НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО”
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ АТОМНОЇ ТА ТЕПЛОВОЇ ЕНЕРГЕТИКИ
кафедра ЦИФРОВИХ ТЕХНОЛОГІЙ В ЕНЕРГЕТИЦІ

“До захисту допущено”
Завідувач кафедри ЦТЕ
_____ Наталія АУШЕВА

“___”_____2024 р.

Дипломна робота

на здобуття ступеня бакалавра
за освітньо-професійною програмою
“Цифрові технології в енергетиці”
зі спеціальності *122 “Комп’ютерні науки”*

на тему: **Веб-додаток моніторингу якості повітря світу**

Виконав (-ла):
студент (-ка) IV курсу, групи ТР-02

ГОНЧАРУК Вадим Юрійович _____
(прізвище, ім’я, по батькові) (підпис)

Керівник:
доцент, канд. техн. наук Вячеслав ЛІСКІН _____
(посада, вчене звання ,науковий ступінь, ім’я, ПРІЗВИЩЕ) (підпис)

Рецензент: _____
(посада, вчене звання, науковий ступінь, ім’я, ПРІЗВИЩЕ) (підпис)

Н.контроль: *асистент Антон ПАСІЧНЮК* _____
(посада, ім’я, ПРІЗВИЩЕ) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент (-ка) _____
(підпис)

Київ – 2024

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО”

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ АТОМНОЇ ТА ТЕПЛОВОЇ ЕНЕРГЕТИКИ

Кафедра ЦИФРОВИХ ТЕХНОЛОГІЙ В ЕНЕРГЕТИЦІ

Рівень вищої освіти – перший (бакалаврський)
спеціальність 122 “Комп’ютерні науки”
Освітньо-професійна програма “Цифрові технології в енергетиці”

ЗАТВЕРДЖУЮ
Завідувач кафедри ЦТЕ
Наталія АУШЕВА

«__»_____202__р.

ЗАВДАННЯ
на дипломну роботу студенту

ГОНЧАРУКУ Вадиму Юрійовичу

(прізвище, ім’я, по батькові)

1. Тема роботи **Веб-додаток моніторингу якості повітря світу**

Керівник роботи
Ліскін Вячеслав Олегович, канд. техн. наук, доцент

(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «___»_____ 202__р. №_____

2. Термін подання роботи студентом ***07.06.2024року***

3. Вихідні дані до роботи JavaScript, TypeScript, React, Redux, HTML, CSS, Axios, Geolib, Google AQ API, Google Maps API, AQP API, React-type-animation, React-slick, use-places-autocomplete, react-d3-speedometer, react-router-dom.

4. Перелік питань, які потрібно розробити
1) провести аналіз серед наявних аналогів веб-систем для моніторингу якості повітря світу

2) аргументувати вибрані інструменти, технології та алгоритми для реалізації програмного забезпечення
3) побудувати архітектуру програмного забезпечення
4) створити програмний веб-інтерфейс
5) представити процес застосування веб-інтерфейсу

5. Орієнтований перелік ілюстративного матеріалу: рисунки для опису роботи програми, таблиці класифікацій AQI у різних країнах світу.

6. Дата видачі завдання «15» вересня 2023 р.

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів дипломної роботи	Примітка
1	Опрацювання літератури за темою дипломної роботи	05.02.2024	
2	Розробка алгоритму роботи веб-додатку, консультація з керівником	26.02.2024	
3	Розробка дизайну інтерфейсу веб-додатку	14.03.2024	
4	Написання базової розмітки веб-додатку відповідно до розробленого дизайну	24.03.2024	
5	Інтегрування відкритих API у веб-додаток: Google Map API, Air Quality Programmatic API.	12.04.2024	
6	Підбір допоміжних бібліотек для роботи додатку та інтерактивного інтерфейсу	22.04.2024	
7	Рефакторинг, покращення коду	25.04.2024	
8	Ручне тестування веб-додатку, повна перевірка усіх компонентів, усунення помилок	01.05.2024	
9	Пошук додаткової інформації за темою дипломної роботи для створення пояснювальної записки	05.05.2024	
10	Написання та оформлення пояснювальної записки, консультація з керівником дипломної роботи	19.05.2024	

Студент _____ **Вадим ГОНЧАРУК**
(підпис) (ім'я, ПРІЗВИЩЕ)

Керівник роботи _____ **В'ячеслав ЛІСКІН**
(підпис) (ім'я, ПРІЗВИЩЕ)

АНОТАЦІЯ

Дипломна робота виконана на 69 сторінках, містить 8 ілюстрацій, 5 таблиць, 2 формули, 1 додаток та 21 джерело в переліку посилань.

Мета роботи – висвітлювання проблеми забруднення повітря у світі та створення веб-застосунку для моніторингу якості повітря світу.

Методи та засоби: HTML, CSS, JS, TS, React, Redux, Google Maps API, AQP API та всілякі додаткові допоміжні бібліотеки, такі як Geolib, react-d3-speedometer, react-type-animation, react-slick та інші.

Результат – веб-додаток розроблений мовами React/TypeScript для моніторингу якості повітря у світі, що висвітлює підняту проблему. У роботі проведено аналіз теперішньої ситуації із забруднення повітря, досліджено основні забруднювачі та їх джерела, а також знайдено рішення проблеми.

Ключові слова: ПОВІТРЯ, ЗАБРУДНЕННЯ, ЗАБРУДНЮВАЧ, ВЕБ-ДОДАТОК, МОНІТОРИНГ.

ABSTRACT

The thesis consists of 69 pages, contains 8 illustrations, 5 tables, 2 formulas, 1 appendix and 21 sources in the list of references.

The purpose of the work: highlighting the problem of air pollution in the world and creating a web application for monitoring the quality of the world's air.

Methods and tools: HTML, CSS, JS, TS, React, Redux, Google Maps API, AQP API and all kinds of additional support libraries like Geolib, react-d3-speedometer, react-type-animation, react-slick and others.

The result: a web application developed in React/TypeScript languages for global air quality monitoring that addresses the problem raised. The paper analyzes the current situation of air pollution, investigates the main pollutants and their sources, and also finds a solution to the problem.

Key words: AIR, POLLUTION, POLLUTANT, WEB APPLICATION, MONITORING.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

API (application programming interface) – набір чітко визначених методів для взаємодії різних компонентів.

AQHI (Air Quality Health Index) – індекс впливу повітря на здоров’я.

AQI (air quality index) – індекс якості повітря.

CSS (Cascading Style Sheets) – спеціальна мова стилю сторінок, що використовується для опису їхнього зовнішнього вигляду.

HTML – Hyper Text Markup Language.

IT (Information Technology) – інформаційні технології.

JS – JavaScript.

ppb – одиниця вимірювання концентрації, мільярдна частка.

ppm – одиниця вимірювання концентрації, мільйонна частка.

px (Pixel) – одиниця вимірювання у растровій графіці.

SPA (Single Page Application) – веб-застосунок, який вміщується на одній сторінці з метою забезпечити користувачу досвід близький до користування настільною програмою.

TS – TypeScript.

μg/m³ – мікрограм на кубічний метр.

ОС – операційна система.

ВСТУП

За останнє століття людство зробило величезний крок у розвитку, досягши колосальних успіхів у технологіях, які принесли нам немало позитивних змін у повсякденному житті. Але є і зворотній бік монети — ми нищимо нашу планету, її біорізноманіття та екосистеми в небаченому історичному масштабі. Кількість викидів шкідливих речовин у атмосферу загрожують нашому здоров'ю та Землі в цілому.

Для того, аби наочно донести суспільству проблему сьогодення, було прийнято рішення розробити веб-додаток, де кожен охочий міг би побачити рівень забруднення повітря в будь-якій країні світу в даний момент.

Це дослідження має на меті дослідити рівень викидів речовин-забруднювачів у атмосферу, усі тонкощі їх впливу на наше здоров'я та природу планети, способи подолання проблеми і їх інтеграція в повсякдення.

1 ПОСТАНОВКА ЗАДАЧІ

Об’єктом дослідження є розробка веб-додатку моніторингу якості повітря світу, заснованого на сучасних технологіях.

Предметом дослідження є аналіз методів вимірювання та класифікації забруднення повітря, використання технологій React, TypeScript, Redux, CSS, react-router-dom, axios, Google Map API, AQI API для розробки веб-додатку.

Для досягнення мети роботи потрібно вирішити наступні завдання:

– провести огляд існуючих методів вимірювання якості повітря та аналіз існуючих веб-додатків для моніторингу якості повітря світу;

– обґрунтувати та вибрати технології для розробки веб-додатку, що найкращим чином відповідають поставленим вимогам;

– розробити архітектуру веб-додатку та його основні функції, зокрема, інтеграцію з Google Map API для візуалізації даних про якість повітря;

– реалізувати функціональність веб-додатку з використанням обраного стеку технологій, забезпечивши зручний та інформативний інтерфейс для користувачів;

– провести тестування розробленого веб-додатку з метою перевірки його працездатності та відповідності функціональним вимогам.

Метою роботи є розробка ефективного та зручного веб-додатку для моніторингу якості повітря світу, який надасть користувачам актуальну та достовірну інформацію про стан довкілля та сприятиме екологічній свідомості.

Кінцевим результатом дипломної роботи є функціонуючий веб-додаток для моніторингу якості повітря світу, який може бути використаний користувачами для отримання інформації про якість повітря та її вплив на здоров'я та навколишнє середовище.

2 ОГЛЯД ТА АНАЛІЗ ІСНУЮЧИХ СИСТЕМ МОНІТОРИНГУ ЯКОСТІ ПОВІТРЯ

Для того, аби створити зручний додаток для моніторингу якості повітря необхідно провести аналіз та дослідити уже існуючі аналоги нашого застосунку та відмітити їх переваги та недоліки, аби далі врахувати ці зауваження при створенні власного сайту. Потрібно звернути увагу на виміри повітря, які надають аналоги, які речовини забруднювачі вимірюють, де знаходяться станції моніторингу, їх інтерфейс та логіку.

2.1 Вступ до огляду існуючих систем моніторингу якості повітря

Проблема забруднення повітря є однією з найсерйозніших та одночасно найстаріших. Це питання розпочалось з далекої промислової революції у 18 столітті, коли люди почали активно використовувати у виробництві транспорт, робочі машини та механізми. З того часу було багато спадів та загострень забруднення повітря світу, одним з таких прикладів є 1952 році, коли рекордне забруднення повітря призвело до тисяч смертей людей у Лондоні через густий смог. Для сьогодення ця проблема все-ще залишається, прямо зараз у мегаполісах Китаю та Індії через інтенсивний, високий рівень промислової діяльності, використання величезної кількості вугілля в енергетиці, рівень забруднення повітря небезпечний, смертельний для здоров’я. В цілому рівень забруднення повітря у різних країнах можна побачити, наприклад, по графіку (рисунок 2.1), який надає OECD Data[1], на якому відображено рівень PM2,5 у різних країнах.

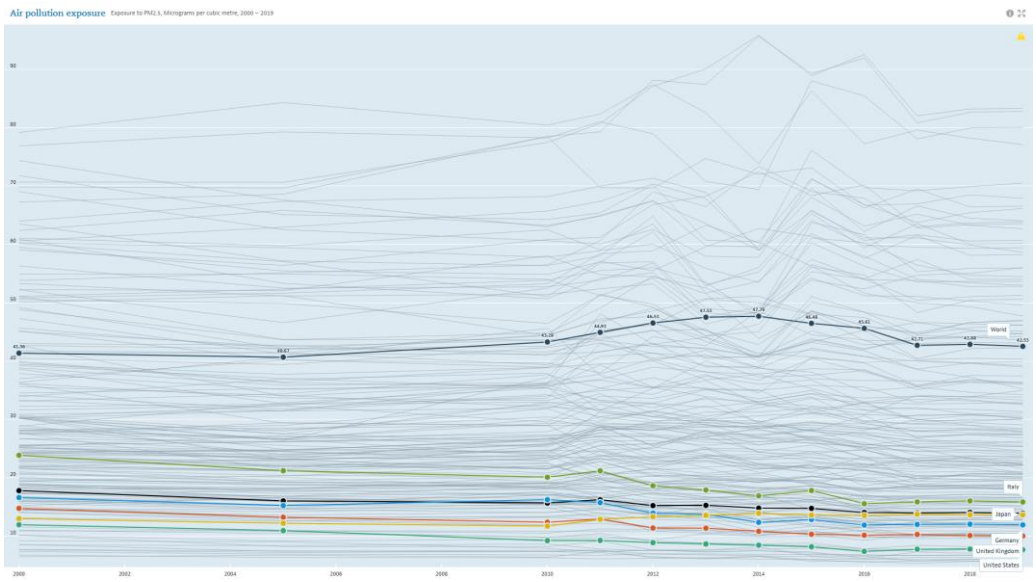


Рисунок 2.1 – PM2,5 μg/m³ у країнах світу, 2000-2019

В середньому для планети цей показних на рівні 42.53 μg/m³, за різними класифікаціями цей рівень відноситься або до задовільної, або до шкідливої для певних груп людей, якщо їх поділити на категорій за здоров'ям.

2.2 Огляд існуючих систем моніторингу якості повітря

SaveEcoBot [2] – українська платформа, що надає можливість досліджувати забруднення повітря в Україні, радіаційний фон, лісні пожежі, напрям та силу вітру. Платформа зручна, але має декілька значних недоліків. Перш за все це те, що побачити забруднення повітря можна лише в Україні, а також є декілька точок у Грузії, Білорусі та Чорногорії. Також не до кінця зрозуміло рівень забруднення, адже платформа надає інформацію лише по твердим частинкам (PM1, PM2.5, PM10).

Eco-city [3] – ще одна українська платформа, яка доволі схожа на попередній аналог SaveEcoBot. Цей проект також розробляє стації моніторингу якості повітря за допомогою робототехніки (Arduino). На сайті можемо побачити карту, на якій відображені точки моніторингу по Україні. Із переваг можна виділити те, що на деяких точках можна побачити рівень забруднення не лише

твердих частинок PM, але й, наприклад, формальдегід (H_2CO), аміак (NH_3), діоксид азоту (NO_2), монооксид вуглецю (CO). Із недоліків – «моніторити» повітря маємо змогу лише по території України.

