційної роботи було розглянуто чотири розділи та було використано 28 літературних джерел.

1 АНАЛІЗ РОЗВИТКУ СИСТЕМ ДЛЯ АНАЛІЗУ Й МОНІТОРИНГУ ЕКОЛОГІЧНОЇ СИТУАЦІЇ ТА ПОСТАНОВКА ЗАДАЧ ДОСЛІДЖЕННЯ

1.1 Аналіз розвитку систем для аналізу й моніторингу екологічної ситуації

На сучасному етапі розвитку суспільства питання екології набули надзвичайної актуальності. Зростання рівня забруднення повітря, ґрунтів, а також частота екологічних інцидентів призводять до потреби постійного моніторингу стану навколишнього середовища на локальному рівні, особливо в межах мікрорайонів великих міст. При цьому традиційні методи обробки екологічної інформації, які часто базуються на застарілих або фрагментованих джерелах, не дозволяють забезпечити своєчасний доступ громадян до актуальних даних, що стримує розвиток свідомого екологічного мислення [7].

Особливу роль у вирішенні цієї проблеми відіграють цифрові технології, а саме — вебзастосунки та мобільні системи екологічного моніторингу. Завдяки інтеграції відкритих API, геолокації, аналітики та засобів візуалізації даних, такі системи здатні не лише інформувати населення, а й допомагати місцевим громадам і органам влади приймати рішення щодо запобігання або мінімізації шкоди для довкілля. Система, що пропонується в даній роботі, забезпечує збір, обробку, зберігання та представлення інформації про якість повітря і стан ґрунтів у зручному та інтуїтивному форматі.

Серед ключових функцій системи — персоналізований доступ до даних через авторизацію, аналітичні графіки змін у довкіллі та можливість залишати власні повідомлення про порушення. Таким чином, користувач не тільки отримує корисну інформацію, а й стає учасником спільного екологічного моніторингу, що підвищує рівень громадської відповідальності і сприяє розвитку екокультури [8].

Розробка інтерактивного інтерфейсу, адаптованого під мобільні пристрої, надає змогу максимально наблизити екологічну інформацію до кінцевого користувача — жителів мікрорайону, студентів, місцевих активістів та працівників комунальних служб. А використання PWA-технологій дозволяє працювати із системою навіть за умов нестабільного або повністю відсутнього інтернет-з'єднання, що є особливо важливим у кризових ситуаціях.

На сьогодні існують окремі сервіси для відображення екологічних показників, однак вони здебільшого або занадто загальні, або не передбачають активної участі користувачів і локальної деталізації. Таким чином, розробка комплексної системи локального моніторингу з можливістю інтеграції у глобальні аналітичні платформи є надзвичайно актуальним напрямом, який відповідає сучасним потребам у сфері екологічної безпеки та сталого розвитку громад.

Отже, проведений аналіз предметної області демонструє доцільність створення та впровадження інноваційної веб- та мобільної системи екологічного моніторингу мікрорайону. Вона дозволяє поєднати можливості цифрових технологій, відкритих екологічних API та інструментів аналітики в єдину ефективну платформу, що сприятиме підвищенню екологічної обізнаності та залученості населення. Реалізація такої системи стане суттєвим кроком на шляху до екологічної цифровізації [9] міст і населених пунктів України.

1.2 Порівняльний аналіз аналогів вебсистеми аналізу й моніторингу екологічної ситуації

З використанням мобільних технологій та сенсорних систем для моніторингу екологічної ситуації стає все більш популярним у сучасному світі. Така технологія легко інтегрується в різні додатки і системи, включаючи системи вимірювання рівня забрудненості повітря і землі. Розробка програмних модулів системи моніторингу й аналізу екологічної ситуації мікрорайону забезпечить швидку

обробку і візуалізацію екологічних даних, що сприятиме оперативному виявленню проблем і допоможе приймати ефективні рішення для покращення стану довкілля. Розглянемо популярні ресурси як аналоги розроблюваної вебсистеми для аналізу й моніторингу екологічної ситуації мікрорайону.

AirVisual є застосунком, створеним для моніторингу якості повітря в режимі реального часу. Він надає користувачу дані про рівень забруднення і прогнозує зміни якості повітря, а також пропонує рекомендації для захисту здоров'я людей. Додаток використовує дані з офіційних станцій моніторингу, пристроїв користувачів, а також інтегрує погодні умови для більш точного аналізу стану навколишнього середовища [10].

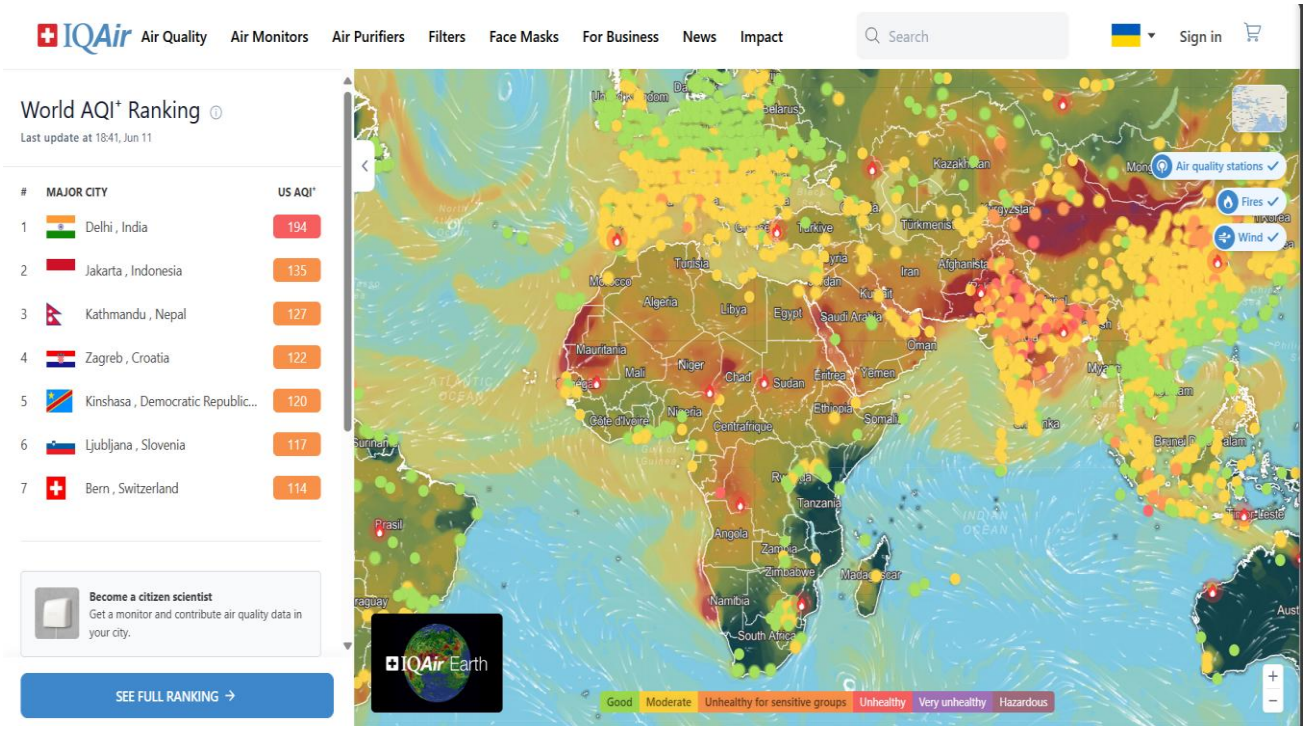


Рисунок 1.1 – Приклад роботи застосунку «AirVisual»

Інший приклад – вебсистема під назвою «Plume Labs» [11].

Plume Labs – це мобільний програмний засіб для моніторингу якості повітря в реальному часі.

Plume Labs оснащений портативним IoT-сенсором, який дозволяє користувачам вимірювати рівень забруднення атмосфери у своєму безпосередньому оточенні.

Flow аналізує концентрацію шкідливих речовин, таких як NO_2, PM2.5 і PM10, а результати відображає у зручному мобільному додатку. Отримані дані можуть бути використані користувачами для створення персоналізованих карт забруднення повітря та прогнозування його змін.

Приклад роботи застосунку зображено на рисунку 1.2.

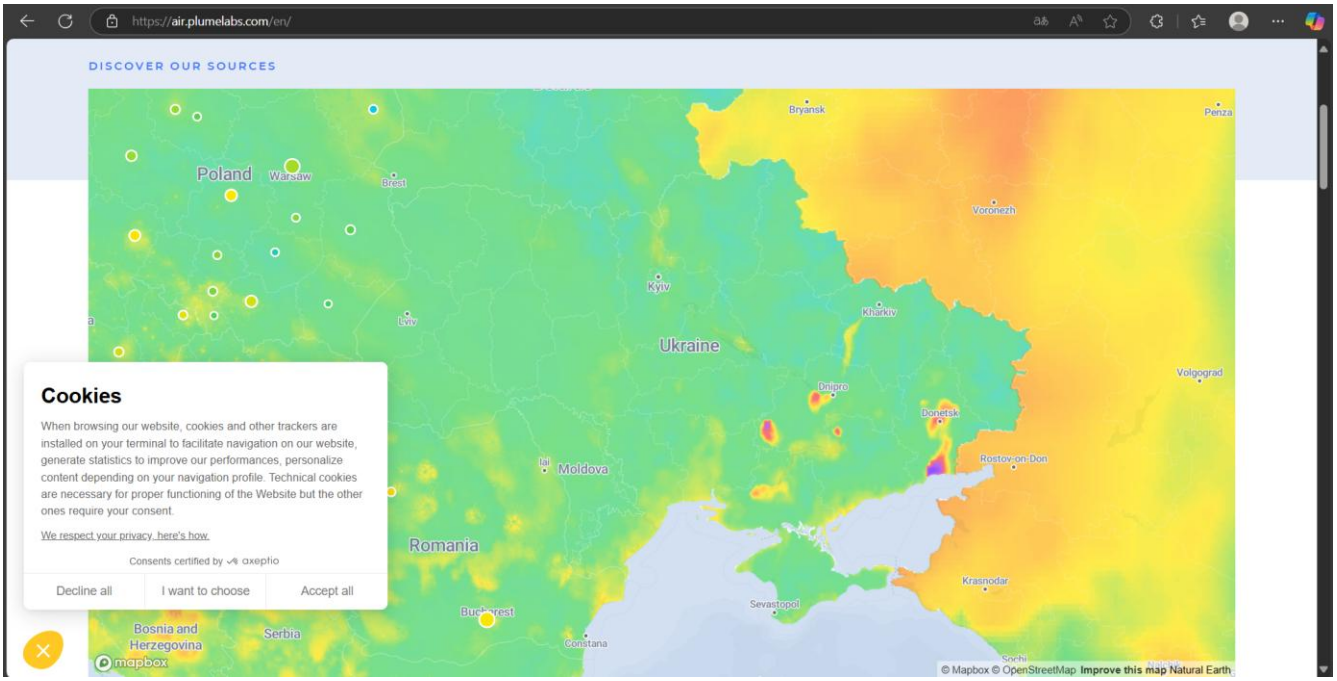


Рисунок 1.2 – Приклад роботи застосунку «PlumeLabs»

ЕкоЗагроза є офіційною мобільною системою міністерства захисту довкілля та природних ресурсів України [12]. Вона надає інформацію про стан повітря, ґрунту та інші дані про довкілля, а також дозволяє користувачу переглядати дані моніторингових систем та актуальні факти наявних екологічних загроз. Приклад роботи зображено на рисунку 1.3.

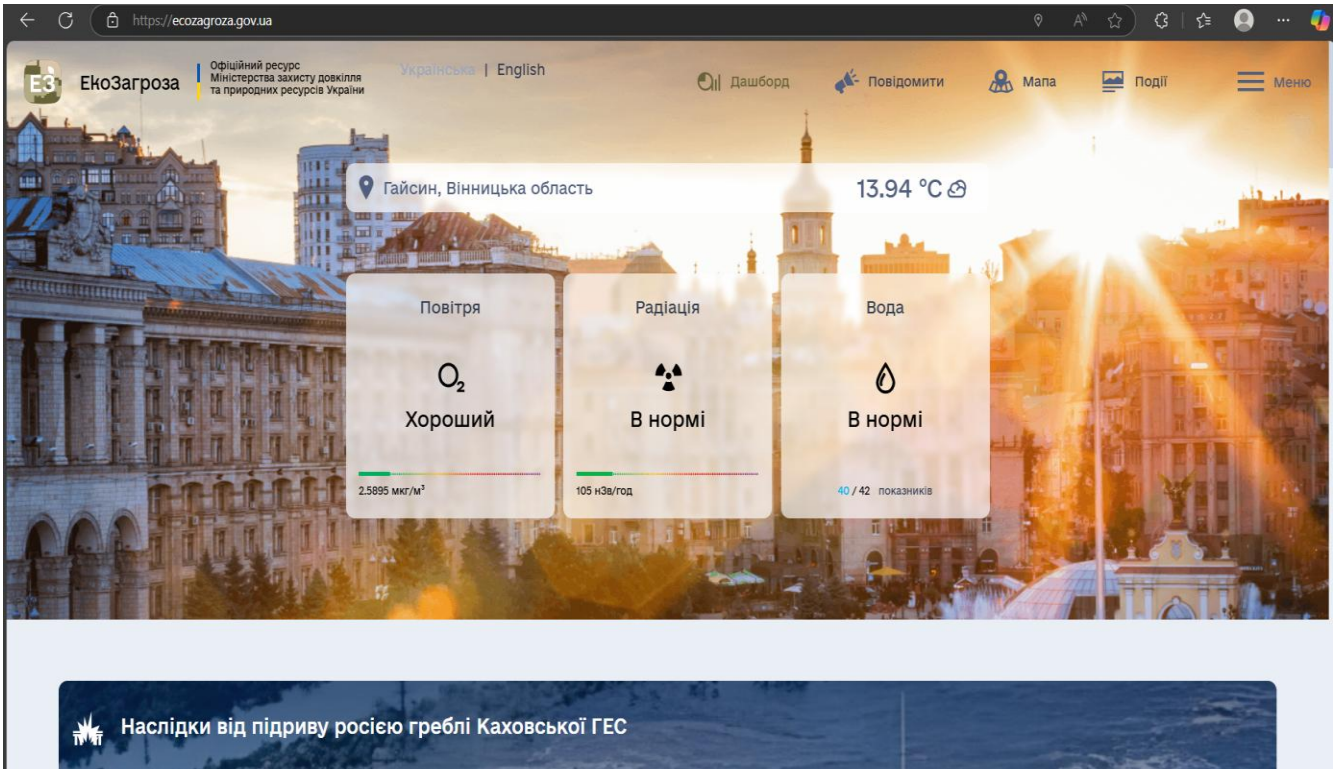


Рисунок 1.3 – Приклад роботи застосунку «Екозагроза»

Отже, після знайомства з усіма аналогами, було детально проаналізовано і вирішено, яким саме функціоналом варто наділити створювану вебсистему для аналізу й моніторингу екологічної ситуації мікрорайону.

Результати порівняння аналогів зведено в таблицю 1.1.

Таблиця 1.1 – Порівняльна характеристика аналогів

Критерій оцінювання	AirVisual	PlumeLabs	Екозагроза	Власна розробка
Наявність карти із зоною моніторингу	1	1	1	1
Автоматизоване отримання даних з API	1	1	0	1
Геолокація користувача	1	0	0	1

Продовження таблиці 1.1

Критерій оцінювання	AirVisual	PlumeLabs	Екозагроза	Власна розробка
Функціонал PWA	0	0	0	1
Збереження історії відвідування в особистому кабінеті	0	0	0	1
Сумарний коефіцієнт	3	2	1	5

Відповідно до таблиці з порівнянням функціональних можливостей аналогів із власною розробкою для аналізу й моніторингу екологічної ситуації із можна підсумувати, що власна розробка є доречною і доцільною для реалізації й впровадження.

Розроблювана система має потенціал покрити недоліки усіх наявних систем й розширити їх можливості, ставши найкращою вебсистемою, де кожен зможе детально бачити стан повітря і грунту у своєму мікрорайоні.

Отже, розробка програмних засобів для аналізу й моніторингу екологічної ситуації мікрорафону відкриває нові горизонти для розвитку громадських ініціатив, локальних екологічних проектів і цифрових сервісів у сфері довкілля.

Такий підхід сприятиме підвищенню екологічної свідомості населення, залученню мешканців до вирішення локальних проблем і створить ефективний інструмент зворотного зв’язку між громадянами та органами місцевої влади. Це є важливим кроком до сталого розвитку, цифровізації екологічного нагляду та покращення якості життя в межах мікрорайонів.

1.3 Аналіз методів розв'язання задачі

Розробка вебсистеми для моніторингу та аналізу екологічної ситуації в мікрорайоні потребує практичного поєднання кількох сучасних технологій, з урахуванням таких вимог, як швидке завантаження, інтерактивність та можливість інтеграції з реальними екологічними API.

У межах проєкту було розглянуто декілька підходів до реалізації застосунку, і після порівняльного аналізу обрано комбінований підхід, який поєднує просту реалізацію, ефективну взаємодію з сервером і високий рівень користувацького досвіду.

Перш за все, було прийнято рішення створити класичну багатосторінкову вебсистему [13] з використанням HTML, CSS, JavaScript, де кожна сторінка виконує свою конкретну функцію. Такий підхід забезпечив:

- простоту реалізації й легке розгортання;
- можливість відображення різного типу контенту без складної маршрутизації;
- гнучкість при оновленні або розширенні функціональності.

Для динамічного виводу даних та обробки запитів використовувався Node.js [14] із фреймворком Express, що дозволяє ефективно обробляти API-запити, забезпечує підключення до MongoDB та реалізацію авторизації через JWT-токени.

На стороні клієнта для виводу аналітичних даних використовувалася бібліотека Chart.js [15], яка дозволяє візуалізувати дані у вигляді графіків і діаграм. Інтеграція з Google Maps API [16] дала змогу відображати геодані в реальному часі, включаючи маркери забруднення й екологічні зони ризику.

Окрему увагу було приділено реалізації PWA функціоналу. Завдяки використанню сервіс-воркерів та створенню Progressive Web App, вебзастосунок отримав такі переваги:

- кешування сторінок, стилів та скриптів;

– швидкий повторний доступ до сторінок;
– підтримка встановлення сайту як мобільного додатку.

Для збору та збереження даних користувача використовувалася MongoDB, а також LocalStorage для тимчасового збереження налаштувань на клієнті.

Таким чином, було обрано оптимальне поєднання класичної архітектури MPA з елементами сучасного веброзроблення, що дозволило досягти високої продуктивності, стабільності в умовах слабкого з'єднання, і водночас зберегти гнучкість у реалізації.

1.4 Постановка задач дослідження

Проаналізувавши переваги і недоліки наявних програмних рішень для аналізу й моніторингу екологічної ситуації мікрорайону з використання вебтехнологій було поставлено наступні цілі, що повинні бути виконані для успішної реалізації задуманого:

– розробити архітектуру клієнт-серверної вебсистеми, що дозволяє збирати та обробляти екологічні дані у режимі реального часу з можливістю масштабування;
– реалізувати систему авторизації користувачів із використанням JWT-токенів, що забезпечує базовий рівень безпеки і персоналізації функціоналу;
– інтегрувати відкриті API для отримання даних про якість повітря та ґрунту з прив'язкою до геолокації користувача;
– розробити модуль аналітики з використанням бібліотеки Chart.js для побудови графіків зміни екологічних параметрів;
– забезпечити швидкий доступ до даних шляхом реалізації PWA із service worker та кешуванням ресурсів;

- створити інтерфейс користувача, який буде адаптований до мобільних пристроїв і забезпечить інтуїтивну взаємодію з системою;
- реалізувати функціонал збереження та перегляду власних сповіщень, які стосуються екологічних проблем у мікрорайоні;
- забезпечити підтримку багатосторінкової навігації з можливістю швидкого переходу між сторінками «Якість повітря», «Якість ґрунту», «Особистий кабінет»;
- протестувати коректність роботи вебсистеми.

Технічне завдання на розробку наведено в додатку А.

1.5 Висновки

У першому розділі було проаналізовано предметну область, що охоплює сучасні підходи до моніторингу екологічної ситуації на рівні мікрорайону з використанням веб- та мобільних технологій. Розглянуто актуальні проблеми в цій галузі, зокрема потребу в доступі до достовірної інформації про якість повітря та ґрунту, а також обмеження існуючих сервісів у контексті локалізації, інтерактивності і персоналізації.

Проведено порівняльну характеристику популярних рішень, таких як AirVisual, IQAir, Plume Labs та вітчизняного рішення ЕкоЗагроза. На основі оцінювання функціональних можливостей було встановлено, що жоден із проаналізованих сервісів не охоплює одночасно всі необхідні компоненти — інтерактивну карту, аналітичні графіки, геолокаційну персоналізацію, історію сповіщень та адаптацію до мобільного середовища.

У межах аналізу методів розв’язання задачі були розглянуті підходи до побудови клієнт-серверної архітектури, зокрема: використання REST API, серверного рендерингу, застосування прогресивного вебдодатку для забезпечення

швидкого доступу, а також вибір оптимальних технологій — HTML/CSS/JavaScript, Node.js, MongoDB, Chart.js, Google Maps API.

Визначено основні функціональні вимоги до вебсистеми, серед яких: реалізація авторизації, отримання та відображення екологічних даних на карті, побудова графіків, формування рекомендацій, підтримка інтерфейсу користувача для персональної взаємодії з даними.

На основі проведеного аналізу сформовано чітку постановку задачі дослідження, яка включає як технічні, так і логічні аспекти реалізації майбутньої системи. Таким чином, було обґрунтовано доцільність розробки власної мобільної екологічної системи, яка здатна заповнити прогалини у функціоналі існуючих аналогів, підвищити екологічну обізнаність користувачів і забезпечити сучасний інструмент моніторингу навколишнього середовища.

Було доведено доцільність власної розробки шляхом аналізу існуючих імплементацій схожих вебсистем і сформовано список завдань, що необхідно виконати.

2 РОЗРОБКА МЕТОДІВ, МОДЕЛЕЙ ТА АЛГОРИТМІВ РОБОТИ СИСТЕМИ

2.1 Аналіз даних

У сучасних системах важливу роль відіграє збирання, обробка та зберігання даних користувача, особливо у випадках, коли йдеться про екологічний моніторинг, де точність, швидкість реакції і персоналізація — критичні показники якості сервісу. Програмна система моніторингу екології мікрорайону, яку було розроблено, базується на ряді взаємопов’язаних модулів і методів, що забезпечують ефективний аналіз, візуалізацію та збереження екологічної інформації, персоналізованої для кожного.

Збір екологічних даних відбувається за допомогою запитів до зовнішніх API, що забезпечують доступ до актуальних показників якості повітря та стану ґрунту. Отримані дані одразу обробляються та зберігаються в базі даних MongoDB у вигляді структурованих об’єктів із параметрами.

Після завершення реєстрації, користувач має персональний акаунт, в який може здійснювати вхід з використанням пошти та паролю.

Кожен із модулів моніторингу реалізує окремий механізм запитів до відповідного API, обробки отриманих значень та їх інтеграції з картою Google Maps.

Вхідними даними є геокоординати користувача або вибраної ділянки, вихідними — маркери на карті з інформаційним блоком про стан середовища, а також додаткові рекомендації. Дані автоматично кешуються для підвищення продуктивності.

Для візуального аналізу змін екологічної ситуації застосовується бібліотека Chart.js, яка дає змогу формувати динамічні графіки згідно відомих значень.

Вхідними даними є масиви значень з MongoDB, вихідними — анімовані графіки у форматі ліній чи стовпців.

Користувачі після авторизації мають доступ до особистого кабінету, де зберігаються та відображаються всі їхні дії: історія переглядів, отримані екосповіщення, обраний регіон моніторингу згідно місцезнаходження, графіки змін, а також профільна інформація. Профіль можна редагувати, дані зберігаються у MongoDB. Кабінет також дозволяє переглядати лише ті сповіщення, які стосуються конкретного користувача чи його локації.

Система підтримує формат Progressive Web App, завдяки чому користувачі можуть встановлювати вебдодаток на мобільні пристрої. PWA-модуль реалізує збереження основних сторінок та кешування даних через Service Worker [17], що дозволяє працювати навіть без входу в браузер.

Додатково реалізовано кнопку «Встановити як додаток», яка доступна постійно.

Всі важливі ресурси попередньо кешуються, включаючи API-відповіді для останніх запитів.

MongoDB використовується як головне сховище для екологічних сповіщень. Кожне повідомлення містить інформацію про джерело, час, рівень небезпеки і тип проблеми.

При кожному вході в систему або при зміні локації користувача, виконується перевірка, чи є сповіщення, що відповідають поточному регіону. Якщо такі є — вони передаються користувачеві у вигляді списку або графічного маркера на карті.

2.2 Розробка загальної моделі вебсистеми

Для глибшого розуміння роботи системи було створено її модель у вигляді діаграми класів з усіма основними сутностями, що використовуються, і зображено на рисунку 2.1.

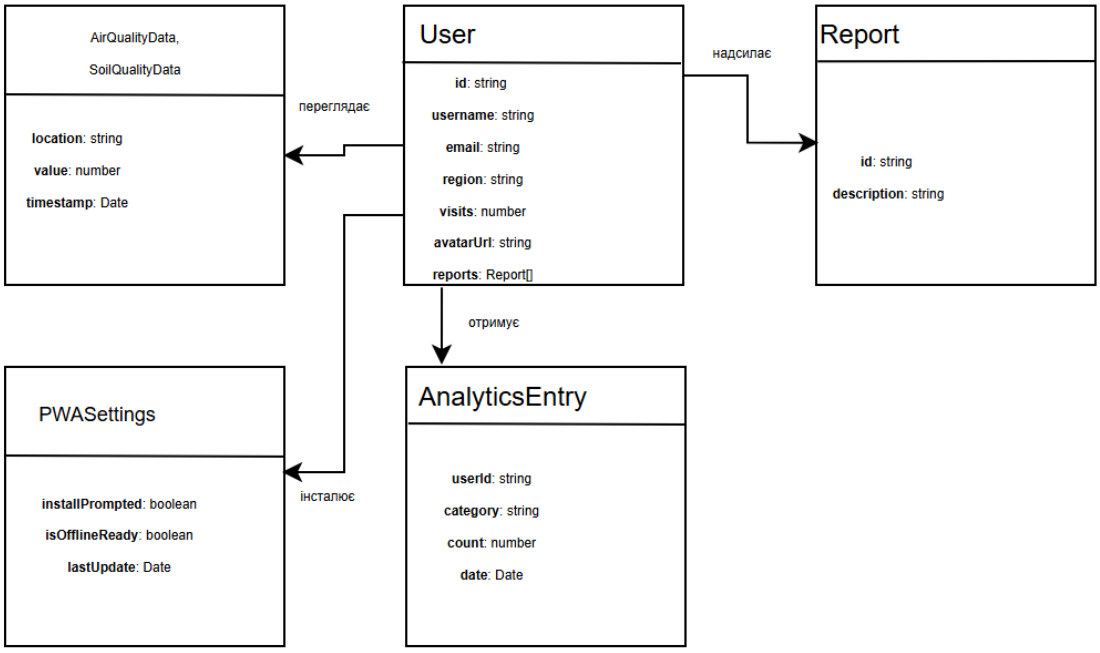


Рисунок 2.1 – Модель вебсистеми

Однією з основних сутностей вебсистеми є клас User, що містить інформацію про користувача. Дана сутність включає поля: id — унікальний ідентифікатор користувача, email — електронна пошта, region — обрана область або місто проживання, avatar — зображення профілю користувача, а також visits — масив дат входів до системи, що зберігається для побудови графіка активності. Клас [18] User має зв’язок типу один-до-багатьох з класом Report, адже кожен користувач може створювати кілька екологічних сповіщень.

Клас Report реалізує механізм повідомлення про локальні екологічні проблеми. Він містить такі поля: id, title — заголовок повідомлення, description — його опис. Клас має зв’язок багато-до-одного із User, оскільки багато сповіщень можуть належати одному користувачу.

Також реалізовано модуль AnalyticsEntry, що ґрунтується на полі visits у класі User. На основі цього масиву будується стовпчиковий графік, що візуалізує

активність користувача за днями. Виведення графіка реалізовано через бібліотеку Chart.js, що генерує візуальний компонент інтерфейсу.

Окремі сутності відповідають за модулі моніторингу якості повітря та ґрунту. У кожному з цих модулів реалізовано механізм отримання екологічних даних з відкритих API, з наступним аналізом і побудовою графіків. Дані не зберігаються в базі, а завантажуються динамічно при відкритті відповідної сторінки. Вони відображаються на карті разом із рекомендаціями щодо дій користувача. В кожному модулі реалізовано пошук за геолокацією користувача та побудову динамічного графіка на основі отриманих показників.

Клас PWASettings реалізує механізми, пов’язані з встановленням сайту [19] як прогресивного додатку. До його складу входять: manifest.json — файл конфігурації застосунку, service-worker.js — файл, що відповідає за кешування сторінок та збереження ресурсів. Додатково реалізовано кнопку, яка дозволяє користувачеві встановити застосунок вручну.

В особистому кабінеті користувача реалізовано окремий модуль відображення персональної інформації, зображення профілю, останніх створених сповіщень та інтерактивного графіка активності. Користувач має можливість редагувати електронну пошту, регіон проживання та змінювати аватар, що зберігається у localStorage та в базі даних MongoDB.

Отже, було розроблено діаграму класів, що охоплює основні сутності вебсистеми моніторингу екологічної ситуації та зв’язки між ними. Усі діаграми та блок-схеми були створені відповідно до реалізованого функціоналу системи в середовищі MongoDB, Node.js та інтерфейсної частини JavaScript/HTML/CSS.

2.3 Розробка загального алгоритму роботи програми

Загальний алгоритм роботи екологічної вебсистеми є основою для її подальшої розробки та підтримки. Оскільки система містить декілька окремих сторінок, обробку геоданих, побудову графіків, а також підтримує персоналізовані сповіщення, було створено узагальнений сценарій її роботи, що демонструє логіку переходів між станами та діями користувача.

Після завантаження сторінки система перевіряє наявність авторизаційного токена у браузері. Якщо токен валідний — користувач автоматично перенаправляється до особистого кабінету, де може переглядати свою статистику, профіль, а також екологічні сповіщення.

Якщо токена немає, або він не дійсний, користувачеві пропонується виконати вхід або пройти реєстрацію. Після успішної авторизації створюється токен, а дані користувача зберігаються у MongoDB. Після цього відкривається доступ до всіх сторінок вебсистеми.

З головної сторінки користувач може перейти на будь-яку з двох тематичних карт: якість повітря або якість ґрунту. На кожній з цих сторінок здійснюється запит до відповідного API, побудова графіка, завантаження Google-карти, а також відображення рекомендацій, залежно від отриманих параметрів.

Крім того, система дозволяє створювати екологічні сповіщення. Ці сповіщення створюються через форму, після чого зберігаються у MongoDB разом із userId. Усі сповіщення користувача доступні в його кабінеті, а загальні сповіщення — адміністраторам або у майбутніх розширеннях.

Також система має PWA-функціональність, яка дозволяє встановити сайт як додаток, а також відображати кнопку встановлення.

Загальний алгоритм роботи вебсистеми екологічного моніторингу:

1) завантаження сторінки;
2) перевірка наявності JWT-токена [20] у localStorage;

3) якщо токен валідний, то відбувається перехід до dashboard.html;
4) якщо токена немає, відбувається редирект на сторінку login.html або register.html;
5) після входу: запит до бекенду /api/profile для завантаження персональних даних;
6) відображення профілю, графіку відвідувань і особистих сповіщень;
7) можливість редагування профілю;
8) перехід на сторінки:
   – /air-quality.html — завантаження API якості повітря, карта, графік, рекомендації;
   – /soil-quality.html — API для ґрунту, карта;
9) додавання екосповіщень через /api/reports;
10) встановлення як PWA-додаток;
11) автоматичне оновлення кешу та ресурсів при новому запуску.

Додатково варто зазначити, що система орієнтована на мобільну та десктопну адаптацію, що забезпечує комфортну роботу з будь-якого пристрою. Завдяки реалізованій кеш-логіці, сторінки системи швидко завантажуються навіть у разі повільного з’єднання.

При наявності інтернету дані з API оновлюються автоматично, забезпечуючи актуальність екологічної інформації.

Під час наступного запуску система перевіряє оновлення, завантажує нові ресурси та оновлює інтерфейс без втручання користувача. Це забезпечує стабільність, автономність та надійність роботи вебзастосунку.

Загальний алгоритм роботи вебсистеми продемонстровано на рисунку 2.2.

