Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра системного аналізу та інформаційних технологій

Бакалаврська кваліфікаційна робота на тему

ІНФОРМАЦІЙНА СИСТЕМА ПРОСТОРОВО-ЧАСОВОГО АНАЛІЗУ КІЛЬКОСТІ ВОДИ У РІЧКОВОМУ БАСЕЙНІ ПІВДЕННОГО БУГУ»

Виконала: студентка 4 курсу 2ІСТ-21б
Спеціальності 126 – «Інформаційні системи та технології»
_____ Каміла ЯНЧЕНКО
Керівник: к.т.н., доц. каф. САІТ
_____ Євгеній КРИЖАНОВСЬКИЙ
«05» 06 2025 р.
Рецензент: к.т.н., ас. каф. КН
Ілона БОГАЧ
«12» 06 2025 р.

Завідувач кафедри САІТ
_____ д.т.н., проф. Віталій МОКІН
«06» 06 2025 р.

Вінниця ВНТУ – 2025 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра системного аналізу та інформаційних технологій
Рівень вищої освіти – перший (бакалаврський)
Галузь знань – 12 Інформаційні технології
Спеціальність 126 – «Інформаційні системи та технології»
Освітньо-професійна програма – Прикладні інформаційні технології

ЗАТВЕРДЖУЮ
Завідувач кафедри САІТ
_____ д.т.н., проф. Віталій МОКІН
"28" 05 2025 року

ЗАВДАННЯ
НА БАКАЛАВРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТЦІ
Янченко Камілі Олексіївні

1. Тема роботи: «Інформаційна система просторово-часового аналізу кількості води у річковому басейні Південного Бугу»
керівник роботи: Крижановський Є.М., доц. каф. САІТ
затверджені наказом ВНТУ від «20» 05 2025 року № 97
2. Термін подання студентом роботи 27.05.2025 р.
3. Зміст текстової частини:
1) Аналіз предметної області
2) Вибір оптимальної ІТ-інфраструктури.
3) Проектування інформаційної системи.
4) Розробка інформаційної системи просторово-часового аналізу кількості води у річковому басейні Південного Бугу.
4. Перелік ілюстративного матеріалу:
1) Діаграма оновлення даних;
2) Діаграма послідовності для керування користувачами;
3) Діаграма класів;
4) Діаграма станів для система моніторингу;
5) Діаграма запуску та ініціалізації системи;

6. Консультанти розділів роботи:

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання 28.03.25 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва та зміст етапу	Термін виконання		Примітка
		початок	закінчення	
1	Аналіз предметної області	01.04	15.04	Вик
2	Вибір оптимальної інфраструктури	15.04	30.04	Вик
3	Проектування архітектури інформаційної системи	01.05	10.05	Вик
4	Розробка інформаційної системи просторово-часового аналізу кількості води у річковому басейні Південного Бугу	10.05	20.05	Вик
5	Оформлення матеріалів до захисту БКР	20.05	30.05	Вик

Студентка _____ Каміла ЯНЧЕНКО

Керівник роботи _____ Євгеній КРИЖАНОВСЬКИЙ

АНОТАЦІЯ

Бакалаврська кваліфікаційна робота містить 84 сторінки формату А4, на яких представлено 49 рисунок; список використаних джерел налічує 21 найменувань.

Метою роботи є створення інформаційної системи у вигляді веб-додатку, що забезпечує просторово-часовий аналіз кількості води в басейні річки Південний Буг з урахуванням геолокації гідрологічних постів, вибраних часових періодів та можливістю формування звітів.

У рамках проєкту розроблено веб-додаток із використанням бібліотеки React та мови програмування JavaScript. Реалізовано зрозумілий для користувачів інтерфейс, що включає інтеграцію з бібліотекою Leaflet для відображення гідрологічних постів на карті, Chart.js для побудови графіків та XLSX для формування звітів. Для зберігання даних використовується хмарна база Supabase.

Система розроблена з урахуванням можливості доступу через браузер на різноманітних пристроях.

Ключові слова: інформаційна система, веб-додаток, React, JavaScript, Leaflet, Chart.js, Supabase, гідрологічний аналіз, басейн Південного Бугу.

ABSTRACT

The bachelor's qualification thesis consists of 84 A4-format pages and includes 49 figures; the list of references contains 21 sources.

The purpose of the work is to create an information system in the form of a web application that provides spatio-temporal analysis of the amount of water in the Southern Bug River basin, taking into account the geolocation of hydrological posts, selected time periods, and the ability to generate reports.

As part of the project, a web application was developed using the React library and the JavaScript programming language. A user-friendly interface was implemented, featuring integration with the Leaflet library for displaying hydrological posts on a map, Chart.js for chart generation, and XLSX for report creation. Data storage is managed through the cloud-based Supabase service.

The system is designed to be accessible via a browser on various devices.

Key words: information system, web application, React, JavaScript, Leaflet, Chart.js, Supabase, hydrological analysis, Southern Bug river basin.

ЗМІСТ

ВСТУП

Актуальність теми. У сучасному світі питання ефективного управління водними ресурсами стає все більш актуальним через вплив кліматичних змін, антропогенних факторів та необхідність дотримання принципів сталого розвитку. Річковий басейн Південного Бугу є одним із ключових водних об’єктів України, який потребує постійного моніторингу та аналізу кількості води для забезпечення екологічної безпеки, раціонального водокористування та попередження надзвичайних ситуацій, таких як повені чи посухи. Розробка інформаційної системи, що дозволяє здійснювати просторово-часовий аналіз водних ресурсів, стане важливим інструментом для екологів, фахівців моніторингу та органів управління, забезпечуючи швидкий доступ до актуальних даних, їх обробку та візуалізацію.

Мета і завдання дослідження. Метою роботи є створення інформаційної системи у вигляді веб-додатку, що забезпечує просторово-часовий аналіз кількості води в басейні річки Південний Буг з урахуванням геолокації гідрологічних постів, вибраних часових періодів та можливістю формування звітів.

Для досягнення цієї мети необхідно виконати такі завдання:

- провести аналіз предметної області;
- дослідити та визначити необхідний функціонал інформаційної системи;
- розробити прототип та реалізувати інформаційну систему.

Об’єкт дослідження є процес створення інформаційної технології для просторово-часового аналізу кількості води у річковому басейні Південного Бугу.

Предмет дослідження є методи та програмні засоби розробки інформаційних систем, що забезпечують збір, обробку, аналіз і візуалізацію гідрологічних даних, а також формування звітів для ефективного моніторингу водних ресурсів.

Публікації. За результатами виконання даної роботи опубліковано тези доповідей на тему: « Інформаційна система моніторингу кількості вод річкового басейну Південного Бугу» на LIV Всеукраїнській науково-технічній конференції

факультету інтелектуальних інформаційних технологій та автоматизації (2025) [1].

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Проблематика просторово-часового аналізу кількості води

У сучасних умовах, на тлі кліматичних змін, зростаючого антропогенного впливу та потреби в сталому розвитку, ефективне управління водними ресурсами набуває особливої актуальності. Систематичний моніторинг і аналіз кількості води в річкових басейнах, зокрема Південного Бугу, відіграє важливу роль у визначенні стану водних ресурсів, прогнозуванні повеней і посух, плануванні водокористування та зниженні екологічних ризиків. Однак однією з основних перешкод для якісного аналізу є відсутність зручних та спеціалізованих інструментів, які б забезпечували обробку, аналіз і візуалізацію гідрологічних даних [2].

Фахівці — екологи, оператори моніторингу, управлінці — часто стикаються з труднощами, пов’язаними з розпорошеністю джерел, браком доступу до актуальної інформації в режимі реального часу та відсутністю комплексних рішень. Це ускладнює оперативну оцінку ситуації та прийняття управлінських рішень, що може негативно впливати на ефективність управління водними ресурсами.

Традиційні підходи до аналізу ґрунтувалися на ручному зборі даних, використанні звітів гідрологічних станцій або локальних баз даних. Вони мають низку обмежень — зокрема, відсутність централізованого доступу до інформації, складність поєднання просторових і часових характеристик, а також обмеженість у фільтрації, обробці та представленні результатів у зручному вигляді[3,21].

З розвитком цифрових технологій з’явилася можливість вирішити ці проблеми, однак аналіз існуючих платформ свідчить, що більшість з них орієнтовані на загальний моніторинг довкілля, моделювання кліматичних змін або прогнозування погоди. При цьому інструменти просторово-часового аналізу зазвичай недостатньо розвинені: системи надають лише загальні звіти без можливості деталізації за регіонами, періодами чи показниками, а інтерактивна візуалізація часто відсутня.

Ті користувачі, які потребують глибшого аналізу, змушені використовувати застарілі звіти, шукати фрагментарні дані у відкритих джерелах або створювати власні рішення, що є не лише трудомістким, а й малоефективним підходом через неповноту або застарілість інформації.

У зв’язку з цим виникає очевидна потреба у створенні інформаційної системи у вигляді веб-додатку, який дозволить здійснювати просторово-часовий аналіз кількості води в басейні Південного Бугу. Такий інструмент значно підвищить якість управлінських рішень і стане необхідним помічником для фахівців, зацікавлених у збереженні водних ресурсів та забезпеченні екологічної безпеки.

1.2 Аналіз ринку інформаційних систем

Ринок інформаційних систем для аналізу водних ресурсів є достатньо розвиненим і продовжує активно зростати. Серед відомих веб-платформ, що забезпечують доступ до гідрологічних даних, варто виокремити HydroDesktop, WaterML, Google Earth Engine, ArcGIS та інші (рис. 1.1–1.2). Проте більшість із них орієнтовані переважно на загальний моніторинг водних ресурсів, кліматичне прогнозування або обробку великих обсягів інформації. Повноцінна реалізація просторово-часового аналізу, особливо для локальних басейнів, часто або обмежена, або вимагає високого рівня технічної підготовки від користувача.

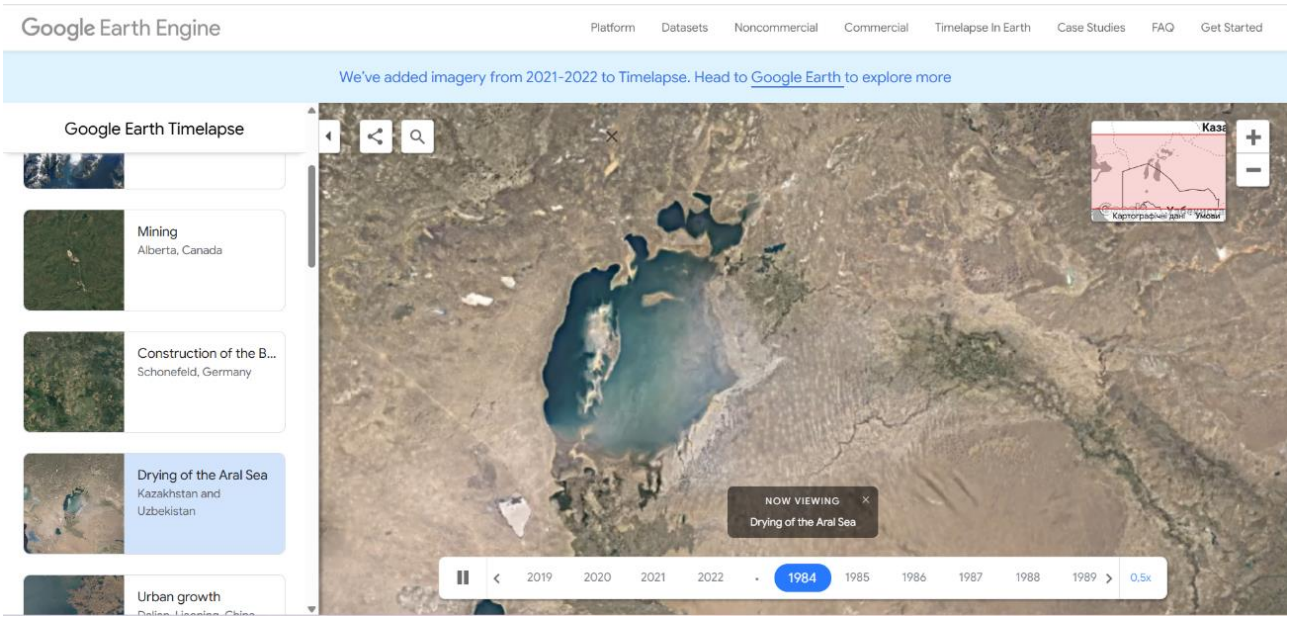

Рисунок 1.1 – Веб-платформа Google Earth Engine

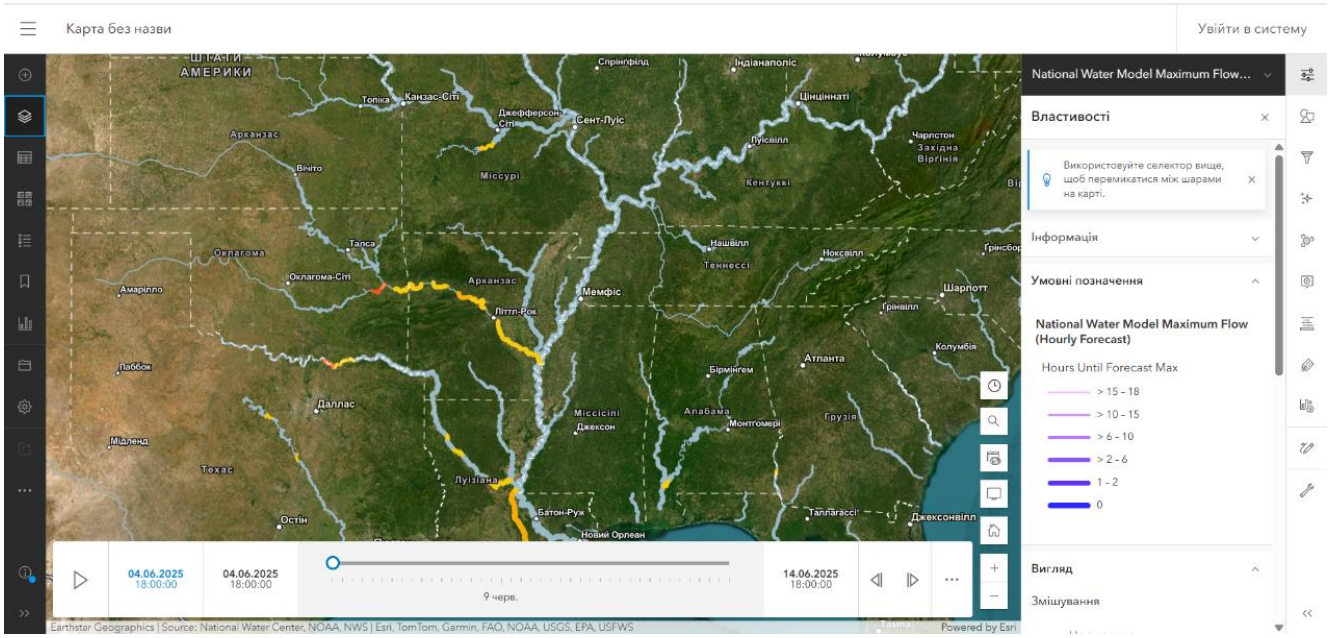

Рисунок 1.2 – Веб-платформа ArcGIS

Світовий ринок інформаційних систем для управління водними ресурсами демонструє стійку тенденцію до зростання. Це зумовлено зростаючою потребою в ефективному та сталому використанні водних ресурсів, а також активним розвитком сучасних цифрових технологій. Зокрема, все більшого значення набувають інструменти геопросторового аналізу, хмарні сервіси для зберігання та обробки даних, а також платформи для інтерактивної візуалізації інформації. Сучасні користувачі очікують від таких систем не лише базового моніторингу, а

й наявності гнучких засобів для просторово-часового аналізу, які поєднуються з доступним та зрозумілим інтерфейсом [4].

В умовах України спеціалізовані веб-сервіси для гідрологічного аналізу поки що не набули широкого поширення. У більшості випадків користувачі змушені користуватися застарілими звітами, відкритими джерелами або локальними програмами, які не забезпечують достатньої оперативності та зручності для ефективного прийняття рішень.

У зв’язку з цим, створення сучасної веб-платформи з простим інтерфейсом, розширеними фільтрами для аналізу, інтеграцією з картою гідрологічних постів та хмарною базою даних Supabase є доцільним і перспективним напрямом. Такий продукт може задовольнити потреби як фахівців, так і широкого кола користувачів у сфері управління водними ресурсами.

1.3 Огляд існуючих інформаційних систем

Огляд існуючих інформаційних систем є важливим етапом у розробці нових програмних рішень, оскільки дозволяє виявити сильні та слабкі сторони наявних сервісів, врахувати перевірені підходи та унікальні функції, які можуть бути адаптовані до локальних потреб користувачів.

Одним із сучасних онлайн-ресурсів для роботи з гідрологічними та кліматичними даними є Copernicus Climate Data Store (CDS) — платформа, яка надає вільний доступ до кліматичних змінних, зокрема до даних про опади, температуру, вологість ґрунту та інші гідрометеорологічні параметри (рис. 1.3–1.4). CDS дозволяє здійснювати просторово-часовий аналіз із використанням історичних і прогнозованих даних, а також завантажувати їх у різних форматах для подальшої обробки. Користувачі мають змогу створювати власні запити через інтерактивний інтерфейс або API, що робить платформу корисною як для науковців, так і для практиків [5].

Переваги: доступ до широкого спектру змінних, API, підтримка форматів NetCDF та CSV.

Недоліки: інтерфейс може бути складним для непідготовленого користувача, обмежена візуалізація.