WAQI.info [4] – найбільша платформа моніторингу якості повітря світу, що також надає API для цього. У проекту сотні партнерів із всього світу. Хоча дизайн веб-додатку трохи застарілий, але технічна частина вражає. Зручна мапа, де одразу видно точки з найбільшим забрудненням, охоплює майже кожну країну, а також надає детальну, розгорнуту інформацію майже про кожну точку. Варто зазначити і те, що на сайті можна детально розібрати ризики для здоров'я відповідно до кожного рівня забруднення.

iQAir [5] – ще одна світова платформа, де будь-хто може побачити актуальну інформацію по забрудненню повітря у будь-якій точці світу. Із переваг саме цього додатку можна віднести гарний дизайн, актуальність даних, активну підтримку, а також те, що ви можете побачити багато додаткової інформації, наприклад, погоду, вологість, напрям та силу вітру, розгорнуту інформацію по кожному забруднювачу. Цей проєкт також надає API, яким може скористатись будь-хто, проте, на превеликий жаль, у безкоштовній версії не дуже багато «ендпоінтів», у платних підписках API є деякі цікавіші можливості, серед них – список-топ по індексу якості повітря по країнах та містах світу.

3 ІНДЕКС ЯКОСТІ ПОВІТРЯ СВІТУ ТА ЗАБРУДНИКИ ПОВІТРЯ

Для подальшої розробки веб-додатку моніторингу якості повітря слід дослідити наявні індекси повітря, їх класифікації, а також виділити основні речовини забруднювачі, які ми будемо отримувати пізніше з API, а також обрати ті класифікації та речовини, які ми будемо показувати користувачу застосунку.

3.1 Індекс якості повітря світу – AQI

AQI (Air quality index) – індекс якості повітря, що показує рівень забруднення атмосфери у конкретний час, у конкретному місці. Чим більший цей індекс – тим більше забруднене повітря, чим менший цей показник – тим повітря частіше.

Аби розрахувати індекс якості повітря – потрібно зібрати немало інформації. Перш за все, слід визначити середні значення концентрації речовин-забруднювачів за певний період. Зазвичай країни та окремі організації слідкують за кількома окремими, головними забруднювачами, а саме: тверді частинки (PM10), дрібні тверді частинки (PM2.5), оксид вуглецю (CO), діоксид сірки (SO_2), діоксид азоту (NO_2), озон (O_3).

AQI обраховується за кусково-лінійною функцією від концентрації забруднювачів. Для перетворення концентрації індексу використовують таке рівняння:

$$I = \frac{I_{high} - I_{low}}{C_{high} - C_{low}} \times (C - C_{low}) + I_{low}\ , \quad (1)$$

де I = індекс якості повітря,

C = концентрація забруднювальної речовини,

C_{low} = концентрація зупинки, $\leq C$,

C_{high} = концентрація зупинки, $\geq C$,

I_{low} = точка зупинки індексу, що відповідає C_{low},

I_{high} = точка зупинки індексу, що відповідає C_{high}.

У таблиці 3.1 наведено граничні точки зупину для кожного з забруднювачів, що використовуються у формулі.

Таблиця 3.1 Таблиця точок зупину [6]

O_3 (ppb)	**O_3 (ppb)**	**$PM_{2.5}$ (µg/m^3)**	**PM_{10} (µg/m^3)**	**CO (ppm)**	**SO_2 (ppb)**	**NO_2 (ppb)**	**AQI**	**AQI**
C_{low} - C_{high} *(avg)*	C_{low} - C_{high} *(avg)*	C_{low}- C_{high} *(avg)*	C_{low} - C_{high} *(avg)*	C_{low} - C_{high} *(avg)*	C_{low} - C_{high} *(avg)*	C_{low} - C_{high} *(avg)*	I_{low} - I_{high}	**Категорія**
0-54 (8-hr)	-	0.0-12.0 (24-hr)	0-54 (24-hr)	0.0-4.4 (8-hr)	0-35 (1-hr)	0-53 (1-hr)	0-50	Добрий
55-70 (8-hr)	-	12.1-35.4 (24-hr)	55-154 (24-hr)	4.5-9.4 (8-hr)	36-75 (1-hr)	54-100 (1-hr)	51-100	Задовільний
71-85 (8-hr)	125-164 (1-hr)	35.5-55.4 (24-hr)	155-254 (24-hr)	9.5-12.4 (8-hr)	76-185 (1-hr)	101-360 (1-hr)	101-150	Шкідливий для групи ризику
86-105 (8-hr)	165-204 (1-hr)	55.5-150.4 (24-hr)	255-354 (24-hr)	12.5-15.4 (8-hr)	186-304 (1-hr)	361-649 (1-hr)	151-200	Шкідливий
106-200 (8-hr)	205-404 (1-hr)	150.5-250.4 (24-hr)	355-424 (24-hr)	15.5-30.4 (8-hr)	305-604 (24-hr)	650-1249 (1-hr)	201-300	Дуже шкідливий
-	405-504 (1-hr)	250.5-350.4 (24-hr)	425-504 (24-hr)	30.5-40.4 (8-hr)	605-804 (24-hr)	1250-1649 (1-hr)	301-400	Небезпечний
-	505-604 (1-hr)	350.5-500.4 (24-hr)	505-604 (24-hr)	40.5-50.4 (8-hr)	805-1004 (24-hr)	1650-2049 (1-hr)	401-500	

Припустимо, що пристрій для вимірювання зафіксував середнє значення концентрації частинок PM10 за 24 години 37 мікрограми на кубічний метр (µg/m^3), отже, можемо вирахувати індекс забруднення повітря підставивши значення у попередньо зазначену формулу:

$$\frac{50-0}{54-0} \times (37-0) + 0 = 34{,}259 \quad (2)$$

Цей результат знаходиться у діапазоні 0-50, що означає «Добре».

3.2 Місцеві класифікації індексу якості повітря

У кожної країни є свої загальні правила по визначенню забруднення повітря та його класифікації. Різниця класифікації полягає зазвичай у використанні різних індексів, наприклад, AQI або AHQI, а також розмежуванні рівнів забруднення та кількості речовин-забруднювачів, десь індекс можуть вимірювати за шістьма речовинами, а десь за вісьма. Давайте розглянемо деякі з них.

3.2.1 Місцевий індекс якості повітря у США

УЗОНС (Управління з охорони навколишнього середовища) США створило окрему, власну класифікацію діапазонів для AQI, що використовується для звітності про якість повітря у країні. Індекс розділено на шестеро категорій (таблиця 3.2), що показують рівень охорони здоров'я жителів США. Індекс оснований на п'яти базових критеріях забруднювачів, які спеціально обрані відповідно до закону Про чистоту повітря США.

Таблиця 3.2 – Категорії індексу повітря США [6]

Індекс якості повітря	**Рівні небезпеки для здоров'я**	**Кольори**
Від 0 до 50	Добрий	Зелений
51 до 100	Помірний	Жовтий
101 до 150	Шкідливий для чутливих груп	Помаранчевий
151 до 200	Шкідливий	Червоний
201 до 300	Дуже шкідливий	Фіолетовий
301 до 500	Небезпечний	Бордовий

Туди входять уже знайомі нам: тверді частинки (PM10, PM2.5), озон (O_3), диоксид азоту (NO_2), діоксид сірки (SO_2), оксид вуглецю (CO). Для кожного з цих небезпечних забруднювачів у країні були встановлені національні норми (стандарти) для охорони здоров'я.

3.2.2 Місцевий індекс якості повітря в Індії

Індійський національний індекс якості повітря запровадив міністр навколишнього середовища 17 вересня 2014 року. Попередньо у країні спеції індекс якості повітря вимірювався не досить точно, адже раніше до уваги бралися лише три показники, а цього, очевидно, мало. Теперішнє вимірювання в рази краще, але до нього додали 5 параметрів, що в сумі дає 8 параметрів-забруднювачів для обчислення індексу. Окрім речовин, які були згадані раніше, Індія моніторить такі речовини, як: аміак (NH_3) та свинець/плюмбум (Pb). Слід також зазначити, що Індія займає одне з перших місць по забрудненню у світі. Жити у містах Індії небезпечно для здоров'я. В Індії існує шість рівнів AQI та охоплює вісім речовин-забруднювачів (таблиця 3.3). Кожен рівень вказує на певні наслідки для здоров'я: «мінімальний вплив» (0-50), «може викликати незначний дискомфорт при диханні у чутливих людей» (51-100), «може спричинити дискомфорт при диханні у людей із захворюваннями легень, таких як астма, а також у людей з серцевими захворюваннями, дітей і літніх людей» (101-200), «може викликати дискомфорт при диханні за тривалого впливу, а також дискомфорт у людей із захворюваннями серця» (201-300), «може викликати респіраторні захворювання у людей при тривалому впливі» (301-400), «може вплинути навіть на здорових людей, і спричинити серйозні наслідки для здоров'я людей із захворюваннями легень чи серця. Негативні наслідки можуть виникнути навіть під час легкої фізичної активності» (401-500). Впровадження індексу якості повітря в Індії є важливим кроком у боротьбі із забрудненням навколишнього середовища. Незважаючи на покращення системи моніторингу, проблема забруднення залишається гострою. Високий рівень забруднення повітря має значний вплив на здоров'я населення, особливо у великих містах, де щільність населення є найвищою. Заходи щодо зниження забруднення, такі як обмеження

використання транспортних засобів на викопному паливі, впровадження більш жорстких стандартів викидів для промислових підприємств і просування зелених технологій, мають стати пріоритетними для індійського уряду.

Таблиця 3.3 – Категорії індексу повітря Індії [6]

Категорія AQI (Діапазон)	**PM_{10}, 24год**	**$PM_{2.5}$, 24год**	**NO_2, 24год**	**O_3, 8год**	**CO, 8год**	**SO_2, 24год**	**NH_3, 24год**	**Pb, 24год**
Хороший (0-50)	0-50	0-30	0-40	0-50	0-1.0	0-40	0-200	0-0.5
Задовільний (51-100)	51-100	31-60	41-80	51-100	1.1-2.0	41-80	201-400	0.5-1.0
Помірно забруднений (101-200)	101-250	61-90	81-180	101-168	2.1-10	81-380	401-800	1.1-2.0
Високий (201-300)	251-350	91-120	181-280	169-208	10-17	381-800	801-1200	2.1-3.0
Дуже високий (301-400)	351-430	121-250	281-400	209-748	17-34	801-1600	1200-1800	3.1-3.5
Небезпечний (401-500)	430+	250+	400+	748+	34+	1600+	1800+	3.5+

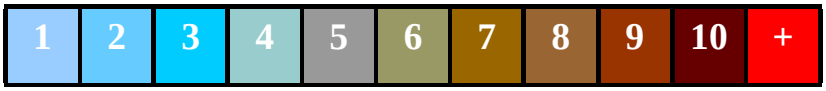
Також важливо підвищувати обізнаність громадськості про небезпеку забруднення повітря та заохочувати людей до дій, спрямованих на поліпшення якості повітря, адже тільки спільними зусиллями можна досягти значного покращення ситуації та забезпечити здорове майбутнє для наступних поколінь.

3.2.3 Місцевий індекс якості повітря у Канаді

Варто згадати країни, які використовують дещо не традиційний індекс якості повітря – не AQI, а так званий AQHI – індекс здоров'я за якість повітря. Це шкала, що дає зрозуміти який вплив на здоров'я дає забруднення атмосфери. Це ніяке не спрощення AQI, а навіть навпаки – на відміну від традиційного індексу, який визначається за максимальним AQI окремого забруднювача, цей визначає

загальний ризик для здоров’я, враховуючи не лише максимальний рівень забруднювача, а повністю усі речовини, для кожної з яких є свій множник. Шкала має 10 рівнів (таблиця 3.4).

Таблиця 3.4 – Категорії індексу повітря Канади [6]



Кожен рівень характеризує ризик небезпеки для здоров’я населення: низький (1–3), середній (4–6), високий (7–10), дуже високий (10+).

3.2.4 Місцевий індекс якості повітря у Великій Британії

Індекс якості повітря у Великій Британії теж представлений десятьма пунктами – це Щоденний Індекс якості повітря, який був розроблений місцевим Комітетом з медичного впливу з забруднювачів повітря. Індекси розподілені у чотири групи ризику (таблиця 3.5) і для кожної є відповідні рекомендації для населення країни. Цей індекс допомагає населенню орієнтуватися у рівні забрудненості повітря та відповідно планувати свою діяльність, зокрема для людей, які страждають на респіраторні та інші захворювання, чутливі до якості повітря.

Таблиця 3.5 – Категорії індексу повітря Великій Британії [6]

Забруднення повітря	Значення
Низький	1-3
Помірний	4-6
Висока	7-9
Дуже Висока	10

Для низького рівня – це «можна вести звичайну активну діяльність на відкритому повітрі», а для дуже високого - це «знизити фізичні навантаження на

свіжому повітрі, особливо якщо ви відчуваєте такі симптоми, як кашель або біль у горлі».

3.3 Основні речовини-забруднювачі повітря

Виходячи з попередніх пунктів можна було зрозуміти, що більшість країн та окремі організації відслідковують декілька конкретних речовин-забруднювачів на основі яких можна вирахувати індекс якості повітря світу (AQI). Чому саме ці речовини, адже є багато інших забруднювачів? Тут все просто – це основні речовини-забруднювачі повітря, які мають найбільш значний вплив на здоров'я людей. Їх вибір ґрунтується саме на їх потенційному впливі на екологію та самопочуття живих організмів. Давайте розберемо окремо кожну із головних речовин, звідки вони беруться, в яких кількостях та яку загрозу собою несуть.

3.3.1 Тверді речовини PM10

PM10 – дуже шкідливі тверді частинки, що знаходяться у повітрі, їх діаметр не перевищує 10 мікрометрів [7]. Їх шкідливість виходить з того, що вони містять фурани, бензопірени, діоксини, а це все – канцерогенні речовини, які викликають ризик утворення ракових клітин в організмі.