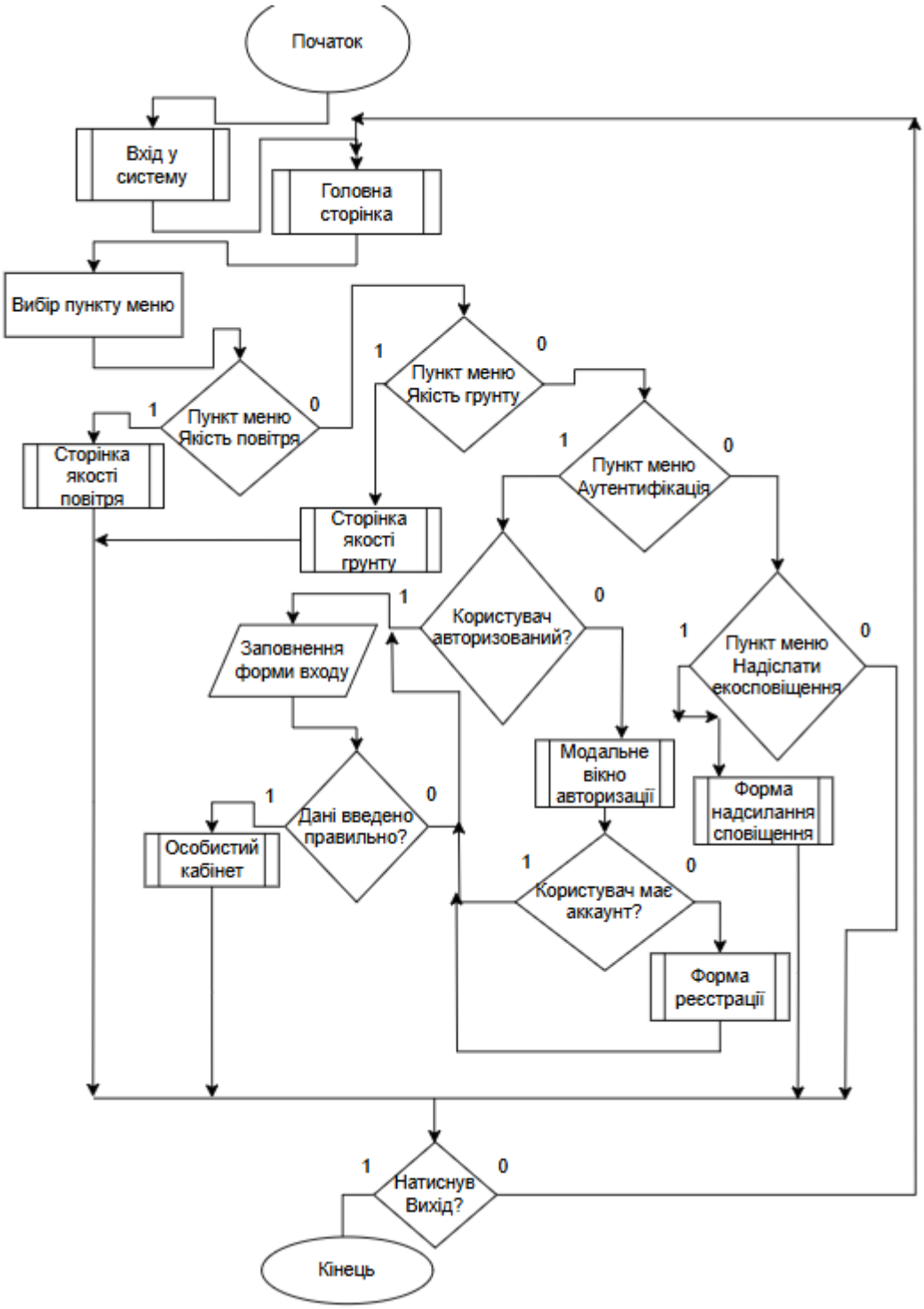


Рисунок 2.2 – Загальний алгоритм роботи вебсистеми екологічного моніторингу

Було продемонстровано загальний алгоритм роботи програми, де можна побачити увесь функціонал, що містить система.

2.4 Розробка методу збору екологічного сповіщення

Екологічна вебсистема надає можливість кожному авторизованому користувачу створювати власні екологічні сповіщення, які інформують про забруднення повітря і ґрунту або інші екологічні проблеми. Створення сповіщення дозволяє системі накопичувати локальні дані, пов’язані з довкіллям, а також вести історію активності користувача в особистому кабінеті.

Метод збору екологічного сповіщення виконується у декілька послідовних кроків:

1) на сторінці створення сповіщення користувач заповнює форму, що складається з таких полів, як title – короткий заголовок проблеми і description – опис ситуації;
2) після заповнення форми відбувається її перевірка на валідність: усі поля мають бути заповнені, інакше система виводить повідомлення про помилку;
3) якщо форма пройдена успішно, її дані передаються у функцію onSubmit(), яка формує JSON-об’єкт [21] і відправляє POST-запит на бекенд за маршрутом POST /api/reports;
4) на серверній стороні Express.js [22] бекенд:
   – приймає запит;
   – перевіряє авторизацію за токеном;
   – створює новий документ у колекції reports у базі даних MongoDB з усіма даними форми;
   – додає мітку часу та унікальний ідентифікатор.
5) у разі успіху сервер відповідає статусом 200 OK, і користувач отримує повідомлення: «Сповіщення успішно надіслано»;

6) усі надіслані сповіщення автоматично додаються до особистого кабінету користувача у вигляді списку, що завантажується при кожному вході через маршрут GET /api/reports;
7) кожне сповіщення відображається зі статусом, типом, заголовком та кнопкою для детальнішого перегляду або редагування, але це буде у майбутніх оновленнях;
8) для збереження швидкої роботи застосунку, сповіщення тимчасово кешуються у localStorage, а при відновленні підключення – синхронізуються з сервером.

Технологічні особливості методу:

- реалізовано з використанням JavaScript на клієнті, Node.js + Express на сервері;
- для збереження даних використовується MongoDB, де зберігається колекція reports;
- HTTP-запити виконуються через нативний fetch() або бібліотеку Axios;
- захист реалізовано через JWT-автентифікацію, тобто лише авторизовані користувачі можуть створювати сповіщення;
- передбачено обробку помилок, порожніх полів, дублювання та неправильного формату.

Таким чином, описаний метод дає змогу кожному авторизованому користувачеві:

- зафіксувати локальну екопроблему;
- зберегти її у базу;
- переглядати власну історію сповіщень у кабінеті;
- працювати навіть без входу у браузер.

Блок – схему роботи алгоритму роботи методу збору екологічного сповіщення зображено на рисунку 2.3.

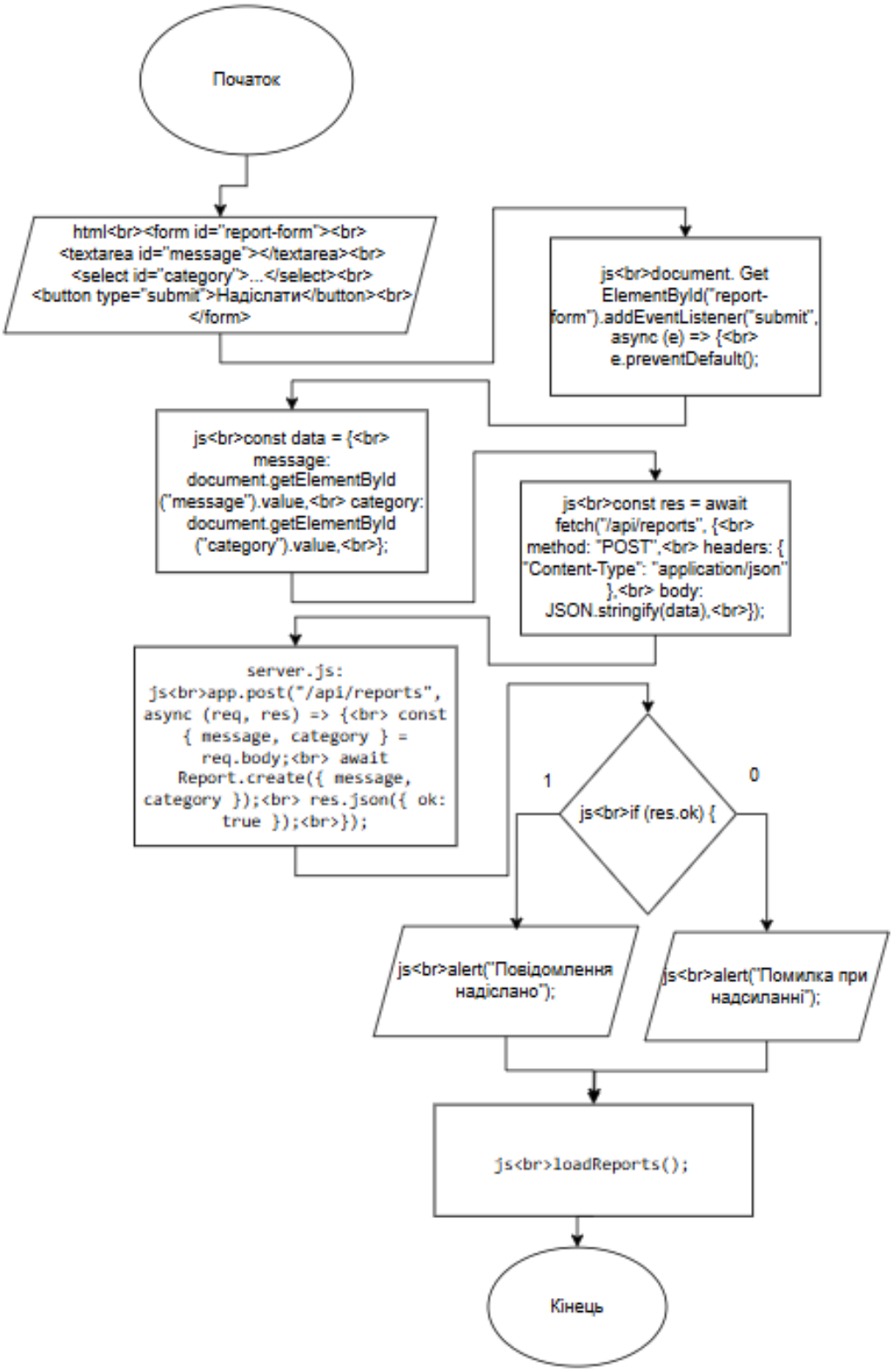


Рисунок 2.3 – Блок-схема алгоритму роботи методу збору екологічного сповіщення у програмній системі

2.5 Розробка методу побудови динамічних графіків та візуалізації даних

У вебсистемі екологічного моніторингу реалізовано функціонал динамічної візуалізації екологічних показників за допомогою графіків та інтерактивних елементів, що дозволяє користувачам зручно переглядати динаміку змін якості повітря і ґрунту. Графіки використовуються як у загальнодоступних розділах сайту, так і в особистому кабінеті користувача для відображення історії відвідувань і сповіщень.

Для реалізації методу використовується відкрита JavaScript-бібліотека Chart.js, яка забезпечує можливість побудови стовпчикових графіків на основі даних, отриманих з API або бази даних.

Метод побудови динамічних графіків та візуалізації даних складається з таких кроків:

1) збір даних - для графіків, пов’язаних з екологією, дані запитуються через маршрут GET /api/air або /api/soil – залежно від тематики сторінки, дані запитуються з Node.js серверу, який взаємодіє або з MongoDB, або з зовнішнім API, у разі персоналізованої аналітики запит надсилається на GET /api/profile і містить JWT-токен користувача для авторизації;
2) обробка даних – отримані дані перетворюються у формат, придатний для побудови графіка, за допомогою JavaScript, з використанням методів map() та reduce() створюється масив значень та міток часу, які будуть відображені на осі X і Y, при потребі значення округлюються до цілих чисел або форматуються за допомогою Math.round();
3) побудова графіка – за допомогою Chart.js створюється новий екземпляр графіка:

```
new Chart(ctx, {
  type: 'bar',
  data: {
    labels: [...],
    datasets: [{
      label: 'Індекс якості',
      data: [...],
    }]
  },
  options: {
    responsive: true,
    scales: {
      y: { beginAtZero: true }
    }
  }
});
```

Рисунок 2.4- Екземпляр графіка, частина коду

Графік розміщується у відповідному HTML-контейнері - <canvas id="chart">, завдяки опції responsive, графік адаптується до розміру пристрою;

1) візуалізація в особистому кабінеті – у dashboard.html для кожного користувача відображається графік відвідувань або сповіщень, дані беруться з MongoDB через маршрут /api/reports і фільтруються за автором, далі обробляються та подаються у вигляді стовпчикового графіка кількості активностей по днях/тижнях;
2) кешування – завдяки реалізованому service worker та підтримці localStorage, графік може побудуватись без входу в браузер на основі збережених раніше даних.

Особливості методу:

- chart.js дозволяє швидко генерувати інтерактивні графіки без складної візуальної логіки;

- дані з API обробляються динамічно у браузері;
- структура адаптована до прогресивного вебзастосунку;
- дані представлені у вигляді JSON, а запити виконуються через fetch() або Axios;

Таким чином, побудова динамічних графіків дає змогу візуально аналізувати екологічну ситуацію, відстежувати зміни за часом, отримувати персональні дані у зручній формі та підтримувати стабільність роботи навіть у нестабільних мережевих умовах. Блок – схему роботи методу побудови динамічних графіків та візуалізації даних показано на рисунку 2.5.

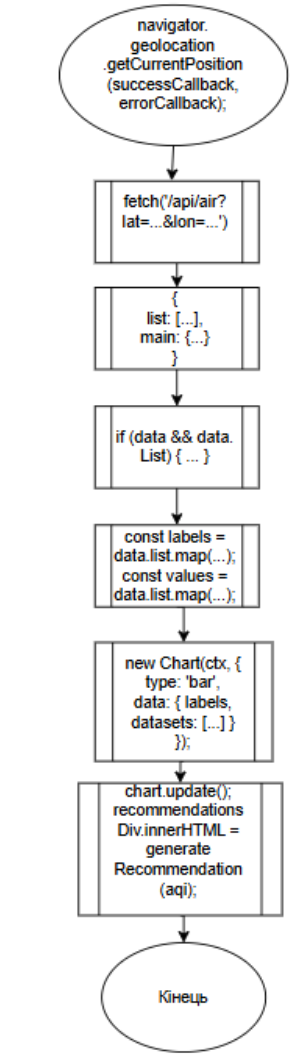


Рисунок 2.5 – Блок-схема алгоритму роботи методу побудови динамічних графіків та візуалізації даних у вебсистемі

2.6 Висновки

У другому розділі було проведено аналіз вхідних та вихідних даних вебсистеми аналізу й моніторингу екологічного стану мікрорайону. Створено та пояснено загальний алгоритм роботи вебсистеми. Розроблено такі алгоритми і методи:

1. Алгоритм збору екологічних сповіщень – базується на дії користувача, що надсилає повідомлення про проблему. Дані обробляються сервером та зберігаються у базі MongoDB.

2. Алгоритм побудови динамічних графіків – включає етапи запиту, обробки й візуалізації екологічних показників за допомогою бібліотеки Chart.js. Розроблено методи: автоматизованого підбору контенту, збору інформації про користувача та формування аналітичних графіків. Логіку обох методів візуалізовано у вигляді блок-схеми.

3. Метод побудови динамічних графіків та візуалізації даних – дозволяє авторизованим користувачам бачити свою екологічну активність у вигляді графіка.

4. Метод збору екологічного сповіщення – дозволяє користувачам надсилати повідомлення про екологічні проблеми у своєму мікрорайоні.

Усі діаграми, включаючи модель роботи вебсистеми, було розроблено за допомогою «UML».

3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ СИСТЕМИ

3.1 Варіантний аналіз і обґрунтування вибору засобів для реалізації вебсистеми

У процесі створення системи для моніторингу та аналізу екологічної ситуації мікрорайону було особливо важливо обрати такі інструменти, які дозволяють ефективно реалізувати інтерфейс, забезпечити взаємодію з базою даних у реальному часі, а також створити масштабований бекенд і підтримку швидкого доступу без входу в браузер завдяки PWA - функціональності.

Для реалізації клієнтської частини було використано мови HTML, CSS та JavaScript, які забезпечують кросбраузерну сумісність і високий рівень кастомізації. Основною бібліотекою для побудови аналітичних графіків обрано Chart.js – популярну JavaScript-бібліотеку, яка дозволяє створювати динамічні та адаптивні діаграми на основі екологічних даних.

У таблиці 3.1 наведено порівняння популярних бібліотек для візуалізації даних.

Таблиця 3.1 – Порівняльна мов програмування

Критерії	Chart.js	D3.js	Google Charts
Простота у використанні	1	0	1
Легкість інтеграції у HTML	1	0	1
Гнучкість налаштування	1	1	0
Підтримка адаптивності	1	1	1
Сумарний коефіцієнт	4	2	3

На основі наведених критеріїв Chart.js було обрано як оптимальне рішення для побудови графіків.

Для створення картографічної візуалізації інтегровано Google Maps JavaScript API, який забезпечує гнучкість у відображенні геолокаційних даних, таких як зони забруднення, рівень якості довкілля та рекомендації в реальному часі.

Для забезпечення бекенд-функціоналу було обрано Node.js разом із фреймворком Express, що дозволяє швидко створювати REST API, зручно обробляти запити, підключати базу даних та реалізовувати авторизацію. Запити від клієнта до сервера здійснюються через такі стандартні HTTP-методи як GET, POST, PUT, DELETE з використанням функції fetch.

Для зберігання та обробки даних було обрано MongoDB, яка дозволяє ефективно працювати з JSON-подібними структурами, є гнучкою до змін у структурі даних і підтримує горизонтальне масштабування. Хостинг бази даних реалізовано через Railway, який також використовується для хостингу самого Node.js-сервера.

Важливою частиною системи є можливість встановлення сайту як мобільного застосунку. Для цього реалізовано Progressive Web App з використанням manifest.json, service-worker.js і кешування ресурсів. Це дозволяє користувачам працювати з системою навіть не заходячи в браузер.

Для розгортання фронтенду було обрано GitHub Pages, що забезпечує безкоштовний і швидкий хостинг із доступом до сайту за публічним URL. Для підключення фронтенду до бекенду було здійснено заміну всіх локальних запитів на публічний URL Railway-сервера.

Середовище розробки — Visual Studio Code [23], яке надає підтримку JavaScript/Node.js, підключення до Git, зручне налагодження коду та широкі можливості через розширення.

Таким чином, обраний стек технологій включає:

- Frontend - HTML, CSS, JavaScript, Chart.js, Google Maps API;
- Backend - Node.js, Express.js;

- база даних - MongoDB;
- хостинг - GitHub Pages, Railway;
- PWA-компоненти - manifest.json, service worker;
- інструменти розробки - Visual Studio Code, GitHub.

Поєднання зазначених інструментів дозволило реалізувати сучасну, динамічну, екологічно орієнтовану вебсистему з високим рівнем взаємодії з користувачем, адаптацією до мобільних пристроїв та підтримкою функції швидкого доступу. Це забезпечує надійність, швидкість та масштабованість платформи.

3.2 Розробка модулів якості повітря і грунту

Модулі оцінки якості повітря та ґрунту є ключовими елементами вебсистеми екологічного моніторингу. Кожен із них реалізований як окрема сторінка з унікальним дизайном, власною логікою завантаження даних, обробкою геолокації, відображенням графіків та рекомендацій. Основною метою модулів є забезпечення користувача актуальною інформацією щодо стану навколишнього середовища у його мікрорайоні в режимі реального часу.

Основні функції модулів описані нижче.

Кожен модуль має однакову загальну структуру:

- інтерактивна карта, що відображає забруднені ділянки;
- геолокаційний запит [24];
- отримання даних через API;
- візуалізація даних у вигляді стовпчикових графіків через бібліотеку Chart.js;
- панель з рекомендаціями щодо поведінки у разі виявлення забруднення;
- можливість зміни анімованого фону.

Усі модулі мають схожу архітектуру — клієнтська частина взаємодіє з бекендом, який, у свою чергу, опрацьовує запит до зовнішніх API, зберігає дані в MongoDB та повертає користувачу відповідь.

1. Модуль якості повітря. Сторінка air-quality.html реалізована як окрема SPA-компонента [25]. Вона завантажується за маршрутом /air-quality і одразу виконує геолокаційний запит до браузера. Після визначення координат формується запит на маршрут /api/air, що обробляється сервером на Node.js.

На стороні сервера цей запит надсилає координати до OpenWeather Air Pollution API, звідки отримуються дані за PM2.5, PM10, CO, O_3 і подібних. Отримані значення передаються на клієнтську частину, де вони будуються у графік за допомогою Chart.js.

Паралельно створюються круги на Google Maps, радіус яких пропорційний рівню забруднення.

Приклад події:

– якщо data.list порожній — відображається повідомлення про недоступність даних.

Інтерфейс модуля якості повітря з графіком та картою Google Maps показано на рисунку 3.1.

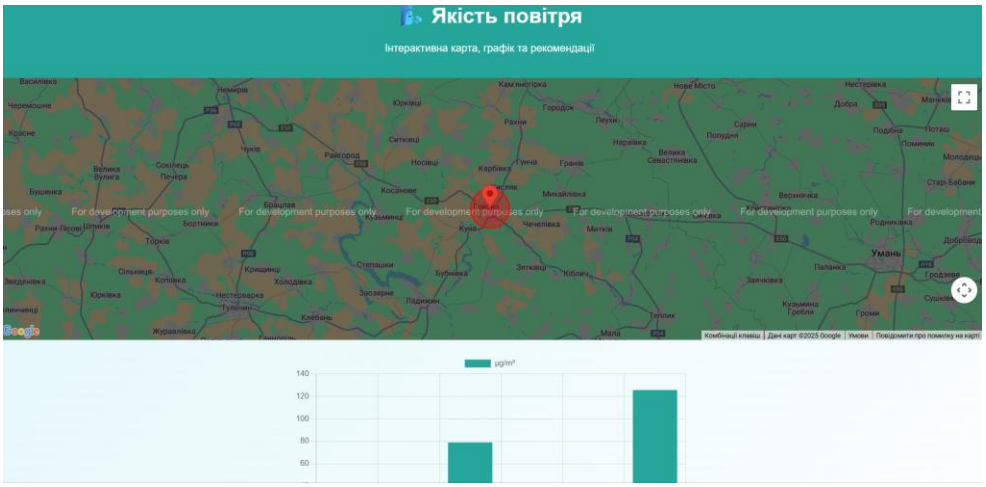


Рисунок 3.1 – Інтерфейс модуля якості повітря з графіком та картою Google Maps

2. Модуль стану ґрунту. Сторінка soil-quality.html реалізована аналогічно модулю якості повітря, проте отримує дані або з внутрішньої бази, або з SaveEcoBot. Ключові показники: вміст важких металів, рівень нітратів, pH, забруднення пестицидами.

На мапі відображаються маркери, які класифікуються кольором залежно від рівня забруднення. Дані зберігаються також у базі даних MongoDB для повторного використання.

Інтерфейс користувача показано на рисунку 3.2.

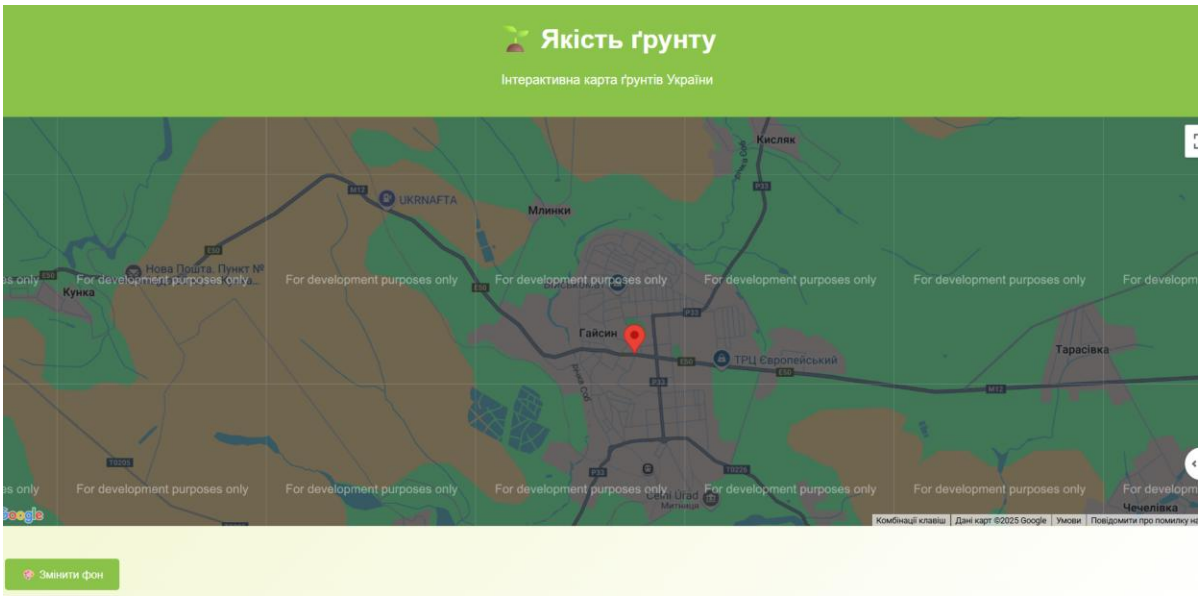


Рисунок 3.2 – Інтерфейс модуля оцінки якості ґрунту з картою

Усі два модулі мають:

– відповідність принципам доступності: використовуються семантичні теги, кольорова контрастність та адаптивний дизайн;
– веалізовану підтримку швидкого доступу через PWA;
– стратегію оновлення через fetch + cache, тобто дані зберігаються локально, але при повторному відкритті порівнюються з API;
– зв’язок із обліковим записом користувача: запити прив’язуються до регіону, вказаного в профілі, а отримані дані зберігаються в історії сповіщень у dashboard.html.

Розробка модулів якості повітря й ґрунту дозволила створити універсальні інтерфейси для екологічного моніторингу, орієнтовані на зручність користувача, динамічне оновлення даних та гнучке масштабування.

Поєднання Google Maps API, Chart.js, серверної обробки на Node.js та зовнішніх API створює надійне підґрунтя для оцінки стану навколишнього середовища в умовах урбанізованого мікрорайону.

3.3 Розробка модуля інтеграції PWA

Надзвичайно важливо надати користувачеві змогу користуватися системою за умов термінової необхідності.. Саме тому було реалізовано інтеграцію технології PWA, що перетворює звичайний вебсайт на майже повноцінний мобільний застосунок, з встановленням на пристрій та локальним кешуванням сторінок.

Вигляд вебсистеми як повноцінного декстопного додатку продемонстровано на рисунку 3.3.

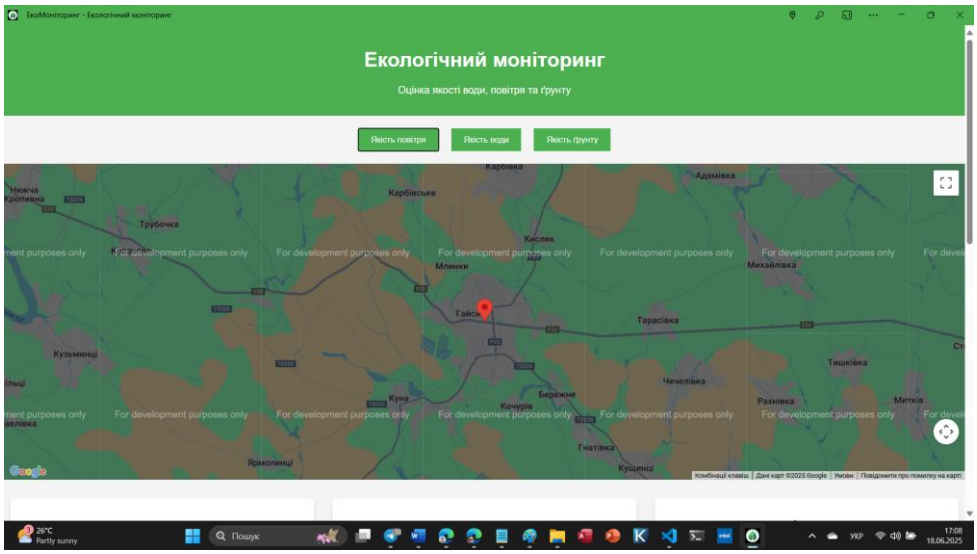


Рисунок 3.3 – Відкрита сторінка додатку на Робочому Столі

Загальний алгоритм роботи PWA в екологічній вебсистемі продемонстровано на рисунку 3.4.

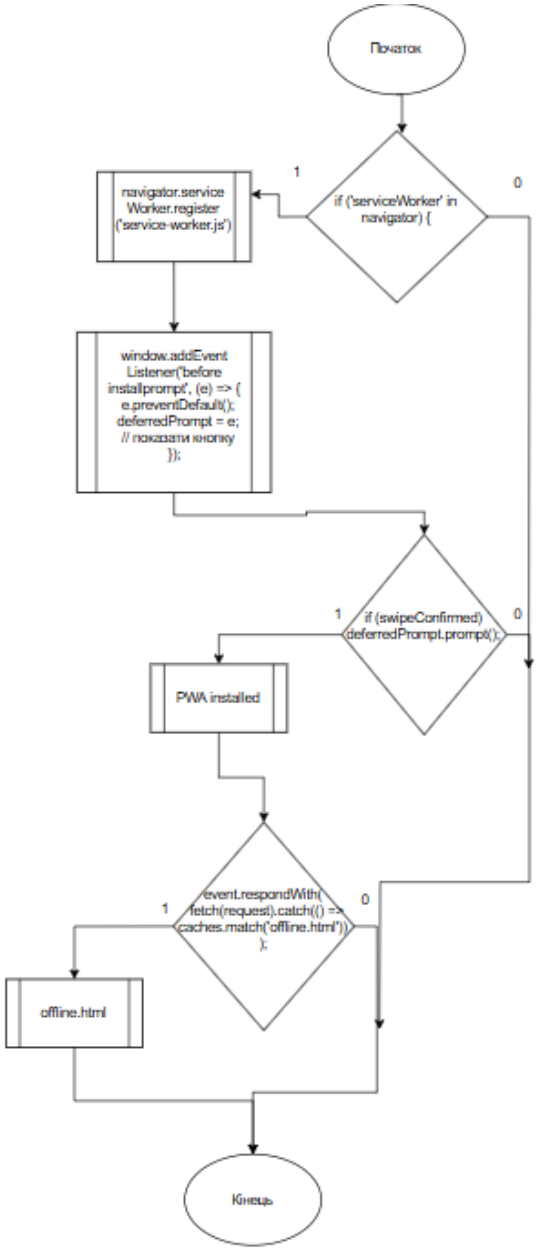


Рисунок 3.4 – Алгоритм роботи модуля інтеграції PWA

Основними компонентами PWA у реалізованому проєкті стали:

– Файл manifest.json, що визначає назву додатка, іконку, кольори теми та режим відображення.
– Скрипт service-worker.js, який обробляє кешування сторінок, ресурсів, зображень та API-відповідей, дозволяючи працювати в екстренному режимі.

– Кнопка встановлення застосунку, реалізована через подію beforeinstallprompt, яка ініціює процес встановлення прогресивного вебдодатка на пристрій користувача.

У ролі локального сховища даних використовувалось Cache Storage API [26], за допомогою якого кешувалися всі основні сторінки:

– /index.html, /dashboard.html, /air-quality.html, /soil-quality.html.

Задля підвищення доступності системи в умовах обмеженого зв'язку, також було реалізовано автоматичне оновлення кешу при повторному завантаженні сторінки, через метод cache.addAll() та fetch + networkFirst стратегію.

На рисунку 3.5 продемонстровано схему інтерфейсу користувача зі встановленням PWA-додатку.

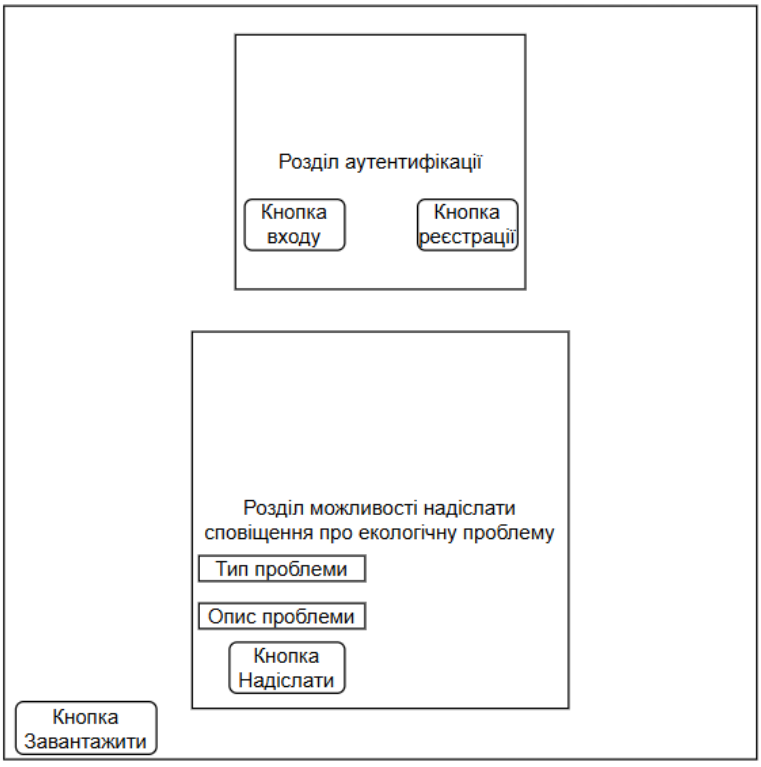


Рисунок 3.5 – Схема взаємодії користувача із модулем встановлення PWA

На рисунку 3.6 зображено готову реалізацію інтерфейсу — кнопки встановлення додатку, яка замінює звичайну кнопку підтвердження. Користувач

повинен натиснути пальцем або мишкою по кнопці, щоб активувати подію installPrompt.prompt().