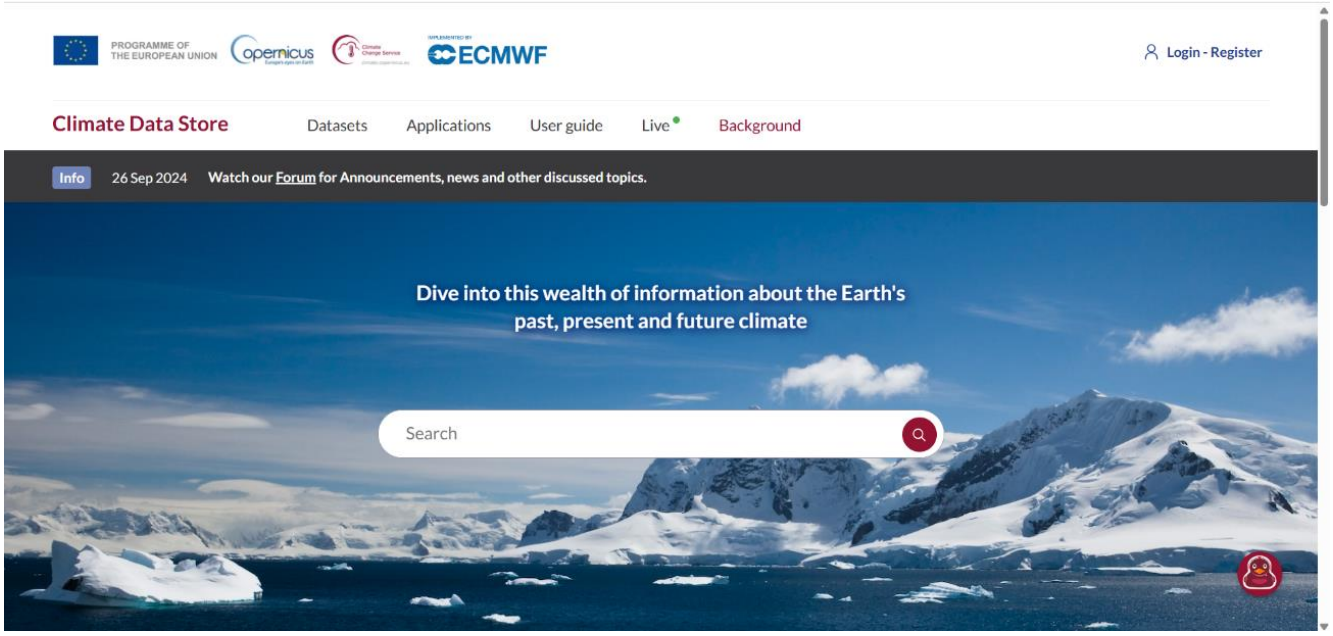

Рисунок 1.3 – Інтерфейс Copernicus Climate Data Store

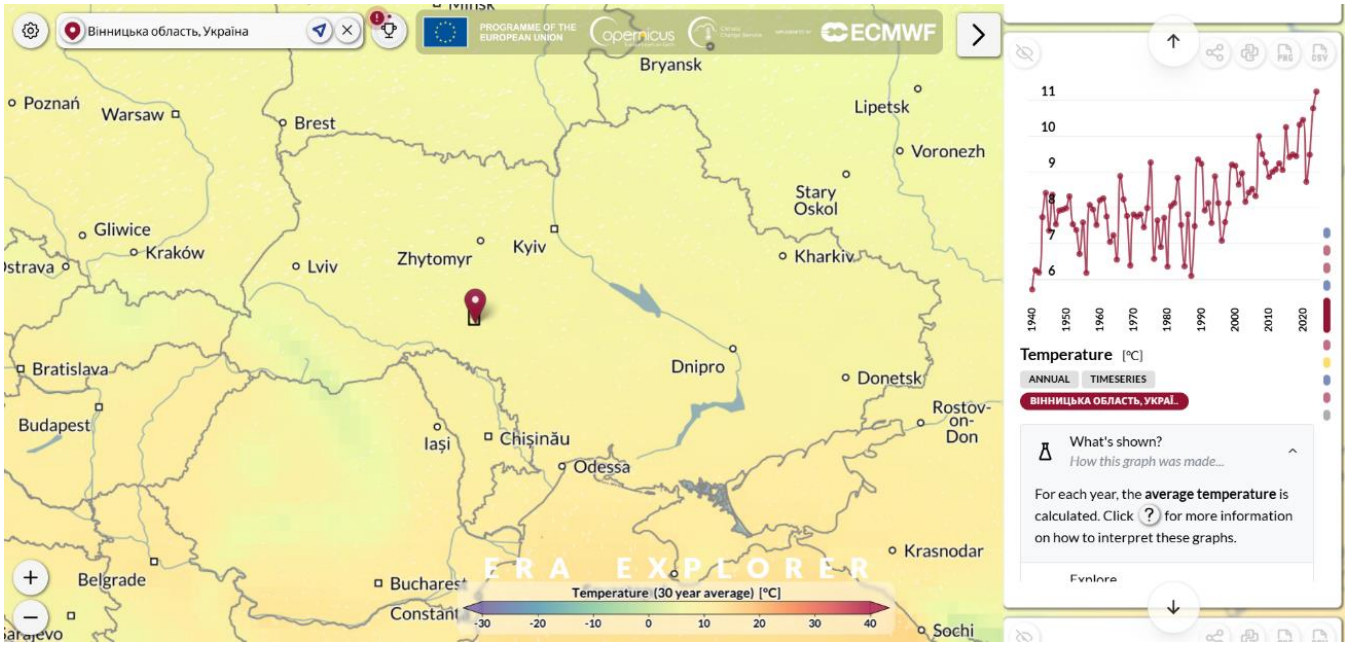

Рисунок 1.4 – Застосування веб-платформа Copernicus Climate Data Store

Веб-сервіс Ventusky поєднує метеорологічні моделі з інтерактивною картою для відображення опадів, хмарності, вітру, атмосферного тиску та рівня води. Платформа має привабливий інтерфейс, зручний для візуалізації погодних

умов у режимі реального часу (рис. 1.5–1.6). Завдяки високій деталізації й анімації даних Ventusky широко використовується як звичайними користувачами, так і фахівцями у сфері навколишнього середовища [6].

Переваги: зручна візуалізація, висока деталізація, доступність у браузері та мобільному додатку.

Недоліки: обмежена можливість завантаження даних, відсутність інструментів для складного аналізу.

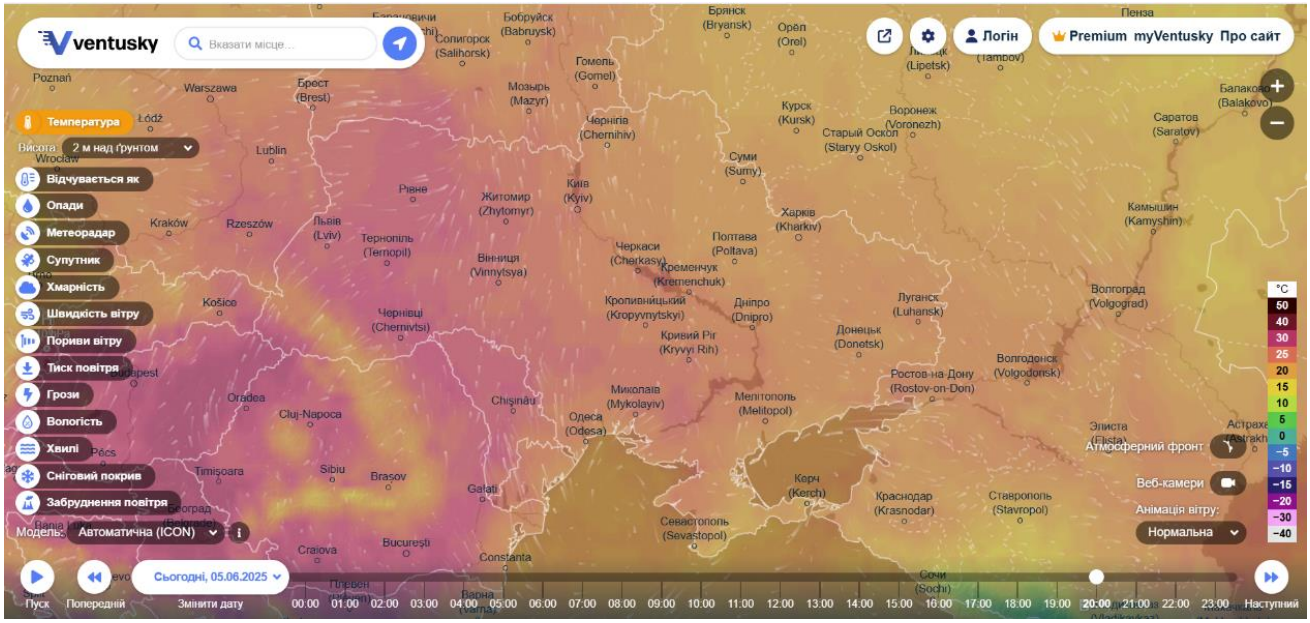

Рисунок 1.5 – Інтерфейс Ventusky

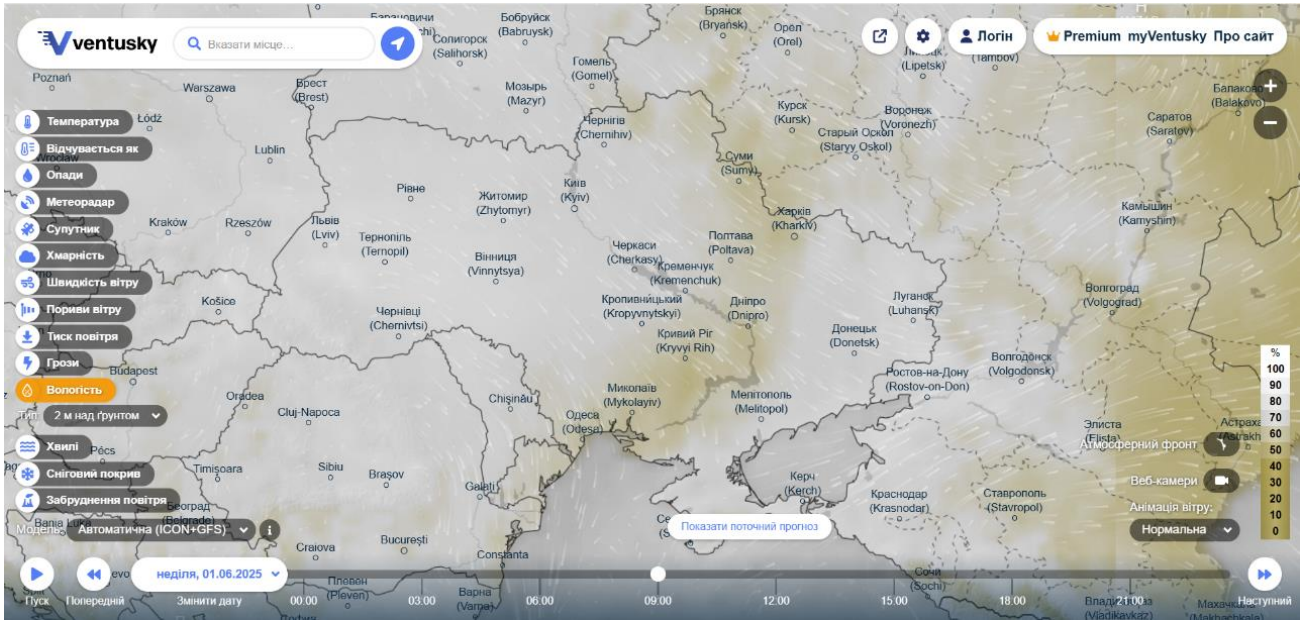

Рисунок 1.6 – Візуалізація опадів і рівня води у Ventusky

Google Earth Engine — потужна платформа для аналізу супутникових знімків і геопросторових даних. Вона дозволяє працювати з великими масивами даних у хмарі, застосовувати алгоритми обробки, аналізувати довгострокові тренди та будувати складні аналітичні моделі (рис. 1.7–1.8). Для гідрологічних задач тут доступні шари про річки, водосховища, вологість ґрунту, а також індекси водяного покриву [7].

Переваги: масштабованість, підтримка мов програмування JavaScript і Python, автоматизація обробки.

Недоліки: крута крива навчання, необхідність програмних навичок.

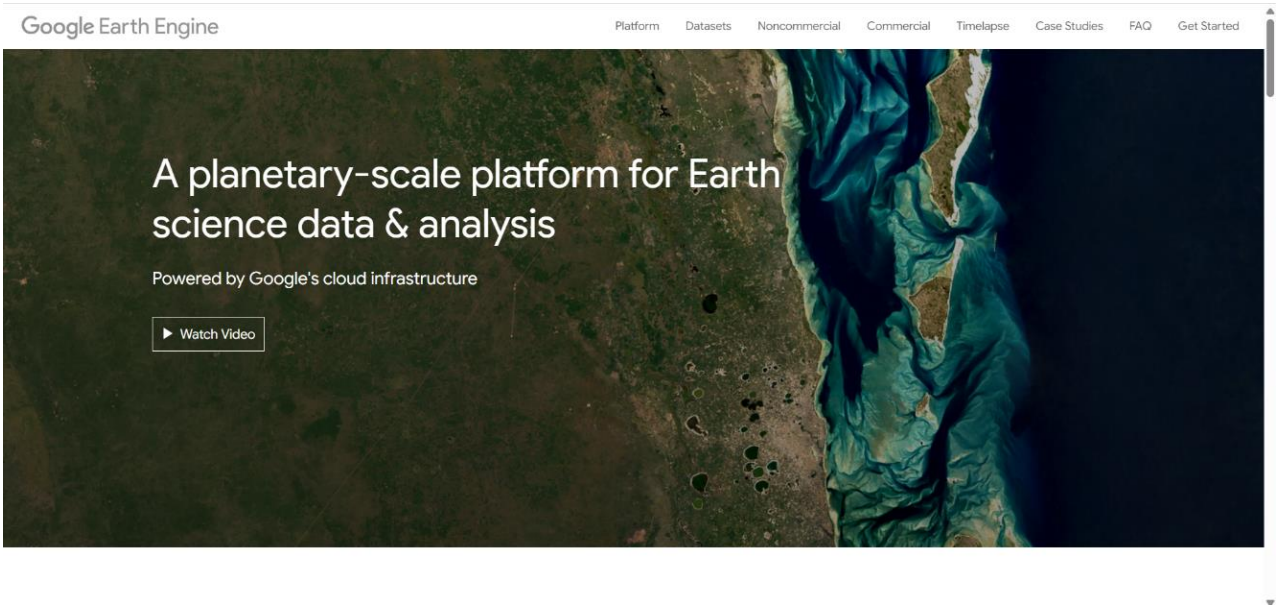

Рисунок 1.7 – Інтерфейс Google Earth Engine

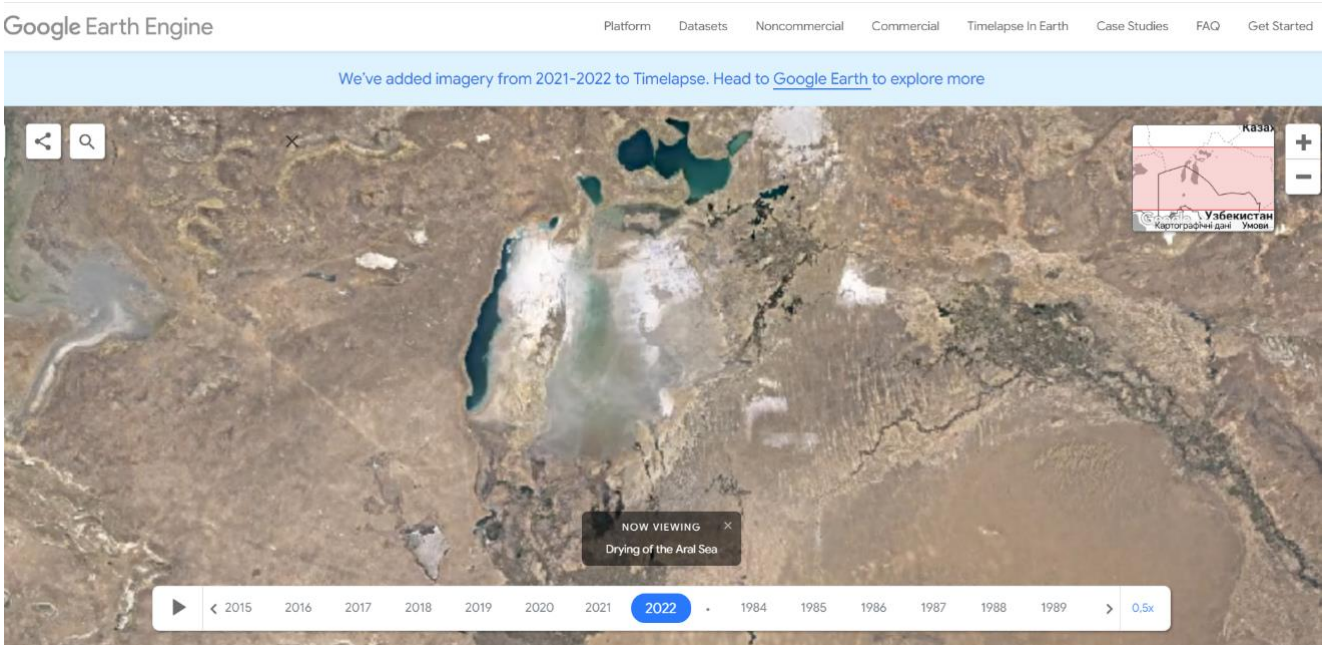

Рисунок 1.8 – Візуалізація гідрологічних шарів у Google Earth Engine

Також широко використовується система ArcGIS, яка включає спеціалізовані інструменти для аналізу водозборів, побудови цифрових моделей рельєфу, обчислення стоку та інтеграції з іншими джерелами гідрологічної інформації (рис. 1.9–1.10). ArcGIS дозволяє виконувати детальний просторово-часовий аналіз і створювати інтерактивні карти з прив’язкою до атрибутивних даних [8].