Високий рівень PM10 у повітрі негативно впливає на дихальну систему людського організму, він може викликати напади кашлю, хрипіння, а також є особливо небезпечним для людей, що хворіють астмою або ж гострим бронхітом. Також важливо зауважити, що речовина впливає не лише на дихальну систему, але й на увесь організм, збільшуючи ризик виникнення інсульту та інфаркту міокарда.

Ці тверді речовини мають два основних шляхи походження: антропогенні та природні джерела. До антропогенних можна віднести промислові виробництва, дорожні транспортні потоки, домашнє опалення. Це заводи та фабрики, які виділяють значну частку забруднювача у повітря у результаті згоряння палива, обробки матеріалів та інше. Це автомобілі, що викидають велику частину PM10

при русі, коли у них згорає паливо, особливо дизельне. Це й звичайні дрова, газ, вугілля, які ми використовуємо для опалення приміщень та виробництва електроенергії. В свою чергу до природніх джерел можемо відносити звичайний пил землі, що утворюється внаслідок природніх процесів, наприклад, ерозія ґрунту, вітряні бурі, піщані бурі, також це вулканічна діяльність, тобто виверження вулканів, що може викидати в атмосферу пил та інші матеріали, що утворюють тверді частинки PM10, а наостанок це біологічні джерела – рослини та тварини.

3.3.2 Тверді речовини PM2.5

PM2.5 – шкідливі частинки, що в цілому схожі на PM10, але суттєва різниця у їх розмірах [7]. Якщо PM10 має в діаметрі до 10 мікрометрів, то PM2.5 – всього 2.5 мікрометра, а це в свою чергу призводить до того, що ці частинки в рази небезпечніші, адже через їх розмір вони можуть проникати безпосередньо у кров живих істот. Ця тонка природа забруднювача прямо впливає на ризик небезпеки для здоров’я, адже PM2.5 відповідає за такі проблеми зі здоров’ям, як: аритмія, загострення астми, зниження функцій легень, рак гортані, горла, легень, васкуліт, атеросклероз, посилення симптомів захворювань, що пов’язані з дихальною та кровоносною системами. Небезпеку важко переоцінити.

Варто вказати, що є деякі порогові значення кількості PM2.5 у повітрі, і коли цей поріг перевищується – видається сигнал тривоги у багатьох додатках, але в цьому і проблема, адже люди думають, що якщо немає тривоги – їм нічого не загрожую, а це не так, адже якщо PM2.5 є у повітрі, хоча і його концентрації не перевищує того порогу - все рівно впливає на здоров’я, у такому випадку рекомендується носити маски проти смогу. Зараз у світі загальною нормою концентрації твердих частинок є 25 мікрограмів на кубічний метр, а річний – 10 мікрограмів на кубічний метр.

Джерела утворення твердих частинок PM2.5 такі ж самі, як і джерела утворення PM10 – це дим, викиди від згоряння палива, промислова діяльність, автомобілі, вулканічна діяльність, лісові пожежі. Тим не менш, PM2.5 в рази небезпечніші за PM10, тому давно варто пробувати мінімізувати їх утворення, а

розпочати потрібно з себе кожному з нас. Це дуже просто – перестати спалювати сміття та осіннє листя у себе вдома, відмовитись від дров та іншого палива, а перейти на, наприклад, сонячну електроенергію, бути обережними у лісі з вогнем, аби лісних пожеж було менше.

3.3.3 Оксид азоту NO_2

Оксид азоту NO_2 – один з основних забруднювачів повітря, який виникає в наслідок різних процесів і може мати серйозні наслідки для людського здоров'я, а також цілих екосистем [8]. Ця речовина, як і попередні, подразнює дихальні шляхи, а особливо у людей з астмою та іншими захворюваннями дихальних шляхів, що може призвести до погіршення симптомів та загального самопочуття. Довготривалий вплив оксид азоту, на жаль, може призвести до таких хронічних захворювань, як бронхіт, пневмонія, а також серцево-судинних захворювань.

Більшість оксид азоту виробляємо ми – люди. Він утворюється внаслідок спалення палива при високих температурах, наприклад, бензин, дизель, а також різне паливо на промислових об'єктах і навіть звичайне домашнє опалення. Із природніх джерел найбільший вплив дають вулканічні виверження. Неможливо не згадати, що саме NO_2 є ключовим компонентом через яких утворюються кислотні дощі, а також смог. Вони призводять до страшенно-негативних наслідків для рослин, ґрунтових вод та екосистем у цілому. Для запобігання отруєнню нашого повітря цією речовиною слід дотримуватись тих же рекомендацій, що й з попередніми речовинами, а саме впроваджувати заходи, що зможуть мінімізувати викиди транспорту та промисловості, використання більш екологічно чистих джерел енергії. Перше, що приходить на думка – альтернативні методи отримання електроенергії, а також перехід великих світових автомобільних концернів на електрокари, аби зменшити кількість бензинових та дизельних авто, що негативно впливають на екологію планети.

3.3.4 Озон O_3

Озон O_3 – газ блакитного кольору, що має запах так би мовити "свіжості" [9]. Саме він знаходить у верхніх шарах атмосфери та захищає живі організми від

ультрафіолетового випромінювання Сонця, проте цей же газ має зовсім інші властивості у приземних, нижніх шарах атмосфери, тут він стає отруйним для нас. Озон є ще одним з основних забруднювачів повітря на Землі, він утворюється внаслідок хімічних реакції між вуглеводними сполуками та оксидами азоту, все ще, звісно, не береться нізвідки, а з того автотранспорту, промислових установ і так далі. Як і попередні забруднювачі, озон подразнює дихальні шляхи і може призвести до погіршення симптомів астми, а також інших захворювань пов’язаних з дихальними шляхами. У місцях, де концентрація озону вища безпечного рівня – смертність набагато вища через респіраторні захворювання, показують дослідження. Також не слід забувати про шкоду довкіллю, озон несе серйозний негативний вплив рослина та екосистемі, він пошкоджує листя, що призводить до зменшення врожаю та загибелі рослин, це надзвичайно погано впливає на екосистемні баланси та біорізноманіття нашої планети. Із запобіжних заходів знову можна відзначити перехід на більш екологічно чисті джерела енергії та підтримку транспортних мереж із низьким рівнем викидів у атмосферу.

3.3.5 Монооксид вуглецю CO

Вуглекислий газ (CO) є одним з основних забруднювачів повітря [10], який виникає внаслідок неповного згорання вуглеводнів і може мати серйозні наслідки для здоров'я та навколишнього середовища, всі ми не одноразово чули про нього, адже про вуглекислий газ починають активно говорити ще в школі, тому навіть діти уже обізнані про небезпеку цього газу. Моноксид вуглецю виникає внаслідок згорання вуглеводнів, таких як паливо для автомобілів, домашніх опалювальних систем, промислові викиди та інші джерела. Великі концентрації CO спостерігаються в міських районах з великою кількістю автотранспорту та інших джерел викидів. Вуглекислий газ може мати серйозний вплив на здоров'я людей. При вдиханні великих концентрацій CO він конкурує з киснем за зв'язування з гемоглобіном у крові, що може призвести до гіпоксії (недостатнього кисню) та серйозних наслідків, включаючи отруєння, втрату свідомості та смерть. Крім того, викиди CO можуть мати негативний вплив на навколишнє середовище. Вони можуть призводити до забруднення атмосфери та глобального потепління,

сприяючи зміні клімату та екологічних дисбалансів, що в сучасному світі є величезною проблемою, так як наслідки забруднення уже видно, процес глобального потепління запущений, нам потрібно максимізувати зусилля для подолання проблеми.

3.3.6 Діоксид сірки SO_2

Діоксид сірки SO_2 – один з головних забруднювачів повітря, який, як і деякі попередні, виникає внаслідок спалювання нафти, вугілля, промислових процесів та деяких природних джерел [11]. Небезпека для здоров'я така ж, як і в інших забруднювачів повітря – при вдиханні великий концентрацію діоксиду сірки, він може подразнювати легені та шляхи дихання, а довготривалий вплив високих рівнів SO_2 тягне за собою хронічні захворювання та погіршення якості життя у цілому. Екологічний вплив діоксиду сірки теж несе страшні наслідки – це можуть бути кислотні дощі, які негативно впливають на ґрунти, водойми та екосистеми, пошкоджують рослини та все біорізноманіття.

3.3.7 Аміак NH3

Аміак NH3 – один з головних забруднювачів, який вимірюють в багатьох країнах світу [12]. На відміну від попередніх забруднювачів, які виникали в більшій частині через спалювання нафти, вугілля і так далі, головним джерелом аміаку є сільське господарство. Він утворюється через використання азотних добрив, розкладання тваринних відходів та органічних матеріалів у воді та ґрунті, також значна частина аміаку утворюється через відходи великої рогатої худоби, птиці, свиней та інших тварин. В деяких країнах навіть запровадили податок на тварин саме через їх випорожнення, що несуть собою забруднення повітря. Для людини NH_3 має кілька негативних впливів, а саме подразнення очей, шкіри, дихальних шляхів. Контакт з аміаком може викликати негативні ефекти для слизових оболонок, очей та шкіри. А при вдиханні – викликати кашель, хрипи та інші симптоми респіраторних захворювань, високі концентрації речовини можуть викликати серйозні проблеми з диханням.

4 МЕТОДИ ТА ТЕХНОЛОГІЇ СТВОРЕННЯ ВЕБ-ДОДАТКУ

Проаналізувавши обрану задачу, а також провівши огляд можливих готових рішень поставленої задачі, було вирішено створювати додаток на основі веб-технологій. Якщо порівнювати інші можливі варіанти з обраним – веб-застосунок має низку переваг. Основною перевагою є доступність, адже нічого скачувати не потрібно, а також не потрібно мати якусь окрему операційну систему, адже на сайт можна знайти з будь-якого браузера будь-якої ОС.

Способів для створення веб-застосунків у нинішніх реаліях доволі багато, але ми обираємо головних лідерів – HTML, CSS, JavaScript. Звісно ж, що у сучасності ніхто не пише на «нативних», так би мовити «чистих» технологіях, тому ми використаємо один з найпопулярніших фреймворків для створення SPA (Single Page Application) – React, також з ним у комплекті використаємо State Manager під назвою Redux та інші бібліотеки, які нам знадобляться в роботі і, звісно ж, розширення, модифікацію JavaScript (JS) – TypeScript (TS), який нині є навіть популярнішим, ніж, власне, сам JS.

4.1 Мова розмітки HTML

HTML (HyperText Markup Language) – мова розмітки веб-сторінок у браузері [13]. Сервер передає документ HTML за протоколами HTTP/HTTPS браузеру, а той в свою чергу «читає» їх та перетворює код в інтерфейс, який ви бачите на веб сторінці. Кожен елемент в HTML є своєрідною цеглиною з яких будується сторінка – багатоповерхівка. Цими цеглинами є теги, наприклад, , <section>, <h1> і так далі, кожен з них несе собою якийсь семантичний зміст та розмітку. HTML напряму взаємодіє з мовою каскадних стилів CSS, вона надає змогу оформлювати кожну цеглину по-своєму, створюючи красивий інтерфейс. Мова розмітки також надає багато інтерактивних елементів, наприклад <input/>, <button> та інші. Вони дають змогу взаємодіяти з користувачем – загрузити

картинку, ввести свої дані, відправити їх. Усі ці елементи можна відслідковувати за допомогою JS і, відповідно, якось обробляти їх.

4.2 Мова стилю сторінок CSS

CSS (Cascading Style Sheets) – спеціальна мова для стилізації веб-сторінок, вона працює в парі з HTML [14], стилізуючи його окремі елементи. Ця мова є однією з основних технологією павутини інтернету поряд з JS та HTML. Цей інструмент надає нам майже безграничні можливості в стилізуванні, будь-який дизайн, який ви розробите можна створити завдяки правильній комбінації застосованих правил CSS, при цьому мова доволі легка у вивченні і всі її базові правила можна освоїти навіть за тиждень, якщо не поглинатись дуже глибоко. Крім того завдяки таблиці каскадних стилів можна зробити інтерактивні елементи та різні анімації, а також адаптивний або «резиновий» інтерфейс під будь-який пристрій. Кожне правило ми можемо застосувати по-різному, наприклад, аби для усіх тегів <p> задати розмір шрифту в 15 пікселів, ми правило пишемо в CSS файлі, який підключили до HTML документу – «p {font-size: 15px}». В разі, якщо нам потрібно задати якийсь стиль для конкретного тегу <p>, ми можемо надати якийсь «клас» для цього тегу, назвавши його, наприклад, «text», а в CSS пишемо – «.text {color: red}», тобто ставимо крапу і пишемо назву класу, а далі надаємо якийсь набір правил. Також є більш значимі селектори, типу ID. Це те ж саме що й клас, але має більшу значимість, тобто, якщо для елемента задати і клас, і ID, то до елемента в першу чергу будуть застосовуватись саме правила з ID, а уже до них додаватись правила з класу. Також правила можна записати абсолютно до всіх елементів сторінки написавши «*». Є ще й так звані «псевдо-класи» - дуже корисна річ. Наприклад, в тезі <p> ми хотіли б зробити першу букву великою і червоною. Для цього потрібно просто написати «p:first-letter{text-transform: uppercase; color: red}». Назвати усі можливості CSS важко, але деякі з них ми розглянули.

4.3 Мова програмування JavaScript

JavaScript – одна з найпопулярніших мов програмування сьогодення [15]. Її називають динамічною, об’єктно-орієнтованою прототипною мовою. Найчастіше використовується для створення різних сценаріїв веб-застосунків, надання користувачу можливості взаємодіяти з сайтом. Також її використовують для створення односторінкових (SPA) сайтів, для цього застосовують фреймворки React, Vue.js, Svelte, Angular, нею програмують і серверну частину (Node js), роблять мобільні додатки (React Native або Cordova) та багато іншого. Варто зазначити, що ця мова дуже вплинула на створення та розвиток інших мов програмування. З «родзинок» цієї мови можна виділити динамічну типізацію, тобто вам не потрібно конкретно вказувати що буде знаходитись у створеній змінній – String чи Number, це неважливо, адже мова сама підбере правильний тип в залежності від того, що ви в неї положите, це дуже зручно, хоча є багато скептиків, які таке не вітають. Також використання разом з HMTL – ще одна «родзинка» цієї мови, адже ви можете створювати інтерактивні інтерфейси сайтів, міняти сторінку «на ходу», а також код виконується прямо в браузері, більше того – JS це по-суті є основною мовою браузерів і є прямо вбудованим у них. У мові є все, що потрібно для розробки будь-чого, багато уже влаштованих функцій з «коробки», а чого немає – легко написати самому або отримати зі сторонніх бібліотек.