Рисунок 3.6 – Інтерфейс кнопки встановлення застосунку

Інтерфейс користувача також інформує, чи додаток вже встановлено. Якщо PWA вже інстальовано, кнопка змінює вигляд на «☑ Встановлено» і більше не реагує на клікання. У разі деінсталяції застосунку кнопка знову активується.

Крім того, реалізовано:

– локальне кешування JSON-відповідей API, що дозволяє зберігати останні дані про якість повітря і ґрунту;
– іконка додатку, яка генерується з каталогу icons/ для різних розмірів пристроїв.

Якщо браузер підтримує PWA, користувач може встановити вебсистему на головний екран пристрою, користуватися нею у повноекранному режимі та без адресного рядка.

У разі, якщо браузер або пристрій не підтримують PWA, наприклад, Safari на iOS, система все одно залишається доступною, але встановлення не пропонується.

3.4 Розробка модуля особистого кабінету користувача

Особистий кабінет є центральним елементом взаємодії користувача з екологічною вебсистемою. У ньому відображається профіль користувача, його обраний регіон, статистика відвідувань, список надісланих екологічних сповіщень, а також інтерактивний графік та можливість редагування персональних даних.

Модуль було реалізовано на сторінці dashboard.html з використанням HTML[27], CSS, JavaScript, Chart.js і взаємодії з серверною частиною через REST

API. Основна мета — забезпечити інтуїтивний інтерфейс з можливістю редагування і збереження даних у MongoDB.

На рисунку 3.7 продемонстровано загальний алгоритм роботи модуля особистого кабінету користувача.

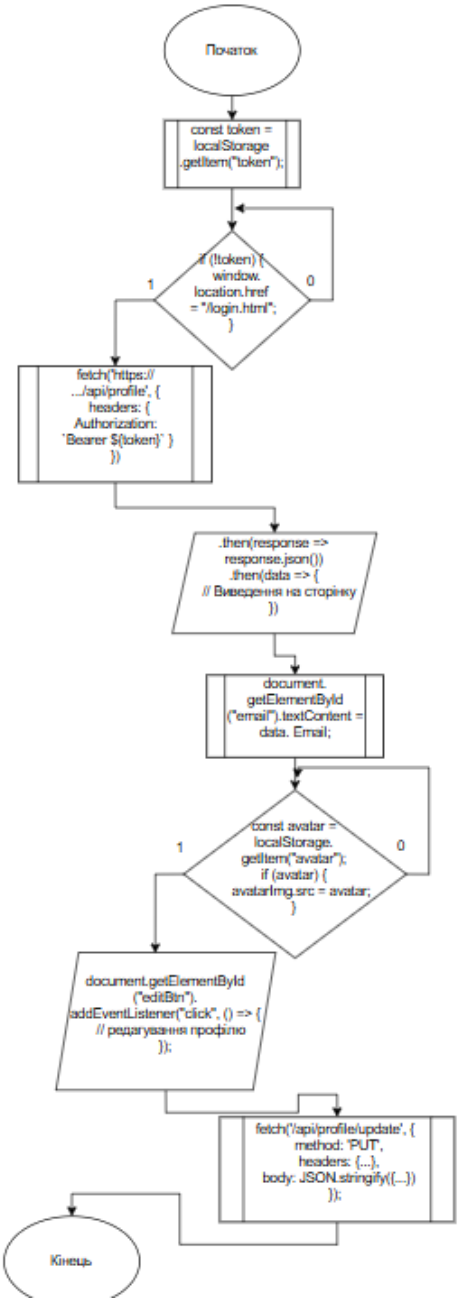


Рисунок 3.7 – Алгоритм роботи особистого кабінету користувача

Основні функціональні блоки модуля:

1. Виведення профілю користувача:

- після авторизації на сторінці dashboard.html надсилається запит до /api/profile, у відповідь повертаються дані з MongoDB - ім’я, email, регіон, аватар;
- аватар завантажується з локального сховища localStorage або з серверної БД, якщо наявний;
- якщо аватар відсутній, відображається зображення за замовчуванням.

2. Редагування профілю:
    - при натисканні кнопки «Редагувати» користувач може змінити email, регіон та обрати новий аватар;
    - аватар відображається у реальному часі до збереження;
    - зміни надсилаються на сервер через PUT-запит до /api/profile/update;
    - дані одразу зберігаються і в localStorage, щоб відображатися без входу в браузер автоматично.
3. Графік активності:
    - побудовано за допомогою бібліотеки Chart.js;
    - відображає статистику відвідувань чи взаємодії користувача зі сторінками якості повітря та ґрунту;
    - графік автоматично оновлюється при кожному заході користувача.
4. Список екосповіщень:
    - виводиться через запит до /api/reports;
    - містить перелік надісланих користувачем проблем із позначкою дати та типом, якщо вказані;
    - якщо повідомлень немає — виводиться повідомлення «У вас ще немає надісланих сповіщень».

Виведення токена:

- у верхній частині сторінки відображається JWT-токен для знеособленого підтвердження авторизації;

– дані із console.log(req.user) виводяться під час запиту для діагностики.

5. Інтерактивний інтерфейс:
   – Аватар можна змінити кліком або перетягуванням файлу.

Схему інтерфейсу користувача особистого кабінету подано на рисунку 3.8.

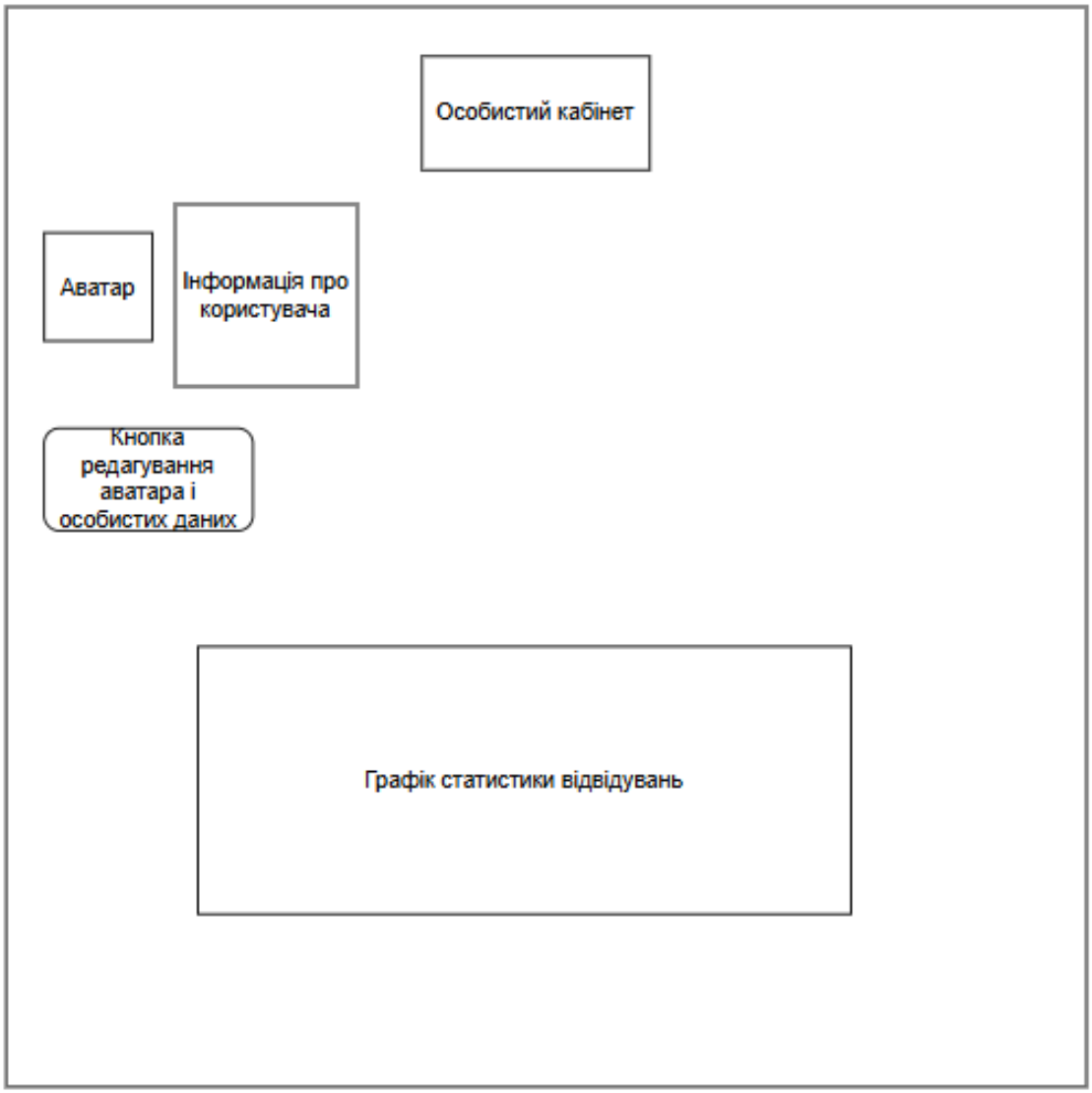


Рисунок 3.8 – Схема інтерфейсу модуля особистого кабінету

Інтерфейс сторінки dashboard.html реалізований на рисунку 3.9.

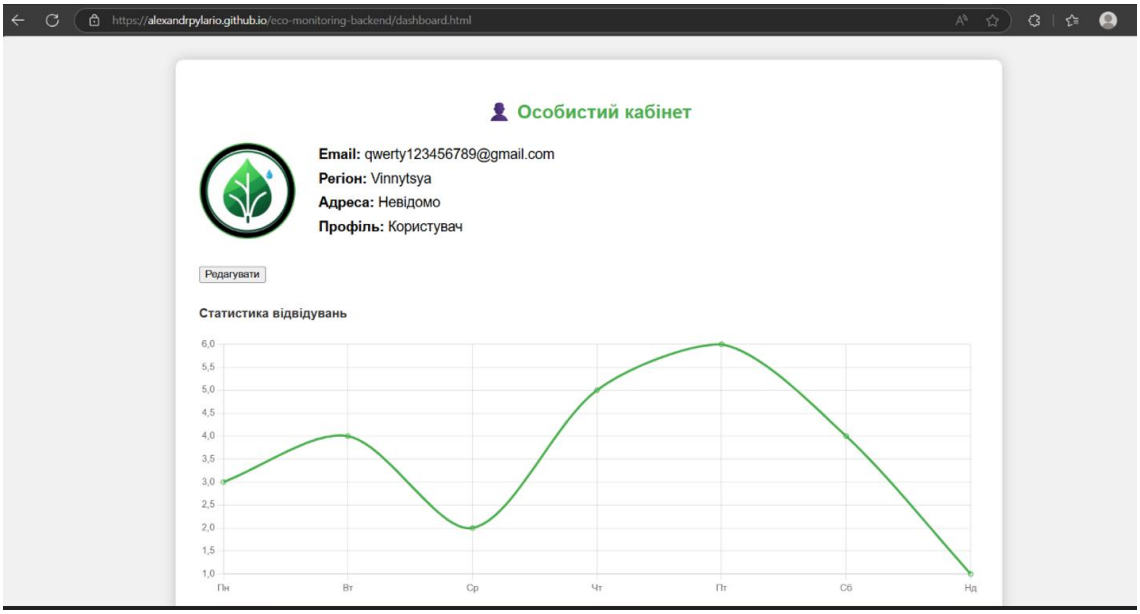


Рисунок 3.9 – Інтерфейс особистого кабінету користувача

Додаткові особливості:

– автокешування даних, завдяки service-worker.js;
– збереження аватара у localStorage навіть без доступу до серверної БД;
– валідація форм редагування: email перевіряється на валідність перед збереженням.

У майбутньому передбачено розширення модуля – додавання розділу «Мої сповіщення» з можливістю перегляду надісланих сповіщень, їх редагування/видалення.

3.5 Розробка інтерфейсу користувача вебсистеми

Під час розробки вебсистеми екологічного моніторингу важливо було створити інтерфейс, що поєднує функціональність, доступність і атмосферу турботи про довкілля. Основним кольором було обрано зелений відтінок #4CAF50, який асоціюється з природою, чистотою та екологічною безпекою. Світло-сірий фон #f4f4f4 використано для основного полотна сторінок, що забезпечує контрастність та полегшує зчитування інформації.

Допоміжні кольори — білий для блоків із даними та чорний для тексту — гарантують зручність читання та акцентують увагу на головному. Така палітра дозволила досягти візуального балансу та сучасного, естетично привабливого вигляду системи.

Після запуску вебсайту користувач потрапляє на головну сторінку з інтерактивною навігацією. Головне меню містить кнопки переходу до таких основних модулів: «Якість повітря», «Якість ґрунту» та «Особистий кабінет». Головну сторінку вебсистеми у браузері продемонстровано на рисунку 3.10.

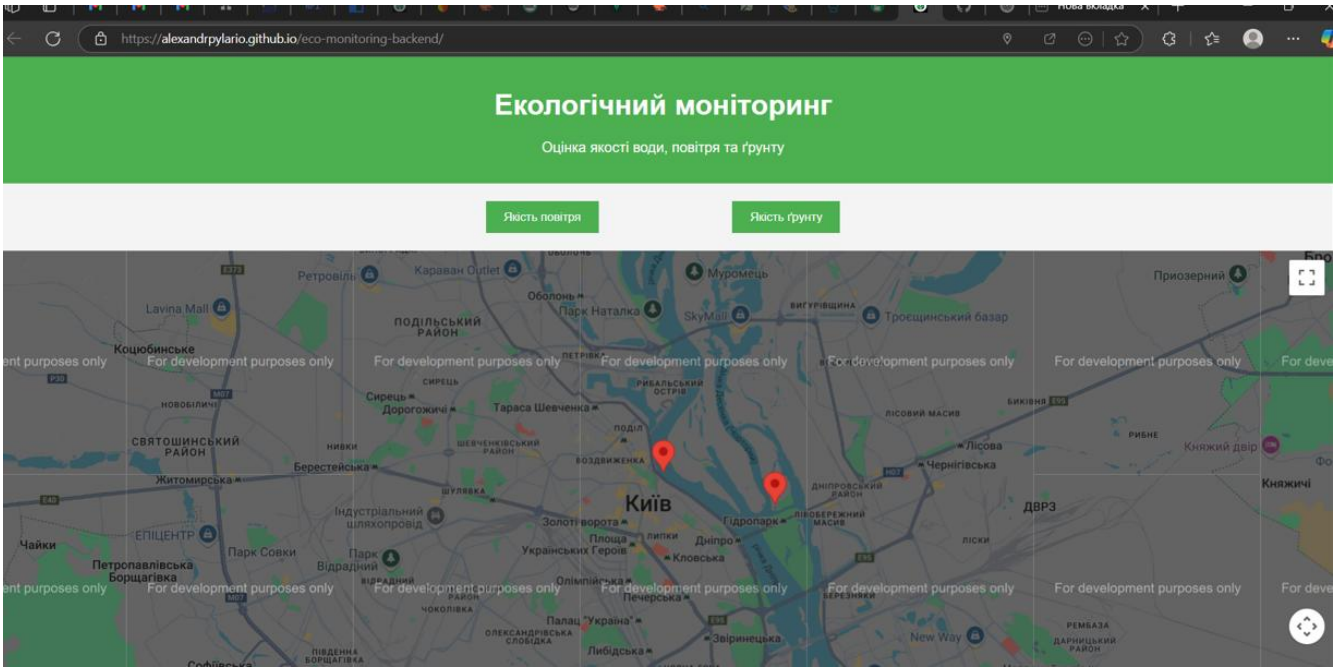


Рисунок 3.10 – Головна сторінка системи екологічного моніторингу в браузері

На сторінці кожного з модулів якості повітря та ґрунту користувач бачить інтерактивну карту Google Maps, яка відображає актуальні зони екологічного стану. Нижче виводиться динамічний графік на основі даних API, а праворуч — блок із корисними рекомендаціями. Приклад модуля якості повітря продемонстровано на рисунку 3.11.

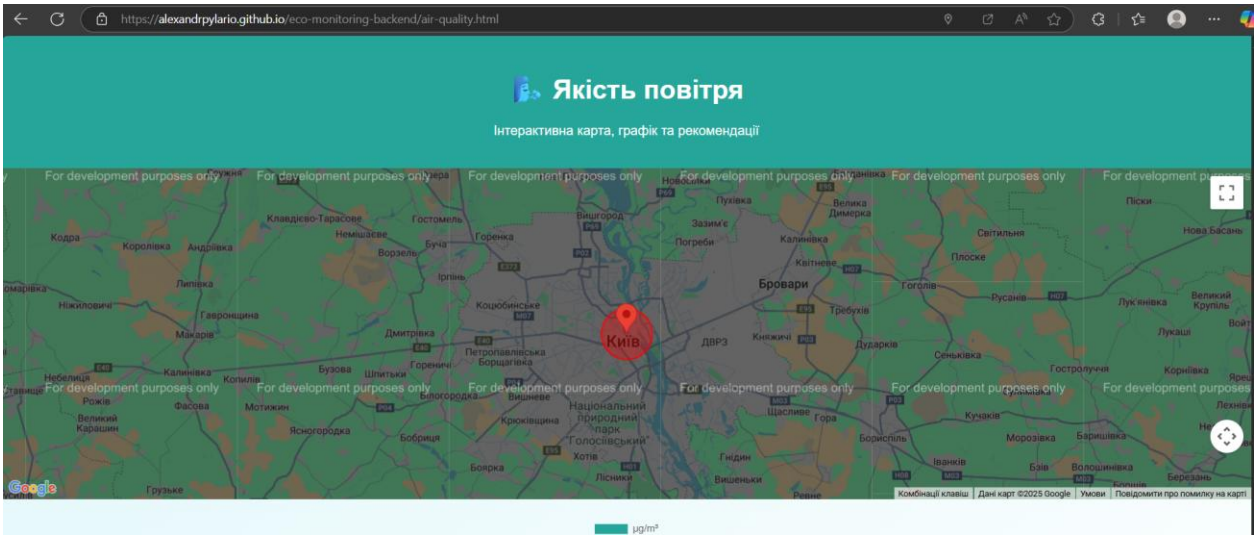


Рисунок 3.11– Інтерфейс модуля моніторингу якості повітря

Для доступу до персоналізованих функцій, таких як збереження аватару, регіону проживання, історії сповіщень і статистики, користувач повинен пройти реєстрацію або авторизацію. Форма авторизації виконана у мінімалістичному стилі з валідацією введених даних. Повідомлення про помилки підсвічуються червоним кольором. Форму входу проілюстровано на рисунку 3.12.

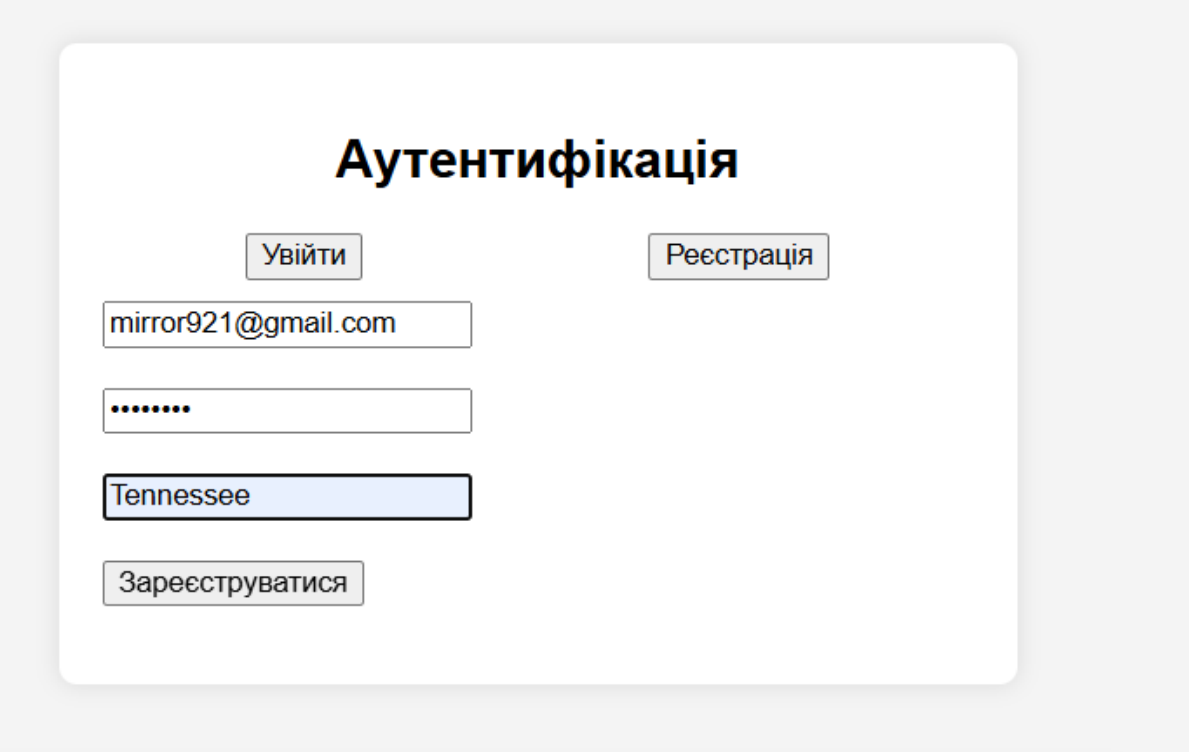


Рисунок 3.12 – Форма авторизації користувача

Після входу користувач отримує доступ до особистого кабінету, де зображено його ім’я, email, регіон проживання, аватар та екологічну статистику у вигляді графіка. Також є розділ зі списком отриманих сповіщень та можливістю редагування профілю. Інтерфейс особистого кабінет продемонстровано на рисунку 3.13.

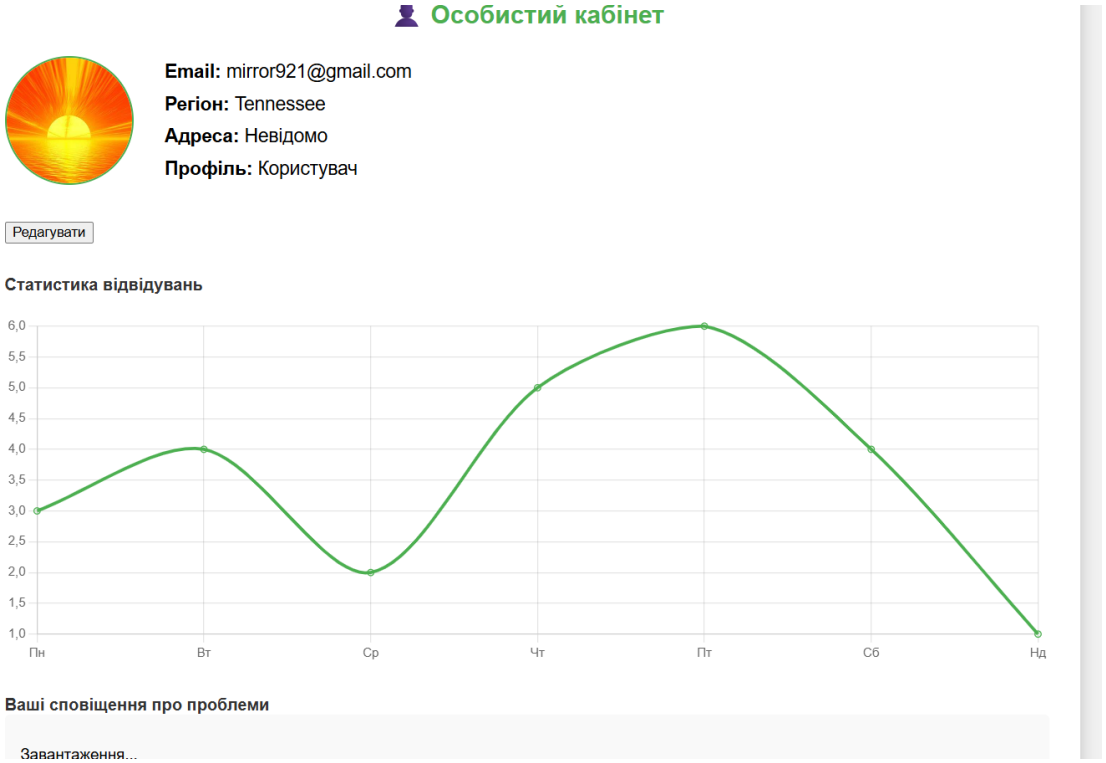


Рисунок 3.13 – Особистий кабінет користувача

Редагування профілю відбувається без перезавантаження сторінки. Користувач може змінити email, регіон або фото — вибраний аватар одразу відображається та зберігається у localStorage і MongoDB. На рисунку 3.14 зображено форму редагування профілю.

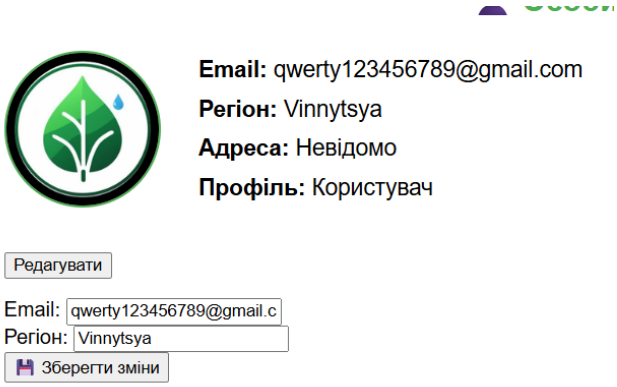


Рисунок 3.14 – Форма редагування профілю

Інтерфейс вебсистеми адаптований під мобільні пристрої завдяки медіа-запитам CSS, а також повністю відповідає принципам доступності — контрастність кольорів, зрозуміла структура, логічні HTML-розмітки.

3.6 Висновки

Отже, у третьому розділі було обґрунтовано вибір технологій, бібліотек і підходів, використаних під час розробки вебсистеми моніторингу й аналізу екологічної ситуації мікрорайону.

Для реалізації клієнтської частини використано HTML, CSS та JavaScript, зокрема бібліотеку Chart.js — для побудови динамічних графіків екологічних показників, а також Google Maps API — для візуалізації стану довкілля на інтерактивній карті. Окремі елементи карти адаптуються до обраного модуля.

Бекенд частину розроблено мовою JavaScript з використанням середовища Node.js та фреймворку Express. Для збереження профілів користувачів, екологічних сповіщень і історії даних застосовано MongoDB.

Пояснено реалізацію модуля особистого кабінету, що дозволяє користувачам переглядати та редагувати свої профілі, завантажувати аватар, слідкувати за отриманими екологічними сповіщеннями та переглядати статистику.

Описано розробку PWA-модуля: створено manifest.json, service-worker.js, реалізовано кешування і можливість встановлення сайту як мобільного застосунку.

Крім того, було розглянуто структуру та реалізацію інтерфейсу користувача — кольорова палітра, адаптивна верстка, тематичні анімації фону та елементи доступності.

Таким чином, у третьому розділі детально розглянуто технологічну базу вебзастосунку, розробку основних функціональних модулів і візуальну реалізацію системи.

4 ТЕСТУВАННЯ ПРОГРАМИ

4.1 Тестування функціоналу вебсистеми

Тестування програмного забезпечення [26] – це процес перевірки роботи ПЗ на відповідність вимогам, специфікаціям і очікуваним результатам. Це важливий етап розробки, який допомагає виявляти помилки та дефекти, покращувати якість продукту, підвищувати безпеку та впевненість у його роботі перед початком експлуатації.

Для того, аби процес тестування приніс результати, необхідно запустити серверну частину, що містить в собі логіку по обробці даних користувача із клієнта та інші модулі, а після – клієнтську. Перед початком тестування надзвичайно важливо надати документацію, вимоги до функціоналу та загальної роботи програмного продукту, що тестуватиметься.

Для перевірки коректності роботи основних функцій програми було обрано підхід, що має назву «Smoke Testing» [27]. Цей вид тестування можна визначити, як деякий короткий цикл тестів, який здійснюються після виходу чергового білду. Виконується це для перевірки роботи основного функціоналу розроблюваної програмної системи.

Під час тестування методом «Smoke Testing» буде перевірено програму на обробку невалідних вхідних даних від користувача, поведінку при відсутності даних та загальна поведінка відповідно до заявленого функціоналу.

Першим етапом тестування вебсистеми перевіряється обробка невалідних вхідних даних від користувача під час реєстрації та авторизації. Коли користувач вводить у поле для електронної пошти рядок неправильного формату, то програма повинна повідомити про це його у такий спосіб, щоб це було помітно. На рисунку4.1 продемонстровано повідомлення від програми про неправильний формат введених даних.

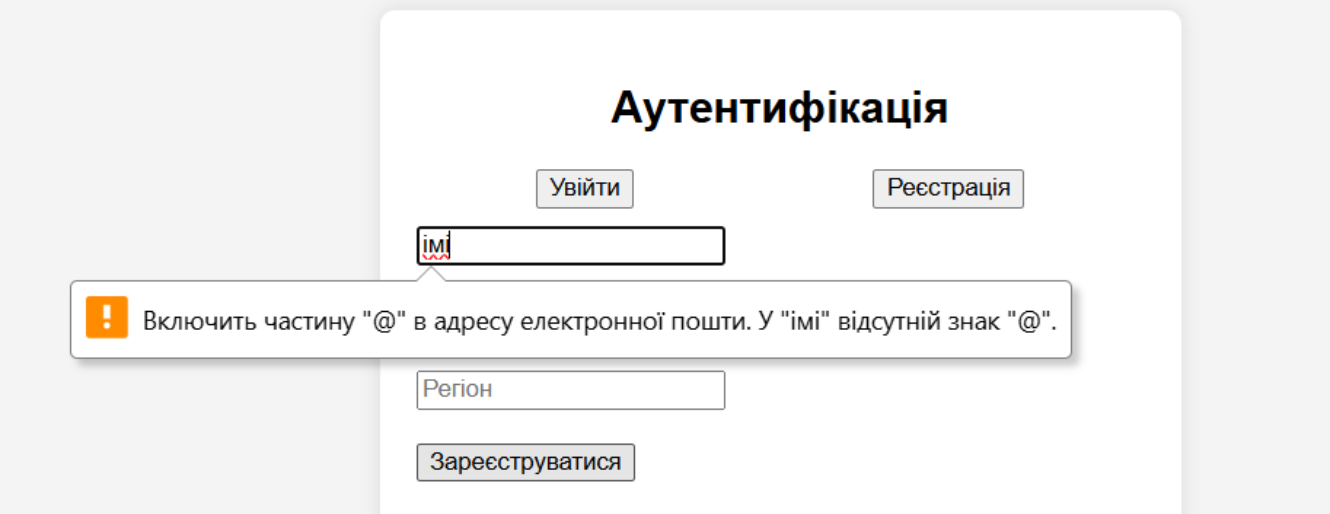


Рисунок 4.1 – Повідомлення про помилки

Тестування вебсистеми — це важливий етап, що дозволяє перевірити її функціональність, відповідність вимогам, стійкість до помилок і надійність у реальних умовах експлуатації. Для цього було обрано методику Smoke Testing — поверхневу перевірку основних функцій після кожного релізу, яка дозволяє впевнитися, що базова функціональність працює.

Перед тестуванням було запущено серверну частину, а також клієнтську, розгорнуту на GitHub Pages. Було підключено базу даних на Railway, перевірено змінні середовища JWT_SECRET та MONGO_URI, а також доступність API-запитів через backend.

Першим етапом було протестовано систему реєстрації та входу користувача. Якщо сервер не працює по причині технічної несправності або оновлення, то користувач отримує сповіщення про невдалу спробу взаємодії з сервером. Це продемонстровано на рисунку 4.2.