Переваги: потужний набір інструментів, висока точність аналізу, підтримка 3D-візуалізації.

Недоліки: платна ліцензія, висока ресурсомісткість.

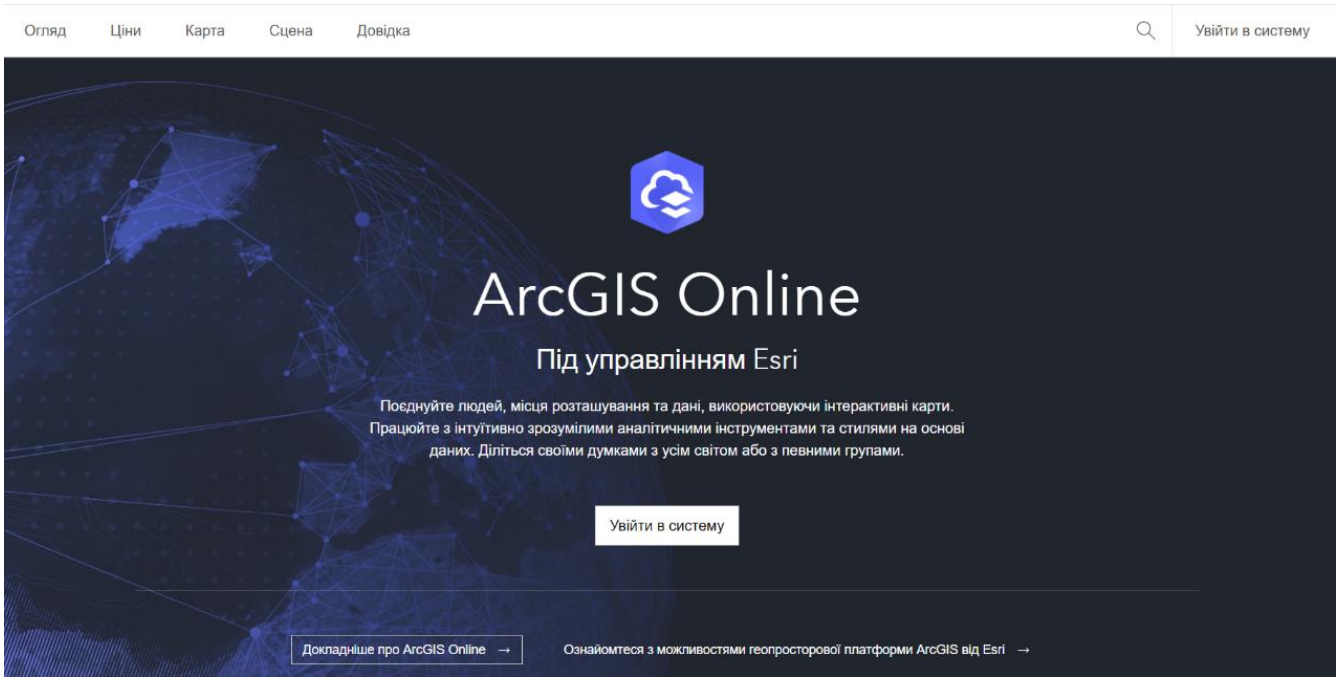

Рисунок 1.9 – Інтерфейс ArcGIS

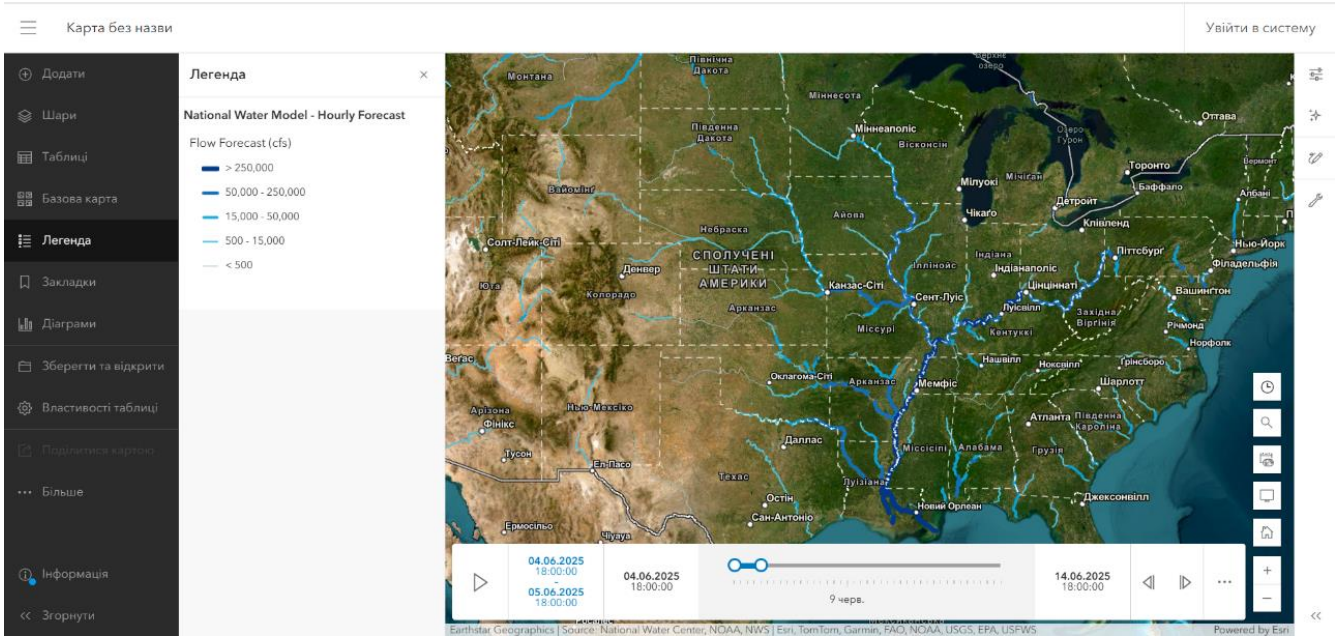

Рисунок 1.10 – Приклад гідрологічного аналізу в ArcGIS

Проте жодна з розглянутих платформ не пропонує повного рішення для локального аналізу басейну Південного Бугу з можливістю збереження даних, налаштування фільтрів, інтеграції з картою гідропостів та генерації звітів. Це обґрунтовує необхідність розробки нової інформаційної системи з урахуванням локальних особливостей, зберіганням даних у хмарній базі (наприклад, Supabase) та підтримкою розширеного функціоналу для кінцевого користувача/

1.4 Висновки

Проведений аналіз показав актуальність створення сучасної веб-платформи для просторово-часового аналізу кількості води в басейні Південного Бугу. Існуючі системи не повністю задовольняють потреби у локальному аналізі, зручному інтерфейсі та інтеграції з актуальними джерелами даних. Розробка нового рішення з використанням хмарної бази даних та інтерактивної карти є доцільною та перспективною.

2 ВИБІР ОПТИМАЛЬНОЇ ІТ-ІНФРАСТРУКТУРИ

2.1 Формування системних вимог

Оптимальна інфраструктура для створення інформаційної системи визначається її функціональними потребами. На основі проведеного аналізу доцільно сформувати мінімально життєздатний продукт (MVP) — базову версію системи, що включає лише ключові функції для задоволення перших користувачів.

Основні функціональні вимоги до інформаційної системи просторово-часового аналізу об’ємів води у басейні річки Південний Буг включають:

- пошук даних за заданими фільтрами (басейн, рік, місяць, тип графіка);
- побудову графіків для зручної візуалізації;
- інтеграцію карти з маркерами гідрологічних постів;
- відображення детальної інформації при натисканні на маркер;
- генерацію звітів у форматі Excel з графіком і таблицею даних.

Відповідно до підходу MVP, остаточний функціонал визначається на основі зворотного зв’язку від перших користувачів[9]. Потенційне розширення системи може включати:

- сповіщення про оновлення даних;
- інтерактивний дашборд для аналізу динаміки у різні періоди;
- функцію прогнозування на основі історичних даних;
- інтеграцію з зовнішніми API для отримання актуальної інформації;
- можливість експорту звітів у формати PDF, CSV тощо.

Ці вимоги орієнтовані на реальні потреби користувачів і можуть слугувати основою для прийняття рішень щодо подальшого розвитку проєкту.

Найкращим рішенням для реалізації такої системи є веб-архітектура, яка забезпечує доступність, масштабованість та зручність роботи з великими обсягами даних. Сучасні аналітичні інструменти часто реалізуються саме як веб-додатки, оскільки це дає змогу користувачам отримувати доступ до системи з будь-якого пристрою з інтернетом.

Ключова перевага веб-архітектури — незалежність від операційної системи чи пристрою. Враховуючи специфіку користувачів (екологи, адміністратори, оператори моніторингу), які можуть працювати як в офісі, так і в польових умовах, веб-додаток є найзручнішим рішенням. Він дає змогу швидко фільтрувати дані, переглядати графіки, створювати звіти та аналізувати інформацію без потреби в установці додаткового ПЗ.

Крім того, веб-інтерфейс дозволяє реалізувати інтерактивні елементи (графіки, карти), а його адаптивність забезпечує комфортну роботу як на ПК, так і на мобільних пристроях. Це сприяє покращенню взаємодії з системою та користувацькому досвіду загалом.

З технічної точки зору веб-архітектура забезпечує інтеграцію з хмарними базами даних (наприклад, Supabase), картографічними платформами (Google Maps, OpenStreetMap) та можливість обробки великих обсягів інформації. Такий підхід забезпечує гнучкість і масштабованість залежно від потреб користувачів і подальших функціональних розширень.

Не менш важливим є й аспект безпеки: веб-архітектура дозволяє впровадити захист API, шифрування даних, керування правами доступу та безпечне зберігання даних у хмарі. Це критично важливо для захисту гідрологічної інформації в межах екологічного моніторингу.

Отже, веб-архітектура є найбільш доцільним варіантом для реалізації інформаційної системи просторово-часового аналізу водних ресурсів у басейні Південного Бугу, оскільки вона максимально відповідає як потребам користувачів, так і сучасним технічним вимогам.

2.2 Вибір програмного забезпечення

Для успішної розробки інформаційної системи необхідно коректно визначити складові програмного забезпечення. Вибір середовища розробки (IDE), мови програмування, способу збереження та відображення даних, а також інтеграції додаткового функціоналу (наприклад, Chart.js для побудови графіків) залежить від заздалегідь визначених вимог до системи. Оскільки ці вимоги вже

сформульовано, наступним кроком є вибір інструментів і технологій, які забезпечать реалізацію функціоналу інформаційної системи у вигляді веб-додатку.

Серед основних технологій, які зазвичай використовуються для створення веб-додатків, можна виділити:

- UI-фреймворки та бібліотеки, які забезпечують адаптивний і зручний інтерфейс (наприклад, React, Vue.js, Angular, Tailwind CSS);
- Бібліотеки для побудови графіків (Chart.js, D3.js), що дозволяють ефективно візуалізувати дані;
- Системи зберігання даних, зокрема хмарні бази даних (Supabase, Firebase, PostgreSQL), які дають змогу зберігати та обробляти гідрологічну інформацію;
- Картографічні сервіси (Google Maps API, OpenStreetMap) для відображення розташування гідрологічних постів;
- API та засоби інтеграції (REST API, Fetch API, Axios) для обміну даними з сервером;
- Інструменти керування станом (React Context, Redux, Zustand) для керування даними інтерфейсу;
- Системи контролю версій, такі як Git, для зберігання історії змін, командної роботи та версіювання коду;
- Інструменти для тестування (Jest, React Testing Library) для перевірки коректності роботи додатку.

Вибір конкретних технологій залежить як від функціональних потреб проєкту, так і від досвіду команди розробників. Full-stack спеціалісти можуть охоплювати як фронтенд, так і бекенд розробку, що сприяє ефективності, хоча й потребує постійного оновлення знань у широкому спектрі технологій.

Інтегроване середовище розробки (IDE) — це набір інструментів для створення програмного забезпечення, який включає редактор коду, відлагоджувач, засоби роботи з версіями та інше. Серед популярних IDE для веб-розробки:

- Visual Studio Code — універсальне середовище від Microsoft із підтримкою JavaScript, TypeScript, Python тощо, а також багатьма розширеннями для React, Git, автодоповнення коду та ін [10].
- WebStorm — спеціалізоване IDE для JavaScript і фреймворків React, Angular, Vue.js [11].
- PyCharm — зосереджене на Python, зокрема для розробки на Flask або Django [12].
- IntelliJ IDEA — багатофункціональне IDE, що підтримує повноцінну full-stack розробку [13].

Visual Studio Code — один із найкращих варіантів для веб-розробки завдяки своїй легкості, гнучкості та великій спільноті (рис. 2.1) [10].

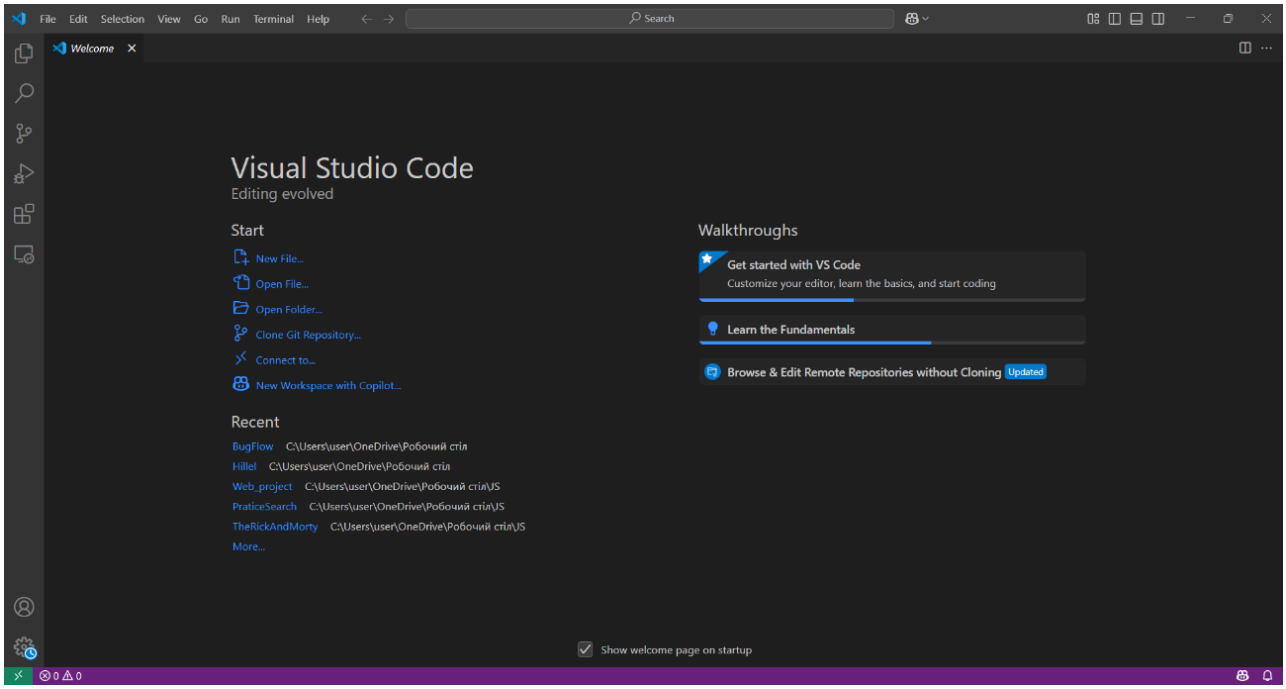

Рисунок 2.1 – Огляд IDE Visual Studio Code

Для розробки інформаційної системи просторово-часового аналізу гідрологічних даних у басейні річки Південний Буг було обрано React з мовою програмування JavaScript[20]. React — це популярна бібліотека для побудови інтерфейсів, яка завдяки компонентній архітектурі забезпечує високу гнучкість і масштабованість [14].

Переваги використання React і JavaScript:

- Модульність: розробка окремих компонентів для графіків, фільтрів, карт тощо;
- Інтерактивність: швидке оновлення інтерфейсу завдяки Virtual DOM;
- Широка екосистема: доступність численних бібліотек (Chart.js, Axios, React Router);
- Асинхронна робота з API: ефективне завантаження та оновлення даних;
- Сумісність із сучасними сервісами, зокрема Supabase для роботи з базами даних.

Оскільки додаток працюватиме з великими обсягами даних, ключову роль відіграватиме інтеграція з хмарною базою даних Supabase, яка побудована на PostgreSQL. Supabase дозволяє зберігати гідрологічні дані у хмарі, забезпечуючи масштабованість, доступність і безпеку [15].

Структура бази даних:

- unit — для зберігання ідентифікаторів і назв одиниць вимірювання.
- river — для збереження ідентифікаторів і назв річок.
- region — містить ідентифікатори та назви регіонів.
- indicator — слугує для зберігання ідентифікаторів і назв показників із зв’язком до одиниць вимірювання.
- hydrological_post — для збереження ідентифікаторів постів, їх назв, площі басейну та зв’язків із річками й регіонами.
- hydrological_data — для зберігання часових рядів гідрологічних даних із зв’язками до постів і показників.

Переваги Supabase:

- REST API і JavaScript SDK для легкої інтеграції;
- підтримка автентифікації, авторизації та шифрування;
- можливість масштабування зі зростанням обсягів даних.