4.4 Бібліотека React

React – це потужна бібліотека (яку також часто називають фреймворком, хоча на офіційних сторінках вона позначена саме бібліотекою) [16], яку створили та підтримують розробники з компанії Meta (Facebook у минулому), вона спеціалізується на створенні інтерфейсів для користувача, ця бібліотека вирішує проблему оновлення сторінок та покликана створювати SPA сторінки. За допомогою неї можна створювати величезні веб-додатки, які можуть міняти свій

вміст без додаткових перезавантажень сторінки або отримання інших HTML файлів. Мета React лежить у його назві – бути «реактивним», тобто дуже швидким, оптимізованим, легким та масштабованим. І це йому добре виходить, адже він застосовує так званий «віртуальний DOM (Document Object Model)», який копіює звичайний DOM сторінки, а потім на основі віртуально починає шукати відмінності, які потрібно змінити у справжньому DOM і лише тоді міняє саме їх, окремі маленькі елементи, дозволяючи ефективно оновлювати сторінку. Також React вводить нове поняття – JSX. Це компоненти – цеглини з яких складається сторінка, ці цеглини можна перевикористовувати та розширювати, а саме JSX розширення дає нам змогу об’єднати HTML з JavaScript в одному файлі, що є надзвичайно зручним інструментом. Слід також згадати «Методи життєвого циклу» компонентів – це спеціальні методи, які вбудовує React, вона надають розробнику змогу обробляти дані та динамічно змінювати інтерфейс відносно них або відслідковувати якісь події, наприклад, коли компонент відрендерився на сторінці або коли навпаки – пропав зі сторінки, ці події можна відслідкувати і написати якусь логіку, яку варто виконати при цих сценаріях. Отже, React – це потужний інструмент, що дає змогу нам об’єднати HTML та JS в одне ціле за допомогою JSX розширення та зручну керувати даними з якими ці компоненти взаємодіють, викликати «ререндери» та інші функції для відображення актуальних даних на SPA сторінці.

4.5 Допоміжні інструменти

4.5.1 Мова програмування TypeScript

TypeScript – це розширена версія JavaScript, що має декілька значних переваг над мовою оригіналом [17], а саме – можливість визначення типів змінних або ж «статична типізація», використання повноцінних класів, можливість підключення модулів. Також за допомогою цих нововведень, як наслідок, з’являються інші переваги, такі як: легка читаємість коду, полегшення пошуку помилок на етапі розробки проекту та рефакторинг, пришвидшення та

оптимізація компіляції, а також сама швидкодія застосунків, що написані на ньому. Із плюсів також є те, що будь-який звичайний JavaScript код є сумісним з TS, тобто раніше написанні бібліотеки або фреймворки на JS будуть підтримуватись у TypeScript проєктах, хоча буде бажано типізувати усе, адже якщо цього не робити – втрачається сенс мови.

4.5.2 Бібліотека Redux

Redux – бібліотека, що майже завжди встановлюється у парі з React [18], її називають «state manager» і цими словами легко її описати, адже виконує вона саме таку роль – покращує керування станам React. За допомогою неї можна створити окремий контейнер станів (Store) для застосунку, він допомагає оптимізувати код. Головною перевагою цього «Store» є те, що нам більше не потрібно «перекидувати» state за допомогою Props через безліч компонентів, якщо нам потрібний цей стейт у двох далеких одне від одного компонентах, через що приходилось піднімати цей стан до найближчого спільного родича цих компонентів. За допомогою Redux ми можемо отримати стейт у будь-якій точці проекту за допомогою кількох простих кроків, що вирішує проблему постійних «перекидань».

4.5.3 Бібліотека Axios

Axios – це бібліотека, яка спрощує створення асинхронних запитів на сервер, вона надає API на основі Promise з JavaScript для асинхронних HTTP-запитів та відповідей [19], бібліотека пропонує широкі можливості для виявлення та обробки помилок, у ній також влаштований захист, який дає нам, як користувачам, змогу додати додаткову автентифікацію в запити.

4.5.4 Google Maps API

Google Maps API – зручне API для розробників, яке надає можливість вбудовувати у свої додатки карти Google [20] з безліччю різного функціоналу, наприклад, з такого, що нам знадобиться це – маркери, які можна ставити на карті, позначаючи ними певні об’єкти, полілінії – лінії, якими можна, наприклад,

з’єднати два маркери або виділити якусь площа на карті, відслідковування координат – маємо змогу при натисканні на будь-яку точку на карті отримати точні координати точки, назву країни, регіону, міста і так далі. Весь функціонал цих карт неможливо назвати, тут є всілякі функції під будь-який проект, я перезвав лише те, що буде використовуватись у проекті дипломної роботи. Також треба згадати про той факт, що це API є наполовину безкоштовним, тобто ви можете використовувати його без оплати, роблячи обмежену кількість запитів на їх сервер, а також не використовуючи карти в комерційних проектах, а навіть якщо ви вийшли за норму запитів на день для безкоштовного користувача – компанія Google покриває перші витрати (до $200) замість вас, API ключ можна отримати безкоштовно у кабінеті розробника Google. Для Google Maps API створено безліч різних налаштувань та інших API, які дають змогу відобразити погоду на карті, дороги і так далі.

4.5.5 Air Quality Programmatic API

Air Quality Programmatic API – це API, що дозволяє нам отримувати актуальні дані забруднення повітря у світі з точок моніторингу проєкту [21]. Ми можемо звернутись до нього, передавши геолокацію – довгото та широту точки, інформацію по вашому IP розташуванню або ж передавши назву міста, або назву конкретної станції збору забруднювачів повітря. У відповідь ми отримаємо широку інформацію у JSON вигляді, а саме – індекс станції моніторингу, AQI – індекс якості повітря, час збору, назва міста та локації, де знаходиться ця станція, також AQI по кожному окремому забруднювачу, який збирає ця станція, більше того – можна отримати навіть передбачення по можливому забрудненню у майбутньому у обраній місцевості. Головна переваг цього API – воно повністю безкоштовне, а кількість запитів, яку ми можемо виконати просто вражає – 1000 запит на секунду, для безкоштовного API така квота це рідкість. Із мінусів сервісу – немає підтримки по всім країнам, проте велика частина охоплюється, а проект інформує, що в багатьох країнах станції моніторингу скоро з’являться, в тому числі Україна.

4.5.6 Бібліотека Geolib

Geolib – невелика проста бібліотека, яка надає нам функції базових геопросторових операції, таких як перетворення десяткових координат у шістдесятичні та навпаки, обрахування відстані між двома точками (що нам і знадобиться у роботі) та всілякі інші корисні функції.

4.5.7 Бібліотека React-type-animation

React-type-animation – маленька бібліотека, яка дає змогу додати гарну анімацію друку. Її використовуємо лише для інтерфейсу «домашньої» сторінки, адже аби зацікавити користувача у проекті – потрібна гарна обкладинка, тому що більшість людей судять «по обгортці».

4.5.8 Бібліотека React-slick / Slick-carousel

React-slick – бібліотека, яка надає анімацію гортання картинок (слайдер). В ній є безліч корисних налаштувань – слайдер може відображати одразу багато фото, прогортати багато фото, слайдер може бути як ручний, так і автоматичний, підтримує горизонтальний та вертикальний напрямок. Його теж використаємо для «домашньої» сторінки, аби вона приваблювала користувача.

4.5.9 Бібліотека use-places-autocomplete

Use-places-autocomplete – бібліотека, яка надає нам однойменний хук для робити з Google Maps, а саме допомагає розробити поле вводу, при введенні в яке користувач зможе отримати місця з гугл мапи, хук як би «доповнює» нас і закінчує за нас. Його використовуємо для того, аби користувач не шукав в ручну конкретне місце на карті, а міг просто ввести його назву і якщо таке місце є – отримуємо координати цього місця, а відповідно і забруднення повітря у цій точці.

4.5.10 Бібліотека react-d3-speedometer

React-d3-speedometer – невеличка зручна бібліотека для відображення даних, яка надає інтерфейс спідометра в який ми можемо передати поділки,

підписати їх, а також передати значення для стрілочки. У нашому випадку будемо передавати індекс якості повітря певної точки, а поділки на шкалі та їх кольори будуть відповідати таблиці розподілу AQI за рівнем небезпеки для здоров'я – від «добре» до «небезпечно». Тобто ця бібліотека теж слугує помічником у створенні красивого інтерфейсу, а відповідно кращого, зручного сприйняття користувачем і його враженням від додатку в цілому.

4.5.11 Google Air Quality API

Google Air Quality API – API від Google, що надає можливість отримати забруднення повітря у обраній точці по координатам або назві міста. У відповідь від API ми можемо отримати розгорнуті дані, до того ж ми самі маємо змогу вказати які саме дані ми хочемо отримати, для цього є спеціальні налаштування запиту, який ми виконуємо. Із плюсів можна виділити велике охоплення країн світу, розгорнуті та повні дані. Із мінусів – не зовсім зрозуміло звідки ці дані API отримує, адже воно передає дані по будь-якій точці, навіть в якомусь безкрайому полі ці дані є, а це означає, що API власноруч обраховує/передбачує дані на основі даних з найближчої точки моніторингу, що робить ці дані не до кінця вірними. Це API ми будемо використовувати виключно для України, тому що попереднє API, що надає якість повітря, не охоплює Україну.

4.5.12 Бібліотека React-router-dom

Ця бібліотекою є основною бібліотекою для роботи з шляхами (Route) у React. Вона є базовою і йде в комплекті, коли ми створюємо React проєкт за допомогою create-react-app. Вона надає потужні інструменти для «роутингу» веб-застосунку, наприклад, компонент <Route>, який дозволяє нам вказувати певний шлях та компонент, який буде відображатись на сторінці при ньому. Також хук, який надає змогу нам змінювати самому шлях при натисканні на кнопку або ще якихсь діях користувача. Або ж компонент <Link>, де непотрібно самому обробляти якісь події, а треба просто передати туди шлях, і при натисканні на цей компонент – шлях зміниться на вказаний.

5 ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

Перейдемо до створення веб-додатку моніторингу якості повітря світу, усі інструменти, технології, бібліотеки та API для розробки уже обрані та описані у четвертому розділі. Тепер потрібно їх грамотно об’єднати у середовищі Visual Studio Code та описати кожен етап розробки та функціонал функцій та усіх компонентів, які будуть створені під час роботи.

5.1 Створення проєкту React.js

У попередньому розділі було обрано бібліотеку React.js для розробки веб-застосунку. Це дуже зручний та потужний інструмент для створення проєкту, який ми обрали. Звісно ж встановлювати вручну усі залежності для роботи бібліотеки надто складно і довго, тому є готове стандартне рішення, а саме – create-react-app. Він дозволяє однією командою в терміналі створити уже готовий макет початкового проекту з основною структурою і дає змогу розпочати нашу розробку. Для цього потрібно виконати ряд послідовних команд, а саме:

– npx create-react-app World-air-quality-monitoring --template typescript;

– cd ./World-air-quality-monitoring;

– npm start.

Перша команда створює у обраній папці нову папку з React проектом, який має усі «дефолтні» налаштування та файли, варто зауважити, що у команді ми також використовуємо «флажок» --template typescript, адже ми заранння знаємо, що буде використовувати TS, пишучи веб-додаток, це доповнення відразу додає типізацію для всіх React файлів, а також змінює «дефолтні» JSX файли на TSX (те саме, що й JSX, але на TypeScript), а JS файли на TS файли. Другою командою ми змінюємо обрану папку в терміналі, відкриваючи папку з проєктом, яку ми створили попередньою командою. Третя вказівка безпосередньо запускає наш проект, запускає сервер розробки, який дозволяє нам бачити нашу роботу. Аби це побачити, нам всього потрібно відкрити браузер та ввести http://localhost:3000/ в адресну стрічку (це адреса за «дефолтом», але якщо на цій адресі у вас уже щось

запущено, то скоріше всього ваш IDE про це повідомить вас і запропонує змінити порт з 3000 на 3001, наприклад). Готово, ви можете розпочати розробку. Далі нам потрібно встановити усі залежності, які потрібні для роботи. Цими залежностями є саме ті інструменти для розробки, які були описані у розділі 4 – це бібліотеки. Аби їх завантажити у наш проект потрібно перейти на сервіс npm (node packet manager), знайти там потрібну нам бібліотеку, скопіювати команди для встановлення, вставити у термінал та, власне, виконати їх. Коли ми виконуємо команду – усі необхідні файли самі скачуються та встановлюються у папки проєкту. Часто буває ситуація, коли залежності теж потребують якихсь залежностей і якщо їх немає – вони просять вас їх встановити, тому що без них вони працювати не будуть. Також є інша підступна ситуація, наприклад, коли різні залежності використовують іншу спільну залежність, але через те, що вони були розроблені у різний час, вони просять різні версії цієї залежності. Зазвичай в такій ситуації потрібно просто додати «флажок» -force до команди інсталяції, він «насильно» встановить усі потрібні пакети, не дивлячись на версії потрібних залежностей, це працює через те, що майже завжди нові версії бібліотек, які виходять – приносять з собою нові можливості, а не забирають старі, тому випадків, коли це не працювало, я поки не зустрічав. Отже, аби встановити усі наші бібліотеки, потрібно виконати ряд команд у терміналі:

– npm i geolib

– npm i use-places-autocomplete

– npm i redux

– npm i axios

– npm i @react-google-maps/api

– npm i @types/google.maps

– npm i react-d3-speedometer

– npm i react-slick

– npm install slick-carousel

– npm i react-type-animation

– npm i react-router-dom

Після виконання усіх цих команд, слід перевірити, чи вони встановились та потрапили у список залежностей, для цього потрібно перевірити package.json файл. Там зберігають назви встановлених пакетів, а також їх версії. Якщо у файлі наявні усі встановлені залежності – можемо розпочинати роботу.