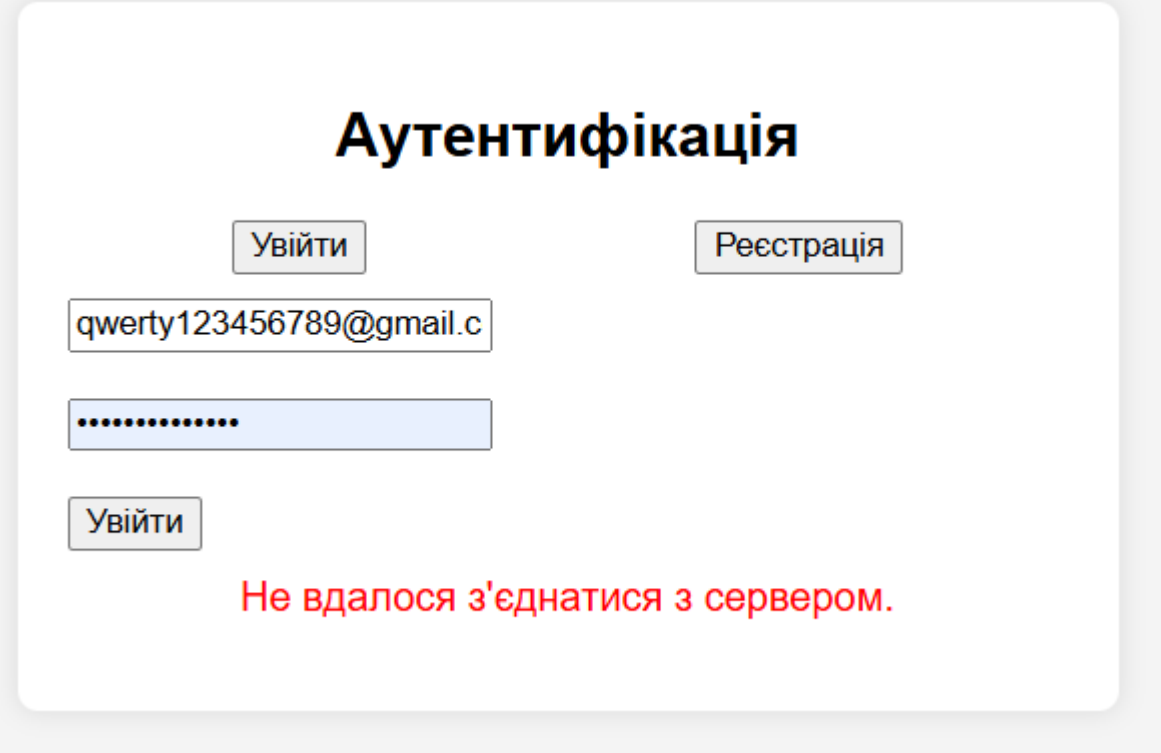


Рисунок 4.2 – Повідомлення про помилку роботи сервера

У разі правильно введених даних, сервер генерує JWT-токен, який зберігається в localStorage та дозволяє користувачу переходити до особистого кабінету. Після авторизації користувач має доступ до розширених функцій, таких як перегляд сповіщень, редагування профілю, графік активності.

Також було перевірено завантаження даних користувача через /api/profile та вивід їх на сторінці dashboard.html. У разі відсутності токену користувач перенаправляється на головну. Також перевірено редагування email, регіону та завантаження аватара з локального пристрою. Редагування профілю користувача зображено на рисунку 4.3.

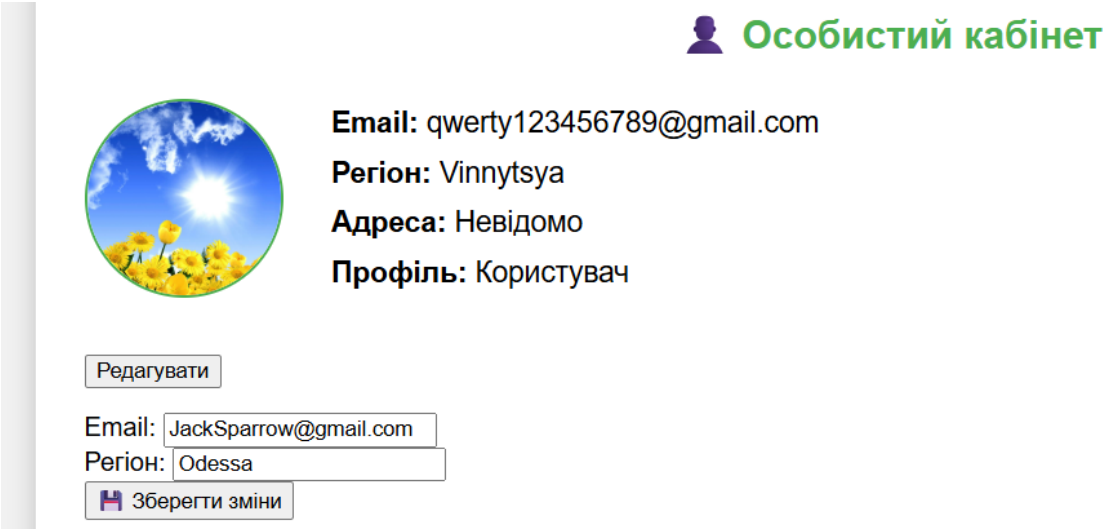


Рисунок 4.3 – Редагування профілю користувача

На рисунку 4.4 зображено результат редагування даних і картинки профілю.

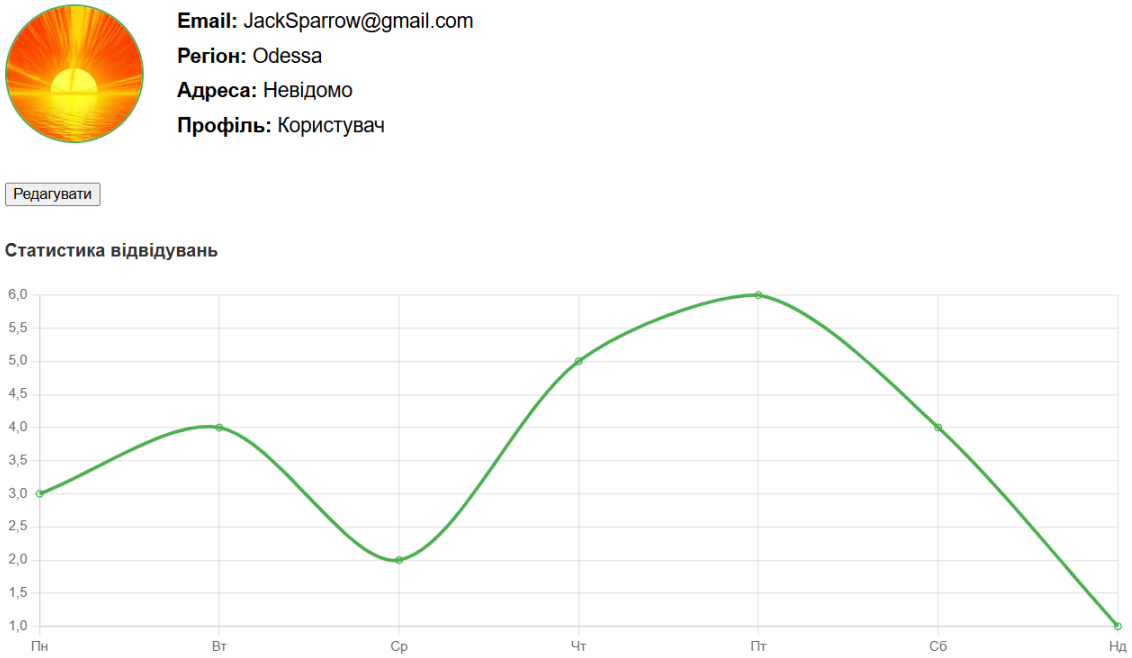


Рисунок 4.4 – Відредагований профіль користувача

Під час тестування кнопки «Зберегти» було перевірено оновлення даних на сервері через PUT /api/profile/update та їх миттєве відображення на сторінці.

Також було перевірено роботу сторінок:

- air-quality.html;

– soil-quality.html.

На кожній сторінці автоматично отримується геолокація користувача. У разі дозволу — на карту наноситься поточна позиція, виконується запит до відповідного API та виводиться графік з даними, що зображено на рисунку 4.5.

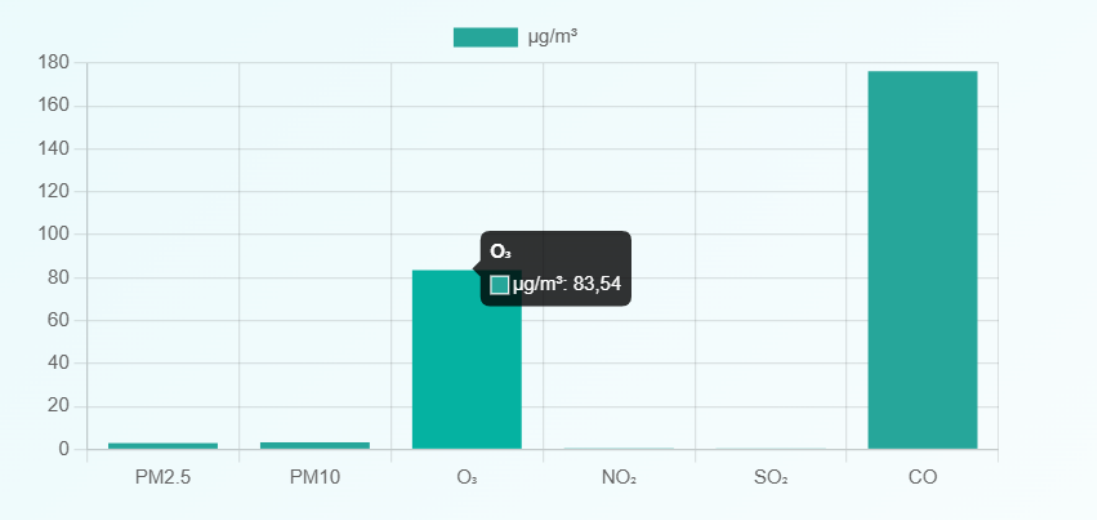


Рисунок 4.5 – Графік якості повітря

Також перевірено реакцію системи у разі помилки геолокації при відсутності підключення, що продемонстровано на рисунку 4.6.

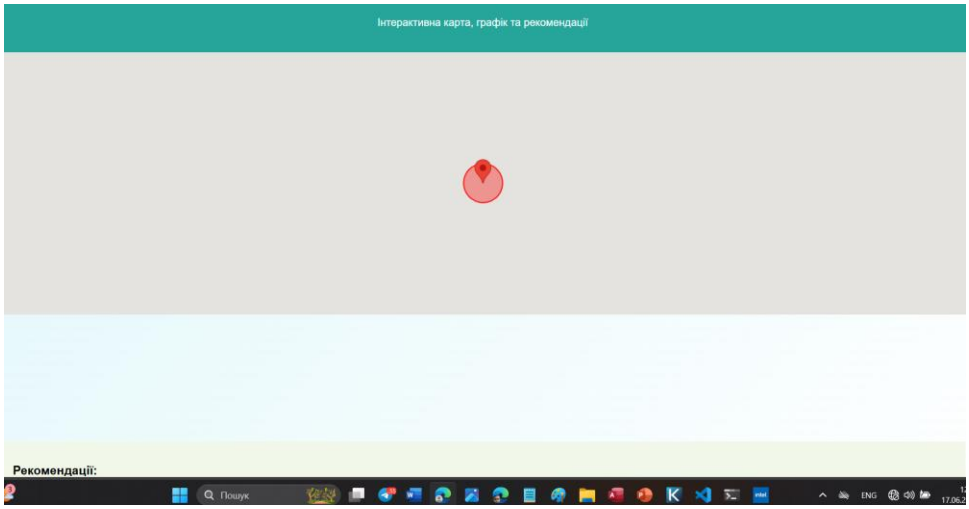


Рисунок 4.5 – Повідомлення про відмову в доступі до геолокації через відсутність підключення до Інтернету

Було перевірено інтерактивність карти на кожному модулі. Система правильно відображає позицію користувача. При зміні координат або оновленні —

карта автоматично адаптується до нових умов, що продемонстровано на рисунку 4.6.

Рисунок 4.6 – Google-карта з геолокацією

Усі API-запити було протестовано через backend з обробкою помилок. Система коректно реагує у разі:

– відсутності з’єднання з API;
– невірних координат;
– нульових даних, а саме – list == null.

Також було перевірено:

– встановлення сайту як додатку;
– роботу кнопки «Встановити» із вікном підтвердження на рисунку 4.7;
– кешування основних сторінок;
– появу іконки встановлення на головному екрані на рисунку 4.7.

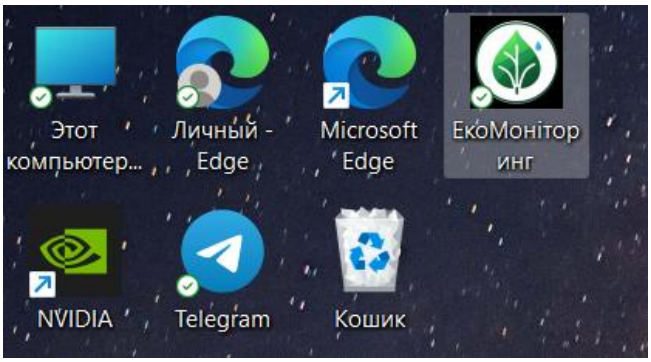


Рисунок 4.7 – Іконка встановлення PWA на робочий стіл користувача

4.2 Розробка інструкції користувача

Інструкція користувача визначає технічні вимоги до середовища запуску, а також умови, за яких вебсистема функціонуватиме належним чином. Програма не потребує складного встановлення та доступна безпосередньо через браузер, однак для стабільної роботи рекомендується використовувати сучасні операційні системи, сучасні браузери типу Google Chrome, Microsoft Edge або Mozilla Firefox останніх версій, а також швидкісне інтернет-з’єднання не нижче 100 Мбіт/сек.

Система є прогресивним вебдодатком, тому її можна встановити як мобільний застосунок або настільну програму. Після встановлення користувач може отримувати працювати з даними в екстреному режимі та мати швидший доступ до основних функцій.

Найбільший вплив на стабільну роботу системи має швидкість та стабільність інтернет-з’єднання, оскільки всі екологічні дані завантажуються з API-серверів у реальному часі. У разі відсутності з’єднання застосовується режим, який відображає раніше кешовані дані або повідомляє про неможливість отримати нову інформацію.

Система підтримує роботу як для гостьових користувачів, так і для зареєстрованих. Після запуску програми користувач потрапляє на головну сторінку, де доступні кнопки переходу до модулів моніторингу якості повітря та ґрунту. Кожен модуль містить інтерактивну карту, динамічний графік забруднення та текстові рекомендації.

Функціонал, який доступний лише для авторизованих користувачів:

- персональний кабінет користувача, де можна переглядати і редагувати електронну пошту, регіон проживання, аватар, а також переглядати власні сповіщення;

- надсилання сповіщень про екологічні проблеми у своєму районі — лише авторизовані користувачі можуть залишати повідомлення, які потрапляють до бази даних;
- історія відвідувань — система фіксує візити авторизованих користувачів для формування аналітичних графіків у кабінеті;
- повноцінні графіки — якщо користувач авторизований, графіки будуються на основі його активності та даних з MongoDB; для гостей графіки можуть бути спрощені або статичні;

Якщо користувач заходить без авторизації, він має доступ до інтерактивних карт та загального екологічного стану по регіонах України, але не може зберігати персональні дані чи надсилати сповіщення.

Для входу в систему необхідно зареєструватися за допомогою форми реєстрації, після чого відкривається доступ до особистого кабінету. Після авторизації користувач має змогу змінювати свій профіль, переглядати сповіщення, отримувати рекомендації відповідно до обраного регіону, а також бачити історію своєї активності у вигляді графіків.

4.3 Висновки

Отже, у четвертому розділі було проведено тестування роботи системи моніторингу екологічного стану мікрорайону. Для перевірки працездатності ключових функцій було використано метод «Smoke Testing». Протестовано модулі авторизації та реєстрації, обробку невалідних даних, а також поведінку системи при відсутності підключення до інтернету.

Підтверджено правильність завантаження особистого кабінету користувача, коректне відображення та редагування електронної пошти, регіону і аватара. Перевірено модуль сповіщень, який зберігає повідомлення про екологічні проблеми в базі даних.

Також протестовано роботу модулів якості повітря та ґрунту: перевірено інтерактивність карт, відображення API-даних, побудову графіків і виведення рекомендацій. Здійснено перевірку роботи системи в режимі роботи з Робочого Столу з використанням кешованих даних.

Розроблено інструкцію користувача з описом вимог до середовища, режимів доступу та функціоналу, що відкривається після авторизації.

ВИСНОВКИ

У бакалаврській кваліфікаційній роботі було розроблено мобільну та вебсистему для моніторингу й аналізу екологічної ситуації в мікрорайоні. Програмну реалізацію здійснено з використанням середовища розробки Visual Studio Code.

Роботу виконано відповідно до методичних рекомендацій, затверджених кафедрою [28].

У процесі дослідження було проведено аналіз проблеми екологічного моніторингу на рівні мікрорайонів, розглянуто приклади існуючих систем, визначено їхні переваги й недоліки. На основі цього сформульовано мету та завдання бакалаврської роботи.

У межах технічного обґрунтування було обрано такі засоби реалізації:

- HTML, CSS і JavaScript для клієнтської частини;
- Node.js для серверної логіки;
- MongoDB як база даних;
- Chart.js – для побудови аналітичних графіків;
- Google Maps API та OpenWeather API – для візуалізації екологічних даних;
- PWA-технології – для встановлення системи як додатку.

У результаті виконання роботи було досягнуто таких результатів:

– проведено аналіз стану та потреб екологічного моніторингу на локальному рівні;

– реалізовано інтерфейс із розділами «Якість повітря», та «Якість ґрунту» з використанням інтерактивної карти, графіка та рекомендацій;

– розроблено особистий кабінет користувача з можливістю редагування даних та перегляду власних сповіщень;

– імплементовано механізм реєстрації/авторизації з JWT-аутентифікацією;

– реалізовано збереження сповіщень про екологічні проблеми в MongoDB;

– протестовано працездатність функціоналу з використанням підходу «Smoke Testing»;

– розроблено інструкцію користувача з описом функціональних режимів і вимог до системи.

Тестування системи підтвердило правильну роботу реалізованих модулів. Система є повноцінним прикладом прогресивного вебзастосунку для моніторингу стану довкілля з можливістю використання як на мобільних, так і десктопних пристроях.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Що таке API? [Електронний ресурс] – Режим доступу до ресурсу: API для початківців: що це таке і чому це важливо – Agiliway (дата звернення: 03.05.2025).

2. PWA-функціональність [Електронний ресурс] – Режим доступу до ресурсу: Progressive web app: що це і приклади його використання | Блог LTESocks (дата звернення: 25.03.2025).

3. Github [Електронний ресурс] – Режим доступу до ресурсу: GitHub — Вікіпедія (дата звернення: 03.05.2025).

4. Railway [Електронний ресурс] – Режим доступу до ресурсу: Railway (дата звернення: 03.05.2025).

5. MongoDB [Електронний ресурс]. – Режим доступу до ресурсу: MongoDB: The World's Leading Modern Database | MongoDB (дата звернення: 03.05.2025).

6. Войтко В.В. Розробка методу і засобів реалізації мобільної системи моніторингу й аналізу екологічної ситуації мікрорайону / В.В. Войтко, С.М.Бурбело, О.І. Пислар // Матеріали Міжнародної науково-практичної інтернет-конференції "Молодь в науці: дослідження, проблеми, перспективи (МН-2025)". [Електронний ресурс] – Режим доступу до ресурсу: https://conferences.vntu.edu.ua/index.php/mn/mn2025/paper/viewFile/24116/19978 (дата звернення: 03.05.2025).

7. Екологічне мислення [Електронний ресурс] – Режим доступу до ресурсу: https://uahistory.co/pidruchniki/zadorozhnij-biology-and-ecology-11-class-2019-standard-level/46.php (дата звернення: 04.05.2025).

8. Екокультура [Електронний ресурс] – Режим доступу до ресурсу: Екологічна культура — Вікіпедія (дата звернення: 04.05.2025).

9. Цифровізація [Електронний ресурс] – Режим доступу до ресурсу: ЦИФРОВІЗАЦІЯ (Діджиталізація) — це що таке, визначення, (дата звернення: 07.05.2025).

10. AirVisual [Електронний ресурс] – Режим доступу до ресурсу: Live Animated Air Quality Map (AQI, PM2.5...) | IQAir (дата звернення: 07.05.2025).

11. PlumeLabs [Електронний ресурс] – Режим доступу до ресурсу: Plume Labs: Be empowered against air pollution (дата звернення: 07.05.2025).

12. ЕкоЗагроза [Електронний ресурс] – Режим доступу до ресурсу: ЕкоЗагроза (дата звернення: 07.05.2025).

13. MPA – Multi Page Application [Електронний ресурс] – Режим доступу до ресурсу: Багатосторінкові та односторінкові додатки, їх переваги та недоліки | DOU (дата звернення: 29.03.2025).

14. Node.js [Електронний ресурс] – Режим доступу до ресурсу: Що таке Node.js та для чого він потрібен? | Блог HyperHost.UA (дата звернення: 10.05.2025).

15. Chart.js [Електронний ресурс] – Режим доступу до ресурсу: JavaScript Chart.js. Уроки для початківців. W3Schools українською (дата звернення: 30.03.2025).

16. Google Maps API [Електронний ресурс] – Режим доступу до ресурсу: Що таке Google Maps API (Google Maps Platform) ☑ Новини компанії «Wise IT» (дата звернення: 10.05.2025).

17. Service Woker [Електронний ресурс] – Режим доступу до ресурсу: Що таке service worker в javascript? - javascript.org.ua - JS Communities (дата звернення: 10.05.2025).

18. Що таке клас? [Електронний ресурс] – Режим доступу до ресурсу: » Поняття класу C++ програмування (дата звернення: 15.05.2025).

19. Сайт [Електронний ресурс] – Режим доступу до ресурсу: Вебсайт — Вікіпедія (дата звернення: 15.05.2025).

20. JWT-токен [Електронний ресурс] – Режим доступу до ресурсу: JSON Web Token — Вікіпедія (дата звернення: 15.05.2025).

21. JSON [Електронний ресурс] – Режим доступу до ресурсу: JSON — Вікіпедія (дата звернення: 16.05.2025).

22. Express.js [Електронний ресурс] – Режим доступу до ресурсу: Express js: встановлення, налаштування та особливості (дата звернення: 16.05.2025).

23. Visual Studio Code [Електронний ресурс] – Режим доступу до ресурсу: Visual Studio Code - Code Editing. Redefined (дата звернення: 18.05.2025).

24. Геолокаційний запит [Електронний ресурс] – Режим доступу до ресурсу: Що таке Геозалежний запит - Глосарій веб студії VOLL (дата звернення: 18.05.2025).

25. SPA [Електронний ресурс] – Режим доступу до ресурсу: Що таке spa в веб-розробці (дата звернення: 20.05.2025).

26. CacheStorage [Електронний ресурс] – Режим доступу до ресурсу: CacheStorage - Интерфейсы веб API | MDN (дата звернення: 24.05.2025).

27. HTML [Електронний ресурс] – Режим доступу до ресурсу: HTML — Вікіпедія (дата звернення: 24.05.2025).

28. Методичні вказівки до виконання бакалаврських кваліфікаційних робіт для студентів студентів спеціальності 121 "Інженерія програмного забезпечення" / Уклад. О. Н. Романюк, Г. О. Черноволик – Вінниця: ВНТУ, 2022. – 52 с (дата звернення: 27.05.2025).

Додатки

65

Додаток А. Технічне завдання

Міністерство освіти і науки України

Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

УЗГОДЖЕНО
НАЧАЛЬНИК ДЕРЖАВНОЇ
ЕКОЛОГІЧНОЇ ІНСПЕКЦІЇ У
ВІННИЦЬКІЙ ОБЛАСТІ
Дубовий Ю.В.
«25» березня 2025 р.

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
Романюк О. Н.
«25» березня 2025 р.

Технічне завдання

на бакалаврську кваліфікаційну роботу

«Розробка методу і засобів реалізації програмної системи моніторингу й аналізу екологічної ситуації мікрорайону»

за спеціальністю 121 – Інженерія програмного забезпечення

Керівник бакалаврської кваліфікаційної роботи:
к.т.н., доц. Войтко В. В.
«24» березня 2025 р.

Виконав:
студент гр. 4ПІ-21б Пислар О.І.
«24» березня 2025 р.

Вінниця – 2025 року

1. Найменування та галузь застосування

Бакалаврська кваліфікаційна робота: «Розробка методу і засобів реалізації програмної системи моніторингу й аналізу екологічної ситуації мікрорайону».

Галузь застосування – екологічний моніторинг навколишнього середовища.

2. Підстава для розробки.

Підставою для виконання бакалаврської кваліфікаційної роботи (БКР) є індивідуальне завдання на БКР та наказ №97 від 20 березня 2025 року про закріплення тем БКР.

3. Мета та призначення розробки.

Метою дослідження є покращення можливостей моніторингу й аналізу екологічної ситуації навколишнього середовища за рахунок розробки і використання спеціалізованої програмної системи, яка дозволить в інтерактивному режимі відображати стан повітря та ґрунту, надавати рекомендації користувачам, зберігати дані в базі, забезпечувати підтримку PWA-функціоналу.

Призначення роботи — розробка інтерактивної мобільної та вебсистеми для збору, аналізу та візуалізації екологічних даних із використанням API, геолокації, карт, графіків і персонального кабінету користувача.

4. Вихідні дані для проведення НДР

1. AirVisual [Електронний ресурс] – Режим доступу до ресурсу: Live Animated Air Quality Map (AQI, PM2.5...) | IQAir

2. Visual Studio Code [Електронний ресурс] – Режим доступу до ресурсу: Visual Studio Code - Code Editing. Redefined

3. PWA-функціональність [Електронний ресурс] – Режим доступу до ресурсу: Progressive web app: що це і приклади його використання | Блог LTESocks

5. Технічні вимоги

Комп’ютер з 64-розрядним (x64) або ж 32-розрядним процесором та тактовою частотою 2 ГГц. Достатньо всього 4 ГБ оперативної пам’яті, а місця на жорсткому диску – 10 ГБ. Операційна система Windows 10 або нижче (включно до Windows 8). З клієнтського боку: комп’ютер з підключенням до мережі «Інтернет» та сучасний браузер.

6. Конструктивні вимоги.

Система повинна повністю відповідати загальноприйнятим вимогам щодо розробки front-end за стосунків.

Інтерфейс користувача повинен бути інтуїтивно зрозумілим, так як система може бути використана користувачами з будь-яким рівнем володіння комп’ютером.

7. Перелік технічної документації, що пред’являється по закінченню робіт:

- пояснювальна записка до бакалаврської кваліфікаційної роботи;
- технічне завдання;
- лістинги програми.

8. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

9. Стадії та етапи розробки

№ з/п	Назва етапів бакалаврської кваліфікаційної роботи	Строк виконання етапів роботи
1	Аналіз розвитку систем для аналізу й моніторингу екологічної ситуації	25.03.25 – 29.03.25
2	Розробка методів та алгоритмів роботи системи	30.03.25 – 10.04.25
3	Розробка програмного забезпечення системи	11.04.25 – 10.05.25
4	Тестування програми	11.05.25 – 14.05.25
5	Оформлення матеріалів до захисту БКР	15.05.25 – 30.05.25

10. Порядок контролю та прийняття

Всі етапи бакалаврської кваліфікаційної роботи контролюються науковим керівником згідно плану по виконанню роботи. Прийняття бакалаврської кваліфікаційної роботи здійснюється ДЕК, затвердженою зав. кафедрою згідно з графіком захисту.

69

Додаток Б. Протокол перевірки бакалаврської кваліфікаційної роботи на наявність текстових запозичень

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: «Розробка методу і засобів реалізації програмної системи моніторингу й аналізу екологічної ситуації мікрорайону»

Тип роботи: бакалаврська кваліфікаційна робота
(бакалаврська кваліфікаційна робота / магістерська кваліфікаційна робота)

Підрозділ кафедра програмного забезпечення, ФІТКІ, 4ПІ-21б
(кафедра, факультет, навчальна група)

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism 8,5 %

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

☐ Запозичення, виявлені у роботі, є законними і не містять ознак плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту

☐ У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.

☐ У роботі виявлено ознаки плагіату та/або текстових маніпуляцій як спроб укриття плагіату, фабрикації, фальсифікації, що суперечить вимогам законодавства та нормам академічної доброчесності. Робота до захисту не приймається.

Експертна комісія:

_____ _____ _____
(прізвище, ініціали, посада) (підпис)

_____ _____ _____
(прізвище, ініціали, посада) (підпис)

Особа, відповідальна за перевірку _____ Черноволик Г. О.

З висновком експертної комісії ознайомлений(-на)

Керівник _____ Войтко В.В., ктн., доц. каф. ПЗ
(підпис) (прізвище, ініціали, посада)

Здобувач _____ Пислар О.І.
(підпис) (прізвище, ініціали)

70

Додаток В. Акт впровадження (Державна екологічна інспекція у Вінницькій області)

АКТ

про впровадження результатів бакалаврської кваліфікаційної роботи «Розробка методу і засобів реалізації програмної системи моніторингу й аналізу екологічної ситуації мікрорайону»

студента ВНТУ Пислара Олександра Івановича

Результати бакалаврської кваліфікаційної роботи студента Вінницького національного технічного університету Пислара О.І, що полягають у розробці методу і засобів реалізації програмної системи моніторингу й аналізу екологічної ситуації мікрорайону, прийняті до використання у Державній екологічній інспекції у Вінницькій області.

Начальник Державної екологічної інспекції у Вінницькій області — Дубовий Ю.В.