Для візуалізації даних буде використано Chart.js, що дозволить створювати:

- різні типи графіків (лінійні, стовпчикові, кругові);
- адаптивні графіки з налаштуванням кольорів, підписів, легенд;
- інтерактивні елементи, які реагують на дії користувача.

Chart.js легко інтегрується з React, що робить його ідеальним для аналітичного відображення гідрологічних даних [16].

У підсумку, визначено ключові елементи програмного забезпечення, які дадуть змогу реалізувати веб-додаток для просторово-часового аналізу водних ресурсів річки Південний Буг відповідно до сформ ульованих вимог.

2.3 Вибір апаратного забезпечення

У процесі проєктування та розробки інформаційної системи просторово-часового аналізу кількості води у річковому басейні Південного Бугу важливим аспектом стало визначення апаратних вимог до майбутнього програмного забезпечення. Система реалізується у вигляді веб-додатку, що забезпечує кросплатформену сумісність, масштабованість та доступність для кінцевих користувачів без необхідності встановлення спеціального програмного забезпечення.

Уся бізнес-логіка, функціональність інтерфейсу, візуалізація просторових та часових даних, а також базовий аналіз інформації здійснюються на стороні клієнта — у браузері користувача. Такий підхід дозволяє зменшити навантаження на серверну частину та пришвидшити взаємодію з системою. Основні функціональні компоненти, як-от фільтрація даних за часовими та просторовими критеріями, побудова графіків за допомогою бібліотеки Chart.js, інтерактивне відображення гідрологічних постів за допомогою Leaflet та формування звітів у форматі Excel за допомогою бібліотеки XLSX, працюють у режимі реального часу без значної затримки, що забезпечує комфортну роботу навіть на пристроях із середнім рівнем продуктивності [16 – 18].

Для забезпечення актуальності даних та підтримки зберігання інформації у системі застосовується хмарна платформа Supabase, побудована на основі PostgreSQL [19]. Вона виконує функції серверної частини — обробляє запити до бази даних, забезпечує збереження, фільтрацію та запитування гідрологічної інформації. Наявність стабільного інтернет-з’єднання є обов’язковою умовою

для коректної роботи системи, оскільки обмін даними з Supabase та завантаження карт із зовнішніх ресурсів здійснюється у хмарному середовищі [15].

Розроблений веб-додаток є платформонезалежним і сумісним з більшістю сучасних операційних систем та пристроїв. Підтримуються наступні платформи:

- Настільні комп’ютери з операційними системами Windows, macOS, Linux;
- Ноутбуки на базі Windows, macOS, Linux;
- Планшети та смартфони на платформах iOS та Android за умови використання сучасних браузерів;
- Віртуальні середовища тестування, зокрема BrowserStack, що дозволяє перевірити коректність роботи додатку на різних пристроях і конфігураціях.

Для коректного відображення інтерфейсу та повної функціональності веб-додатку потрібні браузери, що підтримують HTML5, CSS3 та JavaScript ES6[21]. Станом на 2023 рік такими браузерами є:

- Google Chrome (версія 58 і новіші),
- Mozilla Firefox (версія 54 і новіші),
- Microsoft Edge (версія 15 і новіші),
- Safari (версія 11 і новіші),
- Opera (версія 45 і новіші).

Використання бібліотеки React забезпечує ефективну оптимізацію рендерингу, що дозволяє веб-додатку працювати швидко навіть на пристроях зі скромними технічними характеристиками (наприклад, смартфонах або бюджетних ноутбуках).

На етапі розробки та тестування система перевіряється на локальному комп’ютері розробника з операційною системою Windows 11, процесором Intel Core i5, 16 ГБ оперативної пам’яті та стабільним підключенням до Інтернету зі швидкістю не менше 25 Мбіт/с. Також тестування відбувається у віртуальних середовищах, що дозволяє моделювати роботу додатку на інших платформах і пристроях.

У перспективі система може бути розгорнута на сучасних платформах хостингу для веб-додатків, таких як Vercel або Netlify. Вони забезпечують:

- автоматичне деплоймент-розгортання з GitHub-репозиторію;
- безперервну інтеграцію та оновлення змін;
- масштабовану інфраструктуру з CDN-підтримкою;
- можливість збору аналітики та зворотного зв’язку від користувачів;
- інтеграцію з REST API для взаємодії з іншими сервісами.

У майбутньому можливе розширення функціональності системи, що також впливатиме на вимоги до апаратного забезпечення. Зокрема, планується реалізація таких функцій:

- Синхронізація даних між користувачами для групової роботи;
- Централізоване зберігання історичних гідрологічних даних з можливістю аналізу тенденцій;
- Отримання актуальної інформації через зовнішні API, наприклад, від гідрометеорологічних служб;
- Формування звітів у декількох форматах одночасно;
- Розширення клієнтської частини із додатковими фільтрами та інструментами для дослідження.

Таким чином, обрана архітектура та апаратно-програмне забезпечення дозволяють створити доступну, масштабовану та ефективну систему, яка забезпечує стабільну роботу на різних типах пристроїв та відкриває широкі можливості для подальшого розвитку.

2.4 Висновки

Для створення інформаційної системи аналізу об’ємів води в басейні річки Південний Буг доцільно використати веб-архітектуру, яка забезпечує доступність, масштабованість і зручність роботи. Система реалізується за допомогою JavaScript і React для інтерфейсу, Supabase для зберігання даних, а також бібліотек Chart.js і картографічних сервісів для візуалізації. Основний функціонал включає фільтрацію даних, графіки, карту з маркерами, детальну

інформацію та експорт звітів. Такий підхід дозволяє створити зручну, сучасну та функціональну систему з можливістю подальшого розширення.

3 ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

3.1 Характеристика архітектури інформаційної системи

Інфраструктура веб-додатку, який виконує просторово-часовий аналіз об’ємів води у басейні річки Південний Буг, побудована на основі сучасної клієнт-серверної архітектури. Вона реалізована з використанням React та компонентного підходу, що є рекомендованим стандартом при розробці масштабованих веб-застосунків.

Така архітектура забезпечує чіткий поділ функціональності між компонентами, що сприяє зручності підтримки, розширення та тестування системи. Основна логіка взаємодії з користувачем — фільтрація, побудова графіків, візуалізація даних на карті — виконується на клієнтському боці у браузері, тоді як зберігання та оновлення інформації здійснюється через хмарну платформу Supabase, що передбачає обов’язкове підключення до інтернету.

Архітектура системи умовно поділяється на три рівні:

1. Рівень представлення (UI)

Цей рівень реалізований у вигляді React-компонентів з використанням CSS-модулів, styled-components або SCSS. Він відповідає за інтерфейс користувача: вибір фільтрів (басейн, рік, місяць), перегляд графіків, інтерактивну карту та генерацію звітів. Компоненти UI не містять логіки обробки даних, а лише відображають їх і реагують на дії користувача.

2. Рівень бізнес-логіки (State Management)

Цей рівень забезпечує логіку взаємодії між UI та джерелами даних. Він обробляє введені параметри, надсилає запити до Supabase, перетворює дані для візуалізації (наприклад, для Chart.js) і керує станом додатку. Для управління станом використовуються React Context або бібліотеки типу Zustand, що дозволяє автоматично оновлювати інтерфейс при зміні вхідних параметрів.

3. Рівень доступу до даних (Data Layer)

Цей рівень відповідає за зберігання, отримання та обробку даних. Інформація зберігається у хмарній базі Supabase (на основі PostgreSQL). Доступ

до даних забезпечується через REST API або офіційний JavaScript-клієнт. Усі запити винесені в окремі сервіси для централізованої обробки. Такий підхід також дозволяє в майбутньому інтегрувати додаткові джерела даних.

Структура бази даних включає основні таблиці:

- unit — зберігає ідентифікатори та назви одиниць вимірювання;
- river — містить ідентифікатори та назви річок;
- region — для зберігання регіонів;
- indicator — включає показники з прив’язкою до одиниць вимірювання;
- hydrological_post — описує пости, їх площі басейну, належність до річок і регіонів;
- hydrological_data — містить часові ряди гідрологічних спостережень, пов’язані з постами та показниками.

Використання зовнішніх ключів гарантує цілісність даних, а індекси підвищують ефективність фільтрації.

На етапі розробки система повністю інтегрована із Supabase, проте в майбутньому можливе масштабування — підключення додаткових хмарних сервісів для реалізації нових можливостей: прогнозування, синхронізація користувачів, push-сповіщення тощо.

Для візуалізації даних на карті використовується бібліотека Leaflet. Вона легко інтегрується з React через react-leaflet і надає такі функції[17]:

- масштабування мапи;
- маркери гідрологічних постів;
- інформаційні вікна при взаємодії з маркерами;
- підтримка власних тайлових шарів.

Leaflet — легка та незалежна від сторонніх сервісів бібліотека, що робить її оптимальним рішенням для відображення просторових даних.

Щодо безпеки, Supabase забезпечує базовий рівень конфіденційності — авторизацію та шифрування даних. Після масштабування додатку будуть впроваджені додаткові заходи: шифровані з’єднання (HTTPS, SSL/TLS), обмеження доступу до API через токени, налаштування прав доступу до таблиць у Supabase.

3.2 UML-діаграми

Для кращого розуміння архітектури веб-додатку для просторово-часового аналізу кількості води у річковому басейні Південного Бугу та взаємодії між його компонентами, було розроблено низку UML-діаграм. Вони дозволяють наочно представити структуру додатку, логіку аналізу даних і обробки запитів.

Діаграма випадків використання (Use Case Diagram)

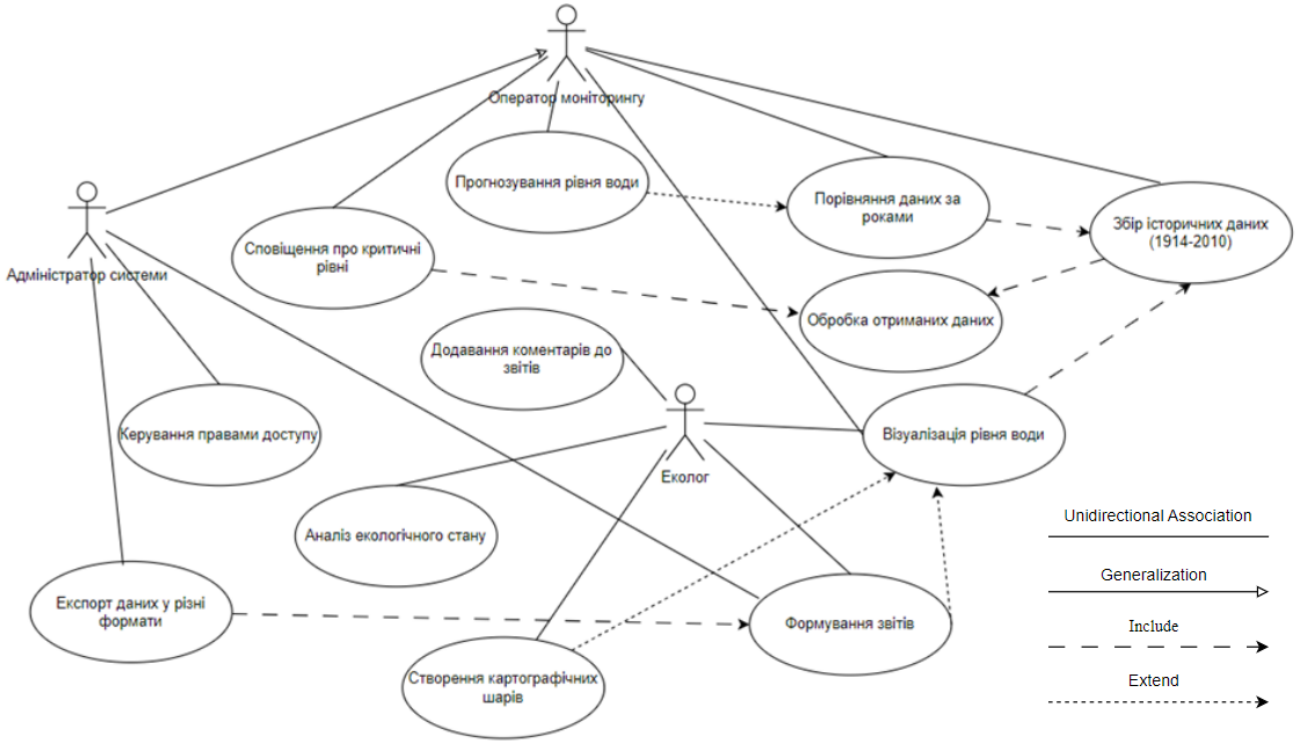

Рисунок 3.1 – Діаграма випадків використання

На рисунку 3.1 представлено діаграму випадків використання, яка ілюструє взаємодію основних акторів із системою: оператор моніторингу, адміністратор системи та еколог.

Згідно з діаграмою, кожен актор має доступ до визначеного набору функцій:

Оператор моніторингу:

- Переглядати рівень води.
- Формувати звіти.

- Отримувати сповіщення про критичний рівень води.

Адміністратор системи:

- Керувати користувачами.
- Встановлювати критичні рівні.
- Управляти правами доступу.

Еколог:

- Аналізувати екологічні дані.
- Переглядати історичні показники рівня води.
- Досліджувати вплив змін.

Типи зв’язків:

Unidirectional Association – пряма асоціація, наприклад, оператор переглядає рівень води.

Generalization – узагальнення, наприклад, адміністратор успадковує можливості оператора.

Extend Relationship – альтернативний сценарій, наприклад, сповіщення розширює перегляд рівня води при критичних значеннях.

Include Relationship – обов’язковий підпроцес, наприклад, формування звітів включає перегляд рівня води.

Ця діаграма демонструє розмежування функціоналу залежно від ролі, забезпечуючи гнучкість і безпеку системи.

Діаграма послідовності: «перегляд рівня води»

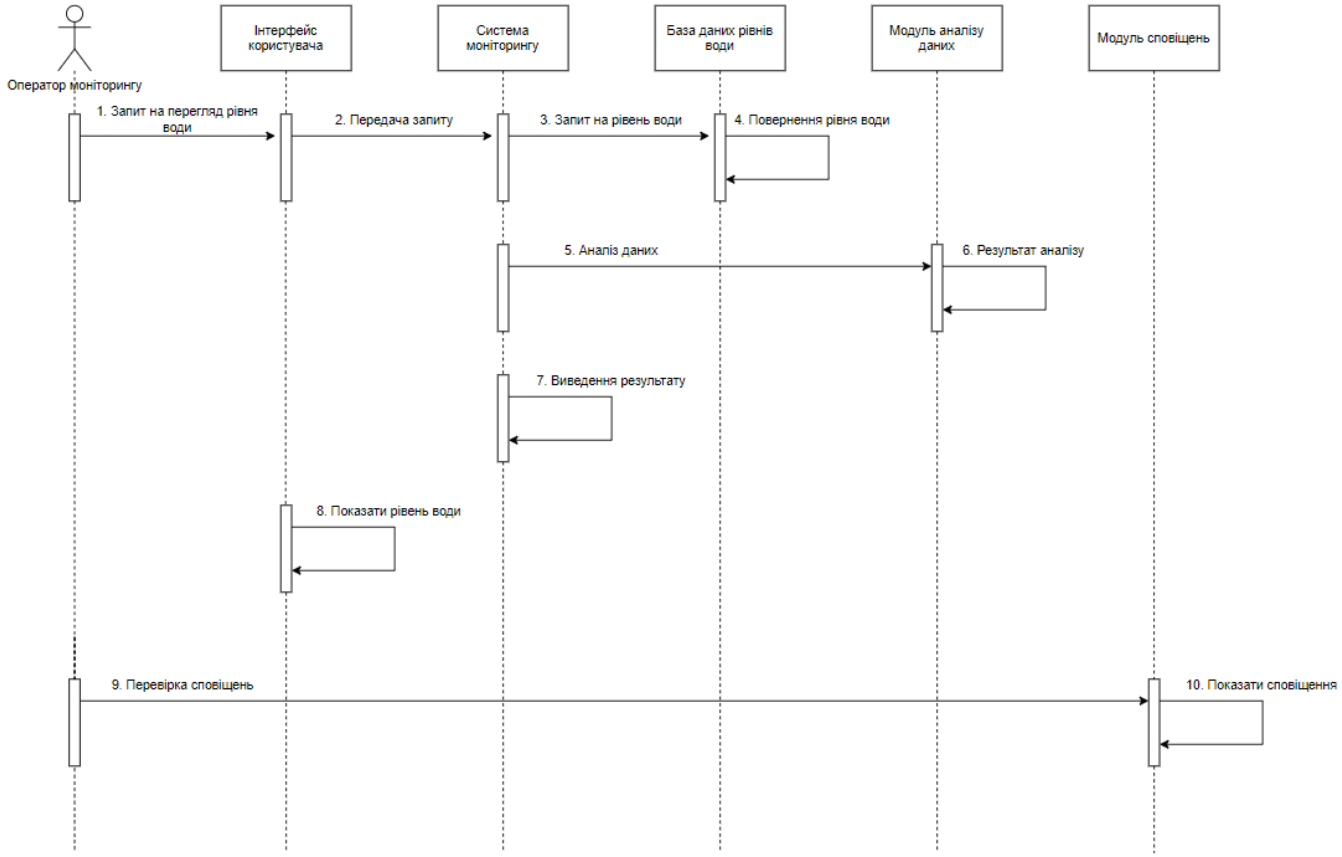

Рисунок 3.2 – Діаграма послідовності: «перегляд рівня води»

На рисунку 3.2 представлено діаграму послідовності, що демонструє процес введення даних про рівень води оператором з повідомленням про критичний рівень.