5.2 Файлова структура веб-застосунку

Отже, враховуючи базовий файлову структуру після створення React.js проекту за допомогою create-react-app нам потрібно видалити зайві файли з проекту, а також додати .git, аби проєкт зберігався у репозиторії на GitHub. Наша структура буде мати приблизно такий вигляд: папки node_modules та public (у яких розташовані усі файли залежностей, базовий HTML файл документу, у якому будуть рендеритись наші компоненти), файл package.json (у ньому знаходиться увесь список залежностей, які ми попередньо встановили), файл .env (файл середовища, у який можна виносити певні зміни, у нашому випадку це будуть API keys – ключі доступу до Google Maps API, Google Air Quality AOI та Air Quality Programmatic API), файл .git та .gitignore (в цей файл обов’язково потрібно внести .env файл, аби наші API ключі не потрапили у відкритий доступ, якщо репозиторій публічний). Також тут буде знаходитись папка src у якій ми й будемо працювати більшу частину часу. У ній буде ряд папок, а саме: components – для наших React-компонентів, fonts – для завантажених шрифтів, images – для картинок, які будуть використовуватись у компонентах або CSS файлах, redux – окрема директорія для Redux, у якій буде зберігатись наш головний, спільний для усього проєкту store, а також дві папки actions та reducers, де будуть створюватись відповідні файли для кожного окремого стану, utils – окрема папка для допоміжних функцій. Також в папці src будуть базові загальні файли, такі як: index.ts, index.css, App.ts, App.css, Globals.d.ts.

5.3 Компоненти та функції веб-застосунку

У цьому підрозділі буде описаний кожен створений файл у src папці, це функціональні TSX компоненти, CSS-модуль файли, допоміжні utils-функції, функції actions та reducers. Розглянемо логіку кожного файлу, як вони працюють і за що відповідають.

5.3.1 Компонент Button

Розпочнемо розробку, що завчасно створимо компонент з кнопкою, яку ми зможемо перевикористовувати будь-де у проєкті. Компонент максимально простий – він отримує всередину себе title та onClick – це вхідні дані, які будуть змінювати компонент, title – назва кнопки, яка буде відображатись всередині, а onClick – це функція, яка власне буде виконуватись при натисканні на кнопку. Компонент повертає тег <button>. До компоненту також створюємо Button.module.css – модульний файл CSS з стилями для кнопки. У всьому проєкті будемо використовувати саме модульні файли стилів, завдяки ним можна не перейматись через те, що якісь із назв селекторів будуть повторюватись та перетинатись з іншими селекторами по усій сторінці і перекривати одне одного. Модульний файл стилю автоматично додає унікальний ідентифікатор

5.3.2 Компонент Header

Оскільки на будь-якій сторінці нашого застосунку ми будемо бачити шапку сторінки, потрібно одразу її створити. Цей компонент буде повертати тег <header> у якому будуть знаходитись ще два компоненти, а саме назва-логотип застосунку, а також компонент Navigation, де, відповідно, знаходиться навігація по сторінці. Цей компонент повертає семантичний тег <nav>, що явно вказує браузеру та пошуковим системам, що в цьому блоці знаходиться навігація. Попередньо створюємо компонент <NavOption>, який буде приймати path та label – тобто шлях і назву навігаційної опції. Компонент буде повертати тег <Link> - це засіб, який надається бібліотекою react-router-dom, дозволяє міняти веб-шлях при натисканні. Також для Link використаємо хук useLocation з тої ж бібліотеки, він

дає змогу відслідковувати теперішній шлях, що введений у адресному рядку. Коли шлях, який зараз введено співпадає із шляхом, який ми передали в компонент – ми додаємо додаткові стилі CSS для компонента, аби відповідна кнопка підсвічувалась, якщо ми зараз на сторінці, на яку ця кнопка веде. Всього кнопок поки що в навігації 3 – це кнопки «Головна», «Карта», «Забруднювачі», які при натисканні міняють теперішній шлях на «/home», «/map», «/pollutants» відповідно.

5.3.3 Компонент Home

Компонент Home – це «домашня» сторінка, на яку буде потрапляти користувач, коли заходить на сайт. Важливо зробити її якомога красивішою в плані дизайну, адже гарне представлення – запорука успіху, потрібно зацікавити користувача у сторінці, як тільки він на неї потрапив, аби він користувався саме нашим додатком, а не якимсь застосунком-аналогом, наприклад, сайтам, які було розглянуто у розділі 2. Домашня сторінка буде складатись із кількох частин – фоновий слайдер, заголовок, підзаголовок та кнопки.

Для нашого слайдера будемо використовувати «кастомні» слайди, створимо для цього компонент CustomSlide, який буде отримувати шлях картинки та alt картинки (якщо картинку через якісь причини буде неможливо відобразити на сторінці), а повертати картинку з CSS стилями, серед цих стилів – розмір картинки на весь екран, прозорість, фільтр «grayscale». Тепер їх вложимо у <Slider> з бібліотеки react-slick, у цей слайдер передаємо налаштування, які прибирають стрілочки для ручного перегортання картинок, автоматично переключають картинки зі швидкість 2000 мілісекунд один раз в 5000 мілісекунд, гортають по одній картинці і одну картинку відображають, роблять це безкінечно та не зупиняються при наведенні миші.

У іншому компоненті створюємо верстку для відображення заголовку, підзаголовку та кнопки. Все це буде відображатись поверх слайдера посередині. У заголовку використаємо бібліотеку react-type-animation, яка дає змогу використати популярну, красиву та завжди актуальну анімацію друку. Для цього використовуємо компонент з бібліотеки – TypeAnimation, у який передаємо кілька

налаштувань, а саме: безкінечне повторювання, швидкість 20 і, власне, наші тези, такі як «Чистота повітря - це важливо», «Чистота повітря - ключ до комфорту» і так далі. Підзаголовок «Дізнайтесь про якість повітря у вашому місті…» обгортаємо у <h2>. Знизу використовуємо попередньо зроблений компонент Button, туди передаємо назву «Розпочати» та функцію, у якій використаємо хук useNavigate з бібліотеки react-router-dom, який дозволить нам змінити адресу на «/map» при натисненні і, відповідно, перейти на сторінку з картою.

5.3.4 Redux store, actions, reducers

У подальшій роботі нам потрібно буде використовувати різноманітні стани, тому можна попередньо створити потрібні нам функції «редюсери», «екшени» і, власне, сам store.

У папці Redux створюємо файл store.ts, у якому і створюємо store за допомогою однойменної функції createStore це наше сховище для станів, які будуть використовуватись у різних місцях проєкту Далі створюємо папки actions та reducers, у яких будуть знаходитись відповідні функції. Там будуть «редюсери» та «екшени» для двох стані, а саме – coord та data. Це стани для координатів, які у майбутньому будуть приходити з Google Map (за дефолтом ці координати дорівнюються координатам міста Київ), а також для даних, які ми будемо отримувати з API якості повітря.

5.3.5 Функція AQItoConc

Ця функція, відповідно до її назви, буде перетворювати індекс якості повітря речовини у її концентрацію у ppb або мікрограми на кубічний метр в залежності від речовини. Аби реалізувати такий задум, потрібно використати формулу перетворення AQI в концентрацію. У цій формулі використовуються «критичні точки» речовин, тому у цьому файлі ми зробимо масив об’єктів з критичними точками для кожної з речовин, які нам можуть прийти. Сама функція буде приймати назву забруднювача і значення індексу якості повітря для нього, далі функція по назві речовини буде обирати який масив критичний точок обрати та передати у формулу. Функція повертає результат обчислення формули.

5.3.6 Функція AQHIcalc

Ця функція призначена для обрахування Air Quality Health Index, який використовується у багатьох країнах. Вона приймає в себе об’єкт з концентрацією забруднювачів PM2.5, NO_2, O_3, якщо речовини немає, то концентрація речовини прирівнюється до нуля. Далі за формулою AQHI вона обраховує індекс на основі цих речовин і повертає його.

5.3.7 Компонент MapPage

MapPage – основна функціональна частина нашого веб-застосунку, у якій користувач зможу отримувати актуальні дані якості повітря у світі. Код компоненту представлений у Додатку А. Цей компонент буде розбитий на дві основні частини – ліва та права, зліва у нас буде знаходитись інформація про забруднення, а справа – Google Map, на якій користувач буде обирати точку, у якій він хочу побачити забруднення.

Почнемо з правої частини – з Google Map компонента. Цей компонент буде повертати великий блок з мапою та поле вводу, де користувач зможе власноруч знайти шукане ним місце на карті. Для початку імпортуємо усі потрібні нам для роботи речі, це Google Map, Marker (компонент, який дає змогу відображати на карті маркери), PolylineF (дає можливість відобразити лінії на карті від точки до точки), useJsApiLoader (хук, який динамічно підгружає дані з Google API), useDispatch, useSelector, setCoord (для управління станами за допомогою Redux), usePlacesAutocomplete, getGeocode, getLetLng (для поля вводу з «аутокомплітом» та для отримання даних за обраним місце), Combobox та інші компоненти з цієї бібліотеки (для створення поля вводу з селектом-аутокомплітом), ну і звісно ж хуки useEffect, useState з React та стилі CSS.

Для початкового відображення карти і подальшої роботи з нею, можемо скопіювати початковий код з документації GoogleMapAPI, а далі налаштовуємо під себе, а саме – пишемо CSS стилі для відображення мапи у кольорах сайту, забороняємо відкрити мапу на повний екран, а також переходити у «перегляд вулиць», задаємо мінімальний, максимальний та початковий рівень «зуму», забороняємо користувачу виходити за певні координати мапи (аби мапа було

однією, а не «скролилась» безкінечно), задаємо координати центру (міста Київ), передаємо onClick функцію – це найважливіше для нас, за допомогою неї будемо отримувати координати точки, на яку користувач натисне на карті.

Для створення поля вводу з «аутомкплітом» використовуємо <Combobox>, у якому комбінуємо Input та «спливаюче» вікно під ним, яке відіграє роль Select. Для управлінням усіх цього поля вводу використовуємо хук usePlacesAutocomplete, який саме для цього і створений. Також створимо функцію, яка буде виконуватись при обиранні місця з псевдо-Select. Ця функція буде заносити обраний адрес у поле вводу, очищати Select, а в стан з координатами заносити координати обраного місця за допомогою послідовних запитів getGeocode та getLatLng. Для Combobox створимо стилі завдяки яким він буде відображатись поверх карти та мати кольори сайту.

Тепер на завершення компоненту слід додати маркери на лінію між ними. Ці маркери будуть відображати дві точки – перша точка це обрані користувачем координати, а друга точка – це координати, які будуть нам повертатись з API забруднення повітря, адже станції моніторингу є не всюди, тому нам потрібно явно вказати наскільки далеко знаходиться точка моніторингу повітря від тієї точки, що обрав користувач на карті. Для цього слід використати хук useEffect, який виконує задану Callback функцію при зміни якогось стану. У масив «стеження» за станами передаємо стан з головного store – pollutionData, тобто тепер хук як би «підписався» на оновлення цього стану. Тому, як тільки з API індексу якості повітря світу прийдуть дані – useEffect виконає для нас функцію, а саме – очистить масив з маркерами, а потім, при умові, що дані про забруднення нам повернулись, створить два нових маркера, перший маркер буде мати координати обрані користувачем на карті, а другий – координати станції моніторингу, які нам повернулись з API. Відповідно, як тільки масив з маркерами зміниться – карти на це зреагує і відрендерить їх, також вона за допомогою PolylineF відобразить лінію між цими точками.

Тепер перейдемо до лівої частини компоненту MapPage – PollutionInfo, в цьому компоненті ми будемо робити запит, аби отримати забруднення повітря у найближчій точці до вибраної, заносити цю інформацію в store і відображати її.

Аби відображати інформацію – нам потрібно її отримати. В цьому разі ми знову використаємо хук useEffect. Ми підписуємось на стан координатів, які змінюються кожного разу, коли користувач клацає по карті або обирає певне місце у полі вводу. Тобто кожного разу, як координати будуть змінені – useEffect виконує функцію fetchData, у якій ми відправляємо за допомогою бібліотеки axios запит на API endpoint передаючи координати довготи та широти точки та наш API ключ. У відповідь ми отримуємо розгорнуту JSON інформацію по забрудненню повітрі (і не тільки) у найближчій точці станції моніторингу якості повітря. Ці дані заносимо у store. Варто зауважити, що AQI Programmatic API не надає інформації по території України, тому в разі, якщо користувач вибере Україну – ми робимо запит на Google Air Quality API, де підтримка України є.

Варто згадати, що до того, як користувач вибере свою першу точку на карті – нам немає що відображати, тому слід про це подбати. Було вирішено відображати «вступну» інформацію, якщо даних немає. Тут мається на увазі роз'яснення для користувача, а саме інформацію про те, що потрібно зробити, аби отримати інформацію про забруднення, а також інформацію по класифікації рівня забруднення за показником AQI – для цього було створено окремий компонент <ColorInfo/>, у нього передаються діапазон, колір фону і підпис-класифікація. Такий компонентів відображаємо шість, адже у нас 6 діапазонів. Також нижче покажемо діапазони класифікації ризику для здоров'я індексу AQHI, який ми теж будемо обраховувати. Уся ця бокова інформація відображається у семантичному тезі <aside>.

Тепер, коли «вступна» інформація є, слід подбати про відображення тієї інформації, яка приходить нам з API запиту. Для гарного відображення вичерпної інформації, було прийнято рішення зазначити відстань від обраної точки до станції моніторингу, «спідометр» на якому стрілочкою буде вказано рівень AQI, а поділками будуть діапазони класифікації забруднення якості повітря, а також концентрацію кожної окремої речовини забруднювача. Також було вирішено показати шкалу AQHI, адже цей показник теж використовується у багатьох країнах світу і може бути корисним, для цього теж використаємо «спідометр» з відповідної бібліотеки, а саме AQHI обраховуємо за допомогою вище описаної

функції AQHIcalc, куди передаємо концентрацію забруднювачів. Отже, як тільки користувач обирає точку і координати потрапляють у стан, ми робимо запит і отримує інформацію про забруднення. Як тільки ця інформацію буде отримана, «вступна» інформація з <aside> зникає, а починає відображатись інформація по забрудненню.