Додаток Г. Лістинг коду

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


  <link rel="preload" as="image" href="icons/icon-192.png">
  <link rel="preload" as="image" href="https://media.giphy.com/media/d2lcHJTG5Tscg/giphy.gif">

  <link rel="manifest" href="manifest.json">
  <link rel="icon" type="image/png" sizes="192x192" href="icons/icon-192.png">
  <meta name="theme-color" content="#4CAF50">

  <title>Екологічний моніторинг</title>
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.15.3/css/all.min.css">
  <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
  <style>
    body {
      font-family: Arial, sans-serif;
      margin: 0;
      padding: 0;
      background-color: #f4f4f4;
    }


    header {
      background-color: #4CAF50;
      color: white;
      text-align: center;
      padding: 15px;
    }

    .nav-buttons {
      display: flex;
      justify-content: center;
```

```
  margin: 20px 0;
}

.nav-buttons button {
  padding: 10px 20px;
  margin: 0 10px;
  background-color: #4CAF50;
  color: white;
  border: none;
  cursor: pointer;
}

.nav-buttons button:hover {
  background-color: #45a049;
}

footer {
  background-color: #333;
  color: white;
  text-align: center;
  padding: 10px;
}

.map-container {
  height: 500px;
  width: 100%;
  margin-top: 20px;
}

.weather-icons {
  display: flex;
  justify-content: space-between;
  margin-top: 20px;
  padding: 10px;
}

.weather-icons .icon-box {
  background-color: white;
  padding: 15px;
```

```
      border-radius: 5px;
      width: 30%;
      box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
      text-align: center;
    }

   #swipe-container {
  position: fixed;
  bottom: 0;
  width: 100%;
  height: 60px;
  background-color: #4CAF50;
  z-index: 9999;
}

#swipe-button {
  width: 60px;
  height: 60px;
  background-color: #fff;
  text-align: center;
  line-height: 60px;
  border-radius: 30px;
  position: absolute;
  left: 0;
  transition: left 0.3s;
  user-select: none;
}


    .weather-icons .icon-box img {
      width: 80%;
      border-radius: 5px;
    }
  </style>
</head>
<body>

<header>
  <h1>Екологічний моніторинг</h1>
```

```
  <p>Оцінка якості води, повітря та ґрунту</p>
</header>

<!-- Навігаційні кнопки -->
<div class="nav-buttons">
  <button onclick="window.location.href='air-quality.html'">Якість повітря</button>
  <button onclick="window.location.href='water-quality.html'">Якість води</button>
  <button onclick="window.location.href='soil-quality.html'">Якість ґрунту</button>
</div>

<!-- Карта -->
<div id="map" class="map-container"></div>

<!-- Анімовані погодні іконки -->
<div class="weather-icons">
  <div class="icon-box">
    <h3>Погода в вашому регіоні</h3>
    <img id="weather-icon" src="https://media.giphy.com/media/l0MYlTV2x6ZjbbkWg/giphy.gif" alt="weather icon">
    <p>Актуальний стан повітря</p>
  </div>

  <div class="icon-box">
    <h3>Вода</h3>
    <img id="water-icon" src="https://media.giphy.com/media/d2lcHJTG5Tscg/giphy.gif" alt="water icon">
    <p>Якість води в регіоні</p>
  </div>

  <div class="icon-box">
    <h3>Ґрунт</h3>
    <img id="soil-icon" src="https://media.giphy.com/media/4yOmeL2hdzv6g/giphy.gif" alt="soil icon">
    <p>Стан ґрунту</p>
  </div>
</div>

<!-- Форма входу та реєстрації -->
<section style="max-width: 400px; margin: 40px auto; background: #fff; padding: 20px; border-radius: 8px; box-shadow: 0
0 10px rgba(0,0,0,0.1);">
  <h2 style="text-align:center;">Аутентифікація</h2>
  <div style="display: flex; justify-content: space-around; margin-bottom: 10px;">
```

```
  <button onclick="toggleForm('login')">Увійти</button>
  <button onclick="toggleForm('register')">Реєстрація</button>
 </div>

 <form id="loginForm" style="display: none;" onsubmit="handleLogin(event)">
  <input type="email" placeholder="Email" id="loginEmail" required><br><br>
  <input type="password" placeholder="Пароль" id="loginPassword" required><br><br>
  <button type="submit">Увійти</button>
 </form>

 <form id="registerForm" style="display: none;" onsubmit="handleRegister(event)">
  <input type="email" placeholder="Email" id="registerEmail" required><br><br>
  <input type="password" placeholder="Пароль" id="registerPassword" required><br><br>
  <input type="text" placeholder="Регіон" id="registerRegion" required><br><br>
  <button type="submit">Зареєструватися</button>
 </form>

 <p id="authMessage" style="color: green; text-align: center; margin-top: 10px;"></p>
</section>
<!-- Форма сповіщення про екологічну проблему -->
<section style="max-width: 500px; margin: 40px auto; background: #fff; padding: 20px; border-radius: 8px; box-shadow: 0
0 10px rgba(0,0,0,0.1);">
 <h2 style="text-align:center;">Надіслати екосповіщення</h2>
 <form id="reportForm" onsubmit="submitReport(event)">
  <label>Тип проблеми:</label><br>
  <input type="text" id="reportType" required><br><br>

  <label>Опис:</label><br>
  <textarea id="reportDescription" rows="4" required></textarea><br><br>

  <button type="submit">Надіслати</button>
 </form>
 <p id="reportMessage" style="text-align:center; color: green; margin-top: 10px;"></p>
</section>

<footer>
 <p>&copy; 2025 Екологічний моніторинг. Всі права захищено.</p>
</footer>
```

```
<!-- ✅ Кнопка встановлення додатку -->
<button id="installBtn" style="display: none; padding: 10px 20px; background: #4CAF50; color: white; border: none;
border-radius: 5px; cursor: pointer;">
  📲 Встановити як додаток
</button>

<script>
  let deferredPrompt;
  const installBtn = document.getElementById("installBtn");

  window.addEventListener("beforeinstallprompt", (e) => {
    e.preventDefault();
    deferredPrompt = e;
    installBtn.style.display = "block";
  });

  // Натискання кнопки завжди показує prompt
  installBtn.addEventListener("click", async () => {
    if (deferredPrompt) {
      deferredPrompt.prompt();
      const { outcome } = await deferredPrompt.userChoice;
      console.log("🧩 Результат встановлення:", outcome);
    } else {
      alert("🚫 Встановлення недоступне. Спробуйте знову пізніше.");
    }
  });

  // Не ховаємо кнопку після встановлення
  window.addEventListener("appinstalled", () => {
    console.log("✅ PWA встановлено");
    installBtn.textContent = "📲 Встановити ще раз";
    installBtn.disabled = false;
    installBtn.style.display = "block";
  });
</script>
```

```
<script
src="https://maps.googleapis.com/maps/api/js?key=AIzaSyAOib_11QyNZYlK2wIjsBEM2fWJf6CYiF0&callback=initMap
" async defer></script>
<script>
  // Карта
  let map;
  function initMap() {
    map = new google.maps.Map(document.getElementById("map"), {
      center: { lat: 50.4501, lng: 30.5736 },
      zoom: 12,
    });

    new google.maps.Marker({
      position: { lat: 50.4501, lng: 30.5736 },
      map: map,
      title: 'Київ',
    });

    map.setOptions({ draggable: true, scrollwheel: true });

    if (navigator.geolocation) {
      navigator.geolocation.getCurrentPosition(function(position) {
        const userLocation = {
          lat: position.coords.latitude,
          lng: position.coords.longitude
        };
        map.setCenter(userLocation);
        new google.maps.Marker({
          position: userLocation,
          map: map,
          title: 'Ваше місцезнаходження',
        });
      });
    } else {
      alert("Геолокація не підтримується вашим браузером.");
    }
  }
```

```
  // Погода
  async function fetchWeatherData() {
    const API_KEY = "c0a820cfc0cc46cc51e67536bf08f5e3";
    const weatherUrl =
`https://api.openweathermap.org/data/2.5/weather?lat=50.4501&lon=30.5736&appid=${API_KEY}&units=metric&lang=u
k`;
    const response = await fetch(weatherUrl);
    const data = await response.json();
    const weatherIcon = document.getElementById("weather-icon");
    if (data.weather && data.weather[0].main) {
      if (data.weather[0].main === "Clear") {
        weatherIcon.src = "https://media.giphy.com/media/3oKIPM7bgJBoK5vJUk/giphy.gif";
      } else if (data.weather[0].main === "Rain") {
        weatherIcon.src = "https://media.giphy.com/media/l0MYlTV2x6ZjbbkWg/giphy.gif";
      } else {
        weatherIcon.src = "https://media.giphy.com/media/d2lcHJTG5Tscg/giphy.gif";
      }
    }
  }
    fetchWeatherData();
</script>

<script>

  function toggleForm(type) {
    document.getElementById("loginForm").style.display = type === 'login' ? 'block' : 'none';
    document.getElementById("registerForm").style.display = type === 'register' ? 'block' : 'none';
    document.getElementById("authMessage").innerText = "";
  }

  async function handleRegister(e) {
    e.preventDefault();

    const email = document.getElementById("registerEmail").value;
    const password = document.getElementById("registerPassword").value;
    const region = document.getElementById("registerRegion").value;
```

```
  console.log("📩 Надсилаємо реєстраційні дані:", { email, password, region });

  try {
    const res = await fetch("https://eco-monitoring-backend-production.up.railway.app/api/register", {

      method: "POST",
      headers: {
        "Content-Type": "application/json"
      },
      body: JSON.stringify({ email, password, region })
    });

    const data = await res.json();
    console.log("📥 Відповідь від сервера:", data);

    if (res.ok) {
      document.getElementById("authMessage").innerText = data.message || "Успішна реєстрація!";
      document.getElementById("authMessage").style.color = "green";
    } else {
      document.getElementById("authMessage").innerText = data.message || "Щось пішло не так.";
      document.getElementById("authMessage").style.color = "red";
    }
  } catch (err) {
    console.error("❌ Помилка під час запиту:", err);
    document.getElementById("authMessage").innerText = "Не вдалося з'єднатися з сервером.";
    document.getElementById("authMessage").style.color = "red";
  }
}

async function handleLogin(e) {
e.preventDefault();

const email = document.getElementById("loginEmail").value;
const password = document.getElementById("loginPassword").value;

console.log("📩 Надсилаємо дані входу:", { email, password });

try {
```

```
    const res = await fetch("https://eco-monitoring-backend-production.up.railway.app/api/login", {
      method: "POST",
      headers: {
        "Content-Type": "application/json"
      },
      body: JSON.stringify({ email, password })
    });

    const data = await res.json();
    console.log("🔐 Відповідь на вхід:", data);

    if (res.ok && data.token) {
      localStorage.setItem("token", data.token);        // ✅ зберігаємо токен
      window.location.href = "dashboard.html";          // ✅ переходимо в кабінет
    } else {
      document.getElementById("authMessage").innerText = data.message || "Невірний email або пароль";
      document.getElementById("authMessage").style.color = "red";
    }
  } catch (err) {
    console.error("❌ Помилка входу:", err);
    document.getElementById("authMessage").innerText = "Не вдалося з'єднатися з сервером.";
    document.getElementById("authMessage").style.color = "red";
  }
}// <- закриття handleLogin

async function submitReport(e) {
  e.preventDefault();

  const type = document.getElementById("reportType").value;
  const description = document.getElementById("reportDescription").value;

  try {
    const res = await fetch("https://eco-monitoring-backend-production.up.railway.app/api/report", {
      method: "POST",
      headers: {
        "Content-Type": "application/json"
      },
      body: JSON.stringify({ type, description })
```

```
    });

    const data = await res.json();
    console.log("📩 Відповідь на сповіщення:", data);

    if (res.ok) {
      document.getElementById("reportMessage").innerText = data.message || "Сповіщення надіслано!";
      document.getElementById("reportMessage").style.color = "green";
      document.getElementById("reportForm").reset();
    } else {
      document.getElementById("reportMessage").innerText = data.message || "Помилка при надсиланні.";
      document.getElementById("reportMessage").style.color = "red";
    }
  } catch (err) {
    console.error("❌ Помилка при надсиланні:", err);
    document.getElementById("reportMessage").innerText = "Не вдалося зв’язатись із сервером.";
    document.getElementById("reportMessage").style.color = "red";
  }
}

</script>

<script>
  if ('serviceWorker' in navigator) {
    window.addEventListener('load', () => {
      navigator.serviceWorker.register('service-worker.js')
        .then(reg => console.log("✅ SW зареєстрований:", reg.scope))
        .catch(err => console.error("❌ Помилка реєстрації SW:", err));
    });
  }
</script>



</body>
</html>//manifest.json
{
  "name": "ЕкоМоніторинг",
```

```
  "short_name": "Εκο",
  "start_url": "index.html",
  "display": "standalone",
  "background_color": "#ffffff",
  "theme_color": "#4CAF50",
  "icons": [
    {
      "src": "icons/icon-192.png",
      "sizes": "192x192",
      "type": "image/png"
    },
    {
      "src": "icons/icon-512.png",
      "sizes": "512x512",
      "type": "image/png"
    }
  ]
}

//package-lock.json
{
  "name": "eco-monitoring",
  "version": "1.0.0",
  "lockfileVersion": 3,
  "requires": true,
  "packages": {
    "": {
      "name": "eco-monitoring",
      "version": "1.0.0",
      "dependencies": {
        "bcryptjs": "^3.0.2",
        "cors": "^2.8.5",
        "dotenv": "^16.5.0",
        "express": "^5.1.0",
        "jsonwebtoken": "^9.0.2",
        "mongoose": "^8.14.3",
        "nodemailer": "^7.0.3"
      }
    },
```

```
    "node_modules/@mongodb-js/saslprep": {
      "version": "1.2.2",
      "resolved": "https://registry.npmjs.org/@mongodb-js/saslprep/-/saslprep-1.2.2.tgz",
      "integrity": "sha512-
EB0O3SCSNRUFk66iRCpI+cXzIjdswfCs7F6nOC3RAGJ7xr5YhaicvsRwJ9eyzYvYRlCSDUO/c7g4yNulxKC1WA==",
      "license": "MIT",
      "dependencies": {
        "sparse-bitfield": "^3.0.3"
      }
    },
    "node_modules/@types/webidl-conversions": {
      "version": "7.0.3",
      "resolved": "https://registry.npmjs.org/@types/webidl-conversions/-/webidl-conversions-7.0.3.tgz",
      "integrity": "sha512-
CiJJvcRtIgzadHCYXw7dqEnMNRjhGZlYK05Mj9OyktqV8uVT8fD2BFOB7S1uwBE3Kj2Z+4UyPmFw/Ixgw/LAlA==",
      "license": "MIT"
    },
    "node_modules/@types/whatwg-url": {
      "version": "11.0.5",
      "resolved": "https://registry.npmjs.org/@types/whatwg-url/-/whatwg-url-11.0.5.tgz",
      "integrity": "sha512-
coYR071JRaHa+xoEvvYqvnIHaVqaYrLPbsufM9BF63HkwI5Lgmy2QR8Q5K/lYDYo5AK82wOvSOS0UsLTpTG7uQ==
",
      "license": "MIT",
      "dependencies": {
        "@types/webidl-conversions": "*"
      }
    },
    "node_modules/accepts": {
      "version": "2.0.0",
      "resolved": "https://registry.npmjs.org/accepts/-/accepts-2.0.0.tgz",
      "integrity": "sha512-
5cvg6CtKwfgdmVqY1WIiXKc3Q1bkRqGLi+2W/6ao+6Y7gu/RCwRuAhGEzh5B4KlszSuTLgZYuqFqo5bImjNKng==",
      "license": "MIT",
      "dependencies": {
        "mime-types": "^3.0.0",
        "negotiator": "^1.0.0"
      },
      "engines": {
```

```
      "node": ">= 0.6"
    }
  },
  "node_modules/bcryptjs": {
    "version": "3.0.2",
    "resolved": "https://registry.npmjs.org/bcryptjs/-/bcryptjs-3.0.2.tgz",
    "integrity": "sha512-
k38b3XOZKv60C4E2hVsXTolJWfkGRMbILBIe2IBITXciy5bOsTKot5kDrf3ZfufQtQOUN5mXceUEpU1rTl9Uog==",
    "license": "BSD-3-Clause",
    "bin": {
      "bcrypt": "bin/bcrypt"
    }
  },
  "node_modules/body-parser": {
    "version": "2.2.0",
    "resolved": "https://registry.npmjs.org/body-parser/-/body-parser-2.2.0.tgz",
    "integrity": "sha512-
02qvAaxv8tp7fBa/mw1ga98OGm+eCbqzJOKoRt70sLmfEEi+jyBYVTDGfCL/k06/4EMk/z01gCe7HoCH/f2LTg==",
    "license": "MIT",
    "dependencies": {
      "bytes": "^3.1.2",
      "content-type": "^1.0.5",
      "debug": "^4.4.0",
      "http-errors": "^2.0.0",
      "iconv-lite": "^0.6.3",
      "on-finished": "^2.4.1",
      "qs": "^6.14.0",
      "raw-body": "^3.0.0",
      "type-is": "^2.0.0"
    },
    "engines": {
      "node": ">=18"
    }
  },
  "node_modules/bson": {
    "version": "6.10.3",
    "resolved": "https://registry.npmjs.org/bson/-/bson-6.10.3.tgz",
```

```
    "integrity": "sha512-
MTxGsqgYTwfshYWTRdmZRC+M7FnG1b4y7RO7p2k3X24Wq0yv1m77Wsj0BzlPzd/IowgESfsruQCUToa7vbOpPQ==
",
    "license": "Apache-2.0",
    "engines": {
     "node": ">=16.20.1"
    }
   },
   "node_modules/buffer-equal-constant-time": {
    "version": "1.0.1",
    "resolved": "https://registry.npmjs.org/buffer-equal-constant-time/-/buffer-equal-constant-time-1.0.1.tgz",
    "integrity": "sha512-
zRpUiDwd/xk6ADqPMATG8vc9VPrkck7T07OIx0gnjmJAnHnTVXNQG3vfvWNuiZIkwu9KrKdA1iJKfsfTVxE6NA==",
    "license": "BSD-3-Clause"
   },
   "node_modules/bytes": {
    "version": "3.1.2",
    "resolved": "https://registry.npmjs.org/bytes/-/bytes-3.1.2.tgz",
    "integrity": "sha512-
/Nf7TyzTx6S3yRJObOAV7956r8cr2+Oj8AC5dt8wSP3BQAoeX58NoHyCU8P8zGkNXStjTSi6fzO6F0pBdcYbEg==",
    "license": "MIT",
    "engines": {
     "node": ">= 0.8"
    }
   },
   "node_modules/call-bind-apply-helpers": {
    "version": "1.0.2",
    "resolved": "https://registry.npmjs.org/call-bind-apply-helpers/-/call-bind-apply-helpers-1.0.2.tgz",
    "integrity": "sha512-
Sp1ablJ0ivDkSzjcaJdxEunN5/XvksFJ2sMBFfq6x0ryhQV/2b/KwFe21cMpmHtPOSij8K99/wSfoEuTObmuMQ==",
    "license": "MIT",
    "dependencies": {
     "es-errors": "^1.3.0",
     "function-bind": "^1.1.2"
    },
    "engines": {
     "node": ">= 0.4"
    }
   },
```

```
    "node_modules/call-bound": {
      "version": "1.0.4",
      "resolved": "https://registry.npmjs.org/call-bound/-/call-bound-1.0.4.tgz",
      "integrity": "sha512-
+ys997U96po4Kx/ABpBCqhA9EuxJaQWDQg7295H4hBphv3IZg0boBKuwYpt4YXp6MZ5AmZQnU/tyMTlRpaSejg==",
      "license": "MIT",
      "dependencies": {
        "call-bind-apply-helpers": "^1.0.2",
        "get-intrinsic": "^1.3.0"
      },
      "engines": {
        "node": ">= 0.4"
      },
      "funding": {
        "url": "https://github.com/sponsors/ljharb"
      }
    },
    "node_modules/content-disposition": {
      "version": "1.0.0",
      "resolved": "https://registry.npmjs.org/content-disposition/-/content-disposition-1.0.0.tgz",
      "integrity": "sha512-
Au9nRL8VNUut/XSzbQA38+M78dzP4D+eqg3gfJHMIHHYa3bg067xj1KxMUWj+VULbiZMowKngFFbKczUrNJ1mg==
",
      "license": "MIT",
      "dependencies": {
        "safe-buffer": "5.2.1"
      },
      "engines": {
        "node": ">= 0.6"
      }
    },
    "node_modules/content-type": {
      "version": "1.0.5",
      "resolved": "https://registry.npmjs.org/content-type/-/content-type-1.0.5.tgz",
      "integrity": "sha512-
nTjqfcBFEipKdXCv4YDQWCfmcLZKm81ldF0pAopTvyrFGVbcR6P/VAAd5G7N+0tTr8QqiU0tFadD6FK4NtJwOA==",
      "license": "MIT",
      "engines": {
        "node": ">= 0.6"
```

```
    }
   },
   "node_modules/cookie": {
    "version": "0.7.2",
    "resolved": "https://registry.npmjs.org/cookie/-/cookie-0.7.2.tgz",
    "integrity": "sha512-
yki5XnKuf750l50uGTllt6kKILY4nQ1eNIQatoXEByZ5dWgnKqbnqmTrBE5B4N7lrMJKQ2ytWMiTO2o0v6Ew/w==",
    "license": "MIT",
    "engines": {
     "node": ">= 0.6"
    }
   },
   "node_modules/cookie-signature": {
    "version": "1.2.2",
    "resolved": "https://registry.npmjs.org/cookie-signature/-/cookie-signature-1.2.2.tgz",
    "integrity": "sha512-
D76uU73ulSXrD1UXF4KE2TMxVVwhsnCgfAyTg9k8P6KGZjlXKrOLe4dJQKI3Bxi5wjesZoFXJWElNWBjPZMbhg==",
    "license": "MIT",
    "engines": {
     "node": ">=6.6.0"
    }
   },
   "node_modules/cors": {
    "version": "2.8.5",
    "resolved": "https://registry.npmjs.org/cors/-/cors-2.8.5.tgz",
    "integrity": "sha512-
KIHbLJqu73RGr/hnbrO9uBeixNGuvSQjul/jdFvS/KFSIH1hWVd1ng7zOHx+YrEfInLG7q4n6GHQ9cDtxv/P6g==",
    "license": "MIT",
    "dependencies": {
     "object-assign": "^4",
     "vary": "^1"
    },
    "engines": {
     "node": ">= 0.10"
    }
   },
   "node_modules/debug": {
    "version": "4.4.0",
```

```
    "resolved": "https://registry.npmjs.org/debug/-/debug-4.4.0.tgz",
    "integrity": "sha512-
6WTZ/IxCY/T6BALoZHaE4ctp9xm+Z5kY/pzYaCHRFeyVhojxlrm+46y68HA6hr0TcwEssoxNiDEUJQjfPZ/RYA==",
    "license": "MIT",
    "dependencies": {
      "ms": "^2.1.3"
    },
    "engines": {
      "node": ">=6.0"
    },
    "peerDependenciesMeta": {
      "supports-color": {
        "optional": true
      }
    }
  },
  "node_modules/depd": {
    "version": "2.0.0",
    "resolved": "https://registry.npmjs.org/depd/-/depd-2.0.0.tgz",
    "integrity": "sha512-
g7nH6P6dyDioJogAAGprGpCtVImJhpPk/roCzdb3fIh61/s/nPsfR6onyMwkCAR/OlC3yBC0lESvUoQEAssIrw==",
    "license": "MIT",
    "engines": {
      "node": ">= 0.8"
    }
  },
  "node_modules/dotenv": {
    "version": "16.5.0",
    "resolved": "https://registry.npmjs.org/dotenv/-/dotenv-16.5.0.tgz",
    "integrity": "sha512-
m/C+AwOAr9/W1UOIZUo232ejMNnJAJtYQjUbHoNTBNTJSvqzzDh7vnrei3o3r3m9blf6ZoDkvcw0VmozNRFJxg==",
    "license": "BSD-2-Clause",
    "engines": {
      "node": ">=12"
    },
    "funding": {
      "url": "https://dotenvx.com"
    }
  },
```

```
    "node_modules/dunder-proto": {
      "version": "1.0.1",
      "resolved": "https://registry.npmjs.org/dunder-proto/-/dunder-proto-1.0.1.tgz",
      "integrity": "sha512-
KIN/nDJBQRcXw0MLVhZE9iQHmG68qAVIBg9CqmUYjmQIhgij9U5MFvrqkUL5FbtyyzZuOeOt0zdeRe4UY7ct+A=="
,
      "license": "MIT",
      "dependencies": {
        "call-bind-apply-helpers": "^1.0.1",
        "es-errors": "^1.3.0",
        "gopd": "^1.2.0"
      },
      "engines": {
        "node": ">= 0.4"
      }
    },
    "node_modules/ecdsa-sig-formatter": {
      "version": "1.0.11",
      "resolved": "https://registry.npmjs.org/ecdsa-sig-formatter/-/ecdsa-sig-formatter-1.0.11.tgz",
      "integrity": "sha512-
nagl3RYrbNv6kQkeJIpt6NJZy8twLB/2vtz6yN9Z4vRKHN4/QZJIEbqohALSgwKdnksuY3k5Addp5lg8sVoVcQ==",
      "license": "Apache-2.0",
      "dependencies": {
        "safe-buffer": "^5.0.1"
      }
    },
    "node_modules/ee-first": {
      "version": "1.1.1",
      "resolved": "https://registry.npmjs.org/ee-first/-/ee-first-1.1.1.tgz",
      "integrity": "sha512-
WMwm9LhRUo+WUaRN+vRuETqG89IgZphVSNkdFgeb6sS/E4OrDIN7t48CAewSHXc6C8lefD8KKfr5vY61brQlow=="
,
      "license": "MIT"
    },
    "node_modules/encodeurl": {
      "version": "2.0.0",
      "resolved": "https://registry.npmjs.org/encodeurl/-/encodeurl-2.0.0.tgz",
```

```
      "integrity": "sha512-
Q0n9HRi4m6JuGIV1eFlmvJB7ZEVxu93IrMyiMsGC0lrMJMWzRgx6WGquyfQgZVb31vhGgXnfmPNNXmxnOkRBrg==
",
      "license": "MIT",
      "engines": {
        "node": ">= 0.8"
      }
    },
    "node_modules/es-define-property": {
      "version": "1.0.1",
      "resolved": "https://registry.npmjs.org/es-define-property/-/es-define-property-1.0.1.tgz",
      "integrity": "sha512-
e3nRfgfUZ4rNGL232gUgX06QNyyez04KdjFrF+LTRoOXmrOgFKDg4BCdsjW8EnT69eqdYGmRpJwiPVYNrCaW3g==
",
      "license": "MIT",
      "engines": {
        "node": ">= 0.4"
      }
    },
    "node_modules/es-errors": {
      "version": "1.3.0",
      "resolved": "https://registry.npmjs.org/es-errors/-/es-errors-1.3.0.tgz",
      "integrity": "sha512-
Zf5H2Kxt2xjTvbJvP2ZWLEICxA6j+hAmMzIlypy4xcBg1vKVnx89Wy0GbS+kf5cwCVFFzdCFh2XSCFNULS6csw==",
      "license": "MIT",
      "engines": {
        "node": ">= 0.4"
      }
    },
    "node_modules/es-object-atoms": {
      "version": "1.1.1",
      "resolved": "https://registry.npmjs.org/es-object-atoms/-/es-object-atoms-1.1.1.tgz",
      "integrity": "sha512-
FGgH2h8zKNim9ljj7dankFPcICIK9Cp5bm+c2gQSYePhpaG5+esrLODihIorn+Pe6FGJzWhXQotPv73jTaldXA==",
      "license": "MIT",
      "dependencies": {
        "es-errors": "^1.3.0"
      },
      "engines": {
```

```
      "node": ">= 0.4"
    }
  },
  "node_modules/escape-html": {
    "version": "1.0.3",
    "resolved": "https://registry.npmjs.org/escape-html/-/escape-html-1.0.3.tgz",
    "integrity": "sha512-
NiSupZ4OeuGwr68lGIeym/ksIZMJodUGOSCZ/FSnTxcrekbvqrgdUxlJOMpijaKZVjAJrWrGs/6Jy8OMuyj9ow==",
    "license": "MIT"
  },
  "node_modules/etag": {
    "version": "1.8.1",
    "resolved": "https://registry.npmjs.org/etag/-/etag-1.8.1.tgz",
    "integrity": "sha512-
aIL5Fx7mawVa300al2BnEE4iNvo1qETxLrPI/o05L7z6go7fCw1J6EQmbK4FmJ2AS7kgVF/KEZWufBfdClMcPg==",
    "license": "MIT",
    "engines": {
      "node": ">= 0.6"
    }
  },
  "node_modules/express": {
    "version": "5.1.0",
    "resolved": "https://registry.npmjs.org/express/-/express-5.1.0.tgz",
    "integrity": "sha512-
DT9ck5YIRU+8GYzzU5kT3eHGA5iL+1Zd0EutOmTE9Dtk+Tvuzd23VBU+ec7HPNSTxXYO55gPV/hq4pSBJDjFpA==",
    "license": "MIT",
    "dependencies": {
      "accepts": "^2.0.0",
      "body-parser": "^2.2.0",
      "content-disposition": "^1.0.0",
      "content-type": "^1.0.5",
      "cookie": "^0.7.1",
      "cookie-signature": "^1.2.1",
      "debug": "^4.4.0",
      "encodeurl": "^2.0.0",
      "escape-html": "^1.0.3",
      "etag": "^1.8.1",
      "finalhandler": "^2.1.0",
```

```
      "fresh": "^2.0.0",
      "http-errors": "^2.0.0",
      "merge-descriptors": "^2.0.0",
      "mime-types": "^3.0.0",
      "on-finished": "^2.4.1",
      "once": "^1.4.0",
      "parseurl": "^1.3.3",
      "proxy-addr": "^2.0.7",
      "qs": "^6.14.0",
      "range-parser": "^1.2.1",
      "router": "^2.2.0",
      "send": "^1.1.0",
      "serve-static": "^2.2.0",
      "statuses": "^2.0.1",
      "type-is": "^2.0.1",
      "vary": "^1.1.2"
    },
    "engines": {
      "node": ">= 18"
    },
    "funding": {
      "type": "opencollective",
      "url": "https://opencollective.com/express"
    }
  },
  "node_modules/finalhandler": {
    "version": "2.1.0",
    "resolved": "https://registry.npmjs.org/finalhandler/-/finalhandler-2.1.0.tgz",
    "integrity": "sha512-
/t88Ty3d5JWQbWYgaOGCCYfXRwV1+be02WqYYlL6h0lEiUAMPM8o8qKGO01YIkOHzka2up08wvgYD0mDiI+q3Q
==",
    "license": "MIT",
    "dependencies": {
      "debug": "^4.4.0",
      "encodeurl": "^2.0.0",
      "escape-html": "^1.0.3",
      "on-finished": "^2.4.1",
      "parseurl": "^1.3.3",
      "statuses": "^2.0.1"
```

```
    },
    "engines": {
      "node": ">= 0.8"
    }
  },
  "node_modules/forwarded": {
    "version": "0.2.0",
    "resolved": "https://registry.npmjs.org/forwarded/-/forwarded-0.2.0.tgz",
    "integrity": "sha512-
buRG0fpBtRHSTCOASe6hD258tEubFoRLb4ZNA6NxMVHNw2gOcwHo9wyablzMzOA5z9xA9L1KNjk/Nt6MT9aYow=
=",
    "license": "MIT",
    "engines": {
      "node": ">= 0.6"
    }
  },
  "node_modules/fresh": {
    "version": "2.0.0",
    "resolved": "https://registry.npmjs.org/fresh/-/fresh-2.0.0.tgz",
    "integrity": "sha512-
Rx/WycZ60HOaqLKAi6cHRKKI7zxWbJ31MhntmtwMoaTeF7XFH9hhBp8vITaMidfljRQ6eYWCKkaTK+ykVJHP2A==
",
    "license": "MIT",
    "engines": {
      "node": ">= 0.8"
    }
  },
  "node_modules/function-bind": {
    "version": "1.1.2",
    "resolved": "https://registry.npmjs.org/function-bind/-/function-bind-1.1.2.tgz",
    "integrity": "sha512-
7XHNxH7qX9xG5mIwxkhumTox/MIRNcOgDrxWsMt2pAr23WHp6MrRlN7FBSFpCpr+oVO0F744iUgR82nJMfG2SA=
=",
    "license": "MIT",
    "funding": {
      "url": "https://github.com/sponsors/ljharb"
    }
  },
  "node_modules/get-intrinsic": {
```

```
      "version": "1.3.0",
      "resolved": "https://registry.npmjs.org/get-intrinsic/-/get-intrinsic-1.3.0.tgz",
      "integrity": "sha512-
9fSjSaos/fRIVIp+xSJlE6lfwhES7LNtKaCBIamHsjr2na1BiABJPo0mOjjz8GJDURarmCPGqaiVg5mfjb98CQ==",
      "license": "MIT",
      "dependencies": {
        "call-bind-apply-helpers": "^1.0.2",
        "es-define-property": "^1.0.1",
        "es-errors": "^1.3.0",
        "es-object-atoms": "^1.1.1",
        "function-bind": "^1.1.2",
        "get-proto": "^1.0.1",
        "gopd": "^1.2.0",
        "has-symbols": "^1.1.0",
        "hasown": "^2.0.2",
        "math-intrinsics": "^1.1.0"
      },
      "engines": {
        "node": ">= 0.4"
      },
      "funding": {
        "url": "https://github.com/sponsors/ljharb"
      }
    },
    "node_modules/get-proto": {
      "version": "1.0.1",
      "resolved": "https://registry.npmjs.org/get-proto/-/get-proto-1.0.1.tgz",
      "integrity": "sha512-
sTSfBjoXBp89JvIKIefqw7U2CCebsc74kiY6awiGogKtoSGbgjYE/G/+l9sF3MWFPNc9IcoOC4ODfKHfxFmp0g==",
      "license": "MIT",
      "dependencies": {
        "dunder-proto": "^1.0.1",
        "es-object-atoms": "^1.0.0"
      },
      "engines": {
        "node": ">= 0.4"
      }
    },
    "node_modules/gopd": {
```

```
    "version": "1.2.0",
    "resolved": "https://registry.npmjs.org/gopd/-/gopd-1.2.0.tgz",
    "integrity": "sha512-
ZUKRh6/kUFoAiTAtTYPZJ3hw9wNxx+BIBOijnlG9PnrJsCcSjs1wyyD6vJpaYtgnzDrKYRSqf3OO6Rfa93xsRg==",
    "license": "MIT",
    "engines": {
      "node": ">= 0.4"
    },
    "funding": {
      "url": "https://github.com/sponsors/ljharb"
    }
  },
  "node_modules/has-symbols": {
    "version": "1.1.0",
    "resolved": "https://registry.npmjs.org/has-symbols/-/has-symbols-1.1.0.tgz",
    "integrity": "sha512-
1cDNdwJ2Jaohmb3sg4OmKaMBwuC48sYni5HUw2DvsC8LjGTLK9h+eb1X6RyuOHe4hT0ULCW68iomhjUoKUqlPQ=
=",
    "license": "MIT",
    "engines": {
      "node": ">= 0.4"
    },
    "funding": {
      "url": "https://github.com/sponsors/ljharb"
    }
  },
  "node_modules/hasown": {
    "version": "2.0.2",
    "resolved": "https://registry.npmjs.org/hasown/-/hasown-2.0.2.tgz",
    "integrity": "sha512-
0hJU9SCPvmMzIBdZFqNPXWa6dqh7WdH0cII9y+CyS8rG3nL48Bclra9HmKhVVUHyPWNH5Y7xDwAB7bfgSjkUMQ
==",
    "license": "MIT",
    "dependencies": {
      "function-bind": "^1.1.2"
    },
    "engines": {
      "node": ">= 0.4"
    }
```