Учасники процесу (об’єкти):

1. Оператор моніторингу – ініціює запит.
2. UserInterface – відображає інтерфейс.
3. DataRequest – обробляє запит до бази.
4. SupabaseClient – взаємодіє з Supabase.
5. HydrologicalData – повертає дані.
6. MapRenderer – відображає дані на карті.
7. NotificationService – сповіщає про критичні значення.

Послідовність дій:

1. Оператор відкриває інтерфейс.
2. UserInterface надсилає запит на перегляд.

3. DataRequest передає запит до SupabaseClient.
4. SupabaseClient виконує SQL-запит.
5. HydrologicalData повертає дані рівня води.
6. DataRequest обробляє дані.
7. MapRenderer відображає дані на карті.
8. UserInterface оновлює екран.
9. Якщо рівень критичний, DataRequest надсилає сигнал.
10. NotificationService надсилає сповіщення.

Ця діаграма ілюструє структурований процес додавання точки спостереження, підкреслюючи автоматизацію та інформування користувачів, що сприяє ефективному управлінню системою.

Діаграма кооперації: “Блокування користувача адміністратором”.

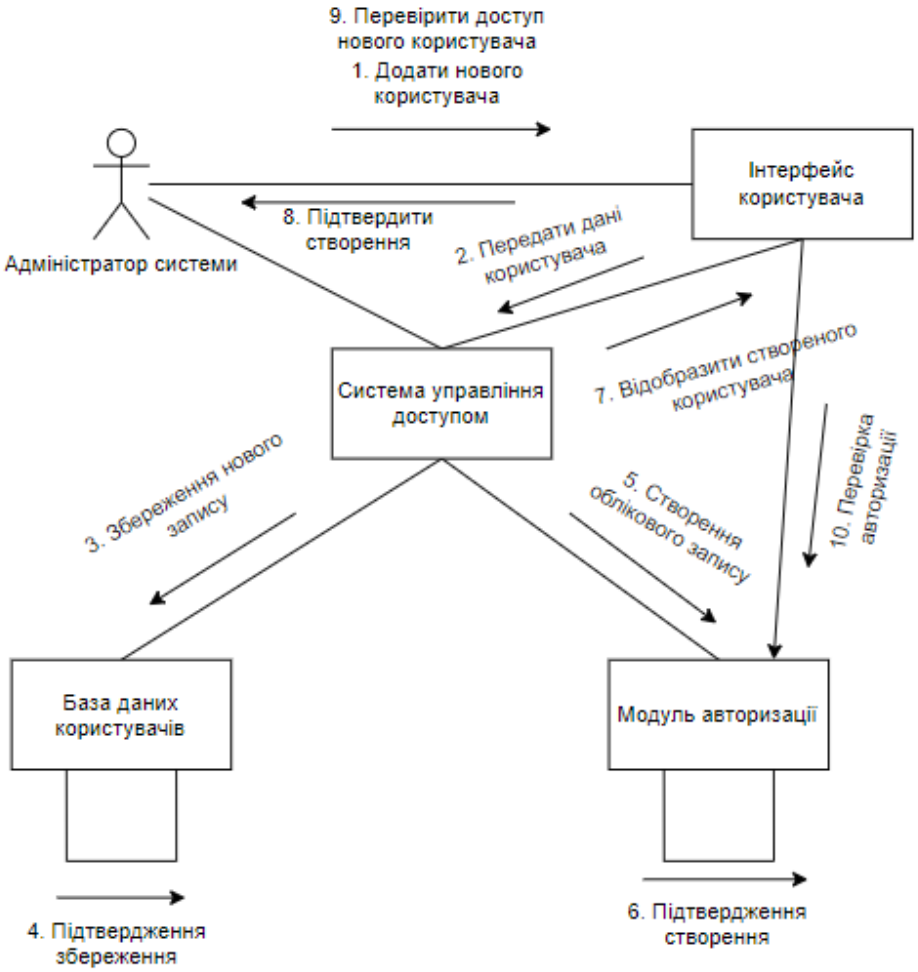

Рисунок 3.3 – Діаграма кооперації: “ Додавання нової точки спостереження адміністратором ”

На рисунку 3.3 представлено діаграму класів, яка описує об’єктно-орієнтовану модель системи для обробки даних про рівень води. Діаграма відображає ключові класи, їхні взаємозв’язки та методи, які забезпечують збір, збереження, аналіз і візуалізацію даних.

Основні учасники процесу:

- UserInterface: клас, відповідальний за взаємодію користувача із системою. Забезпечує відображення даних (графіки, звіти, список вимірювань) та обробку введення (дані про рівень води, вибір точки спостереження).
- MonitoringSystem: центральний клас, який координує збір, обробку та аналіз даних про рівень води. Відповідає за генерацію звітів і взаємодію з іншими модулями.
- WaterLevelDB: база даних, що зберігає інформацію про вимірювання рівня води (значення, дата, точка спостереження). Надає методи для збереження, пошуку та оновлення даних.
- SensorDB: база даних, що містить інформацію про сенсори (наприклад, їхні ідентифікатори та точки спостереження). Використовується для прив’язки вимірювань до конкретних датчиків.
- EcoDataDB: база екологічних даних, що зберігає показники стану водного середовища (наприклад, температура води, рівень pH). Надає дані для аналізу екологами.
- DataAnalysisModule: модуль для аналізу даних про рівень води та екологічні показники. Генерує статистичні звіти та виявляє аномалії (наприклад, перевищення критичних рівнів).
- NotificationModule: модуль для перевірки даних на перевищення критичних рівнів і надсилання сповіщень користувачам (адміністратору або екологу).
- EcoReports: клас для створення, збереження та відображення екологічних звітів. Дозволяє експортувати звіти у текстовому форматі або як графіки.

- GraphGenerator: клас для створення графіків змін рівня води з використанням бібліотеки GraphView. Забезпечує візуалізацію даних за вибраний період.

Основні зв’язки:

1. Асоціація: UserInterface взаємодіє з MonitoringSystem для відображення даних і передачі введених користувачем даних.
2. Композиція: MonitoringSystem включає DataAnalysisModule, NotificationModule і GraphGenerator для виконання основних функцій.
3. Агрегація: EcoReports використовує дані з WaterLevelDB і EcoDataDB для формування звітів.
4. Залежність: GraphGenerator залежить від WaterLevelDB для отримання даних для побудови графіків.
5. Асоціація: SensorDB пов’язаний із WaterLevelDB, оскільки вимірювання прив’язані до конкретних сенсорів.

Ця діаграма класів демонструє модульну структуру системи, де кожен клас має чітко визначені обов’язки. Відсутність авторизації спрощує взаємодію з системою, а зв’язки між класами забезпечують ефективну обробку даних від введення до аналізу та візуалізації. Модель підтримує розширення функціоналу, наприклад, інтеграцію з автоматичними сенсорами або зовнішніми гідрологічними сервісами.

Діаграма станів для процесу аналізу даних.

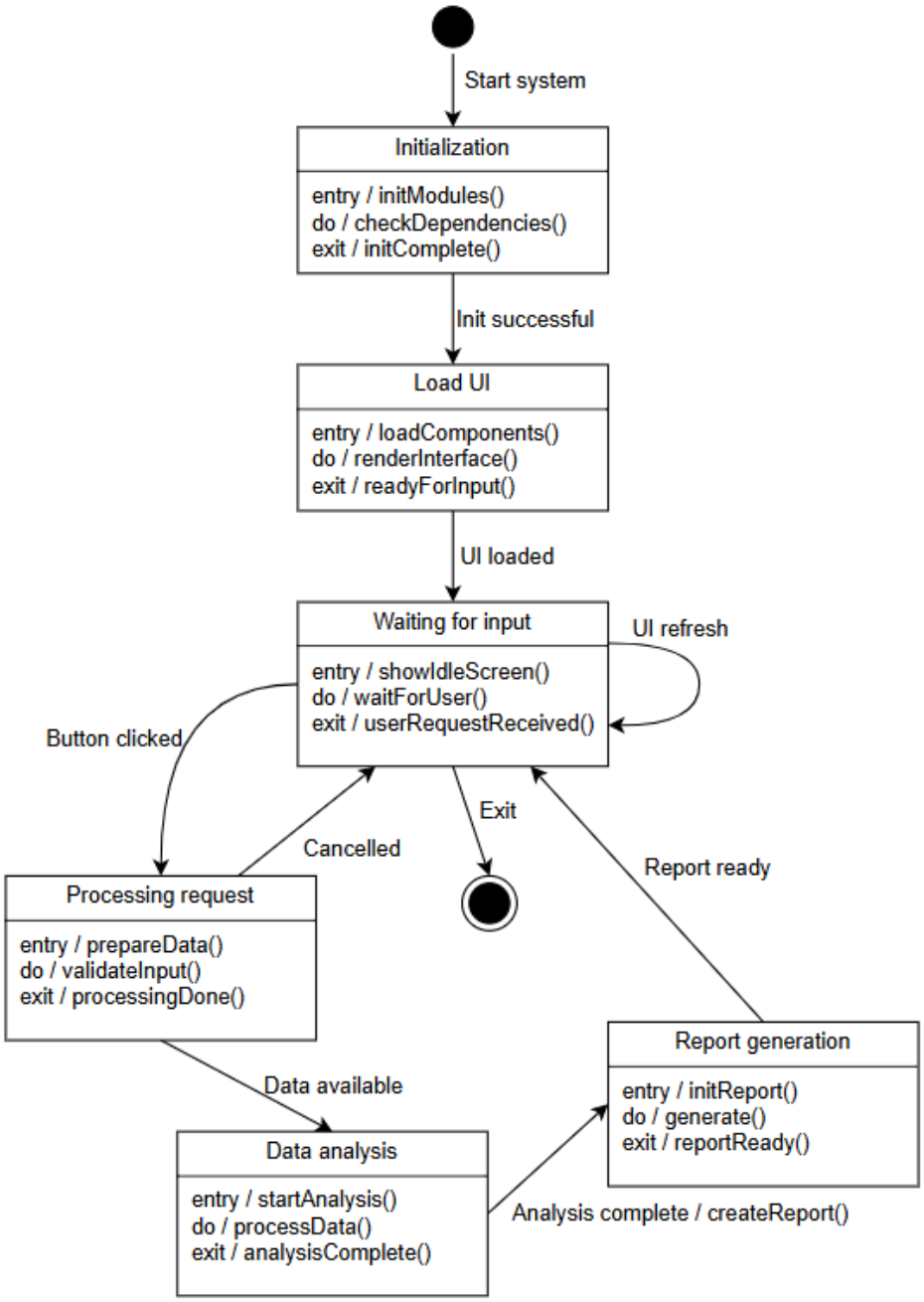

Рисунок 3.4 – Діаграма станів для всієї програми

На рисунку 3.4 зображено діаграму станів, яка описує процес обробки та аналізу гідрологічних даних у системі. Вона відображає всі етапи роботи системи під час виконання аналізу даних про кількість води у басейні Південного Бугу.

Основні гілки:

– Очікування дій користувача («Waiting_for_input») — головний стан, з якого ініціюється обробка запитів.

– Після натискання кнопки користувача перехід у стан обробки запиту («Processing_request»), далі — аналіз даних («Data_analysis») та формування звіту («Report_generation»).

– Після завершення генерації звіту система повертається до очікування.

– У разі скасування обробки — повернення до очікування.

– Передбачено рефлексивний перехід для оновлення інтерфейсу без зміни стану.

На рисунку 3.5 зображено структуру таблиць бази даних інформаційної системи моніторингу кількості вод річкового басейну Південного Бугу.

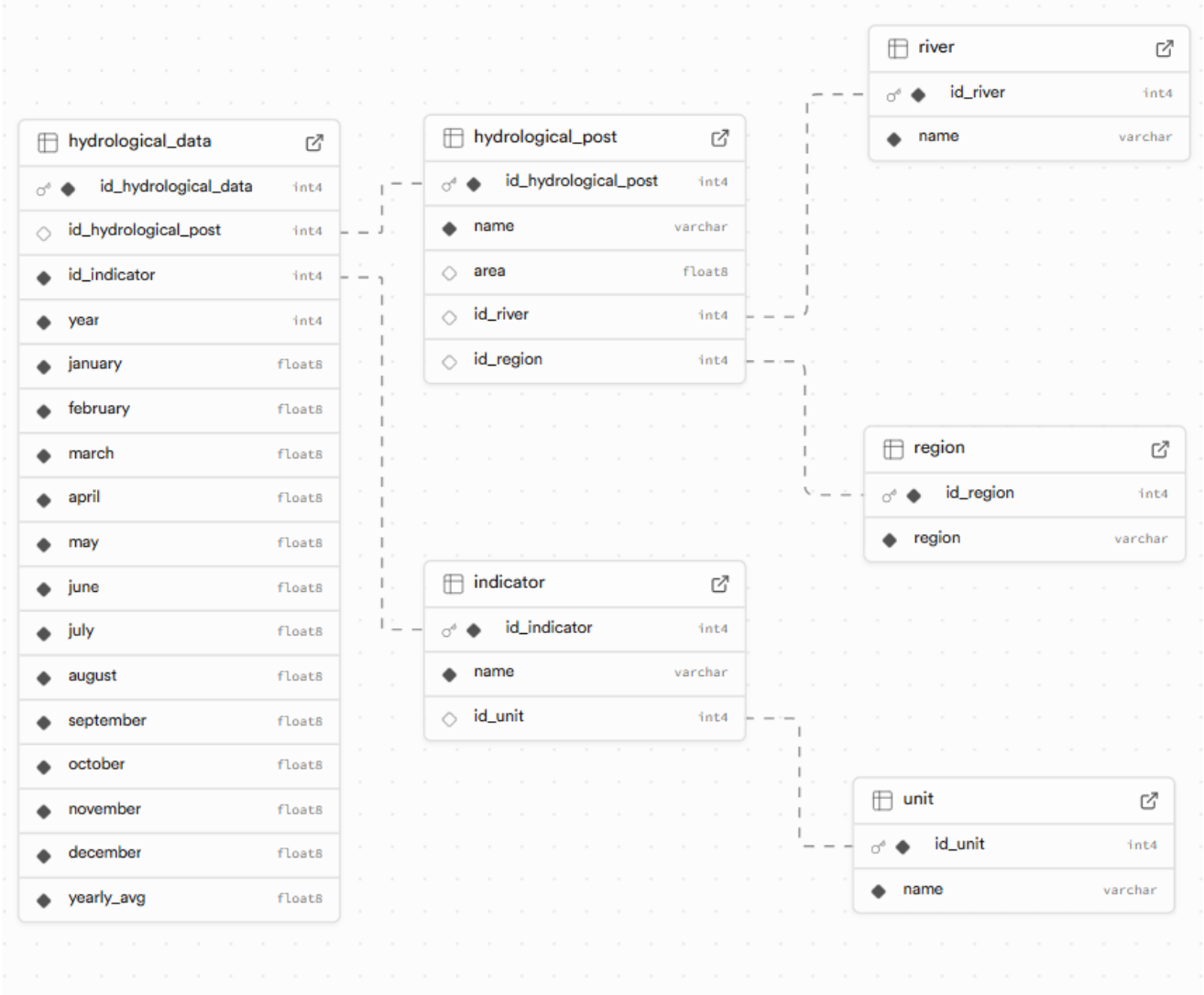

Рисунок 3.5 – Структура бази даних

База даних складається з шести основних таблиць: одиниця (unit), річка (river), область (region), індикатор (indicator), гідрологічний пост (hydrological_post) та гідрологічні дані (hydrological_data). Така структура дозволяє ефективно організувати зберігання та обробку гідрологічної інформації з урахуванням просторових і часових аспектів.

Короткий опис таблиць:

1. unit – таблиця, яка зберігає унікальні ідентифікатори та назви одиниць вимірювання, що використовуються для показників (наприклад, м³/с, мм).
2. river – містить ідентифікатори та назви річок, до яких належать гідрологічні пости. Забезпечує просторову організацію спостережень за водними об’єктами.
3. region – таблиця для зберігання назв адміністративно-територіальних одиниць (областей, районів), що використовується для просторової агрегації та аналізу.
4. indicator – включає перелік показників гідрологічного моніторингу (наприклад, середня витрата води, рівень води), кожен з яких має прив’язку до відповідної одиниці вимірювання з таблиці unit.
5. hydrological_post – описує гідрологічні пости: їх географічне розташування, площу басейну, а також належність до певного регіону та річки. Є центральним елементом для прив’язки просторових даних.
6. hydrological_data – містить часові ряди гідрологічних спостережень. Кожен запис пов’язаний із конкретним постом та показником, що дозволяє здійснювати детальний аналіз змін у водному режимі з часом.

Така структура забезпечує логічну, розширювану та надійну основу для зберігання даних, що є критично важливою для реалізації гнучкої системи фільтрації, побудови аналітичних графіків і просторового відображення даних на карті. Всі таблиці пов’язані зовнішніми ключами, що гарантує цілісність і послідовність даних у системі.

3.3 Висновки

Інформаційна система просторово-часового аналізу об’ємів води у басейні Південного Бугу розроблена на основі сучасної клієнт-серверної архітектури із використанням React для інтерфейсу та Supabase як хмарного сховища даних. Такий підхід забезпечує зручність масштабування, підтримки та подальшого розвитку системи. Компонентна структура дозволяє розмежувати відповідальність між рівнями представлення, логіки та доступу до даних, що сприяє більшій гнучкості та надійності. Реалізовані UML-діаграми ілюструють ключові аспекти роботи системи: ролі користувачів, взаємодію між модулями, процеси обробки та аналізу гідрологічної інформації. База даних побудована на реляційній моделі з підтримкою зовнішніх ключів, що забезпечує цілісність інформації та ефективну роботу з просторово-часовими спостереженнями. Застосування Leaflet дозволяє відображати геодані на карті без залежності від сторонніх сервісів. Проєкт демонструє комплексний підхід до розробки системи моніторингу, що поєднує зручність для користувача, технічну ефективність і потенціал до подальшого вдосконалення.