Перш за все потрібно визначити відстань до станції моніторингу. Для цього ми знову використовуємо useEffect, який буде реагувати на зміну pollutionData і виконувати функцію, де буде перевірятись чи у нас є в наявності координати обраної точки, а також координати, які приходять разом з pollutionData, і якщо вони є – створюємо два об’єкти-точки з координатами та передаємо у функцію getDistance з бібліотеки Geolib, яка може обчислити дистанцію між двома точками. Результат вносимо у стан distance і відображаємо його у нашому компоненті ділячи на 1000, адже результат повертається у метрах, а відображати ми хочем кілометри, також фіксуємо числа після коми до одного числа, аби результат мав, наприклад, такий вигляд – 412.2 км. Під відстанню відображаємо назву міста, де знаходить точка моніторингу. Ця назва нам повертається у об’єкті pollutionData.

Далі для наглядного відображення індексу якості повітря, аби користувач розумів наскільки якість повітря у цій точці безпечна або небезпечна, використаємо компонент <ReactSpeedometer> з бібліотеки react-d3-speedometer. Туди ми передаємо деякі налаштування, а саме – ширину спідометра у пікселях (у нашому випадку це 400 px), висоту стрілки спідометра – 0.8 від загальної висоти, значення спідометра – це якраз загальний AQI (якщо AQI>500, що буває іноді через збій у станціях моніторингу, то призначаємо AQI значення 500), далі передаємо поділки спідометра, це масив значень [0, 50, 100, 150, 200, 300, 500] відповідно до розподілу у класифікації, також мінімальне значення – 0 та максимальне значення – 500, кольори сегментів (для кожного сегменту між значенням поділок) – це масив кольорів у HEX, далі підпис значення – це «AQI», значення швидкості анімації стрілки спідометра при зміні значення в мілісекундах, назву анімації стрілки (анімації нам надає ця ж бібліотека), колір стрілки, а також колір тексту на спідометрі.

Тепер слід відобразити інформацію про концентрацію забруднювачів у повітрі, які нам приходять. Для цього створимо окремий компонент <Option>, яка буде приймати назву забруднювача і його концентрацію та відображати її. Оскільки користувачу буде цікавіше побачити саме концентрацію, а не рівень AQI забруднювача – потрібно перетворити AQI у концентрацію речовини відповідною формулою, для цього була попередньо створена функція AQItoConc, яку ми вже описали вище у цьому розділі. Якщо є відповідний забруднювач – ми передаємо його назву в <Option> та результат виконання функції AQItoConc. Також у компоненті дописується ppb або μg/m3 після концентрації у залежності від назви речовини.

На кінець покажемо рівень AQHI, аби користувач розумів який ризик для здоров'я несе повітря у обраній точці. Для цього спочатку потрібно перевести усі AQI у концентрацію, а потім передати ці показники у функцію AQHIcalc, яка власне і обрахує нам індекс ризику для здоров'я. В цілому наша шкала з react-d3-speedometer буде мати 10 різнокольорових сегментів з поділками: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, які явно вказують на ризику для здоров'я, а саме: 0-3 – низький, 4-7 – середній, 8-10 – високий, а 10+ - дуже високий, хоча показник 10+ це велика рідкість, а у нас в спідометрі максимальне значення буде саме 10. Під спідометром вказуємо обрахований AQHI округлений до 2 цифр після коми.

5.3.8 Компонент Pollutants

Оскільки було б добре, аби веб-застосунок робив ще якусь просвітницьку роботу – було вирішено створити окрему сторінку, де користувач міг би прочитати коротку інформацію про основні забруднювачі повітря, рівень яких ми можемо отримати на мапі забруднення. Для того, аби зручно було відображати інформацію по кожній речовині – створюємо окремий компонент Pollutant, це блок-обгортка для відображення інформації, він буде приймати заголовок, шлях на картинку (зазвичай це картинка молекул речовини), які відображати поряд з текстом і, власне, сам текст про речовину. В результаті у нас вийде сторінка з п'ятьма параграфами та картинками про кожну речовину, це – SO_2, CO, NO_2, O_3, а в останньому параграфі будуть об'єднані PM10 та PM2.5, адже вони схожі.

5.3.9 Компонент TopAQI

TopAQI – компонент, де користувач зможе побачити у виді таблиці забруднення повітря у 25 містах України, а саме 24 обласних центрах та міста Сімферополь. У таблиці є 8 стовпців, а саме назва міста, AQI, CO, NO_2, O_3, PM2.5, PM10, SO_2. Усі ці дані ми отримуємо за допомогою Google Air Quality API, для цього був створений великий масив з об'єктам, у кожному об'єкті є поля з назвою міста та координатами. По кожному об'єкті ми робимо запит на API і «пушимо» ці дані у state топу, далі це топ просто відображаємо у вигляді таблиці. Для зручності також було впроваджено функціонал сортування. Сортувати можна як за спаданням так і за зростанням. Для цього потрібно натиснути на заголовок будь-яку стовпця. Якщо натиснути один раз – сортування за цим стовпцем пройде за спаданням, якщо ще раз натиснути на той же стовпець – за зростанням. Стовпець з назвою міста сортується за алфавітом, інші – за порівнянням чисел.

5.3.10 Компонент NotFound

NotFound – невеличкий простий компонент, який буде відображати напис посередині сторінки про помилку 404 Not Found, яка свідчить про те, що такої сторінки не існує. Вона потрібна для того, аби у випадку, якщо користувач введе власний шлях у адресі сайту, якого просто не існує у Route, на сайті щось відображалось, а самец я сторінка, яка повідомляє нас, що ми потрапили на неіснуючий шлях.

5.3.11 Файл App.tsx

У цьому файлі був створений базовий Routing сайту, який визначає, які компоненти відображати на сторінці в залежності введеного шляху у браузері. Для цього використовується бібліотека react-router-dom. У загальному блоці App, ми створюємо обгортку <BrowserRouter>, у яку заносимо компонент шапки сайту Header, який буде відображатись завжди, незалежно від поточного Route. Поряд з ним створюємо обгортку <Routes> в середині якої уже, власне, пишемо наші шляхи та компоненти, які будуть відображатись, для цього пишемо тег <Route>, а в нього передаємо наш шлях – path (наприклад, '/map') та компонент, який буде

відображатись при даному path – element (наприклад, компонент <MapPage/>). Для того, аби відобразити сторінку NotFound, яку ми описали у попередньому підрозділі, у Route потрібно передати шлях path=’/*’, зірочка у цьому випадку означає будь-який шлях, а в елемент відповідно передаємо наш компонент «404»: element={<NotFound/>}. Таким чином, кожен раз, коли користувач вводить в адресний рядок браузера шлях, що відповідає одному з вказаних у <Route> шляхів, відображається відповідний компонент. Наприклад, при введенні шляху '/about' буде відображено компонент <AboutPage/>. Якщо ж шлях не відповідає жодному з вказаних у <Route>, буде відображено компонент NotFound, що дозволяє уникнути ситуацій, коли користувач бачить порожню сторінку або помилку. Це забезпечує зручну навігацію по сайту і покращує користувацький досвід.

6 ВИКОРИСТАННЯ ВЕБ-ЗАСТОСУНКУ КОРИСТУВАЧЕМ

Після закінчення процесу створення веб-додатку, який був описаний у розділі номер п’ять, слід провести інструктаж з користувачем, аби не залишилось питань по використанню створеного продукту. Потрібно зазначити, як потрапити на сайт, які вимоги до системи, інтернету, а також покроково пояснити сценарій користування системою моніторингу якості повітря світу для людей-користувачів.

6.1 Інсталяція та системні вимоги

Враховуючи те, що наш проєкт є веб-застосунком і, відповідно, працює з використанням веб-технологій, він не вимагає якоїсь конкретної операційної системи або якихось попередніх встановлень на комп’ютер користувача, окрім, звісно ж, браузера. Проте, оскільки веб-додаток робить багато API запитів, для комфортного використання бажано мати швидкість інтернету принаймні 50мбайт на секунду. Якщо у вас повільний інтернет – додаток все рівно буде виконувати свої функції, хоча прийдеться трохи зачекати. При тестуванні додатку на 2G інтернеті – сторінка з мапою отримувалась приблизно 10 секунд, будь-які дії з мапою вимагали зачекати ще секунд 10, аби підвантажилась мапа, запит на якість повітря займав приблизно 4 секунди.

6.2 Інструкції з використання веб-додатку

6.2.1 Головна сторінка веб-застосунку

Коли користувач заходить на сайт, він потрапляє на «домашню» головну сторінку (рисунок 6.1) додатку.

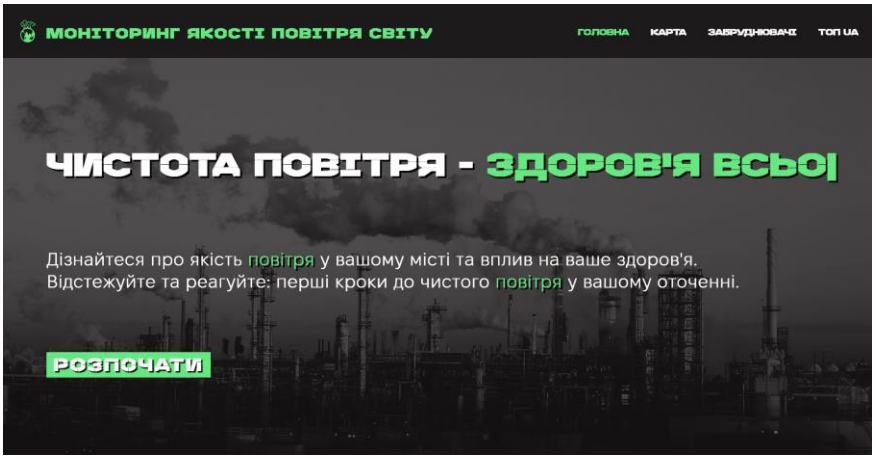


Рисунок 6.1 – Головна сторінка веб-застосунку

На ній зображено гасло сайту, яке змінюється кожні кілька секунд, як і тло сторінки. У підзаголовку коротко описані можливості користувача при роботі з веб-застосунком. Під підзаголовком знаходить кнопка «Розпочати», яка веде нас туди ж, куди і кнопка «карта» у шапці сайту – до головної за функціоналом сторінки, тобто сторінки з мапою, де ми можемо моніторити якість повітря.

6.2.2 Сторінка «Карта»

Коли користувач потрапляє на сторінку мапи – головну за функціоналом сторінку, він бачить карту Google Maps та підказку для початку роботи по моніторингу якості повітря (рисунок 6.2). Після того, як користувач прочитав інформацію для початку роботи з картою, він може перейти до користування нею, тобто просто натиснути на будь-яку точку на карті, після цього користувач отримає дані про забруднення повітря у найближчій точці моніторингу якості повітря (рисунок 6.3), а саме – відстань між точкою, що обрав користувач та найближчою станцією моніторингу якості повітря, відображення цих точок на карті та відрізку між ними, AQI – індекс якості повітря у даній точці, концентрацію речовин-забруднювачів, що вимірюються на станції моніторингу.

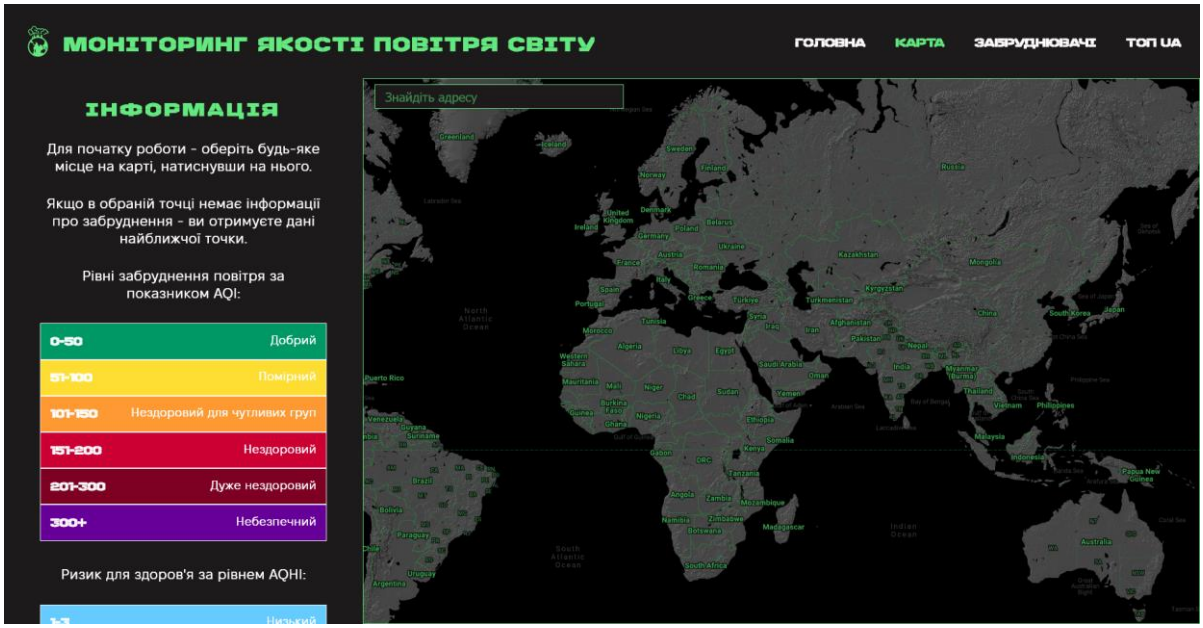


Рисунок 6.2 – Початкова сторінка «Карта»

Індекс якості повітря AQI представлений не просто у числовому виді, а у спідометрі, який явно дає зрозуміти користувачеві наскільки хороша чи погана якість повітря у обраній ним точці на мапі.