```
  },
  "node_modules/http-errors": {
    "version": "2.0.0",
    "resolved": "https://registry.npmjs.org/http-errors/-/http-errors-2.0.0.tgz",
    "integrity": "sha512-
FtwrG/euBzaEjYeRqOgly7G0qviiXoJWnvEH2Z1plBdXgbyjv34pHTSb9zoeHMyDy33+DWy5Wt9Wo+TURtOYSQ==",
    "license": "MIT",
    "dependencies": {
      "depd": "2.0.0",
      "inherits": "2.0.4",
      "setprototypeof": "1.2.0",
      "statuses": "2.0.1",
      "toidentifier": "1.0.1"
    },
    "engines": {
      "node": ">= 0.8"
    }
  },
  "node_modules/iconv-lite": {
    "version": "0.6.3",
    "resolved": "https://registry.npmjs.org/iconv-lite/-/iconv-lite-0.6.3.tgz",
    "integrity": "sha512-
4fCk79wshMdzMp2rH06qWrJE4iolqLhCUH+OiuIgU++RB0+94NlDL81atO7GX55uUKueo0txHNtvEyI6D7WdMw==",
    "license": "MIT",
    "dependencies": {
      "safer-buffer": ">= 2.1.2 < 3.0.0"
    },
    "engines": {
      "node": ">=0.10.0"
    }
  },
  "node_modules/inherits": {
    "version": "2.0.4",
    "resolved": "https://registry.npmjs.org/inherits/-/inherits-2.0.4.tgz",
    "integrity": "sha512-
k/vGaX4/Yla3WzyMCvTQOXYeIHvqOKtnqBduzTHpzpQZzAskKMhZ2K+EnBiSM9zGSoIFeMpXKxa4dYeZIQqewQ=
=",
    "license": "ISC"
  },
```

```
    "node_modules/ipaddr.js": {
      "version": "1.9.1",
      "resolved": "https://registry.npmjs.org/ipaddr.js/-/ipaddr.js-1.9.1.tgz",
      "integrity": "sha512-
0KI/607xoxSToH7GjN1FfSbLoU0+btTicjsQSWQlh/hZykN8KpmMf7uYwPW3R+akZ6R/w18ZlXSHBYXiYUPO3g==",
      "license": "MIT",
      "engines": {
        "node": ">= 0.10"
      }
    },
    "node_modules/is-promise": {
      "version": "4.0.0",
      "resolved": "https://registry.npmjs.org/is-promise/-/is-promise-4.0.0.tgz",
      "integrity": "sha512-
hvpoI6korhJMnej285dSg6nu1+e6uxs7zG3BYAm5byqDsgJNWwxzM6z6iZiAgQR4TJ30JmBTOwqZUw3WlyH3AQ==",
      "license": "MIT"
    },
    "node_modules/jsonwebtoken": {
      "version": "9.0.2",
      "resolved": "https://registry.npmjs.org/jsonwebtoken/-/jsonwebtoken-9.0.2.tgz",
      "integrity": "sha512-
PRp66vJ865SSqOlgqS8hujT5U4AOgMfhrwYIuIhfKaoSCZcirrmASQr8CX7cUg+RMih+hgznrjp99o+W4pJLHQ==",
      "license": "MIT",
      "dependencies": {
        "jws": "^3.2.2",
        "lodash.includes": "^4.3.0",
        "lodash.isboolean": "^3.0.3",
        "lodash.isinteger": "^4.0.4",
        "lodash.isnumber": "^3.0.3",
        "lodash.isplainobject": "^4.0.6",
        "lodash.isstring": "^4.0.1",
        "lodash.once": "^4.0.0",
        "ms": "^2.1.1",
        "semver": "^7.5.4"
      },
      "engines": {
        "node": ">=12",
        "npm": ">=6"
      }
```

```
  },
  "node_modules/jwa": {
   "version": "1.4.2",
   "resolved": "https://registry.npmjs.org/jwa/-/jwa-1.4.2.tgz",
   "integrity": "sha512-
eeH5JO+21J78qMvTIDdBXidBd6nG2kZjg5Ohz/1fpa28Z4CcsWUzJ1ZZyFq/3z3N17aZy+ZuBoHljASbL1WfOw==",
   "license": "MIT",
   "dependencies": {
    "buffer-equal-constant-time": "^1.0.1",
    "ecdsa-sig-formatter": "1.0.11",
    "safe-buffer": "^5.0.1"
   }
  },
  "node_modules/jws": {
   "version": "3.2.2",
   "resolved": "https://registry.npmjs.org/jws/-/jws-3.2.2.tgz",
   "integrity": "sha512-
YHlZCB6lMTllWDtSPHz/ZXTsi8S00usEV6v1tjq8tOUZzw7DpSDWVXjXDre6ed1w/pd495ODpHZYSdkRTsa0HA==",
   "license": "MIT",
   "dependencies": {
    "jwa": "^1.4.1",
    "safe-buffer": "^5.0.1"
   }
  },
  "node_modules/kareem": {
   "version": "2.6.3",
   "resolved": "https://registry.npmjs.org/kareem/-/kareem-2.6.3.tgz",
   "integrity": "sha512-
C3iHfuGUXK2u8/ipq9LfjFfXFxAZMQJJq7vLS45r3D9Y2xQ/m4S8zaR4zMLFWh9AsNPXmcFfUDhTEO8UIC/V6Q==",
   "license": "Apache-2.0",
   "engines": {
    "node": ">=12.0.0"
   }
  },
  "node_modules/lodash.includes": {
   "version": "4.3.0",
   "resolved": "https://registry.npmjs.org/lodash.includes/-/lodash.includes-4.3.0.tgz",
   "integrity": "sha512-
W3Bx6mdkRTGtlJISOvVD/lbqjTlPPUDTMnlXZFnVwi9NKJ6tiAk6LVdlhZMm17VZisqhKcgzpO5Wz91PCt5b0w==",
```

```
    "license": "MIT"
  },
  "node_modules/lodash.isboolean": {
    "version": "3.0.3",
    "resolved": "https://registry.npmjs.org/lodash.isboolean/-/lodash.isboolean-3.0.3.tgz",
    "integrity": "sha512-
Bz5mupy2SVbPHURB98VAcw+aHh4vRV5IPNhILUCsOzRmsTmSQ17jIuqopAentWoehktxGd9e/hbIXq980/1QJg==",
    "license": "MIT"
  },
  "node_modules/lodash.isinteger": {
    "version": "4.0.4",
    "resolved": "https://registry.npmjs.org/lodash.isinteger/-/lodash.isinteger-4.0.4.tgz",
    "integrity": "sha512-
DBwtEWN2caHQ9/imiNeEA5ys1JoRtRfY3d7V9wkqtbycnAmTvRRmbHKDV4a0EYc678/dia0jrte4tjYwVBaZUA==",
    "license": "MIT"
  },
  "node_modules/lodash.isnumber": {
    "version": "3.0.3",
    "resolved": "https://registry.npmjs.org/lodash.isnumber/-/lodash.isnumber-3.0.3.tgz",
    "integrity": "sha512-
QYqzpfwO3/CWf3XP+Z+tkQsfaLL/EnUlXWVkIk5FUPc4sBdTehEqZONuyRt2P67PXAk+NXmTBcc97zw9t1FQrw==",
    "license": "MIT"
  },
  "node_modules/lodash.isplainobject": {
    "version": "4.0.6",
    "resolved": "https://registry.npmjs.org/lodash.isplainobject/-/lodash.isplainobject-4.0.6.tgz",
    "integrity": "sha512-
oSXzaWypCMHkPC3NvBEaPHf0KsA5mvPrOPgQWDsbg8n7orZ290M0BmC/jgRZ4vcJ6DTAhjrsSYgdsW/F+MFOBA=
=",
    "license": "MIT"
  },
  "node_modules/lodash.isstring": {
    "version": "4.0.1",
    "resolved": "https://registry.npmjs.org/lodash.isstring/-/lodash.isstring-4.0.1.tgz",
    "integrity": "sha512-
0wJxfxH1wgO3GrbuP+dTTk7op+6L41QCXbGINEmD+ny/G/eCqGzxyCsh7159S+mgDDcoarnBw6PC1PS5+wUGgw=="
,
    "license": "MIT"
  },
```

```
    "node_modules/lodash.once": {
      "version": "4.1.1",
      "resolved": "https://registry.npmjs.org/lodash.once/-/lodash.once-4.1.1.tgz",
      "integrity": "sha512-
Sb487aTOCr9drQVL8pIxOzVhafOjZN9UU54hiN8PU3uAiSV7lx1yYNpbNmex2PK6dSJoNTSJUUswT651yww3Mg==",
      "license": "MIT"
    },
    "node_modules/math-intrinsics": {
      "version": "1.1.0",
      "resolved": "https://registry.npmjs.org/math-intrinsics/-/math-intrinsics-1.1.0.tgz",
      "integrity": "sha512-
/IXtbwEk5HTPyEwyKX6hGkYXxM9nbj64B+ilVJnC/R6B0pH5G4V3b0pVbL7DBj4tkhBAppbQUlf6F6Xl9LHu1g==",
      "license": "MIT",
      "engines": {
        "node": ">= 0.4"
      }
    },
    "node_modules/media-typer": {
      "version": "1.1.0",
      "resolved": "https://registry.npmjs.org/media-typer/-/media-typer-1.1.0.tgz",
      "integrity": "sha512-
aisnrDP4GNe06UcKFnV5bfMNPBUw4jsLGaWwWfnH3v02GnBuXX2MCVn5RbrWo0j3pczUilYblq7fQ7Nw2t5XKw==
",
      "license": "MIT",
      "engines": {
        "node": ">= 0.8"
      }
    },
    "node_modules/memory-pager": {
      "version": "1.5.0",
      "resolved": "https://registry.npmjs.org/memory-pager/-/memory-pager-1.5.0.tgz",
      "integrity": "sha512-
ZS4Bp4r/Zoeq6+NLJpP+0Zzm0pR8whtGPf1XExKLJBAczGMnSi3It14OiNCStjQjM6NU1okjQGSxgEZN8eBYKg==",
      "license": "MIT"
    },
    "node_modules/merge-descriptors": {
      "version": "2.0.0",
      "resolved": "https://registry.npmjs.org/merge-descriptors/-/merge-descriptors-2.0.0.tgz",
```

```
    "integrity": "sha512-
Snk314V5ayFLhp3fkUREub6WtjBfPdCPY1Ln8/8munuLuiYhsABgBVWsozAG+MWMbVEvcdcpbi9R7ww22l9Q3g==",
    "license": "MIT",
    "engines": {
      "node": ">=18"
    },
    "funding": {
      "url": "https://github.com/sponsors/sindresorhus"
    }
  },
  "node_modules/mime-db": {
    "version": "1.54.0",
    "resolved": "https://registry.npmjs.org/mime-db/-/mime-db-1.54.0.tgz",
    "integrity": "sha512-
aU5EJuIN2WDemCcAp2vFBfp/m4EAhWJnUNSSw0ixs7/kXbd6Pg64EmwJkNdFhB8aWt1sH2CTXrLxo/iAGV3oPQ==",
    "license": "MIT",
    "engines": {
      "node": ">= 0.6"
    }
  },
  "node_modules/mime-types": {
    "version": "3.0.1",
    "resolved": "https://registry.npmjs.org/mime-types/-/mime-types-3.0.1.tgz",
    "integrity": "sha512-
xRc4oEhT6eaBpU1XF7AjpOFD+xQmXNB5OVKwp4tqCuBpHLS/ZbBDrc07mYTDqVMg6PfxUjjNp85O6Cd2Z/5HWA
==",
    "license": "MIT",
    "dependencies": {
      "mime-db": "^1.54.0"
    },
    "engines": {
      "node": ">= 0.6"
    }
  },
  "node_modules/mongodb": {
    "version": "6.16.0",
    "resolved": "https://registry.npmjs.org/mongodb/-/mongodb-6.16.0.tgz",
    "integrity": "sha512-
D1PNcdT0y4Grhou5Zi/qgipZOYeWrhLEpk33n3nm6LGtz61jvO88WlrWCK/bigMjpnOdAUKKQwsGIl0NtWMyYw==",
```

```
"license": "Apache-2.0",
"dependencies": {
  "@mongodb-js/saslprep": "^1.1.9",
  "bson": "^6.10.3",
  "mongodb-connection-string-url": "^3.0.0"
},
"engines": {
  "node": ">=16.20.1"
},
"peerDependencies": {
  "@aws-sdk/credential-providers": "^3.188.0",
  "@mongodb-js/zstd": "^1.1.0 || ^2.0.0",
  "gcp-metadata": "^5.2.0",
  "kerberos": "^2.0.1",
  "mongodb-client-encryption": ">=6.0.0 <7",
  "snappy": "^7.2.2",
  "socks": "^2.7.1"
},
"peerDependenciesMeta": {
  "@aws-sdk/credential-providers": {
    "optional": true
  },
  "@mongodb-js/zstd": {
    "optional": true
  },
  "gcp-metadata": {
    "optional": true
  },
  "kerberos": {
    "optional": true
  },
  "mongodb-client-encryption": {
    "optional": true
  },
  "snappy": {
    "optional": true
  },
  "socks": {
    "optional": true
```

```
      }
    }
  },
  "node_modules/mongodb-connection-string-url": {
    "version": "3.0.2",
    "resolved": "https://registry.npmjs.org/mongodb-connection-string-url/-/mongodb-connection-string-url-3.0.2.tgz",
    "integrity": "sha512-
rMO7CGo/9BFwyZABcKAWL8UJwH/Kc2x0g72uhDWzG48URRax5TCIcJ7Rc3RZqffZzO/Gwff/jyKwCU9TN8gehA==
",
    "license": "Apache-2.0",
    "dependencies": {
      "@types/whatwg-url": "^11.0.2",
      "whatwg-url": "^14.1.0 || ^13.0.0"
    }
  },
  "node_modules/mongoose": {
    "version": "8.14.3",
    "resolved": "https://registry.npmjs.org/mongoose/-/mongoose-8.14.3.tgz",
    "integrity": "sha512-
BiIQK4mZiStUgnNep1YJMMYTiC4K893+Dj/Sr3lvxXutqy4+yZMVhlHq60xRH3r/l6eXkQXO3tXJnVOE5g592Q==",
    "license": "MIT",
    "dependencies": {
      "bson": "^6.10.3",
      "kareem": "2.6.3",
      "mongodb": "~6.16.0",
      "mpath": "0.9.0",
      "mquery": "5.0.0",
      "ms": "2.1.3",
      "sift": "17.1.3"
    },
    "engines": {
      "node": ">=16.20.1"
    },
    "funding": {
      "type": "opencollective",
      "url": "https://opencollective.com/mongoose"
    }
  },
  "node_modules/mpath": {
```

```
    "version": "0.9.0",
    "resolved": "https://registry.npmjs.org/mpath/-/mpath-0.9.0.tgz",
    "integrity": "sha512-
ikJRQTk8hw5DEoFVxHG1Gn9T/xcjtdnOKIU1JTmGjZZlg9LST2mBLmcX3/ICIbgJydT2GOc15RnNy5mHmzfSew==",
    "license": "MIT",
    "engines": {
      "node": ">=4.0.0"
    }
  },
  "node_modules/mquery": {
    "version": "5.0.0",
    "resolved": "https://registry.npmjs.org/mquery/-/mquery-5.0.0.tgz",
    "integrity": "sha512-
iQMncpmEK8R8ncT8HJGsGc9Dsp8xcgYMVSbs5jgnm1lFHTZqMJTUWTDx1LBO8+mK3tPNZWFLBghQEIOULSTHZ
g==",
    "license": "MIT",
    "dependencies": {
      "debug": "4.x"
    },
    "engines": {
      "node": ">=14.0.0"
    }
  },
  "node_modules/ms": {
    "version": "2.1.3",
    "resolved": "https://registry.npmjs.org/ms/-/ms-2.1.3.tgz",
    "integrity": "sha512-
6FlzubTLZG3J2a/NVCAleEhjzq5oxgHyaCU9yYXvcLsvoVaHJq/s5xXI6/XXP6tz7R9xAOtHnSO/tXtF3WRTlA==",
    "license": "MIT"
  },
  "node_modules/negotiator": {
    "version": "1.0.0",
    "resolved": "https://registry.npmjs.org/negotiator/-/negotiator-1.0.0.tgz",
    "integrity": "sha512-
8Ofs/AUQh8MaEcrlq5xOX0CQ9ypTF5dl78mjlMNfOK08fzpgTHQRQPBxcPlEtIw0yRpws+Zo/3r+5WRby7u3Gg==",
    "license": "MIT",
    "engines": {
      "node": ">= 0.6"
    }
```

```
  },
  "node_modules/nodemailer": {
   "version": "7.0.3",
   "resolved": "https://registry.npmjs.org/nodemailer/-/nodemailer-7.0.3.tgz",
   "integrity": "sha512-
Ajq6Sz1x7cIK3pN6KesGTah+1gnwMnx5gKl3piQlQQE/PwyJ4Mbc8is2psWYxK3RJTVeqsDaCv8ZzXLCDHMTZw==",
   "license": "MIT-0",
   "engines": {
    "node": ">=6.0.0"
   }
  },
  "node_modules/object-assign": {
   "version": "4.1.1",
   "resolved": "https://registry.npmjs.org/object-assign/-/object-assign-4.1.1.tgz",
   "integrity": "sha512-
rJgTQnkUnH1sFw8yT6VSU3zD3sWmu6sZhIseY8VX+GRu3P6F7Fu+JNDoXfklElbLJSnc3FUQHVe4cU5hj+BcUg==",
   "license": "MIT",
   "engines": {
    "node": ">=0.10.0"
   }
  },
  "node_modules/object-inspect": {
   "version": "1.13.4",
   "resolved": "https://registry.npmjs.org/object-inspect/-/object-inspect-1.13.4.tgz",
   "integrity": "sha512-
W67iLl4J2EXEGTbfeHCffrjDfitvLANg0UlX3wFUUSTx92KXRFegMHUVgSqE+wvhAbi4WqjGg9czysTV2Epbew==",
   "license": "MIT",
   "engines": {
    "node": ">= 0.4"
   },
   "funding": {
    "url": "https://github.com/sponsors/ljharb"
   }
  },
  "node_modules/on-finished": {
   "version": "2.4.1",
   "resolved": "https://registry.npmjs.org/on-finished/-/on-finished-2.4.1.tgz",
   "integrity": "sha512-
oVlzkg3ENAhCk2zdv7IJwd/QUD4z2RxRwpkcGY8psCVcCYZNq4wYnVWALHM+brtuJjePWiYF/ClmuDr8Ch5+kg==",
```

```
      "license": "MIT",
      "dependencies": {
        "ee-first": "1.1.1"
      },
      "engines": {
        "node": ">= 0.8"
      }
    },
    "node_modules/once": {
      "version": "1.4.0",
      "resolved": "https://registry.npmjs.org/once/-/once-1.4.0.tgz",
      "integrity": "sha512-
lNaJgI+2Q5URQBkccEKHTQOPaXdUxnZZElQTZY0MFUAuaEqe1E+Nyvgdz/aIyNi6Z9MzO5dv1H8n58/GELp3+w=="
,
      "license": "ISC",
      "dependencies": {
        "wrappy": "1"
      }
    },
    "node_modules/parseurl": {
      "version": "1.3.3",
      "resolved": "https://registry.npmjs.org/parseurl/-/parseurl-1.3.3.tgz",
      "integrity": "sha512-
CiyeOxFT/JZyN5m0z9PfXw4SCBJ6Sygz1Dpl0wqjlhDEGGBP1GnsUVEL0p63hoG1fcj3fHynXi9NYO4nWOL+qQ==",
      "license": "MIT",
      "engines": {
        "node": ">= 0.8"
      }
    },
    "node_modules/path-to-regexp": {
      "version": "8.2.0",
      "resolved": "https://registry.npmjs.org/path-to-regexp/-/path-to-regexp-8.2.0.tgz",
      "integrity": "sha512-
TdrF7fW9Rphjq4RjrW0Kp2AW0Ahwu9sRGTkS6bvDi0SCwZlEZYmcfDbEsTz8RVk0EHIS/Vd1bv3JhG+1xZuAyQ==",
      "license": "MIT",
      "engines": {
        "node": ">=16"
      }
    },
```

```
    "node_modules/proxy-addr": {
      "version": "2.0.7",
      "resolved": "https://registry.npmjs.org/proxy-addr/-/proxy-addr-2.0.7.tgz",
      "integrity": "sha512-
llQsMLSUDUPT44jdrU/O37qlnifitDP+ZwrmmZcoSKyLKvtZxpyV0n2/bD/N4tBAAZ/gJEdZU7KMraoK1+XYAg==",
      "license": "MIT",
      "dependencies": {
        "forwarded": "0.2.0",
        "ipaddr.js": "1.9.1"
      },
      "engines": {
        "node": ">= 0.10"
      }
    },
    "node_modules/punycode": {
      "version": "2.3.1",
      "resolved": "https://registry.npmjs.org/punycode/-/punycode-2.3.1.tgz",
      "integrity": "sha512-
vYt7UD1U9Wg6138shLtLOvdAu+8DsC/ilFtEVHcH+wydcSpNE20AfSOduf6MkRFahL5FY7X1oU7nKVZFtfq8Fg==",
      "license": "MIT",
      "engines": {
        "node": ">=6"
      }
    },
    "node_modules/qs": {
      "version": "6.14.0",
      "resolved": "https://registry.npmjs.org/qs/-/qs-6.14.0.tgz",
      "integrity": "sha512-
YWWTjgABSKcvs/nWBi9PycY/JiPJqOD4JA6o9Sej2AtvSGarXxKC3OQSk4pAarbdQlKAh5D4FCQkJNkW+GAn3w=="
,
      "license": "BSD-3-Clause",
      "dependencies": {
        "side-channel": "^1.1.0"
      },
      "engines": {
        "node": ">=0.6"
      },
      "funding": {
        "url": "https://github.com/sponsors/ljharb"
```

```
    }
   },
   "node_modules/range-parser": {
    "version": "1.2.1",
    "resolved": "https://registry.npmjs.org/range-parser/-/range-parser-1.2.1.tgz",
    "integrity": "sha512-
Hrgsx+orqoygnmhFbKaHE6c296J+HTAQXoxEF6gNupROmmGJRoyzfG3ccAveqCBrwr/2yxQ5BVd/GTl5agOwSg==",
    "license": "MIT",
    "engines": {
     "node": ">= 0.6"
    }
   },
   "node_modules/raw-body": {
    "version": "3.0.0",
    "resolved": "https://registry.npmjs.org/raw-body/-/raw-body-3.0.0.tgz",
    "integrity": "sha512-
RmkhL8CAyCRPXCE28MMH0z2PNWQBNk2Q09ZdxM9IOOXwxwZbN+qbWaatPkdkWIKL2ZVDImrN/pK5HTRz2Pc
S4g==",
    "license": "MIT",
    "dependencies": {
     "bytes": "3.1.2",
     "http-errors": "2.0.0",
     "iconv-lite": "0.6.3",
     "unpipe": "1.0.0"
    },
    "engines": {
     "node": ">= 0.8"
    }
   },
   "node_modules/router": {
    "version": "2.2.0",
    "resolved": "https://registry.npmjs.org/router/-/router-2.2.0.tgz",
    "integrity": "sha512-
nLTrUKm2UyiL7rlhapu/Zl45FwNgkZGaCpZbIHajDYgwlJCOzLSk+cIPAnsEqV955GjILJnKbdQC1nVPz+gAYQ==",
    "license": "MIT",
    "dependencies": {
     "debug": "^4.4.0",
     "depd": "^2.0.0",
     "is-promise": "^4.0.0",
```

```
      "parseurl": "^1.3.3",
      "path-to-regexp": "^8.0.0"
     },
     "engines": {
      "node": ">= 18"
     }
    },
    "node_modules/safe-buffer": {
     "version": "5.2.1",
     "resolved": "https://registry.npmjs.org/safe-buffer/-/safe-buffer-5.2.1.tgz",
     "integrity": "sha512-
rp3So07KcdmmKbGvgaNxQSJr7bGVSVk5S9Eq1F+ppbRo70+YeaDxkw5Dd8NPN+GD6bjnYm2VuPuCXmpuYvmCXQ
==",
     "funding": [
      {
       "type": "github",
       "url": "https://github.com/sponsors/feross"
      },
      {
       "type": "patreon",
       "url": "https://www.patreon.com/feross"
      },
      {
       "type": "consulting",
       "url": "https://feross.org/support"
      }
     ],
     "license": "MIT"
    },
    "node_modules/safer-buffer": {
     "version": "2.1.2",
     "resolved": "https://registry.npmjs.org/safer-buffer/-/safer-buffer-2.1.2.tgz",
     "integrity": "sha512-
YZo3K82SD7Riyi0E1EQPojLz7kpepnSQI9IyPbHHg1XXXevb5dJI7tpyN2ADxGcQbHG7vcyRHk0cbwqcQriUtg==",
     "license": "MIT"
    },
    "node_modules/semver": {
     "version": "7.7.2",
     "resolved": "https://registry.npmjs.org/semver/-/semver-7.7.2.tgz",
```

```
    "integrity": "sha512-
RF0Fw+rO5AMf9MAyaRXI4AV0Ulj5lMHqVxxdSgiVbixSCXoEmmX/jk0CuJw4+3SqroYO9VoUh+HcuJivvtJemA==",
    "license": "ISC",
    "bin": {
      "semver": "bin/semver.js"
    },
    "engines": {
      "node": ">=10"
    }
  },
  "node_modules/send": {
    "version": "1.2.0",
    "resolved": "https://registry.npmjs.org/send/-/send-1.2.0.tgz",
    "integrity": "sha512-
uaW0WwXKpL9blXE2o0bRhoL2EGXIrZxQ2ZQ4mgcfoBxdFmQold+qWsD2jLrfZ0trjKL6vOw0j//eAwcALFjKSw==",
    "license": "MIT",
    "dependencies": {
      "debug": "^4.3.5",
      "encodeurl": "^2.0.0",
      "escape-html": "^1.0.3",
      "etag": "^1.8.1",
      "fresh": "^2.0.0",
      "http-errors": "^2.0.0",
      "mime-types": "^3.0.1",
      "ms": "^2.1.3",
      "on-finished": "^2.4.1",
      "range-parser": "^1.2.1",
      "statuses": "^2.0.1"
    },
    "engines": {
      "node": ">= 18"
    }
  },
  "node_modules/serve-static": {
    "version": "2.2.0",
    "resolved": "https://registry.npmjs.org/serve-static/-/serve-static-2.2.0.tgz",
    "integrity": "sha512-
61g9pCh0Vnh7IutZjtLGGpTA355+OPn2TyDv/6ivP2h/AdAVX9azsoxmg2/M6nZeQZNYBEwIcsne1mJd9oQItQ==",
    "license": "MIT",
```

```
    "dependencies": {
      "encodeurl": "^2.0.0",
      "escape-html": "^1.0.3",
      "parseurl": "^1.3.3",
      "send": "^1.2.0"
    },
    "engines": {
      "node": ">= 18"
    }
  },
  "node_modules/setprototypeof": {
    "version": "1.2.0",
    "resolved": "https://registry.npmjs.org/setprototypeof/-/setprototypeof-1.2.0.tgz",
    "integrity": "sha512-
E5LDX7Wrp85Kil5bhZv46j8jOeboKq5JMmYM3gVGdGH8xFpPWXUMsNrlODCrkoxMEeNi/XZIwuRvY4XNwYMJpw
==",
    "license": "ISC"
  },
  "node_modules/side-channel": {
    "version": "1.1.0",
    "resolved": "https://registry.npmjs.org/side-channel/-/side-channel-1.1.0.tgz",
    "integrity": "sha512-
ZX99e6tRweoUXqR+VBrslhda51Nh5MTQwou5tnUDgbtyM0dBgmhEDtWGP/xbKn6hqfPRHujUNwz5fy/wbbhnpw==",
    "license": "MIT",
    "dependencies": {
      "es-errors": "^1.3.0",
      "object-inspect": "^1.13.3",
      "side-channel-list": "^1.0.0",
      "side-channel-map": "^1.0.1",
      "side-channel-weakmap": "^1.0.2"
    },
    "engines": {
      "node": ">= 0.4"
    },
    "funding": {
      "url": "https://github.com/sponsors/ljharb"
    }
  },
  "node_modules/side-channel-list": {
```

```
      "version": "1.0.0",
      "resolved": "https://registry.npmjs.org/side-channel-list/-/side-channel-list-1.0.0.tgz",
      "integrity": "sha512-
FCLHtRD/gnpCiCHEiJLOwdmFP+wzCmDEkc9y7NsYxeF4u7Btsn1ZuwgwJGxImImHicJArLP4R0yX4c2KCrMrTA==",
      "license": "MIT",
      "dependencies": {
        "es-errors": "^1.3.0",
        "object-inspect": "^1.13.3"
      },
      "engines": {
        "node": ">= 0.4"
      },
      "funding": {
        "url": "https://github.com/sponsors/ljharb"
      }
    },
    "node_modules/side-channel-map": {
      "version": "1.0.1",
      "resolved": "https://registry.npmjs.org/side-channel-map/-/side-channel-map-1.0.1.tgz",
      "integrity": "sha512-
VCjCNfgMsby3tTdo02nbjtM/ewra6jPHmpThenkTYh8pG9ucZ/1P8So4u4FGBek/BjpOVsDCMoLA/iuBKIFXRA==",
      "license": "MIT",
      "dependencies": {
        "call-bound": "^1.0.2",
        "es-errors": "^1.3.0",
        "get-intrinsic": "^1.2.5",
        "object-inspect": "^1.13.3"
      },
      "engines": {
        "node": ">= 0.4"
      },
      "funding": {
        "url": "https://github.com/sponsors/ljharb"
      }
    },
    "node_modules/side-channel-weakmap": {
      "version": "1.0.2",
      "resolved": "https://registry.npmjs.org/side-channel-weakmap/-/side-channel-weakmap-1.0.2.tgz",
```

```
    "integrity": "sha512-
WPS/HvHQTYnHisLo9McqBHOJk2FkHO/tlpvldyrnem4aeQp4hai3gythswg6p01oSoTl58rcpiFAjF2br2Ak2A==",
    "license": "MIT",
    "dependencies": {
     "call-bound": "^1.0.2",
     "es-errors": "^1.3.0",
     "get-intrinsic": "^1.2.5",
     "object-inspect": "^1.13.3",
     "side-channel-map": "^1.0.1"
    },
    "engines": {
     "node": ">= 0.4"
    },
    "funding": {
     "url": "https://github.com/sponsors/ljharb"
    }
   },
   "node_modules/sift": {
    "version": "17.1.3",
    "resolved": "https://registry.npmjs.org/sift/-/sift-17.1.3.tgz",
    "integrity": "sha512-
Rtlj66/b0ICeFzYTuNvX/EF1igRbbnGSvEyT79McoZa/DeGhMyC5pWKOEsZKnpkqtSeovd5FL/bjHWC3CIIvCQ==",
    "license": "MIT"
   },
   "node_modules/sparse-bitfield": {
    "version": "3.0.3",
    "resolved": "https://registry.npmjs.org/sparse-bitfield/-/sparse-bitfield-3.0.3.tgz",
    "integrity": "sha512-
kvzhi7vqKTfkh0PZU+2D2PIllw2ymqJKujUcyPMd9Y75Nv4nPbGJZXNhxsgdQab2BmlDct1YnfQCguEvHr7VsQ==",
    "license": "MIT",
    "dependencies": {
     "memory-pager": "^1.0.2"
    }
   },
   "node_modules/statuses": {
    "version": "2.0.1",
    "resolved": "https://registry.npmjs.org/statuses/-/statuses-2.0.1.tgz",
    "integrity": "sha512-
RwNA9Z/7PrK06rYLIzFMlaF+l73iwpzsqRIFgbMLbTcLD6cOao82TaWefPXQvB2fOC4AjuYSEndS7N/mTCbkdQ==",
```

```
    "license": "MIT",
    "engines": {
      "node": ">= 0.8"
    }
  },
  "node_modules/toidentifier": {
    "version": "1.0.1",
    "resolved": "https://registry.npmjs.org/toidentifier/-/toidentifier-1.0.1.tgz",
    "integrity": "sha512-
o5sSPKEkg/DIQNmH43V0/uerLrpzVedkUh8tGNvaeXpfpuwjKenlSox/2O/BTlZUtEe+JG7s5YhEz608PlAHRA==",
    "license": "MIT",
    "engines": {
      "node": ">=0.6"
    }
  },
  "node_modules/tr46": {
    "version": "5.1.1",
    "resolved": "https://registry.npmjs.org/tr46/-/tr46-5.1.1.tgz",
    "integrity": "sha512-
hdF5ZgjTqgAntKkklYw0R03MG2x/bSzTtkxmIRw/sTNV8YXsCJ1tfLAX23lhxhHJlEf3CRCOCGGWw3vI3GaSPw==",
    "license": "MIT",
    "dependencies": {
      "punycode": "^2.3.1"
    },
    "engines": {
      "node": ">=18"
    }
  },
  "node_modules/type-is": {
    "version": "2.0.1",
    "resolved": "https://registry.npmjs.org/type-is/-/type-is-2.0.1.tgz",
    "integrity": "sha512-
OZs6gsjF4vMp32qrCbiVSkrFmXtG/AZhY3t0iAMrMBiAZyV9oALtXO8hsrHbMXF9x6L3grlFuwW2oAz7cav+Gw==",
    "license": "MIT",
    "dependencies": {
      "content-type": "^1.0.5",
      "media-typer": "^1.1.0",
      "mime-types": "^3.0.0"
    },
```