4 РОЗРОБЛЕННЯ ІНФОРМАЦІЙНОЇ СИСТЕМА ПРОСТОРОВО-ЧАСОВОГО АНАЛІЗУ КІЛЬКОСТІ ВОДИ У РІЧКОВОМУ БАСЕЙНІ ПІВДЕННОГО БУГУ

4.1 Програмна реалізація картографічного модуля інформаційної системи.

Ключовим елементом функціональності системи є відображення гідрологічних постів на інтерактивній карті, реалізованій за допомогою бібліотеки Leaflet. Користувач має змогу обирати басейни Південного Бугу зі списку — після вибору відкривається сторінка з картою, на якій розміщені відповідні маркери. Початковий вигляд списку басейнів подано на рисунку 4.1 а також адаптація під смартфон на рисунку 4.2.

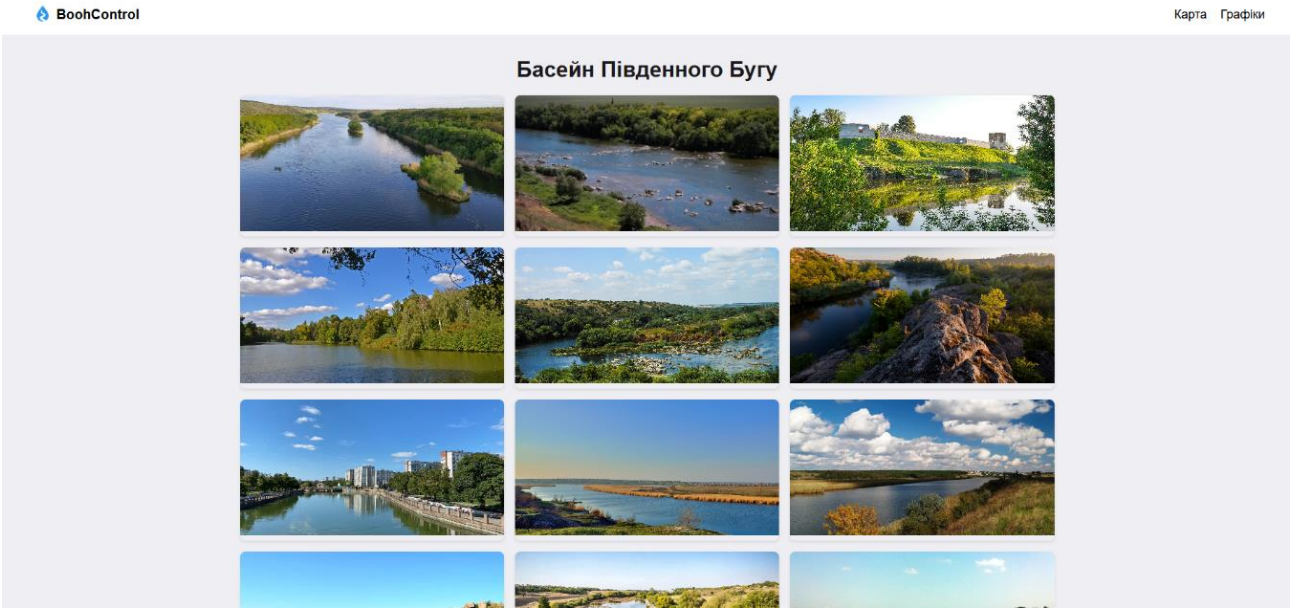

Рисунок 4.1 – Інтерфейс вибору басейнів

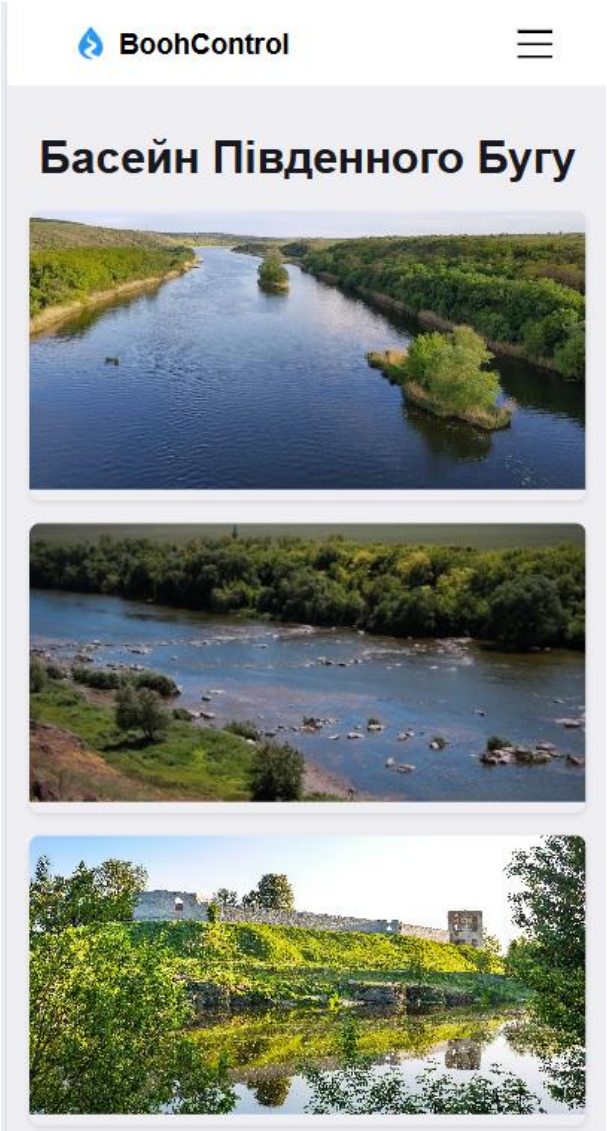

Рисунок 4.2 – Адаптований інтерфейс вибору басейнів

Після натискання на зображення певного басейну користувача перенаправляє на сторінку з картою (компонент Map), де маркери масштабуються відповідно до площі басейну. На карті також відображаються координати, а при взаємодії з маркером відкривається інформаційне спливаюче вікно. Приклад відображення карти наведено на рисунку 4.3, адаптація на рисунку 4.4.

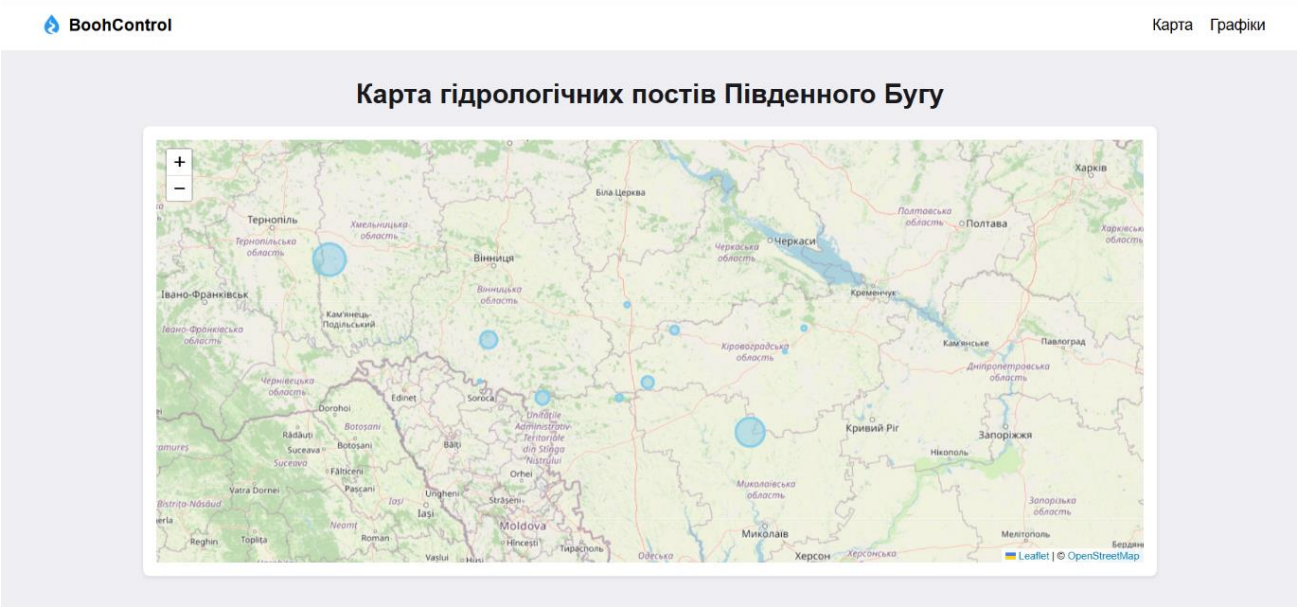

Рисунок 4.3 – Відображення карти з маркерами

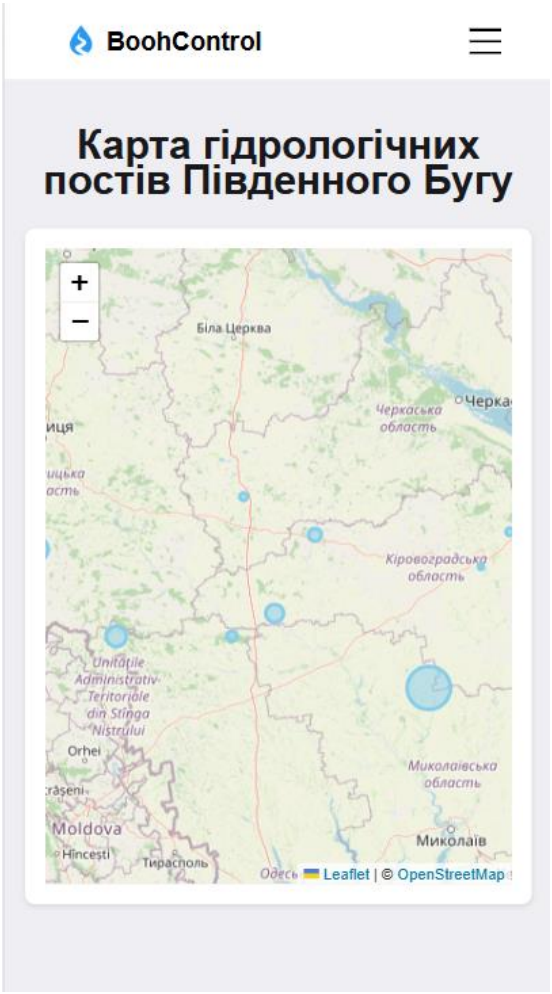

Рисунок 4.4 – Адаптивне відображення карти з маркерами

Реалізація роботи з картою виконана у вигляді React-компонента з інтеграцією Leaflet. Після вибору басейну (ідентифікатора post_id у URL), карта автоматично фокусується на відповідному регіоні. Геодані (координати, площа) отримуються з масиву locations, а радіус маркерів обчислюється пропорційно до площі басейну. Детальніше процес показано на рисунках 4.5–4.8.

```
const handleImageClick = (id) => {
  navigate(`/map?post_id=${id}`);
};
```

Рисунок 4.5 – Обробка переходу до карти

```
useEffect(() => {
  if (!mapRef.current) {
    mapRef.current = L.map("map");

    L.tileLayer("https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png", {
      attribution:
        '© <a href="https://www.openstreetmap.org/copyright">OpenStreetMap</a>',
    }).addTo(mapRef.current);
```

Рисунок 4.6 – Ініціалізація карти Leaflet

```
const maxArea = Math.max(...locations.map((loc) => loc.area));
const minRadius = 1000;
const maxRadius = 15000;
```

Рисунок 4.7 – Розрахунок радіусу маркерів

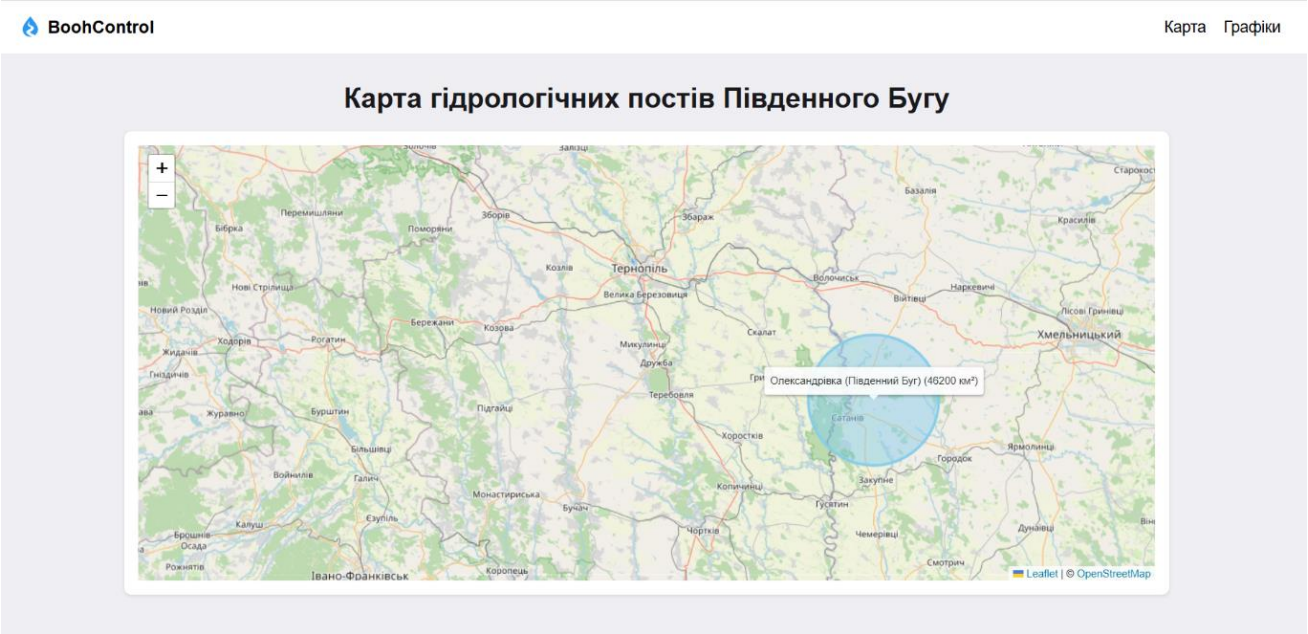

Рисунок 4.8 – Відображення маркера басейну

Ідентифікація басейну здійснюється за допомогою хука useSearchParams, де параметр post_id визначає фокус карти. На рисунках 4.9–4.14 зображено основні етапи цієї реалізації.

```
export default function Map() {
  const mapRef = useRef(null);
  const navigate = useNavigate();
  const [searchParams] = useSearchParams();
  const postId = searchParams.get("post_id");
```

Рисунок 4.9 – Обробка параметра post_id

```
const locations = [
  {
    id: 1,
    name: "Олександрівка (Південний Буг)",
    lat: 49.284426,
    lng: 26.330327,
    area: 46200,
    color: "#87CEEB",
  },
  {
    id: 2,
    name: "Лелітка (Південний Буг)",
    lat: 48.9,
    lng: 30.2,
    area: 4000,
    color: "#87CEEB",
  },
```

Рисунок 4.10 – Додавання маркерів на карту

```
    .addTo(mapRef.current)
    .bindTooltip(`${loc.name} (${loc.area} км²)`, {
      permanent: false,
      direction: "top",
    })
    .bindPopup(
      `<b>${loc.name}</b><br>` +
        `Площа: ${loc.area} км²<br>`
    )
    .on("click", () => {
      navigate(`/graphs?post_id=${loc.id}`);
    });
  return { id: loc.id, marker };
});
```

Рисунок 4.11 – Реалізація спливаючих вікон з даними

```
  if (postId) {
    const selectedLocation = locations.find(
      (loc) => loc.id === parseInt(postId)
    );
    if (selectedLocation) {
      mapRef.current.setView(
        [selectedLocation.lat, selectedLocation.lng],
        10
      );
      const marker = markers.find((m) => m.id === parseInt(postId))?.marker;
      if (marker) marker.openPopup();
    } else {
      mapRef.current.setView([48.5, 30.5], 7);
    }
  } else {
    mapRef.current.setView([48.5, 30.5], 7);
  }
}
```

Рисунок 4.12 – Конфігурація тайлів OpenStreetMap

Рисунок 4.13 – Інтерфейс вибору басейну

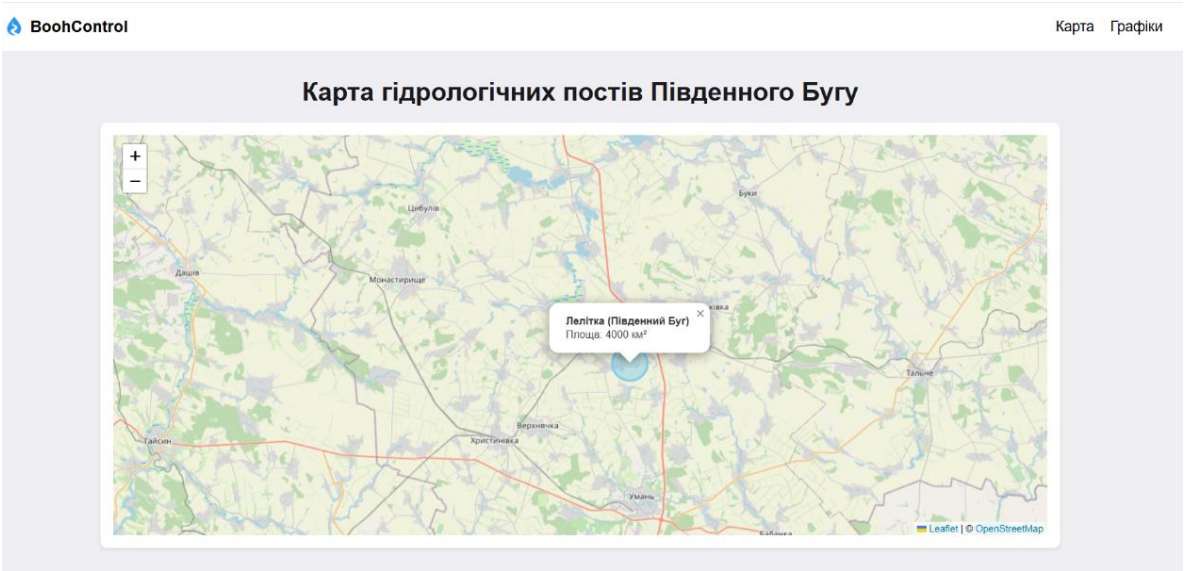

Рисунок 4.14 – Карта з фокусуванням на обраному басейні

Таким чином, реалізований функціонал роботи з картою дозволяє ефективно переглядати гідрологічні локації та переходити до аналізу даних у вигляді графіків, що забезпечує зручність та інформативність користувацької взаємодії.