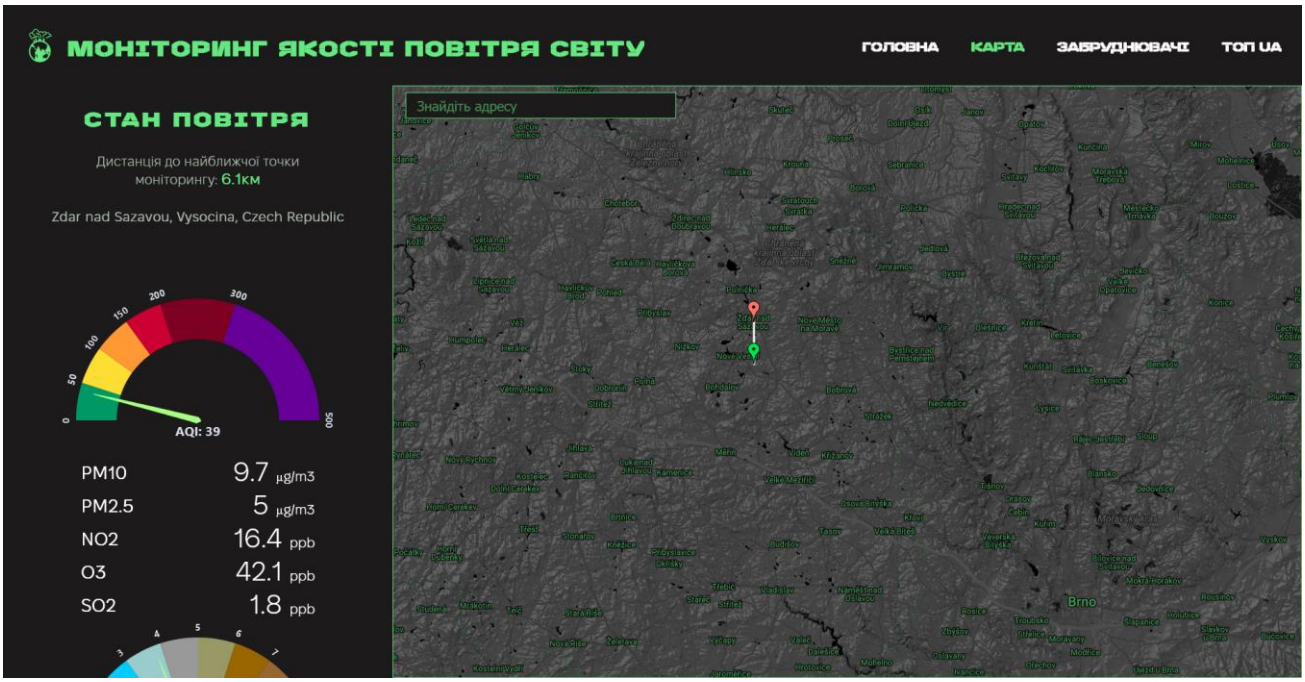


Рисунок 6.3 – Сторінка «Карта» з обраною користувачем точкою на карті

Також для зручності користувача на сторінці знаходиться поле вводу з автодоповненням (рисунок 6.4).

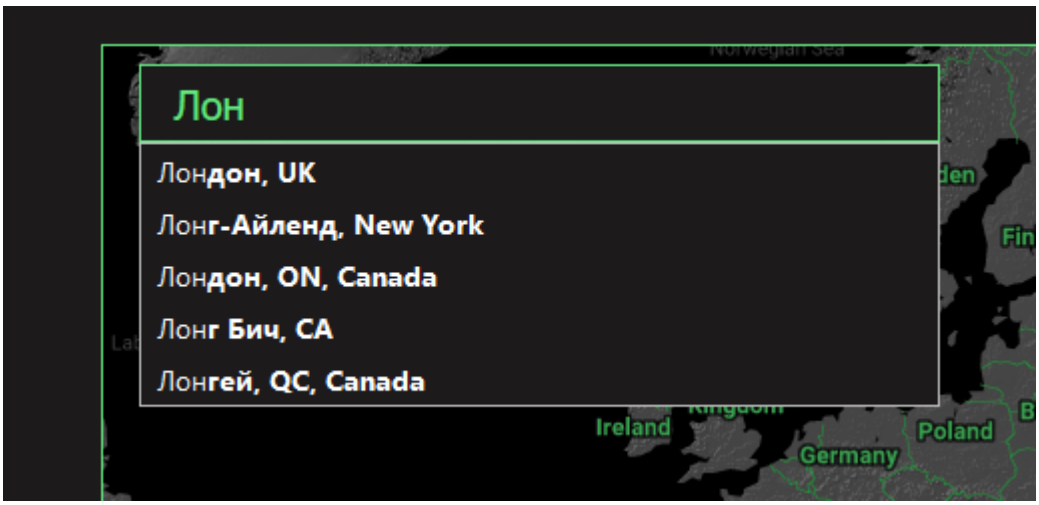


Рисунок 6.4 – Поле вводу з автодоповненням користувача для знаходження точки

Якщо користувач не хочу шукати потрібне йому місце на карті, він може ввести назву міста чи адресу у полі вводу, а додаток сам йому підкаже, доповнить його. Якщо користувач обирає місце з запропонованих йому у підказках – на карті обирається ця точка, а у бічному вікні відображається інформація з найближчої станції моніторингу якості повітря до обраної точки.

6.2.3 Сторінка «Забруднювачі»

Застосунок проводить невелику просвітницьку роботу з користувачами. Якщо ви захотіли більше дізнатись про забруднювачі, які ви бачите у інформації про забруднення повітря у обраній точці на сторінці «Карта», ви можете перейти на сторінку «Забруднювачі» (рисунок 6.5).

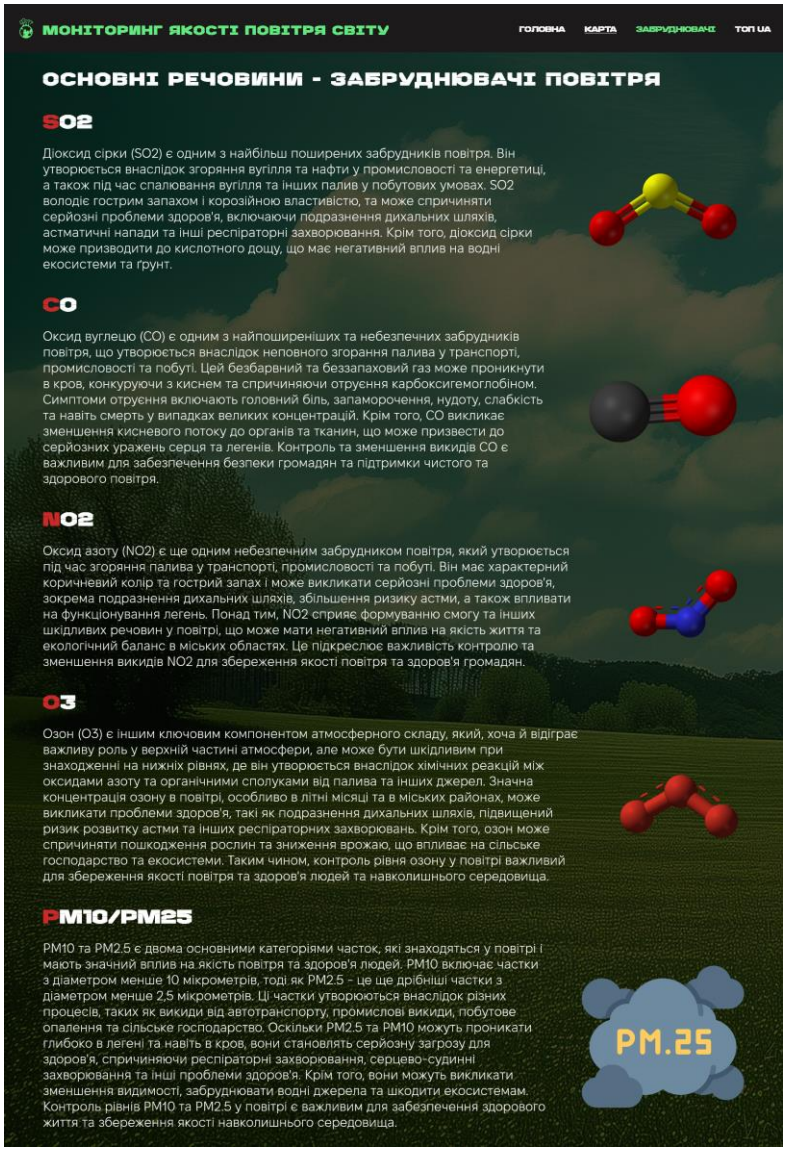


Рисунок 6.5 – Сторінка «Забруднювачі»

На даній сторінці ви можете прочитати коротку інформацію про кожну небезпечну речовину, а конкретно ви можете дізнатись про назву, походження, ризики для здоров’я та наслідки для екології.

6.2.4 Сторінка «Топ UA»

Натиснувши на вкладку «Top UA» у навігації, користувач потрапить на сторінку з таблицею-топом по містах України, а саме обласних центрах та місту Сімферополь (рисунок 6.6).

НАЗВА МІСТА	AQI	CO	NO2	O3	PM2.5	PM10	SO2
Харків	56	144.27	9.37	55.82	18.12	24.93	1.41
Одеса	56	126.31	2.1	54.99	23.75	31.47	1.45
Донецьк	60	122.35	6.84	50.16	20.63	27.19	2.8
Запоріжжя	60	318.74	9.41	40.71	24.87	36.38	7.33
Луганськ	61	138.99	7.69	49.15	18.98	29.38	3.2
Чернівці	62	441.52	9.67	48.07	7.98	16.62	1.53
Миколаїв	63	120.85	6.79	46.8	14.51	21.22	0.63
Дніпро	64	130.68	6.88	45.04	19.7	29.65	3.36
Івано-Франківськ	64	133.08	1.42	45.64	5.26	6	0.24
Полтава	65	124.41	4.9	44.16	14.78	23.03	0.6
Сімферополь	65	116.41	9.99	44.65	14.43	23.39	1
Суми	65	123.35	6.22	44.72	16.5	25.82	1.04
Херсон	65	121.2	5.08	44.24	13.66	20.03	0.85
Черкаси	66	120.16	5.08	43.49	15.13	23.4	2.68
Кропивницький	66	114.18	5.62	42.51	11.36	19.1	0.2
Київ	67	128.52	8.38	41.8	11.77	18.51	0.51
Ужгород	67	206.79	6.6	41.9	12.98	20.91	2.1
Тернопіль	67	118.96	7.37	41.29	11.1	14.38	0.53
Львів	68	133.41	7.5	40.79	10.66	16.17	0.54
Вінниця	68	122.95	6.43	40.52	10.48	14.25	0.2
Хмельницький	68	120.86	6.16	40.24	8.78	12.42	0.13
Чернігів	68	110.65	4.93	40.6	11.99	18.19	0.39
Луцьк	70	125.88	5.36	37.88	7.55	10.72	0.12
Житомир	70	124.37	6.15	37.53	10.05	15.3	0.34
Рівне	72	129.25	8.21	34.88	7.46	10.96	0.4

Рисунок 6.6 – Сторінка «Top UA»

Користувач також може сортувати топ так, як він того захоче. Веб-застосунок дає змогу натиснути на назву будь-якого стовпця один або два рази і вся таблиці буде сортуватись за відповідним стовпцем за зростанням або спаданням у залежності від кількості натискань на назву стовпця. За «дефолтом» стовпець впорядкований за AQI у порядку зростання, тобто зверху у таблиці будуть міста, де забруднення найменше.

6.2.5 Сторінка «404 NotFound»

Якщо користувач самостійно ввів адресу сторінки, якої не існує – він потрапляє на сторінку Not Found (рисунок 6.7).

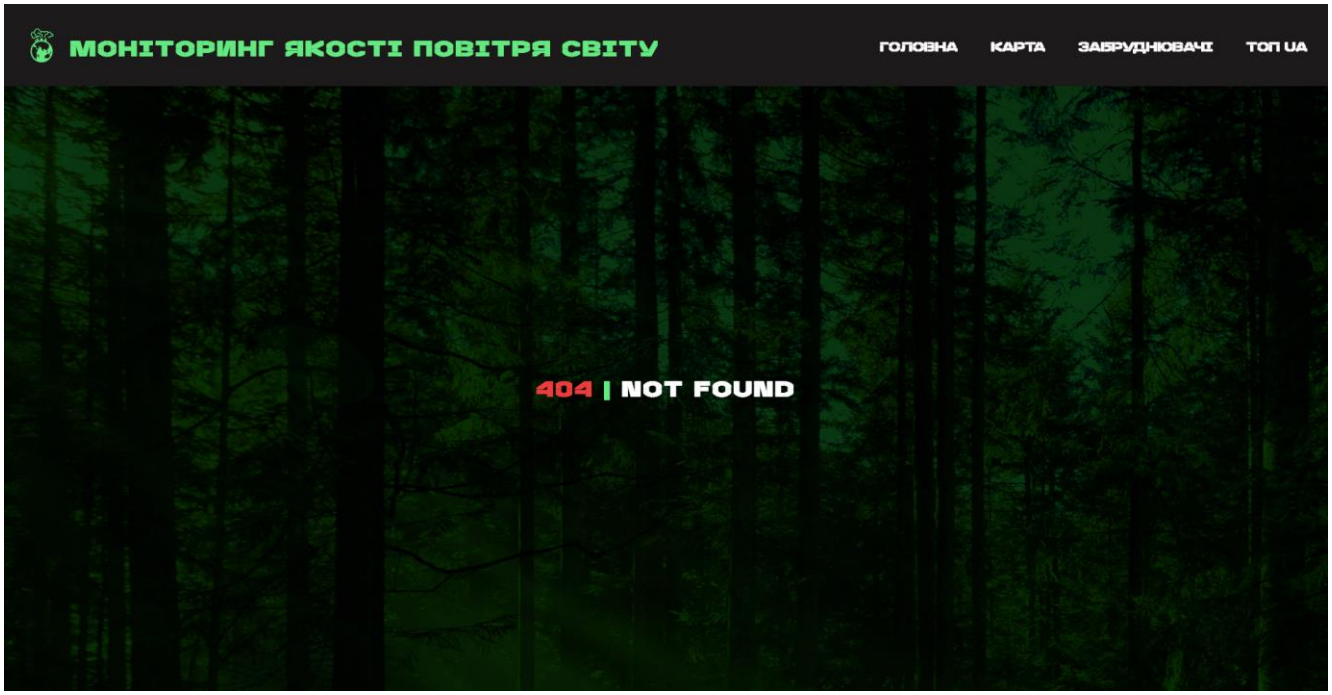


Рисунок 6.7 – Сторінка «404 Not Found»

Ця сторінка дає користувачу зрозуміти, що він навмисно або випадково потрапив на неіснуючу сторінку. Користувач може легко перейти на існуючу, «валідну» сторінку просто натиснувши на назву іншої сторінку у шапці веб-застосунку або увівши «валідний», існуючий шлях до сторінки у адресному рядку. Для звичайного користувача знайти цю сторінку буде складно.