```
    "engines": {
      "node": ">= 0.6"
    }
  },
  "node_modules/unpipe": {
    "version": "1.0.0",
    "resolved": "https://registry.npmjs.org/unpipe/-/unpipe-1.0.0.tgz",
    "integrity": "sha512-
pjy2bYhSsufwWlKwPc+l3cN7+wuJlK6uz0YdJEOlQDbl6jo/YlPi4mb8agUkVC8BF7V8NuzeyPNqRksA3hztKQ==",
    "license": "MIT",
    "engines": {
      "node": ">= 0.8"
    }
  },
  "node_modules/vary": {
    "version": "1.1.2",
    "resolved": "https://registry.npmjs.org/vary/-/vary-1.1.2.tgz",
    "integrity": "sha512-
BNGbWLfd0eUPabhkXUVm0j8uuvREyTh5ovRa/dyow/BqAbZJyC+5fU+IzQOzmAKzYqYRAISoRhdQr3eIZ/PXqg==",
    "license": "MIT",
    "engines": {
      "node": ">= 0.8"
    }
  },
  "node_modules/webidl-conversions": {
    "version": "7.0.0",
    "resolved": "https://registry.npmjs.org/webidl-conversions/-/webidl-conversions-7.0.0.tgz",
    "integrity": "sha512-
VwddBukDzu71offAQR975unBIGqfKZpM+8ZX6ySk8nYhVoo5CYaZyzt3YBvYtRtO+aoGlqxPg/B87NGVZ/fu6g==",
    "license": "BSD-2-Clause",
    "engines": {
      "node": ">=12"
    }
  },
  "node_modules/whatwg-url": {
    "version": "14.2.0",
    "resolved": "https://registry.npmjs.org/whatwg-url/-/whatwg-url-14.2.0.tgz",
```

```
      "integrity": "sha512-
De72GdQZzNTUBBChsXueQUnPKDkg/5A5zp7pFDuQAj5UFoENpiACU0wlCvzpAGnTkj++ihpKwKyYewn/XNUbKw
==",
      "license": "MIT",
      "dependencies": {
        "tr46": "^5.1.0",
        "webidl-conversions": "^7.0.0"
      },
      "engines": {
        "node": ">=18"
      }
    },
    "node_modules/wrappy": {
      "version": "1.0.2",
      "resolved": "https://registry.npmjs.org/wrappy/-/wrappy-1.0.2.tgz",
      "integrity": "sha512-
l4Sp/DRseor9wL6EvV2+TuQn63dMkPjZ/sp9XkghTEbV9KlPS1xUsZ3u7/IQO4wxtcFB4bgpQPRcR3QCvezPcQ==",
      "license": "ISC"
    }
  }
}
//package.json
{
  "name": "eco-monitoring",
  "version": "1.0.0",
  "description": "Екологічний моніторинг сайту",
  "main": "server.js",
  "scripts": {
    "start": "node server.js"
  },
  "dependencies": {
    "bcryptjs": "^3.0.2",
    "cors": "^2.8.5",
    "dotenv": "^16.5.0",
    "express": "^5.1.0",
    "jsonwebtoken": "^9.0.2",
    "mongoose": "^8.14.3",
    "nodemailer": "^7.0.3"
  }
```

```
}
//server.js
console.log("🚀 Сервер запускається...");
const express = require('express');
const mongoose = require('mongoose');
const bcrypt = require('bcryptjs');
const jwt = require('jsonwebtoken');
const dotenv = require("dotenv");
dotenv.config();

console.log("🔐 JWT_SECRET із .env:", process.env.JWT_SECRET);

const cors = require('cors');

const Report = require('./server/models/report');
const profileRoute = require('./server/routes/profile');

const app = express();
app.use(express.json());
app.use(cors());
app.use('/api/profile', profileRoute);

// Підключення до MongoDB
setTimeout(() => {
  mongoose.connect(process.env.MONGO_URI, {
    useNewUrlParser: true,
    useUnifiedTopology: true
  }).then(() => console.log('✅ MongoDB connected'))
    .catch(err => console.error('❌ MongoDB error:', err));
}, 3000); // 3 секунди затримки

const User = require('./server/models/user');


// Реєстрація
app.post('/api/register', async (req, res) => {
  try {
    console.log("🔥 req.body:", req.body); // ← Додай сюди!
```

```
    const { email, password, region } = req.body;
    console.log("📩 Запит на реєстрацію:", req.body); // Можеш залишити також

    const existingUser = await User.findOne({ email });
    if (existingUser) {
      return res.status(400).json({ message: 'Користувач вже існує' });
    }

    const hashedPassword = await bcrypt.hash(password, 10);
    const user = new User({ email, password: hashedPassword, region });
    await user.save();

    res.status(201).json({ message: 'Реєстрація успішна' });
  } catch (err) {
    console.error("❌ Помилка під час реєстрації:", err.message);
    res.status(500).json({
      message: 'Помилка сервера при реєстрації',
      error: err.message
    });
  }
});


// Вхід
app.post('/api/login', async (req, res) => {
  const { email, password } = req.body;
  const user = await User.findOne({ email });
  if (!user || !(await bcrypt.compare(password, user.password))) {
    return res.status(401).json({ message: 'Неправильний email або пароль' });
  }

 const token = jwt.sign(
  {
    userId: user._id,
    email: user.email,
    region: user.region,
    address: user.address,
```

```
    role: user.role
  },
  process.env.JWT_SECRET,
  { expiresIn: '1d' }
);

  res.json({ token });
});

// Сповіщення (імітація)
app.get('/api/alert', async (req, res) => {
  const { region } = req.query;
  if (region === 'Київ') {
    res.json({ message: '⚠️ Високий рівень забруднення повітря в Києві!' });
  } else {
    res.json({ message: '✅ Все спокійно у вашому регіоні' });
  }
});

// Додавання нового сповіщення
app.post('/api/report', async (req, res) => {
  try {
    const { type, description } = req.body;
    const newReport = new Report({ type, description });
    await newReport.save();

    res.status(201).json({ message: 'Сповіщення успішно надіслано!' });
  } catch (err) {
    console.error("❌ Помилка при збереженні сповіщення:", err.message);
    res.status(500).json({ message: 'Помилка сервера при додаванні сповіщення' });
  }
});

// Запуск
const PORT = process.env.PORT || 5000;
app.listen(PORT, '0.0.0.0', () => console.log(`🌍 Сервер запущено на порту ${PORT}`));
//service-worker.js
const CACHE_NAME = 'eco-monitoring-v1';
```

```
const urlsToCache = [
  '/',
  '/index.html',
  '/dashboard.html',
  '/manifest.json',
  '/icons/icon-192.png',
  '/icons/icon-512.png'
];

// Кешування під час встановлення
self.addEventListener('install', (event) => {
  event.waitUntil(
    caches.open(CACHE_NAME).then((cache) => {
      console.log('✅ SW: Кешуємо ресурси');
      return cache.addAll(urlsToCache);
    })
  );
});

// Перехоплення запитів — кеш або мережа
self.addEventListener('fetch', (event) => {
  event.respondWith(
    caches.match(event.request).then((cachedResponse) => {
      return cachedResponse || fetch(event.request);
    })
  );
});

// Видалення старих кешів
self.addEventListener('activate', (event) => {
  event.waitUntil(
    caches.keys().then((keyList) =>
      Promise.all(
        keyList.map((key) => {
          if (key !== CACHE_NAME) {
            console.log('🗑️ SW: Видаляємо старий кеш:', key);
            return caches.delete(key);
          }
```

```
      })
    )
  )
 );
});
//soil-quality. Html
<!DOCTYPE html>
<html lang="uk">
<head>
 <meta charset="UTF-8" />
 <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
 <title>Якість ґрунту</title>
 <style>
  body {
   font-family: Arial, sans-serif;
   margin: 0;
   padding: 0;
   background: linear-gradient(135deg, #f0f4c3, #ffffff);
  }
  header {
   background-color: #8bc34a;
   color: white;
   text-align: center;
   padding: 20px;
  }
  #map {
   width: 100%;
   height: 500px;
  }
  .content {
   padding: 20px;
  }
  #bgToggle {
   margin-top: 20px;
   padding: 10px 20px;
   background-color: #8bc34a;
   color: white;
   border: none;
   border-radius: 4px;
```

```
      cursor: pointer;
    }
  </style>
</head>
<body>
  <header>
    <h1>🌱 Якість ґрунту</h1>
    <p>Інтерактивна карта ґрунтів України</p>
  </header>

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

  <div class="content">
    <button id="bgToggle">🎨 Змінити фон</button>
  </div>

  <script>
    let map;
    function initMap() {
      const ukraineCenter = { lat: 48.3794, lng: 31.1656 };
      map = new google.maps.Map(document.getElementById("map"), {
        zoom: 6,
        center: ukraineCenter,
      });

      // Додавання шару KML з картою ґрунтів
      const kmlLayer = new google.maps.KmlLayer({
        url: 'https://www.etomesto.com/kml/ukraine_soil_map.kml',
        map: map
      });

      // Обробка помилки геолокації
      if (navigator.geolocation) {
        navigator.geolocation.getCurrentPosition(
          (position) => {
            const userPos = {
              lat: position.coords.latitude,
              lng: position.coords.longitude,
```

```
          };
          new google.maps.Marker({ position: userPos, map, title: "Ваше місцезнаходження" });
          map.setCenter(userPos);
        },
        (error) => {
          alert("Не вдалося отримати ваше місцезнаходження. Дані можуть бути неактуальними.");
        }
      );
    } else {
      alert("Геолокація не підтримується вашим браузером.");
    }
  }

  // Зміна фону
  const bgToggle = document.getElementById("bgToggle");
  let isDefault = true;
  bgToggle.addEventListener("click", () => {
    document.body.style.background = isDefault
      ? "linear-gradient(135deg, #dcedc8, #f0f4c3)"
      : "linear-gradient(135deg, #f0f4c3, #ffffff)";
    isDefault = !isDefault;
  });
 </script>

 <script
src="https://maps.googleapis.com/maps/api/js?key=AIzaSyAOib_11QyNZYlK2wIjsBEM2fWJf6CYiF0&callback=initMap
" async defer></script>
</body>
</html>
//firebase.json
{
 "hosting": {
   "public": "public",
   "ignore": [
     "firebase.json",
     "**/.*",
     "**/node_modules/**"
   ]
 }
```

```
}
//dashboard.html
<!DOCTYPE html>
<html lang="uk">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Особистий кабінет</title>
  <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
  <style>
    body {
      font-family: Arial, sans-serif;
      background-color: #f1f1f1;
      margin: 0;
      padding: 0;
    }

    .container {
      max-width: 1000px;
      margin: 30px auto;
      background: white;
      padding: 30px;
      border-radius: 10px;
      box-shadow: 0 0 15px rgba(0, 0, 0, 0.2);
    }

    h2 {
      text-align: center;
      color: #4CAF50;
    }

    .profile {
      display: flex;
      gap: 30px;
      margin-bottom: 30px;
      align-items: center;
    }

    .profile img {
```

```
  width: 120px;
  height: 120px;
  object-fit: cover;
  border-radius: 50%;
  border: 2px solid #4CAF50;
}

.info {
  font-size: 18px;
}

.info p {
  margin: 10px 0;
}

.section-title {
  font-weight: bold;
  margin-top: 30px;
  color: #333;
}

#reportList {
  background: #f9f9f9;
  padding: 15px;
  border-radius: 5px;
  max-height: 200px;
  overflow-y: auto;
  font-size: 15px;
}

.logout-btn {
  display: block;
  margin: 20px auto;
  padding: 10px 20px;
  background-color: #4CAF50;
  color: white;
  border: none;
  border-radius: 5px;
  cursor: pointer;
```

```
    }

    canvas {
      margin-top: 20px;
    }
  </style>
</head>
<body>
  <div class="container">
    <h2>👤 Особистий кабінет</h2>
    <div class="profile">
      <input type="file" id="avatarInput" accept="image/*" style="display: none;">
<img id="avatar" src="https://via.placeholder.com/120" alt="Фото користувача"
onclick="document.getElementById('avatarInput').click()">

      <div class="info">
        <p><strong>Email:</strong> <span id="email">Невідомо</span></p>
        <p><strong>Регіон:</strong> <span id="region">Невідомо</span></p>
        <p><strong>Адреса:</strong> <span id="address">Київ, Україна</span></p>
        <p><strong>Профіль:</strong> <span id="role">Користувач</span></p>
      </div>
    </div>
<button id="editBtn">Редагувати</button>

<div id="editForm" style="display: none; margin-top: 15px;">
  <label>Email: <input type="email" id="editEmail"></label><br>
  <label>Регіон: <input type="text" id="editRegion"></label><br>
  <button id="saveBtn">💾 Зберегти зміни</button>
</div>

    <div class="section-title">Статистика відвідувань</div>
    <canvas id="visitsChart" height="100"></canvas>

    <div class="section-title">Ваші сповіщення про проблеми</div>
    <div id="reportList">
      <p>Завантаження...</p>
    </div>
```

```
  <button class="logout-btn" onclick="logout()">Вийти</button>
 </div>

 <script>
  // Демо-графік відвідувань
  const ctx = document.getElementById('visitsChart').getContext('2d');
  new Chart(ctx, {
   type: 'line',
   data: {
    labels: ['Пн', 'Вт', 'Ср', 'Чт', 'Пт', 'Сб', 'Нд'],
    datasets: [{
     label: 'Відвідування',
     data: [3, 4, 2, 5, 6, 4, 1],
     borderColor: '#4CAF50',
     backgroundColor: 'rgba(76, 175, 80, 0.1)',
     tension: 0.3
    }]
   },
   options: {
    responsive: true,
    plugins: {
     legend: { display: false }
    }
   }
  });

  // Симуляція заповнення даних
  const token = localStorage.getItem("token");
  if (!token) {
   window.location.href = "index.html";
  }

  fetch("https://eco-monitoring-backend-production.up.railway.app/api/profile", {
 headers: { Authorization: `Bearer ${token}` }
})
 .then(res => {
  if (!res.ok) throw new Error("Профіль не знайдено");
  return res.json();
 })
```

```
  .then(data => {
    document.getElementById("email").innerText = data.email || "Невідомо";
    document.getElementById("region").innerText = data.region || "Невідомо";
    document.getElementById("address").innerText = data.address || "Невідомо";
    document.getElementById("role").innerText = data.role || "Користувач";
  })
  .catch(err => {
    console.error("❌ Помилка профілю:", err);
    alert("Не вдалося завантажити профіль користувача.");
  });

    fetch("https://eco-monitoring-backend-production.up.railway.app/api/reports", {
      headers: { "Authorization": `Bearer ${token}` }
    })
      .then(res => res.json())
      .then(data => {
        const list = document.getElementById("reportList");
        list.innerHTML = "";
        if (data.length === 0) {
          list.innerHTML = "<p>Жодного сповіщення ще не надсилалось.</p>";
        } else {
          data.forEach(rep => {
            const p = document.createElement("p");
            p.innerText = `• [${rep.type}] ${rep.description}`;
            list.appendChild(p);
          });
        }
      });

    function logout() {
      localStorage.removeItem("token");
      window.location.href = "index.html";
    }

    // Обробка вибору аватара
const avatarInput = document.getElementById("avatarInput");
const avatarImg = document.getElementById("avatar");
```

```
// Коли користувач обирає файл
avatarInput.addEventListener("change", function () {
  const file = this.files[0];
  if (!file) return;

  const reader = new FileReader();

  reader.onload = function (e) {
    const imageDataUrl = e.target.result;

    // Показуємо одразу на сторінці
    avatarImg.src = imageDataUrl;

    // Зберігаємо в localStorage
    localStorage.setItem("avatar", imageDataUrl);
  };

  reader.readAsDataURL(file);
});

// При завантаженні сторінки — встановити збережене фото
window.addEventListener("DOMContentLoaded", () => {
  const savedAvatar = localStorage.getItem("avatar");
  if (savedAvatar) {
    avatarImg.src = savedAvatar;
    const editBtn = document.getElementById("editBtn");
const editForm = document.getElementById("editForm");
const editEmail = document.getElementById("editEmail");
const editRegion = document.getElementById("editRegion");

editBtn.addEventListener("click", () => {
  // Показати форму
  editForm.style.display = "block";
  editEmail.value = document.getElementById("email").innerText;
  editRegion.value = document.getElementById("region").innerText;
});

document.getElementById("saveBtn").addEventListener("click", () => {
  const newEmail = editEmail.value.trim();
```

```
  const newRegion = editRegion.value.trim();

  if (!newEmail || !newRegion) {
    alert("Усі поля обов’язкові!");
    return;
  }

  // Оновити на сторінці
  document.getElementById("email").innerText = newEmail;
  document.getElementById("region").innerText = newRegion;

  // Можна зберегти в localStorage для демо
  localStorage.setItem("userEmail", newEmail);
  localStorage.setItem("userRegion", newRegion);

  // 🔄 Надіслати зміни на сервер для MongoDB
fetch("https://eco-monitoring-backend-production.up.railway.app/api/profile/update", {
  method: "PUT",
  headers: {
    "Content-Type": "application/json",
    "Authorization": `Bearer ${token}`
  },
  body: JSON.stringify({ email: newEmail, region: newRegion })
})
  .then(res => res.json())
  .then(data => {
    console.log("✅ Профіль оновлено на сервері:", data.message);
  })
  .catch(err => {
    console.error("❌ Помилка при оновленні профілю:", err);
  });

  // Сховати форму
  editForm.style.display = "none";
});
const savedEmail = localStorage.getItem("userEmail");
const savedRegion = localStorage.getItem("userRegion");
```

```
if (savedEmail) document.getElementById("email").innerText = savedEmail;
if (savedRegion) document.getElementById("region").innerText = savedRegion;

  }
});

  </script>
</body>
</html>
//air-quality.html
<!DOCTYPE html>
<html lang="uk">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
  <title>Якість повітря</title>
  <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
  <style>
    body {
      font-family: Arial, sans-serif;
      margin: 0;
      padding: 0;
      background: linear-gradient(135deg, #e0f7fa, #ffffff);
    }
    header {
      background-color: #26a69a;
      color: white;
      text-align: center;
      padding: 20px;
    }
    #map {
      width: 100%;
      height: 400px;
    }
    .content {
      padding: 20px;
    }
    #chartContainer {
      max-width: 600px;
```

```
    margin: 0 auto;
   }
   #recommendations {
    margin-top: 20px;
    background: #f1f8e9;
    padding: 15px;
    border-left: 5px solid #66bb6a;
   }
   #bgToggle {
    margin-top: 20px;
    padding: 10px 20px;
    background-color: #26a69a;
    color: white;
    border: none;
    border-radius: 4px;
    cursor: pointer;
   }
  </style>
</head>
<body>
  <header>
   <h1> Якість повітря</h1>
   <p>Інтерактивна карта, графік та рекомендації</p>
  </header>

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

  <div class="content">
   <div id="chartContainer">
    <canvas id="airChart"></canvas>
   </div>

   <div id="recommendations">
    <h3>Рекомендації:</h3>
    <ul id="adviceList"></ul>
   </div>

   <button id="bgToggle"> Змінити фон</button>
```

```
</div>

<script>
  let map;

  function initMap() {
    const kyiv = { lat: 50.4501, lng: 30.5234 };
    map = new google.maps.Map(document.getElementById("map"), {
      zoom: 10,
      center: kyiv,
    });

    navigator.geolocation.getCurrentPosition(
      pos => {
        const userPos = {
          lat: pos.coords.latitude,
          lng: pos.coords.longitude,
        };

        map.setCenter(userPos);
        new google.maps.Marker({ position: userPos, map, title: "Ваше місцезнаходження" });

        const circle = new google.maps.Circle({
          strokeColor: "#FF0000",
          strokeOpacity: 0.8,
          strokeWeight: 2,
          fillColor: "#FF0000",
          fillOpacity: 0.35,
          map,
          center: userPos,
          radius: 3000,
        });

        fetchAirQuality(userPos.lat, userPos.lng);
      },
      err => {
        alert("Не вдалося отримати місцезнаходження. Дані можуть бути неактуальні.");
        fetchAirQuality(50.4501, 30.5234);
      }
```

```
  );
}

async function fetchAirQuality(lat, lon) {
  const apiKey = "c0a820cfc0cc46cc51e67536bf08f5e3";
  const url = `https://api.openweathermap.org/data/2.5/air_pollution?lat=${lat}&lon=${lon}&appid=${apiKey}`;

  try {
    const res = await fetch(url);
    const data = await res.json();

    if (data && data.list && data.list.length > 0) {
      const aqi = data.list[0].main.aqi;
      const components = data.list[0].components;

      new Chart(document.getElementById("airChart").getContext("2d"), {
        type: "bar",
        data: {
          labels: ["PM2.5", "PM10", "O₃", "NO₂", "SO₂", "CO"],
          datasets: [{
            label: "µg/m³",
            data: [
              components.pm2_5, components.pm10, components.o3,
              components.no2, components.so2, components.co
            ],
            backgroundColor: "#26a69a"
          }]
        },
        options: {
          responsive: true,
          scales: { y: { beginAtZero: true } }
        }
      });

      const recommendations = [
        "Уникайте прогулянок поблизу доріг.",
        "Тримайте вікна зачиненими при високому забрудненні.",
        "Носіть маску, якщо відчуваєте утруднене дихання."
      ];
```

```
        document.getElementById("adviceList").innerHTML = recommendations.map(r => `<li>${r}</li>`).join("");

        // [6] Надсилання в MongoDB (опціонально)
        /*
        await fetch("https://eco-monitoring-backend-production.up.railway.app/api/airdata", {
          method: "POST",
          headers: { "Content-Type": "application/json" },
          body: JSON.stringify({ lat, lon, aqi, components })
        });
        */

      } else {
        alert("Дані про якість повітря недоступні для вашого регіону.");
      }

    } catch (err) {
      console.error("Помилка API:", err);
    }
  }

  //  Зміна фону
  const bgToggle = document.getElementById("bgToggle");
  let isDefault = true;
  bgToggle.addEventListener("click", () => {
    document.body.style.background = isDefault
      ? "linear-gradient(135deg, #ffecb3, #ffe082)"
      : "linear-gradient(135deg, #e0f7fa, #ffffff)";
    isDefault = !isDefault;
  });
 </script>

 <script
src="https://maps.googleapis.com/maps/api/js?key=AIzaSyAOib_11QyNZYlK2wIjsBEM2fWJf6CYiF0&callback=initMap
" async defer></script>
</body>
</html>
//profile.js
const express = require("express");
const router = express.Router();
```

```
const User = require("../models/user");
const auth = require("../middleware/auth");

router.get("/", auth, async (req, res) => {
  try {
    console.log(" Дані з токена:", req.user);
const user = await User.findById(req.user.id).select("email region address role");

    if (!user) return res.status(404).json({ message: "Користувача не знайдено" });
    res.json(user);
  } catch (err) {
    res.status(500).json({ message: "Помилка сервера" });
  }
});
// PUT /api/profile/update
router.put("/update", auth, async (req, res) => {
  try {
    const { email, region } = req.body;
    const userId = req.user.id;

    if (!email || !region) {
      return res.status(400).json({ message: "Email і Регіон обов’язкові" });
    }

    const updatedUser = await User.findByIdAndUpdate(
      userId,
      { email, region },
      { new: true }
    ).select("email region");

    res.json({
      message: "Профіль успішно оновлено",
      email: updatedUser.email,
      region: updatedUser.region,
    });
  } catch (err) {
    console.error("❌ Помилка оновлення профілю:", err);
    res.status(500).json({ message: "Помилка сервера" });
```

```
  }
});

module.exports = router;

// PUT /api/profile/update
router.put("/update", auth, async (req, res) => {
  try {
    const { email, region } = req.body;
    const userId = req.user.id;

    if (!email || !region) {
      return res.status(400).json({ message: "Email і Регіон обов’язкові" });
    }

    const updatedUser = await User.findByIdAndUpdate(
      userId,
      { email, region },
      { new: true }
    ).select("email region");

    res.json({
      message: "Профіль успішно оновлено",
      email: updatedUser.email,
      region: updatedUser.region,
    });
  } catch (err) {
    console.error("❌ Помилка оновлення профілю:", err);
    res.status(500).json({ message: "Помилка сервера" });
  }
});
//update.profile.js
const express = require('express');
const router = express.Router();
const jwt = require('jsonwebtoken');
const User = require('../models/user'); // твоя модель

//  Middleware для перевірки токена
```

```
function authenticate(req, res, next) {
  const authHeader = req.headers.authorization;
  if (!authHeader) return res.status(401).json({ message: "⛔ Токен відсутній" });

  const token = authHeader.split(" ")[1];
  try {
    const decoded = jwt.verify(token, process.env.JWT_SECRET);
    req.user = decoded;
    next();
  } catch (err) {
    return res.status(401).json({ message: "⛔ Невірний токен" });
  }
}

// Оновлення профілю
router.put('/', authenticate, async (req, res) => {
  const { email, region } = req.body;
  try {
    const user = await User.findByIdAndUpdate(
      req.user.userId,
      { email, region },
      { new: true }
    );
    res.json({
      message: "✅ Профіль оновлено",
      email: user.email,
      region: user.region
    });
  } catch (err) {
    res.status(500).json({ message: "❌ Помилка при оновленні профілю", error: err.message });
  }
});

module.exports = router;
//report.js
const mongoose = require('mongoose');

const reportSchema = new mongoose.Schema({
```

```
  type: String,
  description: String,
  createdAt: {
    type: Date,
    default: Date.now
  }
});

module.exports = mongoose.model('Report', reportSchema);
//user.js
const mongoose = require('mongoose');

const userSchema = new mongoose.Schema({
  email: String,
  password: String,
  region: String,
  address: { type: String, default: "Невідомо" },
  role: { type: String, default: "Користувач" }
});

//  Уникнення OverwriteModelError
module.exports = mongoose.models.User || mongoose.model('User', userSchema);
//auth.js
const jwt = require("jsonwebtoken");

module.exports = function (req, res, next) {
  const authHeader = req.headers.authorization;

  if (!authHeader || !authHeader.startsWith("Bearer ")) {
    return res.status(401).json({ message: "⛔ Немає токена" });
  }

  const token = authHeader.split(" ")[1];
  try {
    const decoded = jwt.verify(token, process.env.JWT_SECRET);
    req.user = { id: decoded.userId }; //
    next();
  } catch (err) {
```