4.2 Програмна реалізація аналітичного модуля інформаційної системи.

Одним із ключових інструментів аналізу в системі є графіки, реалізовані за допомогою бібліотеки Chart.js. Вони відображають витрати води залежно від обраних параметрів фільтрації, таких як басейн, рік та місяць. Користувач також має змогу змінювати тип графіка (лінійний, стовпцевий або круговий). На рисунку 4.15 представлено початковий варіант відображення графіка, на рисунку 4.16 адаптивний дизайн.

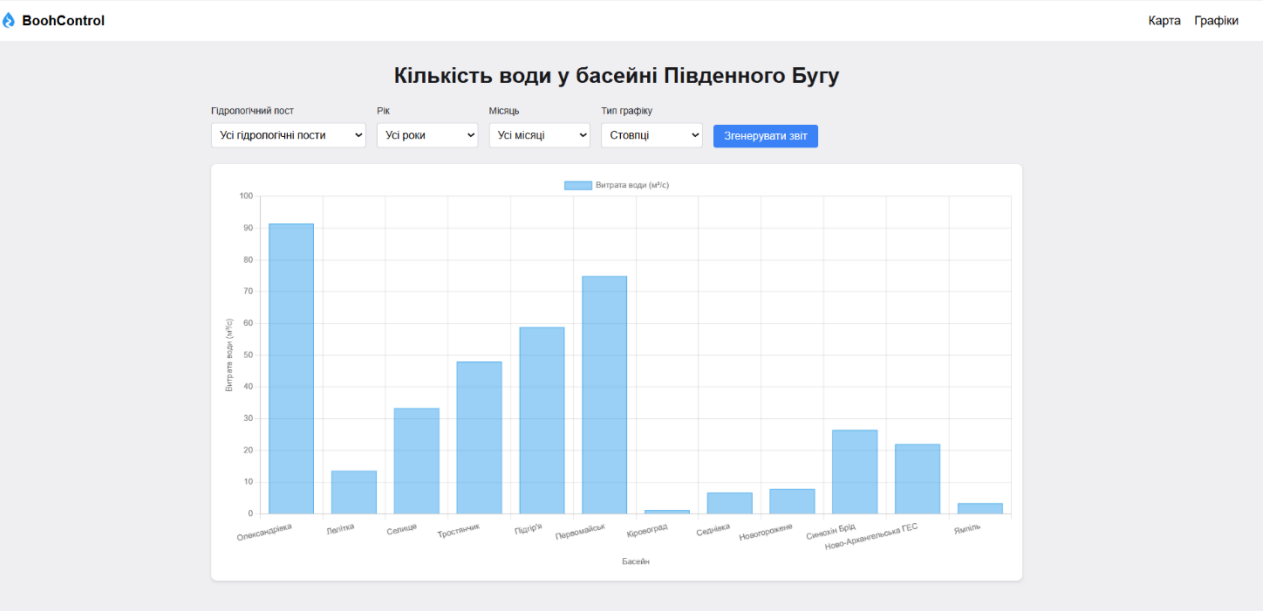

Рисунок 4.15 – Сторінка з даними

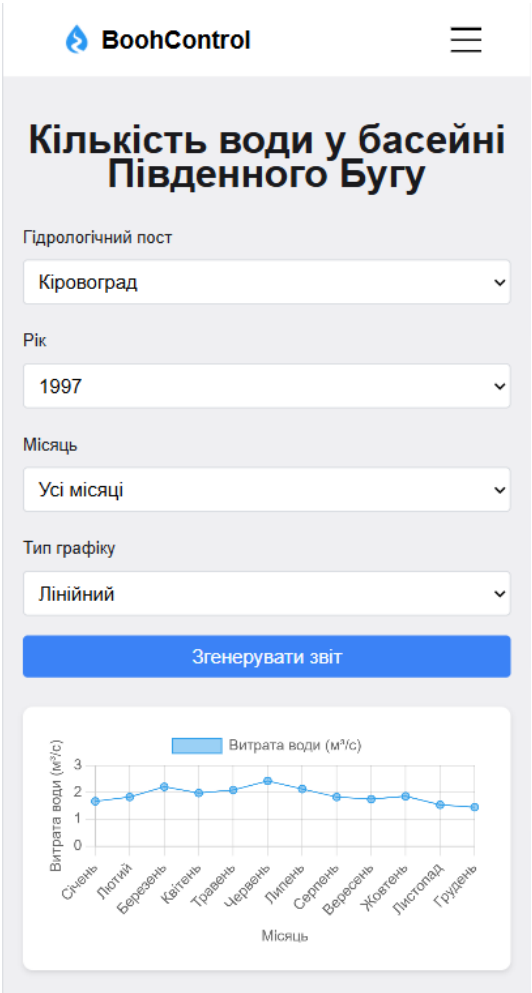

Рисунок 4.16 – Адаптивна сторінка з даними

Графік автоматично оновлюється відповідно до даних, отриманих із бази даних Supabase. Тип графіка обирається через випадаючий список. Вигляд інтерфейсу з графіком наведено на рисунку 4.17.

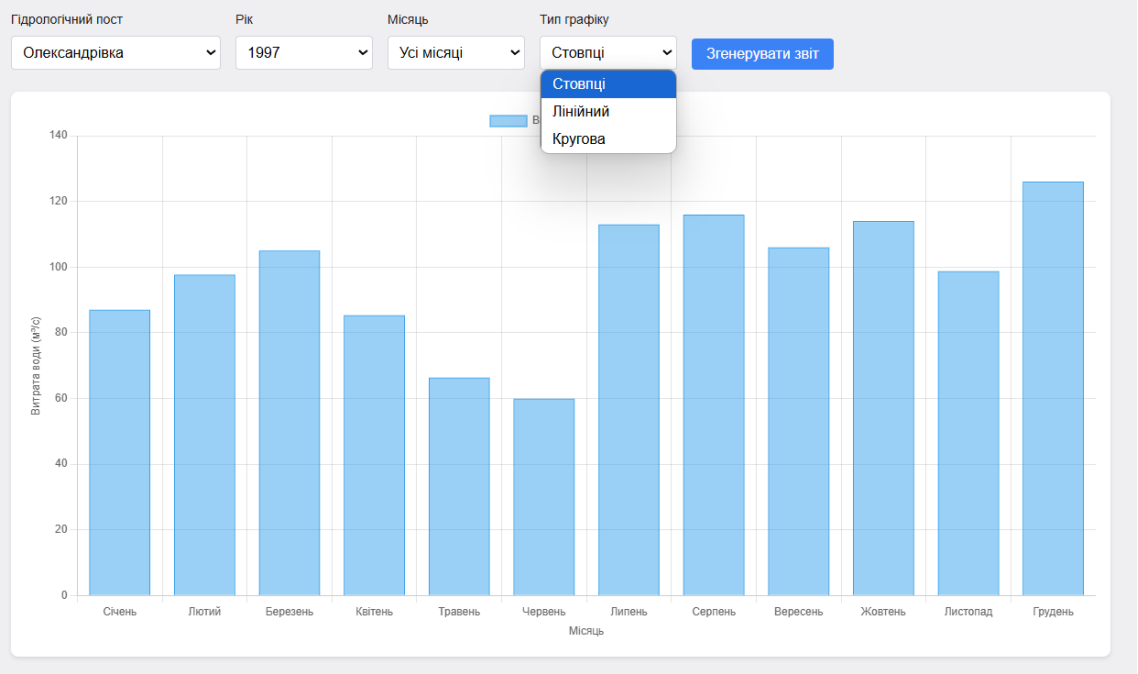

Рисунок 4.17 – Приклад списку з графіками

Реалізація графічного інтерфейсу базується на React-компоненті з використанням Chart.js. Після вибору фільтрів (post_id, year, month, chartType) відбувається запит до Supabase, і отримані дані візуалізуються у вигляді графіка. Деталі цього процесу ілюструють рисунки 4.18–4.21.

hydrological_data?select=*	200	fetch
hydrological_data?select=*	200	fetch
hydrological_data?select=*	200	fetch

Рисунок 4.18 – Обробка вибраних фільтрів

```
useEffect(() => {
  const fetchYears = async () => {
    try {
      const response = await fetch(
        `${SUPABASE_URL}/rest/v1/hydrological_data?select=year&order=year.asc`,
        {
          headers: {
            apikey: SUPABASE_KEY,
            Authorization: `Bearer ${SUPABASE_KEY}`,
          },
        }
      );
      if (!response.ok) {
        throw new Error(`HTTP помилка: ${response.status}`);
      }
      const result = await response.json();
      const uniqueYears = [...new Set(result.map((item) => item.year))].sort(
        (a, b) => b - a
      );
```

Рисунок 4.19 – Запит до бази Supabase

```
export default function Graphs() {
  useEffect(() => {
    if (data.length === 0 && !error) return;

    const { labels, chartData } = prepareChartData();

    if (chartRef.current) chartRef.current.destroy();
    chartRef.current = new Chart(canvasRef.current, {
      type: filters.chartType,
      data: {
        labels,
        datasets: [
          {
            label: "Витрата води (м³/с)",
            data: chartData,
            backgroundColor:
              filters.chartType === "pie"
                ? locations.map((loc) => loc.color)
                : "rgba(54, 162, 235, 0.5)",
            borderColor:
              filters.chartType === "pie"
                ? locations.map((loc) => loc.color)
                : "rgba(54, 162, 235, 1)",
            borderWidth: 1,
          },
        ],
```

Рисунок 4.20 – Налаштування конфігурації Chart.js

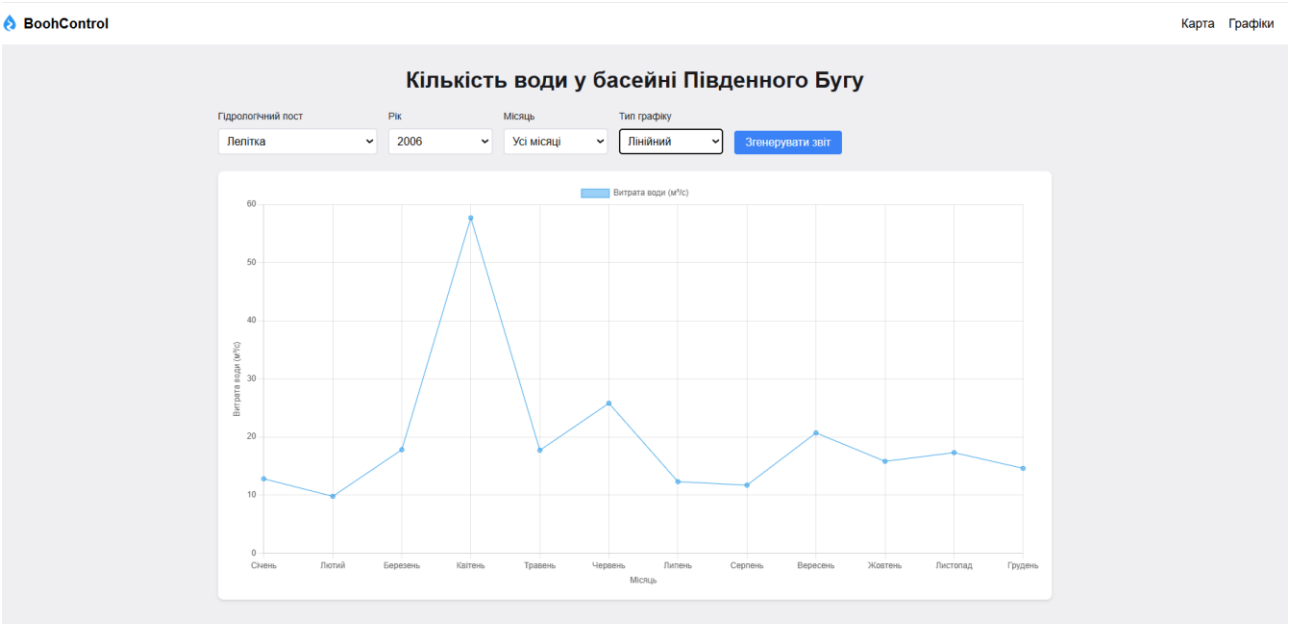

Рисунок 4.21 – Приклад лінійного графіка для заданого року

Таким чином, реалізована система дозволяє зручно переглядати й аналізувати гідрологічні дані в різних форматах, забезпечуючи динамічну побудову графіків та гнучкість візуального представлення інформації.

4.3 Програмна реалізація модуля генерації звітів.

Після проведення аналізу користувач має можливість згенерувати звіт у форматі Excel, який містить як табличні дані, так і візуалізацію у вигляді графіка. На сторінці з графіком доступна кнопка «Згенерувати звіт». Після її натискання автоматично створюється файл, що містить таблицю з даними та зображення графіка. На рисунку 4.22 показано інтерфейс вибору фільтрів перед генерацією звіту.

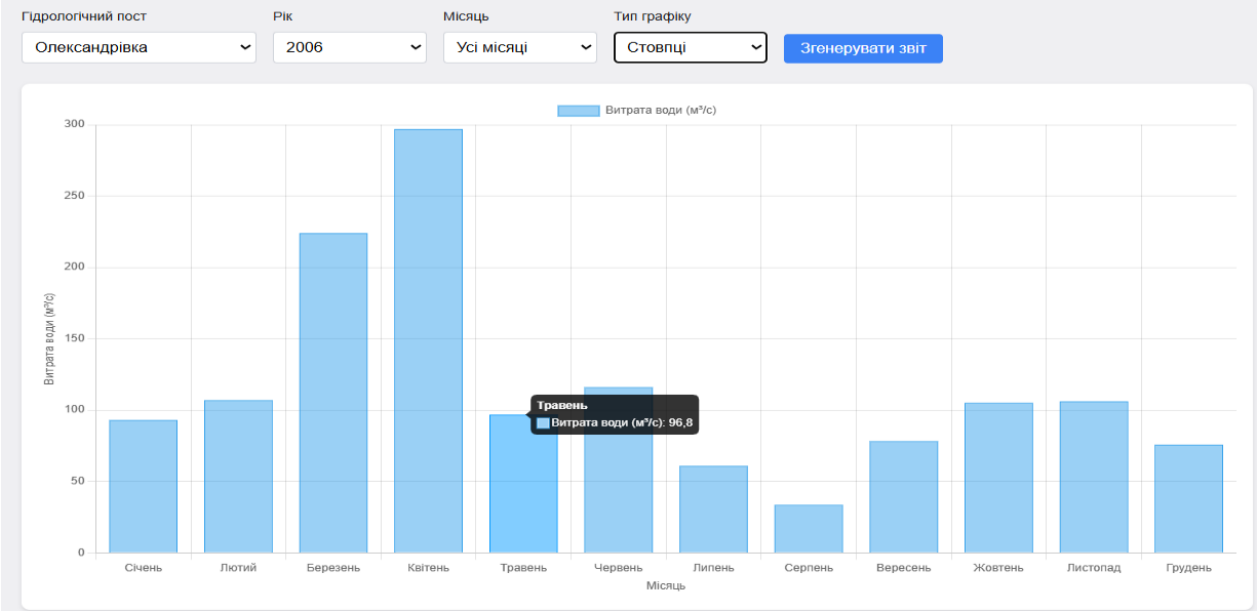

Рисунок 4.22 – Інтерфейс налаштувань фільтрів

Логіка формування звіту полягає в наступному: дані фільтруються відповідно до вибраних параметрів, після чого створюється таблиця за допомогою бібліотеки XLSX, а графік експортується у форматі PNG. Деталі реалізації представлені на рисунках 4.23–4.26.

```
if (filters.post_id === "all") {
  locations.forEach((loc, index) => {
    wsData.push([loc.name, chartData[index]]);
  });
} else if (filters.year === "all") {
  years.forEach((y, index) => {
    wsData.push([y, chartData[index]]);
  });
} else if (filters.month === "all") {
  months.slice(1).forEach((m, index) => {
    wsData.push([m.label, chartData[index]]);
  });
} else {
  data.forEach((d, index) => {
    wsData.push([d.year, chartData[index], d.yearly_avg]);
  });
}
```

Рисунок 4.23 – Підготовка даних для звіту

```
const ws = XLSX.utils.aoa_to_sheet(wsData);
XLSX.utils.book_append_sheet(wb, ws, "Звіт");

const canvas = canvasRef.current;
const chartImage = canvas.toDataURL("image/png");
const imgBlob = dataURLtoBlob(chartImage);
const imgFile = new File([imgBlob], "chart.png", { type: "image/png" });
```