ВИСНОВКИ

У цій роботі було досліджено проблему забруднення повітря, його вплив на здоров'я та довкілля, а також розроблено веб-додаток для моніторингу якості повітря світу мовою React.

Було проведено аналіз наявних можливих схожих рішень створення додатків, а саме SaveEcoBot, Eco-city, WAQI.info, iQAir. Були розглянути їх переваги та недоліки, які були враховані при створенні власного веб-застосунку з моніторингу якості повітря світу.

Було розглянуто теоретичну частину по забрудненню повітря у світі, а саме можливі індекси якості повітря світу, їх класифікації та розподіли у різних країнах світу сучасності, основні речовини-забруднювачі, їх походження, вплив на здоров’я та екосистему планети, можливі способи-рекомендації усунення проблеми.

Для розробки власного проєкту були підібрані відповідні мови програмування та інші інструменти, а саме HTML, CSS, JS, TS, React, Redux, Google Maps API, Google Air Quality API, AQP API та всілякі додаткові допоміжні бібліотеки, такі як Geolib, react-d3-speedometer, react-type-animation, react-slick та інші.

Було створено веб-застосунок за допомогою вище описаних інструментів. Проєкт успішно функціонує і готовий до використання. Він надає змогу будь-кому відслідковувати актуальну якість повітря у будь-якій обраній точці планети, даючи вичерпну, широку інформацію по речовинам забруднювачам та загальному AQI обраної точки. Застосунок також виконує просвітницьку роботу, адже кожен користувач може отримати коротку інформацію по кожному забруднювачу, який він може отримати з карти.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. OECD: The Organization for Economic Co-operation and Development (OECD) is an international organization that works to build better policies for better lives. URL: https://data.oecd.org/air/air-pollution-exposure.html (дата звернення 12.06.2024)

2. SaveEcoBot: Єдина в Україні екологічна система, яка поєднує дані про поточний стан довкілля, про забруднення, забруднювачів та інструменти захисту довкілля. URL: https://www.saveecobot.com (дата звернення 12.06.2024)

3. Eco-city: Карта моніторингу якості. URL: https://eco-city.org.ua/ (дата звернення 12.06.2024)

4. WAQI.info: The World Air Quality Index project is a non-profit project started in 2007. URL: https://waqi.info/ (дата звернення 12.06.2024)

5. iQAir: Explore your Air Quality. URL: https://www.iqair.com (дата звернення 12.06.2024)

6. УкрХімАналіз: хімічний аналіз в Україні. Індекс якості повітря. URL: https://himanaliz.ua/uk/yak-viznachaietsya-indeks-yakosti-povitr/ (дата звернення 12.06.2024)

7. SaveDnipro: найчастіші питання про забруднене повітря. URL: https://www.savednipro.org/pytannia-pro-zabrudnene-povitria/ (дата звернення 12.06.2024)

8. Wikiwand: Онлайн енциклопедія. Оксид азот (IV). URL: https://www.wikiwand.com/uk/Оксид_азот (дата звернення 12.06.2024)

9. StreamOzone. Озон. URL: http://streamozone.com.ua/ozon (дата звернення 12.06.2024)

10. Gastech. Монооксид вуглецю. URL: https://www.gastech.com.ua/chadnyj-gaz-oksyd-vugleczyu-co/ (дата звернення 12.06.2024)

11. Tyhj: Діоксид сірки. URL: http://uk.tyhjgas.com/sulfur-dioxide-so2-product/ (дата звернення 12.06.2024)

12. Cleanair: все про повітря. Аміак. URL: https://cleanair.org.ua/pollutant/ammonia-ua/ (дата звернення 12.06.2024)

13. Mdn web docs: Resources for Developers by Developers, HTML. URL: https://developer.mozilla.org/en-US/docs/Web/HTML (дата звернення 12.06.2024)

14. Mdn web docs: Resources for Developers by Developers, CSS. URL: https://developer.mozilla.org/en-US/docs/Web/CSS (дата звернення 12.06.2024)

15. Mdn web docs: Resources for Developers by Developers, JavaScript. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript (дата звернення 12.06.2024)

16. React: The library for web and native user interfaces. URL: https://react.dev/ (дата звернення 12.06.2024)

17. TypeScript is JavaScript with syntax for types. URL: https://www.typescriptlang.org/ (дата звернення 12.06.2024)

18. Redux: a JS library for predictable and maintainable global state management. URL: https://redux.js.org/ (дата звернення 12.06.2024)

19. Axios: HTTP-клієнт для браузера та node.js на основі Promise. URL: https://axios-http.com/uk/ (дата звернення 12.06.2024)

20. Google Maps Platform: Build awesome apps with Google`s knowledge of the real world. URL: https://developers.google.com/maps (дата звернення 12.06.2024)

21. Air Quality Programmatic APIs. URL: https://aqicn.org/api/ (дата звернення 12.06.2024)

Додаток А

Програмна реалізація <GoogleMap/> компоненту

Програмні засоби:

- Мова програмування TypeScript
- Бібліотеки React, Redux, react-google-maps/api, use-places-autocomplete, Combobox,

```
import React, { useEffect, useState } from "react";
import {
  GoogleMap,
  Marker,
  useJsApiLoader,
  PolylineF,
} from "@react-google-maps/api";
import { useDispatch, useSelector } from "react-redux";
import { setCoord, setCountryShortName } from "../../../redux/actions";
import usePlacesAutocomplete, {
  getGeocode,
  getLatLng,
} from "use-places-autocomplete";
import {
  Combobox,
  ComboboxInput,
  ComboboxPopover,
  ComboboxList,
  ComboboxOption,
} from "@reach/combobox";
import "@reach/combobox/styles.css";
import c from "../MapPage.module.css";
```

```
const containerStyle = {
  width: "70vw",
  height: "88vh",
  border: "1px solid #64ea81",
};

const center = {
  lat: 50.450001,
  lng: 30.523333,
};

const mapStyles = [
  {
    elementType: "geometry",
    stylers: [
      {
        color: "#454545",
      },
    ],
  },
  {
    elementType: "labels.icon",
    stylers: [
      {
        visibility: "off",
      },
    ],
  },
  {
    elementType: "labels.text.fill",
```

```
  stylers: [
    {
      color: "#2e9043",
    },
  ],
},
{
  elementType: "labels.text.stroke",
  stylers: [
    {
      color: "#000000",
    },
  ],
},
{
  featureType: "administrative",
  elementType: "geometry",
  stylers: [
    {
      color: "#21933a",
    },
  ],
},
{
  featureType: "administrative.country",
  elementType: "labels.text.fill",
  stylers: [
    {
      color: "#39a250",
    },
  ],
```

```
},
{
 featureType: "administrative.land_parcel",
 stylers: [
  {
   visibility: "off",
  },
 ],
},
{
 featureType: "administrative.locality",
 elementType: "labels.text.fill",
 stylers: [
  {
   color: "#42a056",
  },
 ],
},
{
 featureType: "poi",
 elementType: "labels.text.fill",
 stylers: [
  {
   color: "#757575",
  },
 ],
},
{
 featureType: "poi.park",
 elementType: "geometry",
 stylers: [
```

```
    {
      color: "#181818",
    },
  ],
},
{
  featureType: "poi.park",
  elementType: "labels.text.fill",
  stylers: [
    {
      color: "#616161",
    },
  ],
},
{
  featureType: "poi.park",
  elementType: "labels.text.stroke",
  stylers: [
    {
      color: "#1b1b1b",
    },
  ],
},
{
  featureType: "road",
  elementType: "geometry.fill",
  stylers: [
    {
      color: "#2c2c2c",
    },
  ],
```

```
},
{
 featureType: "road",
 elementType: "labels.text.fill",
 stylers: [
  {
   color: "#8a8a8a",
  },
 ],
},
{
 featureType: "road.arterial",
 elementType: "geometry",
 stylers: [
  {
   color: "#373737",
  },
 ],
},
{
 featureType: "road.highway",
 elementType: "geometry",
 stylers: [
  {
   color: "#3c3c3c",
  },
 ],
},
{
 featureType: "road.highway.controlled_access",
 elementType: "geometry",
```

```
  stylers: [
    {
      color: "#4e4e4e",
    },
  ],
},
{
  featureType: "road.local",
  elementType: "labels.text.fill",
  stylers: [
    {
      color: "#616161",
    },
  ],
},
{
  featureType: "transit",
  elementType: "labels.text.fill",
  stylers: [
    {
      color: "#757575",
    },
  ],
},
{
  featureType: "water",
  elementType: "geometry",
  stylers: [
    {
      color: "#000000",
    },
```

```
    ],
  },
  {
    featureType: "water",
    elementType: "labels.text.fill",
    stylers: [
      {
        color: "#3d3d3d",
      },
    ],
  },
];

const polylineOptions = {
  strokeColor: "#ffffff",
};

declare var process: {
  env: {
    REACT_APP_GOOGLE_API_TOKEN: string;
  };
};

interface MarkerType {
  lat: number;
  lng: number;
  icon?: string;
}

const TOKEN = process.env.REACT_APP_GOOGLE_API_TOKEN;
```

```
const GoogleMapComp: React.FC = () => {
  const dispatch = useDispatch();
  const coord = useSelector((state: any) => state.coord);
  const pollutionData = useSelector((state: any) => state.data);

  const [markers, setMarkers] = useState<MarkerType[]>([]);

  const { isLoaded } = useJsApiLoader({
    id: "google-map-script",
    googleMapsApiKey: TOKEN,
    libraries: ["places"],
  });

  useEffect(() => {
    console.log("TUT");
    const getCountryName = async () => {
      const response = await fetch(

`https://maps.googleapis.com/maps/api/geocode/json?latlng=${coord.coord
[0]},${coord.coord[1]}&key=${TOKEN}`
      );
      const data = await response.json();
      if (data.status === "OK" && data.results.length > 0) {
        dispatch(setCountryShortName(data.results[0].address_components));
      }
    };
    getCountryName();
  }, [coord]);

  const onLoad = React.useCallback(function callback(map:
google.maps.Map) {
```

```
  if (window.google && window.google.maps) {
    const bounds = new window.google.maps.LatLngBounds(center);
    map.fitBounds(bounds);
  }
}, []);

useEffect(() => {
  setMarkers([]);
  if (
    pollutionData &&
    pollutionData.data &&
    pollutionData.data.city &&
    pollutionData.data.city.geo
  ) {
    const newMarkers: MarkerType[] = [
      {
        lat: pollutionData.data.city.geo[0],
        lng: pollutionData.data.city.geo[1],
        icon: "http://maps.google.com/mapfiles/ms/icons/red-dot.png",
      },
      {
        lat: coord.coord[0],
        lng: coord.coord[1],
        icon: "http://maps.google.com/mapfiles/ms/icons/green-dot.png",
      },
    ];
    setMarkers(newMarkers);
  }
}, [pollutionData]);

return isLoaded ? (
```

```
<>
 <GoogleMap
  mapContainerStyle={containerStyle}
  center={center}
  onLoad={onLoad}
  zoom={5}
  onClick={(e) => {
   if (e.latLng) {
    dispatch(setCoord([e.latLng.lat(), e.latLng.lng()]));
   }
  }}
  options={{
   styles: mapStyles,
   fullscreenControl: false,
   mapTypeControl: false,
   mapTypeId: "terrain",
   streetViewControl: false,
   zoomControl: false,
   minZoom: 3,
   maxZoom: 10,
   restriction: {
    latLngBounds: {
     north: 85,
     south: -85,
     west: -179.9,
     east: 179.9,
    },
    strictBounds: false,
   },
  }}
 >
```

```
        {markers.map((marker, index) => (
          <Marker
            key={index}
            position={{ lat: marker.lat, lng: marker.lng }}
            icon={marker.icon}
          />
        ))}
        {markers.length === 2 && (
          <PolylineF path={markers} options={polylineOptions} />
        )}
      </GoogleMap>
      <PlacesAutocomplete />
    </>
  ) : (
    <></>
  );
};

export default React.memo(GoogleMapComp);

const PlacesAutocomplete: React.FC = () => {
  const {
    ready,
    value,
    setValue,
    suggestions: { status, data },
    clearSuggestions,
  } = usePlacesAutocomplete();
  const dispatch = useDispatch();

  const handleSelect = async (address: string) => {
```

```
      setValue(address, false);
      clearSuggestions();

      const results = await getGeocode({ address });
      const { lat, lng } = await getLatLng(results[0]);
      dispatch(setCoord([lat, lng]));
    };

    return (
      <Combobox onSelect={handleSelect} className={c.combox}>
        <ComboboxInput
          value={value}
          onChange={(e) => setValue(e.target.value)}
          disabled={!ready}
          className={c.comboxInp}
          placeholder="Знайдіть адресу"
        />
        <ComboboxPopover>
          <ComboboxList className={c.list}>
            {status === "OK" &&
              data.map(({ place_id, description }) => (
                <ComboboxOption key={place_id} value={description} />
              ))}
          </ComboboxList>
        </ComboboxPopover>
      </Combobox>
    );
};
```