```
    return res.status(403).json({ message: "⛔ Невірний токен" });
  }
};
```

Додаток Д. Ілюстративна частина

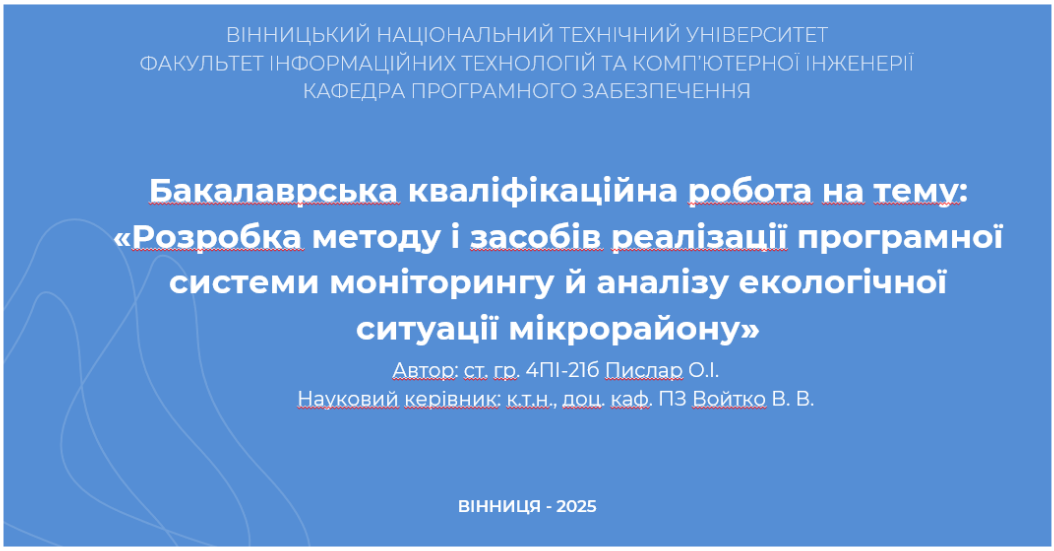


Рисунок Г.1 – Титульний слайд

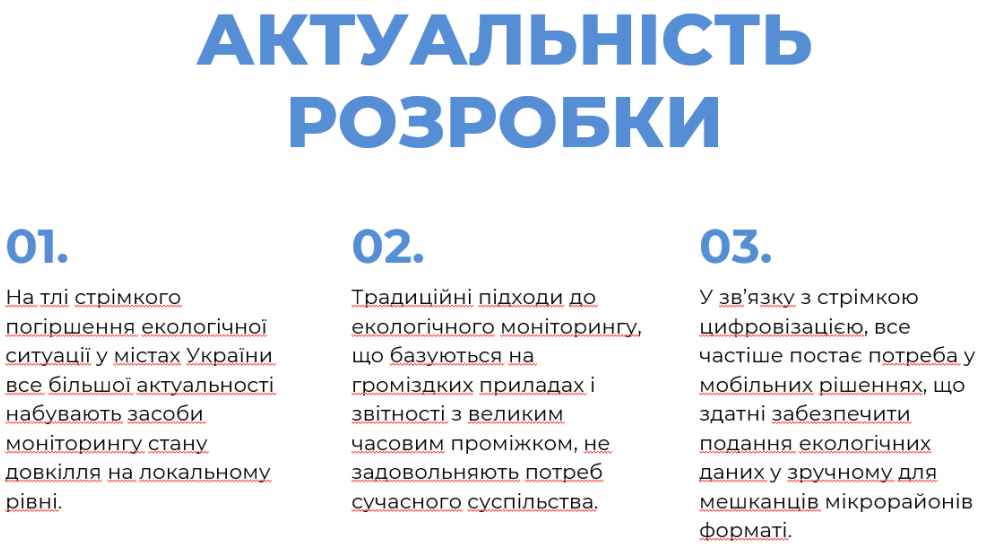


Рисунок Г.2 – Актуальність розробки

Рисунок Г.3 – Мета, предмет та об’єкт дослідження

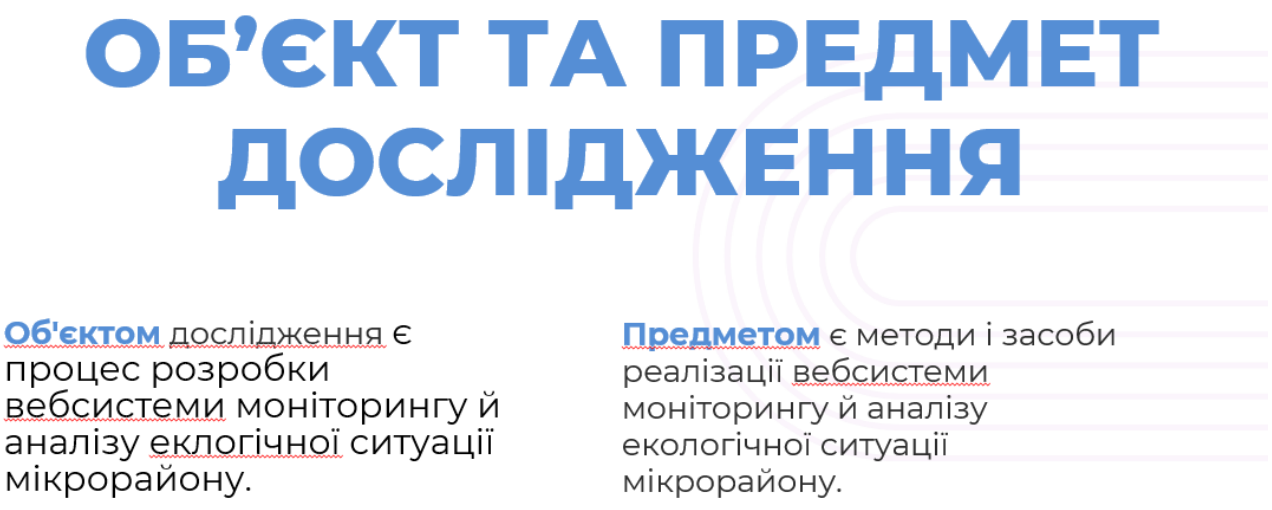


Рисунок Г.4 – Об’єкт та предмет дослідження

ПОСТАНОВКА ЗАДАЧІ

Після аналізу переваг та недоліків існуючих систем аналізу й моніторингу екологічної ситуації було визначено завдання для розробки системи:

1. Розробити архітектуру клієнт-серверної вебсистеми, що дозволяє збирати та обробляти екологічні дані у режимі реального часу з можливістю масштабування.
2. Реалізувати систему авторизації користувачів із використанням JWT-токенів, що забезпечує базовий рівень безпеки і персоналізації функціоналу.
3. Інтегрувати відкриті API для отримання даних про якість повітря та ґрунту з прив'язкою до геолокації користувача.
4. Розробити модуль аналітики з використанням бібліотеки Chart.js для побудови графіків зміни екологічних параметрів.
5. Забезпечити швидкий доступ до даних шляхом реалізації PWA із service worker та кешуванням ресурсів.
6. Провести тестування кожного з програмних модулів з метою забезпечення надійності й коректності роботи вебсистеми.
7. Створити інтерфейс користувача, який буде адаптований до мобільних пристроїв і забезпечить інтуїтивну взаємодію з системою.
8. Реалізувати функціонал збереження та перегляду власних сповіщень, які стосуються екологічних проблем у мікрорайоні.

Рисунок Г.5 – Постановка задачі

НОВИЗНА

01.

Подальшого розвитку отримав метод збору екологічного сповіщення, який, на відміну від існуючих, крім парсингу даних з відкритих джерел, реалізує ідентифікацію користувача на карті з визначенням мікрорайону, що дозволяє провести аналіз моніторингових даних для локалізованого місця.

02.

Подальшого розвитку отримала модель вебсистеми моніторигу екологічного стану мікрорайону, яка, на відміну від існуючих, інтегрує інтерактивні модулі відстеження користувача на карті, отримання моніторингових даних з відкритих джерел, аналіз отриманої інформації з можливістю візуалізації результатів моніторингового аналізу, що дозволяє покращити інтерактивну взаємодію користувача з програмною системою.

Рисунок Г.6 – Наукова новизна

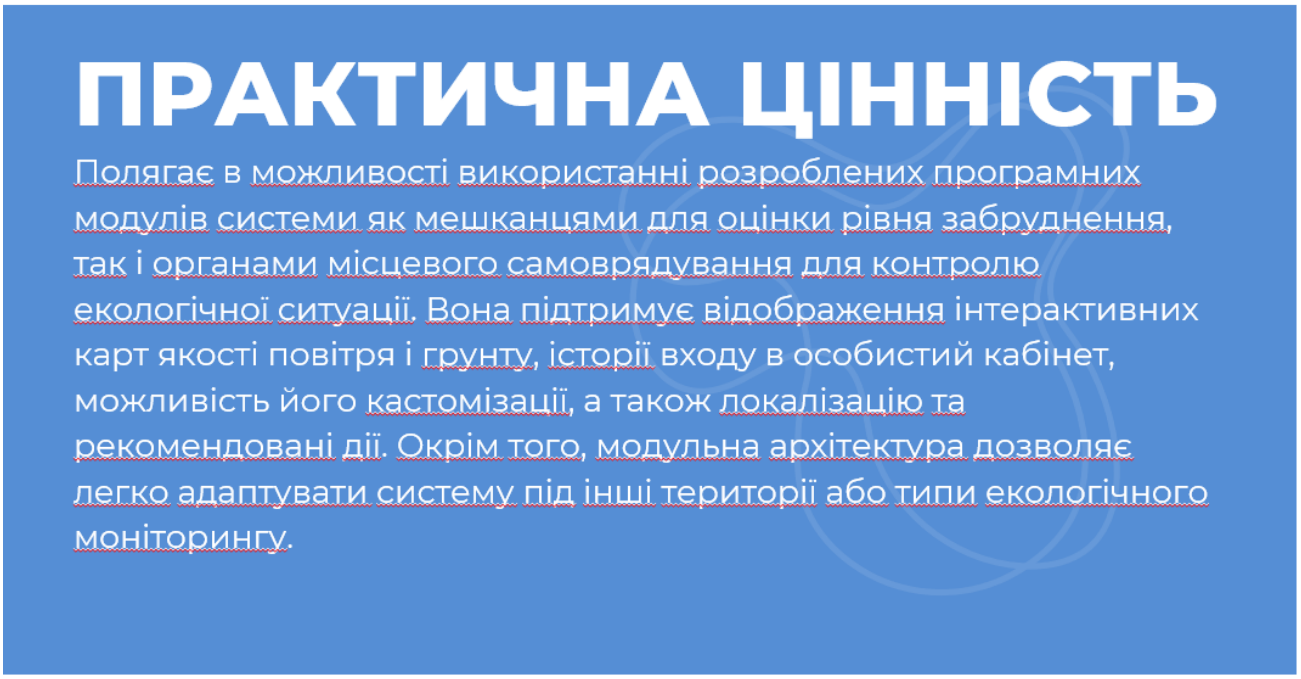


Рисунок Г.7 – Практична цінність

АНАЛІЗ АНАЛОГІВ

Критерій порівняння	AirVisual	Екозагроза	PlumeLabs	Власна розробка
Наявність карти із зоною моніторингу	1	1	1	1
Автоматизоване отримання даних з API	1	0	1	1
Геолокація користувача	1	0	0	1
Функціонал PWA	0	0	0	1
Збереження історії відвідування в особистому кабінеті	0	0	0	1
Сумарний коефіцієнт	**3**	**1**	**2**	**5**

Результати даного етапу дослідження опубліковані в тезах доповіді та апробовані на конференції.

Рисунок Г.8 – Порівняння аналогів

Метод збору екологічного сповіщення

Створення сповіщення дозволяє системі накопичувати локальні дані, пов'язані з довкіллям, а також вести історію активності користувача в особистому кабінеті. Метод збору екологічного сповіщення виконується у декілька послідовних кроків:

1. на сторінці створення сповіщення користувач заповнює форму, що складається з таких полів, як title – короткий заголовок проблеми і description – опис ситуації;
2. після заповнення форми відбувається її перевірка на валідність: усі поля мають бути заповнені, інакше система виводить повідомлення про помилку;
3. якщо форма пройдена успішно, її дані передаються у функцію onSubmit(), яка формує JSON-об'єкт [21] і відправляє POST-запит на бекенд за маршрутом POST /api/reports;
4. на серверній стороні Express.js [22] бекенд:
   - приймає запит;
   - перевіряє авторизацію за токеном;
   - створює новий документ у колекції reports у базі даних MongoDB з усіма даними форми;
   - додає мітку часу та унікальний ідентифікатор.
5. у разі успіху сервер відповідає статусом 200 OK, і користувач отримує повідомлення: «Сповіщення успішно надіслано»;
6. усі надіслані сповіщення автоматично додаються до особистого кабінету користувача у вигляді списку, що завантажується при кожному вході через маршрут GET /api/reports;
7. кожне сповіщення відображається зі статусом, типом, заголовком та кнопкою для детальнішого перегляду або редагування, але це буде у майбутніх оновленнях;
8. для збереження швидкої роботи застосунку, сповіщення тимчасово кешуються у localStorage, а при відновленні підключення – синхронізуються з сервером.

Рисунок Г.9 – Метод збору екологічного сповіщення

Блок-схема алгоритму методу збору екологічного сповіщення

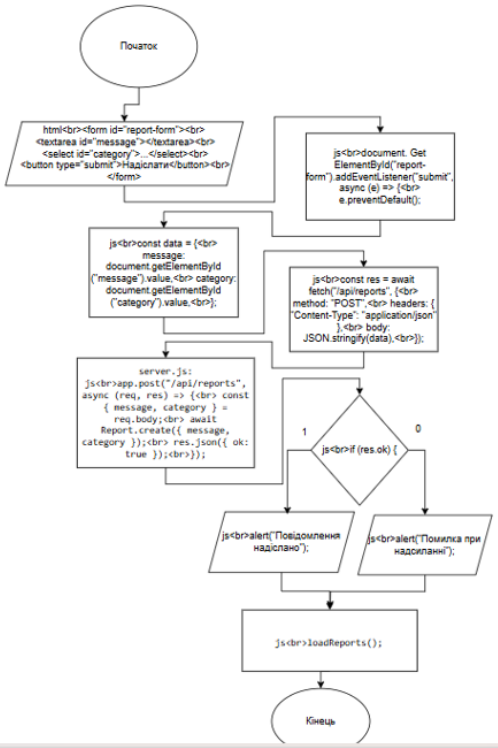


Рисунок Г.10 – Блок – схема алгоритму роботи методу збору екологічного сповіщення

Метод побудови динамічних графіків та візуалізації даних (ч.1)

У вебсистемі екологічного моніторингу реалізовано функціонал динамічної візуалізації екологічних показників за допомогою графіків та інтерактивних елементів, що дозволяє користувачам зручно переглядати динаміку змін якості повітря і ґрунту. Метод побудови динамічних графіків та візуалізації даних складається з таких кроків:

1. збір даних - для г.рафіків, пов'язаних з екологією, дані запитуються через маршрут GET /api/air або /api/soil – залежно від тематики сторінки, дані запитуються з Node.js серверу, який взаємодіє або з MongoDB, або з зовнішнім API, у разі персоналізованої аналітики запит надсилається на GET /api/profile і містить JWT-токен користувача для авторизації;
2. обробка даних – отримані дані перетворюються у формат, придатний для побудови графіка, за допомогою JavaScript, з використанням методів map() та reduce() створюється масив значень та міток часу, які будуть відображені на осі X і Y, при потребі значення округлюються до цілих чисел або форматуються за допомогою Math.round();
3. побудова графіка – за допомогою Chart.js створюється новий екземпляр графіка.

Графік розміщується у відповідному HTML-контейнері - <canvas id="chart">, завдяки опції responsive, графік адаптується до розміру пристрою;

4. візуалізація в особистому кабінеті – у dashboard.html для кожного користувача відображається графік відвідувань або сповіщень, дані беруться з MongoDB через маршрут /api/reports і фільтруються за автором, далі обробляються та подаються у вигляді стовпчикового графіка кількості активностей по днях/тижнях;
5. кешування – завдяки реалізованому service worker та підтримці localStorage, графік може побудуватись без входу в браузер на основі збережених раніше даних.

Рисунок Г.11 – Метод побудови динамічних графіків і візуалізації даних ч.1

Метод побудови динамічних графіків та візуалізації даних (ч.2)

Особливості методу:

- chart.js дозволяє швидко генерувати інтерактивні графіки без складної візуальної логіки;
- дані з API обробляються динамічно у браузері;
- структура адаптована до прогресивного вебзастосунку;
- дані представлені у вигляді JSON, а запити виконуються через fetch() або Axios.

Таким чином, побудова динамічних графіків дає змогу візуально аналізувати екологічну ситуацію, відстежувати зміни за часом, отримувати персональні дані у зручній формі та підтримувати стабільність роботи навіть у нестабільних мережевих умовах.

Рисунок Г.12 – Метод побудови динамічних графіків і візуалізації даних ч.2

Блок-схема алгоритму методу побудови динамічних графіків та візуалізації даних

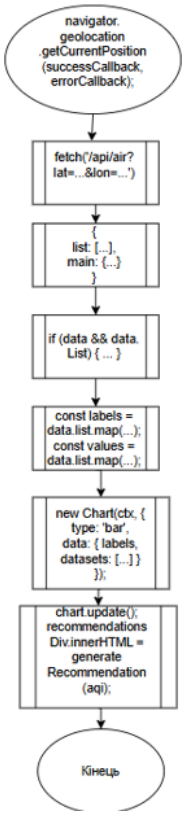


Рисунок Г.13 – Блок-схема алгоритму методу побудови динамічних графіків

МОДЕЛЬ РОБОТИ СИСТЕМИ

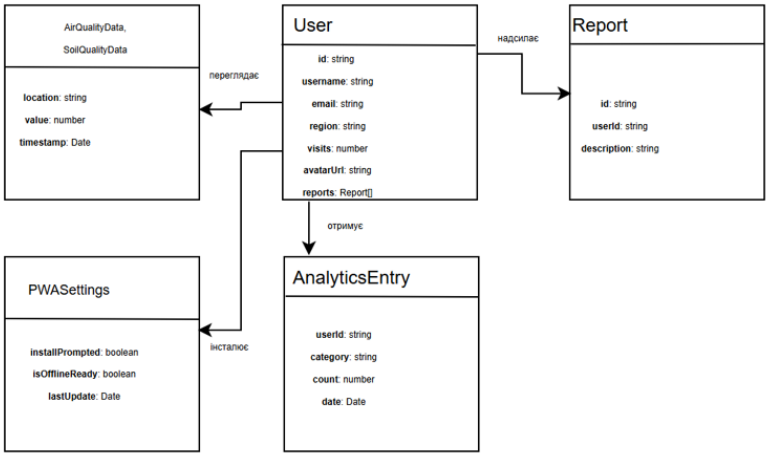


Рисунок Г.14 – Модель роботи системи

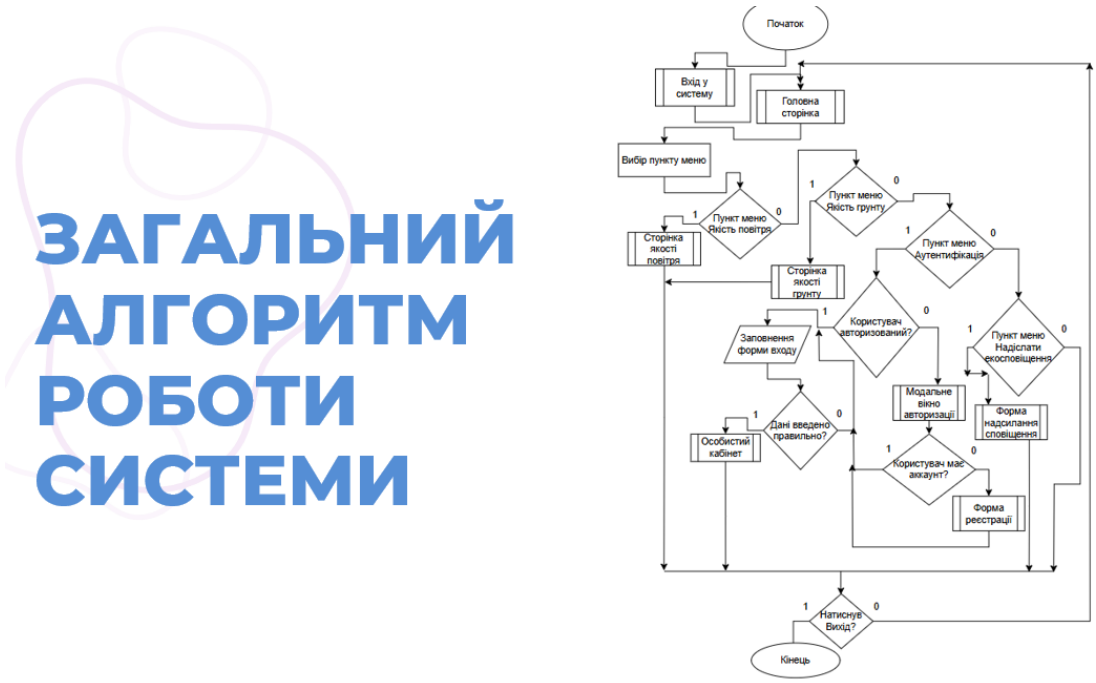


Рисунок Г.15 – Загальний алгоритм роботи системи

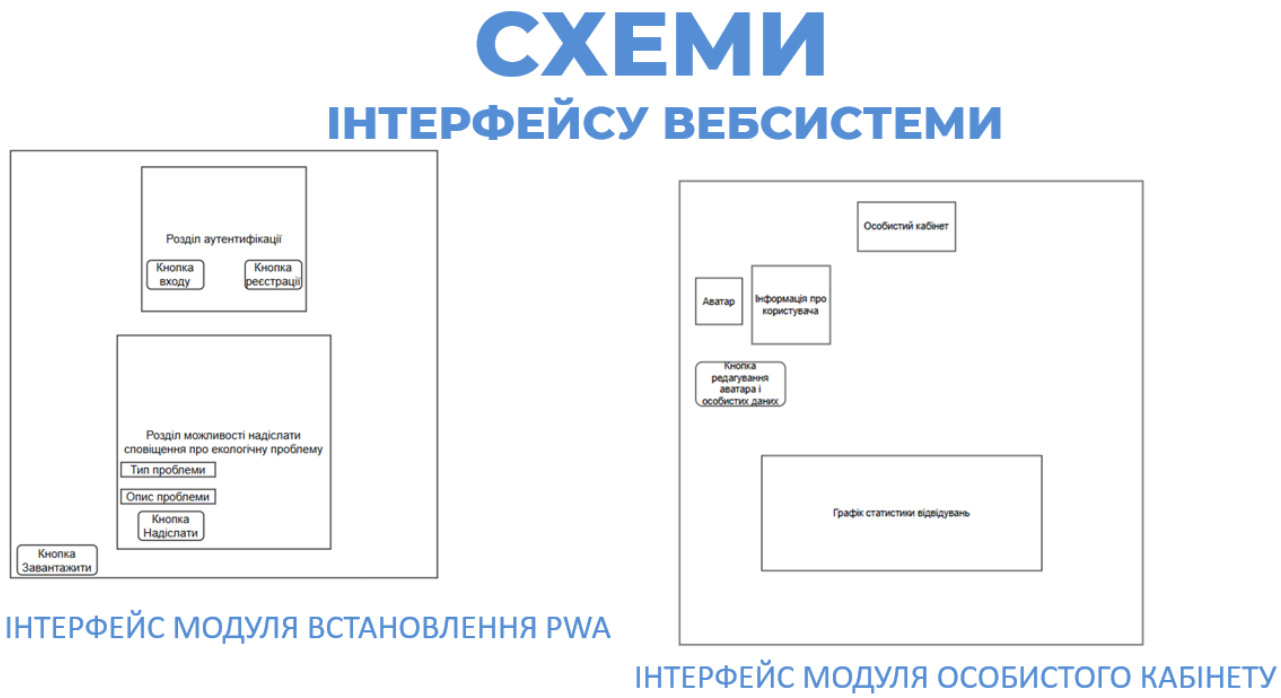


Рисунок Г.16 – Схеми інтерфейсу вебсистеми

Рисунок Г.17 – Використані технології і фреймворки

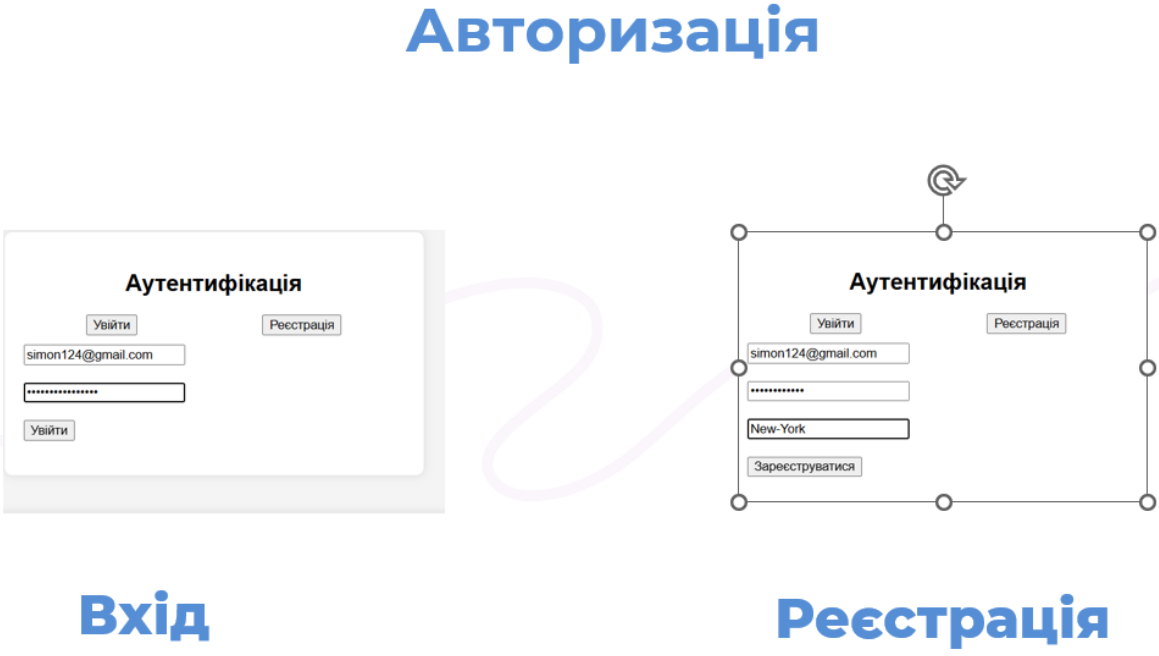


Рисунок Г.18 – Авторизація

Модулі якості повітря і грунту

Модулі оцінки якості повітря та ґрунту є ключовими елементами вебсистеми екологічного моніторингу. Кожен із них реалізований як окрема сторінка з унікальним дизайном, власною логікою завантаження даних, обробкою геолокації, відображенням графіків та рекомендацій. Основною метою модулів є забезпечення користувача актуальною інформацією щодо стану навколишнього середовища у його мікрорайоні в режимі реального часу.

Основні функції модулів описані нижче.

Кожен модуль має однакову загальну структуру:

- інтерактивна карта, що відображає забруднені ділянки;
- геолокаційний запит;
- отримання даних через API;
- візуалізація даних у вигляді стовпчикових графіків через бібліотеку Chart.js;
- панель з рекомендаціями щодо поведінки у разі виявлення забруднення;
- можливість зміни анімованого фону.

Рисунок Г.19 – Модулі якості повітря і грунту

Інтерфейси модулів якості повітря і грунту

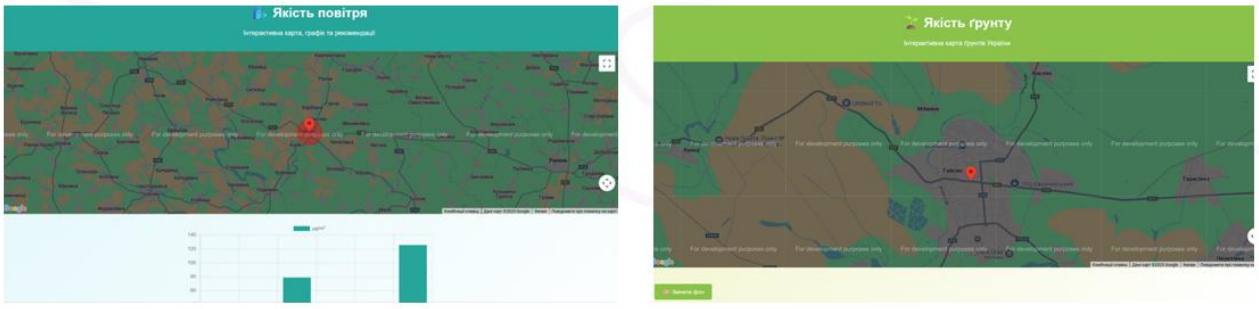


Рисунок Г.20 – Інтерфейси модулів

Модуль інтеграції PWA

Основними компонентами PWA у реалізованому проєкті стали:

– Файл manifest.json, що визначає назву додатка, іконку, кольори теми та режим відображення.

– Скрипт service-worker.js, який обробляє кешування сторінок, ресурсів, зображень та API-відповідей, дозволяючи працювати в екстренному режимі.

– Кнопка встановлення застосунку, реалізована через подію beforeinstallprompt, яка ініціює процес встановлення прогресивного вебдодатка на пристрій користувача.

Інтерфейс користувача також інформує, чи додаток вже встановлено. Якщо PWA вже інстальовано, кнопка змінює вигляд на «✅ Встановлено» і більше не реагує на клікання. У разі деінсталяції застосунку кнопка знову активується. Якщо браузер підтримує PWA, користувач може встановити вебсистему на головний екран пристрою, користуватися нею у повноекранному режимі та без адресного рядка.

У разі, якщо браузер або пристрій не підтримують PWA, наприклад, Safari на iOS, система все одно залишається доступною, але встановлення не пропонується.

Рисунок Г.21 – Модуль інтеграції PWA

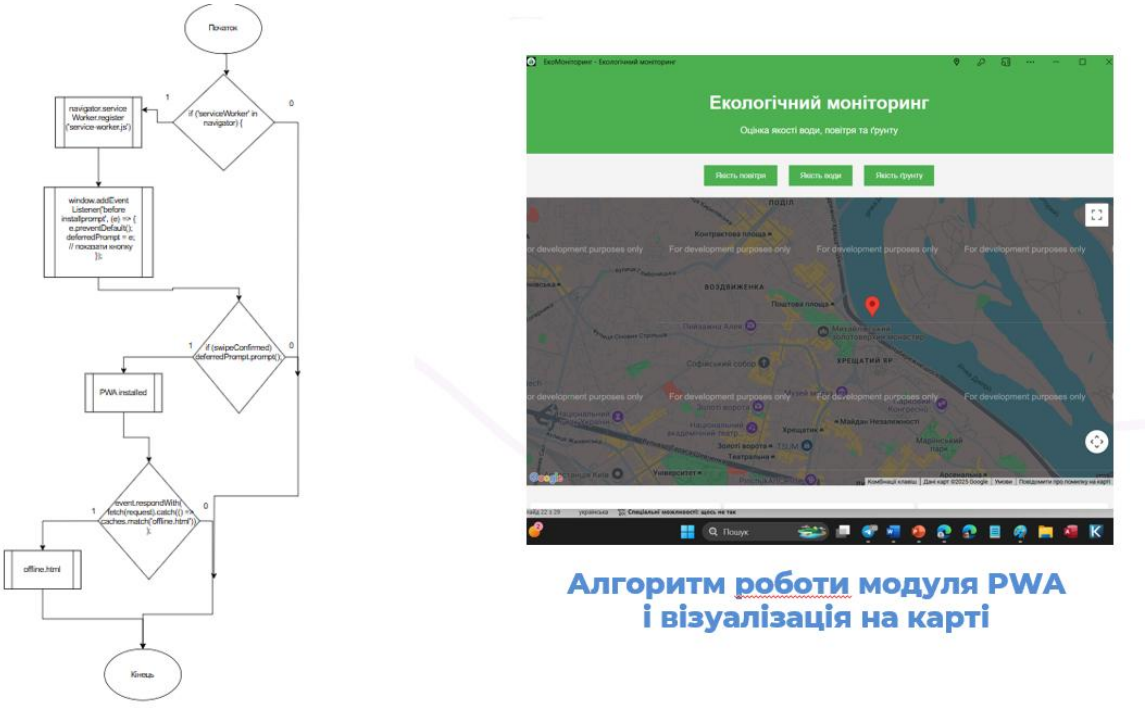


Рисунок Г.22 – Алгоритм роботи і візуалізація модуля на карті

Модуль особистого кабінету користувача

Особистий кабінет є центральним елементом взаємодії користувача з системою. У ньому відображається профіль користувача, його обраний регіон, статистика відвідувань, список надісланих екологічних сповіщень, а також інтерактивний графік та можливість редагування персональних даних.

Модуль було реалізовано на сторінці dashboard.html з використанням HTML, CSS, JavaScript, Chart.js і взаємодії з серверною частиною через REST API. Основна мета — забезпечити інтуїтивний інтерфейс з можливістю редагування і збереження даних у MongoDB. Алгоритм роботи показано на рисунку праворуч

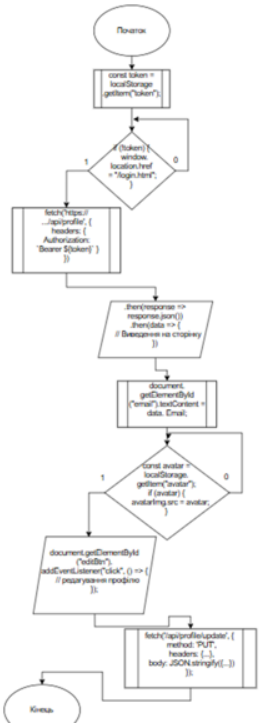
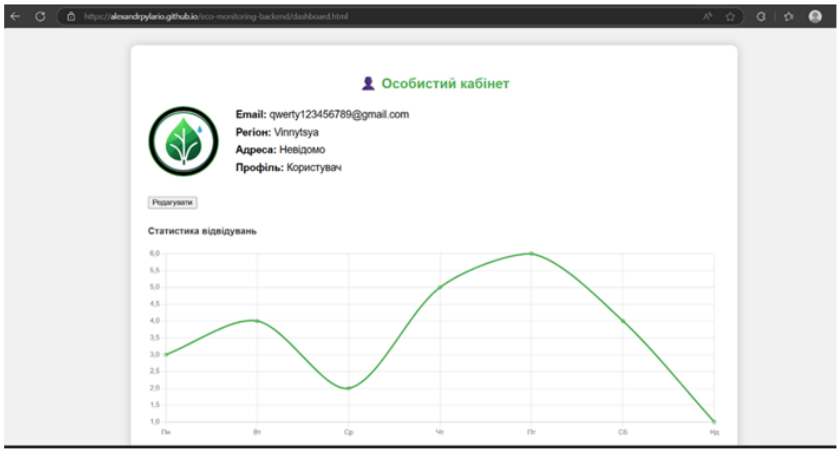


Рисунок Г.23 – Опис і блок-схема роботи модуля особистого кабінету користувача

Модуль особистого кабінету користувача



Інтерфейс особистого кабінету користувача

Рисунок Г.24 – Інтерфейс особистого кабінету користувача

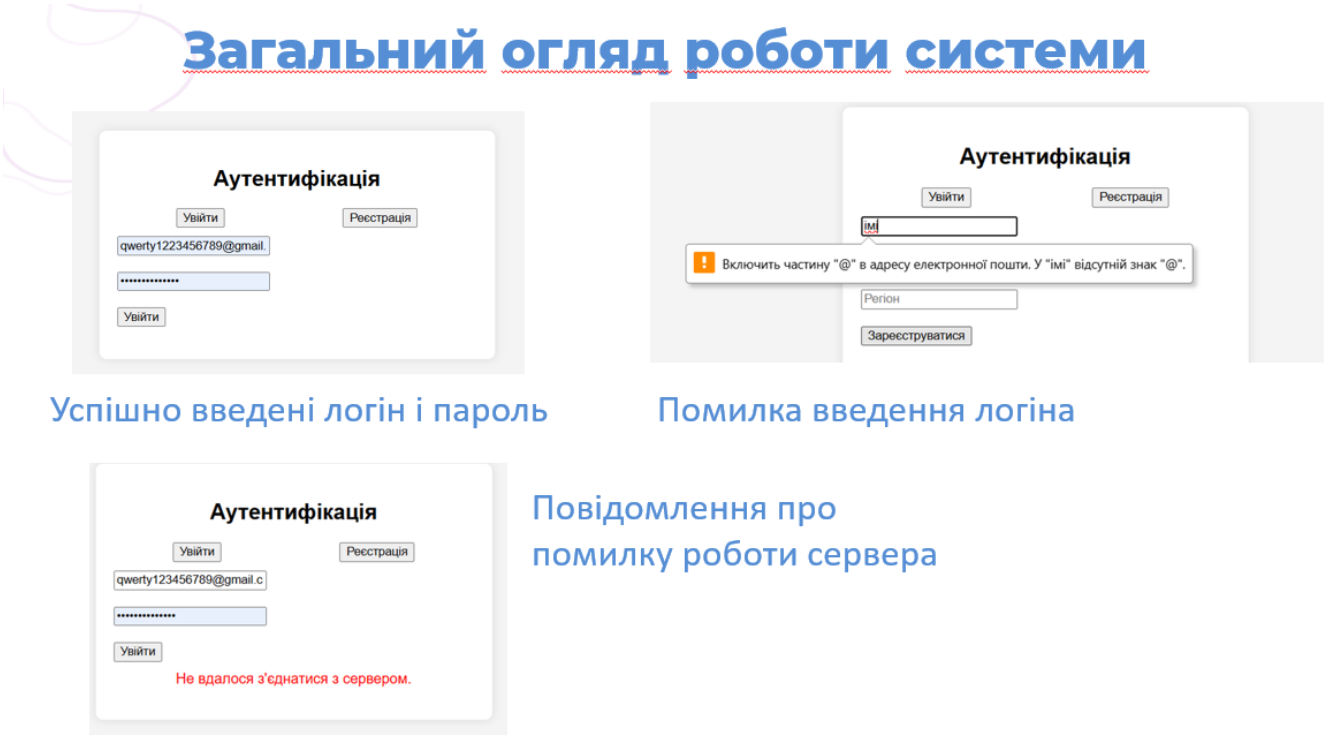


Рисунок Г.25 – Реєстрація в системі

Рисунок Г.26 – Загальний огляд роботи системи

АПРОБАЦІЯ ТА ПУБЛІКАЦІЯ РЕЗУЛЬТАТІВ РОБОТИ

Результати бакалаврської кваліфікаційної роботи обговорювалися на Міжнародній науково-практичній конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2025).

За тематикою дослідження опубліковано 1 наукову працю у збірнику матеріалів Міжнародної науково-практичної конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2025).

Рисунок Г.27 – Апробація та публікація результатів роботи

ВИСНОВКИ

У результаті виконання бакалаврської кваліфікаційної роботи було розроблено програмну систему аналізу й моніторингу екологічної ситуації мікрорайону, яка включає такі функціональні модулі:

- методи та модель роботи системи;
- модулі відображення якості повітря та ґрунту з використанням інтерактивних карт, графіків та рекомендацій;
- особистий кабінет користувача з можливістю редагування профілю, перегляду сповіщень і аналітики;
- систему збору та обробки екологічних сповіщень із збереженням даних у MongoDB;
- механізм побудови динамічних графіків та візуалізації даних з API;
- інтеграцію з PWA та можливості встановлення додатку на пристрій.

Рисунок Г.28 – Висновки