Рисунок 4.24 – Генерація Excel-файлу та зображення графіка

```
const selectedLocation =
  filters.post_id !== "all"
    ? locations.find((loc) => loc.id === parseInt(filters.post_id))?.name ||
      "Усі гідрологічні пости"
    : "Усі гідрологічні пости";
const selectedYear = filters.year !== "all" ? filters.year : "Усі роки";

wsData.push(["Звіт про витрату води"]);
wsData.push([`Гідрологічний пост : ${selectedLocation}`, `Рік: ${selectedYear}`]);
wsData.push([]);


let headers = ["Параметр"];
const { labels, chartData } = prepareChartData();
if (filters.post_id === "all") headers.push("Середня витрата (м³/с)");
else if (filters.year === "all") headers.push("Середня витрата (м³/с)");
else if (filters.month === "all") headers.push("Середня витрата (м³/с)");
else headers.push(filters.month, "Річний середній показник (м³/с)");
wsData.push(headers);
```

Рисунок 4.25 – Формування структури таблиці

	A	B	C	D	E	F	G	H	I	J	K
1	Басейн: Олександрівка	Рік: 2005									
2	Параметр	Середня витрата (м³/с)									
3	Січень	94,5									
4	Лютий	109									
5	Березень	195									
6	Квітень	170									
7	Травень	151									
8	Червень	80,8									
9	Липень	49,5									
10	Серпень	49,9									
11	Вересень	68,6									
12	Жовтень	90,4									
13	Листопад	86,5									
14	Грудень	94,6									
15											
16											

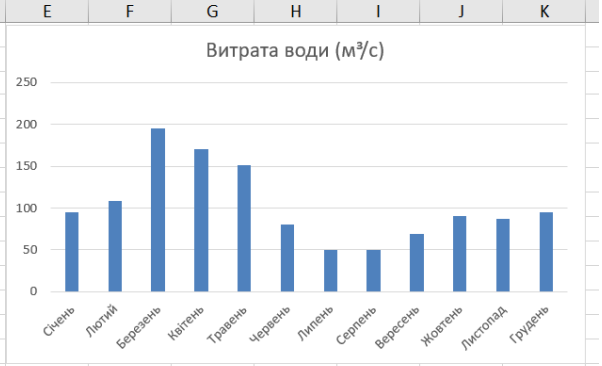

Рисунок 4.26 – Приклад створеного звіту

Функція протестована через unit-тестування. Результати наведено на рисунках 4.27-4.29.

```
import { render, screen, fireEvent } from "@testing-library/react";
import Graphs from "../pages/Graphs";
import * as XLSX from "xlsx";
import { dataURLtoBlob } from "./Graphs";


jest.mock("xlsx", () => ({
  utils: {
    book_new: jest.fn(),
    aoa_to_sheet: jest.fn(),
    book_append_sheet: jest.fn(),
    writeFile: jest.fn(),
  },
}));


const mockCanvas = {
  toDataURL: jest.fn(() => ""),
};
global.HTMLCanvasElement.prototype.toDataURL = mockCanvas.toDataURL;
```

Рисунок 4.27 – Код unit-тестування

```
describe("Graphs Component - generateReport", () => {
  beforeEach(() => {

    render(<Graphs />);

    const mockData = [
      { id_hydrological_post: 1, year: 2023, january: 100, yearly_avg: 120 },
    ];
    jest.spyOn(require("./Graphs"), "setData", "set").mockImplementation(() => mockData);
  });

  test("generateReport створює звіт без помилок", () => {
    const button = screen.getByText("Згенерувати звіт");
    fireEvent.click(button);


    expect(XLSX.utils.book_new).toHaveBeenCalled();
    expect(XLSX.utils.aoa_to_sheet).toHaveBeenCalled();
    expect(XLSX.utils.book_append_sheet).toHaveBeenCalled();
    expect(XLSX.utils.writeFile).toHaveBeenCalledWith(
      expect.anything(),
      expect.stringContaining("water_flow_report")
    );


    expect(mockCanvas.toDataURL).toHaveBeenCalled();
    expect(dataURLtoBlob).toHaveBeenCalled();
  });
});
```

Рисунок 4.28 – Код unit-тестування

hydrological_data?select=*&id_hydrological_post=eq.1	200	preflight	Preflight
hydrological_data?select=*&id_hydrological_post=eq.1	200	fetch	index.jsx:137
hydrological_data?select=*&id_hydrological_post=eq.1&year=eq.2005	200	preflight	Preflight
hydrological_data?select=*&id_hydrological_post=eq.1&year=eq.2005	200	fetch	index.jsx:137

Рисунок 4.29 – Результат unit-тестування

Таким чином, користувач отримує повноцінний звіт із можливістю подальшого збереження, друку або використання в інших аналітичних дослідженнях. Це значно підвищує зручність роботи з гідрологічними даними та сприяє глибшому просторово-часовому аналізу ситуації в басейні Південного Бугу.

4.4 Висновки

У даному розділі представлено розробку інформаційної системи моніторингу кількості вод річкового басейну Південного Бугу, що охоплює три ключові компоненти: інтерактивну карту, графічне відображення даних і генерацію звітів. Інтерактивна карта, реалізована з використанням Leaflet, дозволяє візуалізувати гідрологічні пости та швидко орієнтуватися по регіону. Компонент графіків на базі Chart.js забезпечує зручне представлення та аналіз гідрологічної інформації за різними параметрами. Функція генерації звітів дозволяє експортувати результати у вигляді Excel-файлів з таблицею і графіком, що забезпечує збереження та подальше використання результатів аналізу для прийняття рішень, звітності чи дослідницьких цілей. Усе це робить систему зручною, гнучкою та ефективною для моніторингу стану водних ресурсів.

ВИСНОВКИ

У межах виконання бакалаврської дипломної роботи було проведено аналіз предметної області, визначено сутність проблеми, здійснено огляд існуючих аналогів і підтверджено актуальність створення інформаційної системи для просторово-часового аналізу кількості води в басейні річки Південний Буг.

Описано ІТ-інфраструктуру системи, а також обґрунтовано вибір ефективних технологій і засобів для реалізації проєкту. Зокрема, обрано бібліотеку React та мову JavaScript для розробки веб-додатку, Chart.js — для побудови графіків, Leaflet — для інтеграції карти, а XLSX — для генерації звітів. Також обґрунтовано використання середовища Visual Studio Code, яке забезпечує зручну розробку фронтенду та має підтримку всіх необхідних інструментів, а Supabase (на базі PostgreSQL) — як хмарного сховища для зберігання та обробки даних.

Здійснено проектування інформаційної системи для просторово-часового аналізу кількості води в басейні річки Південний Буг. Розроблено архітектуру інформаційної системи, побудовано UML-діаграми, структуру бази даних і описано сценарії взаємодії користувача з системою.

Створено інформаційну систему просторово-часового аналізу у вигляді веб-додатку за допомогою React та JavaScript у середовищі Visual Studio Code. Реалізовано інтеграцію з Leaflet для відображення гідропостів на карті, Chart.js — для графічного аналізу, а XLSX — для формування звітів. У Supabase створено та організовано хмарну базу даних. Проведено тестування основного функціоналу додатку.

Опубліковано тези доповідей на тему: « Інформаційна система моніторингу кількості вод річкового басейну Південного Бугу» на LIV Всеукраїнській науково-технічній конференції факультету інтелектуальних інформаційних технологій та автоматизації (2025).

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Крижановський Є. М. Інформаційна система моніторингу кількості вод річкового басейну Південного Бугу [Електронний ресурс] / Є. М. Крижановський, Янченко К. О. // Тези доповідей Всеукраїнаська науково-технічна конференція, Вінниця, 21 травня 2025 р. – Електрон. текст. дані. – 2025. – Режим доступу: https://conferences.vntu.edu.ua/index.php/all-fksa/all-fksa-2025/paper/view/24018/19798 (дата звернення: 01.06.2025).

2. Climate change impact on water availability of main river basins in Ukraine – DOAJ. *Directory of Open Access Journals – DOAJ*. URL: https://doaj.org/article/d7a0541dcbb244a1b56440a4ce84d6e2 (дата звернення: 01.06.2025).

3. Мокін В. Б., Крижановський Є. М., Яцолт А. Р., Скорина Л. М. Автоматизація розрахунку водогосподарського балансу ділянок басейнів річок. Водне господарство України. 2017. № 3 (129). С. 25–30.

4. Цифровізація водного циклу – Sigfox. *Sigfox – The Global Communications Service Provider for the Internet of Things (IoT)*. URL: https://sigfox.ua/2024/08/22/water-cycle-digit/ (дата звернення: 01.06.2025).

5. Climate Data Store. *Climate Data Store*. URL: https://cds.climate.copernicus.eu/ (дата звернення: 01.06.2025).

6. Ventusky - Weather Forecast Maps. *Ventusky - Wind, Rain and Temperature Maps*. URL: https://www.ventusky.com/ (дата звернення: 02.06.2025).

7. Google Earth Engine. *Google Earth Engine*. URL: https://earthengine.google.com/ (дата звернення: 02.06.2025).

8. ArcGIS. URL: https://www.arcgis.com/ (дата звернення: 02.06.2025).

9. Minimum Viable Product (MVP): From Validation to MAP Mastery. *Slickplan*. URL: https://slickplan.com/blog/minimum-viable-product (дата звернення: 02.06.2025).

10. Microsoft. Visual Studio Code - Code Editing. Redefined. *Visual Studio Code - Code Editing. Redefined*. URL: https://code.visualstudio.com/ (дата звернення: 02.06.2025).

11. JetBrains. WebStorm: The JavaScript and TypeScript IDE, by JetBrains. *JetBrains*. URL: https://www.jetbrains.com/webstorm/ (дата звернення: 03.06.2025).

12. JetBrains. PyCharm: The only Python IDE you need. *JetBrains*. URL: https://www.jetbrains.com/pycharm/ (дата звернення: 03.06.2025).

13. JetBrains. IntelliJ IDEA – the IDE for Pro Java and Kotlin Development. *JetBrains*. URL: https://www.jetbrains.com/idea/ (дата звернення: 03.06.2025).

14. React. *React*. URL: https://reactjs.org/ (date of access: 17.06.2025). (дата звернення: 03.06.2025).

15. Database | Supabase Docs. *Supabase | The Postgres Development Platform*. URL: https://supabase.com/docs/guides/database/overview (дата звернення: 03.06.2025).

16. Chart.js. *Chart.js | Open source HTML5 Charts for your website*. URL: https://www.chartjs.org (дата звернення: 03.06.2025).

17. Leaflet – an open-source JavaScript library for interactive maps. *Leaflet - a JavaScript library for interactive maps*. URL: https://leafletjs.com (дата звернення: 04.06.2025).

18. Overview | SheetJS Community Edition. *SheetJS Community Edition | SheetJS Community Edition*. URL: https://docs.sheetjs.com/docs/ (дата звернення: 04.06.2025).

19. PostgreSQL. *PostgreSQL*. URL: https://www.postgresql.org (дата звернення: 04.06.2025).

20. Flanagan D. JavaScript: The Definitive Guide. 7th ed. O'Reilly Media, 2020. 706 p

21. Duckett J. HTML & CSS: Design and build websites. Wiley, 2014. 490 p.

Додаток А

(обов’язковий)

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації

ЗАТВЕРДЖУЮ
Завідувач кафедри САІТ
_____ д.т.н., проф. Віталій МОКІН
“____”_____ 2025 р.

ТЕХНІЧНЕ ЗАВДАННЯ
на бакалаврську кваліфікаційну роботу
ІНФОРМАЦІЙНА СИСТЕМА ПРОСТОРОВО-ЧАСОВОГО АНАЛІЗУ КІЛЬКОСТІ ВОДИ У РІЧКОВОМУ БАСЕЙНІ ПІВДЕННОГО БУГУ
08-34.БКР.028.00.000 ТЗ

Керівник: доц.
_____ Євгеній КРИЖАНОВСЬКИЙ
«__» _____ 2025 р.
Розробила студентка гр. 2ІСТ-21б
_____ Каміла ЯНЧЕНКО
«__» _____ 2025 р.

Вінниця 2025

1. Підстава для проведення робіт.

Підставою для виконання роботи є наказ №__ по ВНТУ від «__» _____2025р., та індивідуальне завдання на БКР, затверджене протоколом №__ засідання кафедри САІТ від «__» _____ 2025р.

2. Джерела розробки:

1) Bruce J. Visual Studio Code: End-To-End Editing and Debugging Tools for Web Developers. Wiley & Sons, Incorporated, John, 2019.

2) Boduch A. React and React Native. Packt Publishing - ebooks Account, 2017. 500 p.

3) Lorenz D. Building Production-Grade Web Applications with Supabase. 1st ed. Birmingham : Packt Publishing, 2024. – 534 c.

3. Мета і призначення роботи.

Метою роботи є створення інформаційної системи у вигляді веб-додатку, що забезпечує просторово-часовий аналіз кількості води в басейні річки Південний Буг з урахуванням геолокації гідрологічних постів, вибраних часових періодів та можливістю формування звітів.

4. Методи дослідження:

Аналіз існуючих веб-додатків та платформ для просторово-часового аналізу гідрологічних даних, вивчення потреб користувачів у галузі моніторингу водних ресурсів, моделювання системи за допомогою UML-діаграм для відображення взаємодії її компонентів, а також функціональне тестування системи на різних пристроях та у різних браузерах.

5. Етапи роботи і терміни їх виконання:

a) Аналіз предметної області _____ – _____

b) Вибір оптимальної інфраструктури _____ – _____

c) Проектування інформаційної системи _____ – _____

d) Розроблення інформаційної система просторово-часового аналізу кількості води у річковому басейні Південного Бугу. _____ – _____

e) Оформлення матеріалів до захисту БКР _____ – _____

7. Очікувані результати та порядок реалізації

Отримання інформаційної системи у вигляді веб-додатку, який забезпечить просторово-часовий аналіз обсягів води в басейні річки Південний Буг.

8. Вимоги до розробленої документації

Текстова та ілюстративна частини роботи оформлені у відповідності до вимог «Методичних вказівок до виконання бакалаврських кваліфікаційних робіт для студентів спеціальностей: 124 «Системний аналіз», 126 «Інформаційні системи та технології» (освітня програма «Прикладні інформаційні технології»)».

9. Порядок приймання роботи

Публічний захист «___» _____ 2025 р.

Початок розробки «___» _____ 2025 р.

Граничні терміни виконання БКР «___» _____ 2025 р.

Розробила студентка групи 2ІСТ-21б _____ Каміла ЯНЧЕНКО

Додаток Б

(обов'язковий)

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: «Інформаційна система просторово-часового аналізу кількості води у річковому басейні Південного Бугу»

Тип роботи: бакалаврська кваліфікаційна робота

Підрозділ: кафедра САІТ, ФІІТА, гр. 2ІСТ-21б

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism 0,74 %

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне):

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

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

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

Експертна комісія:

Віталій МОКІН, зав. каф. САІТ _____
(підпис)

Євгеній КРИЖАНОВСЬКИЙ, доц. каф. САІТ _____
(підпис)

Особа, відповідальна за перевірку _____Сергій ЖУКОВ
(підпис)

З висновком експертної комісії ознайомлений(-на)

Керівник _____ Євгеній КРИЖАНОВСЬКИЙ, к.т.н., доц. каф. САІТ
(підпис)

Здобувач _____ Каміла ЯНЧЕНКО
(підпис)

Додаток В

(довідниковий)

Фрагмент лістингу програми

Фрагмент коду Main:

```
import { useNavigate } from "react-router-dom";
import styled from "./mein.module.scss";

export default function Mein() {
  const navigate = useNavigate();

  const basins = [
    {
      id: 1,
      name: "Олександрівка",
      territory: "upper",
      region: "Хмельницька",
      river: "Південний Буг",
      area: 46200,
      image: "/images/aleksandrovska.jpg",
    },
    {
      id: 2,
      name: "Лелітка",
      territory: "middle",
      region: "Вінницька",
      river: "Південний Буг",
      area: 4000,
      image: "/images/lelitka.jpg",
    },
    {
      id: 3,
      name: "Селище",
```

```
  territory: "lower",
  region: "Миколаївська",
  river: "Південний Буг",
  area: 9100,
  image: "/images/seleshe.jpg",
},
{
  id: 4,
  name: "Тростянчик",
  territory: "upper",
  region: "Вінницька",
  river: "Південний Буг",
  area: 17400,
  image: "/images/trostjantunivka.jpg",
},
{
  id: 5,
  name: "Підгір'я",
  territory: "middle",
  region: "Вінницька",
  river: "Південний Буг",
  area: 24600,
  image: "/images/pidgirja.jpg",
},
{
  id: 6,
  name: "Первомайськ",
  territory: "lower",
  region: "Миколаївська",
  river: "Південний Буг",
  area: 44000,
  image: "/images/pervomausk.jpg",
},
{
  id: 7,
```

```
  name: "Кіровоград",
  territory: "upper",
  region: "Кіровоградська",
  river: "Інгул",
  area: 840,
  image: "/images/kirovograt.jpg",
},
{
  id: 8,
  name: "Седнівка",
  territory: "middle",
  region: "Вінницька",
  river: "Інгул",
  area: 4770,
  image: "/images/sednivka.jpg",
},
{
  id: 9,
  name: "Новогорожене",
  territory: "lower",
  region: "Одеська",
  river: "Інгул",
  area: 6670,
  image: "/images/novogorozene.jpg",
},
{
  id: 10,
  name: "Синюхін Брід",
  territory: "upper",
  region: "Вінницька",
  river: "Синюха",
  area: 16700,
  image: "/images/shuhin_brid.jpg",
},
{
```

```
    id: 11,
    name: "Ново-Архангельська ГЕС",
    territory: "middle",
    region: "Кіровоградська",
    river: "Синюха",
    area: 9600,
    image: "/images/snuha.gif",
  },
  {
    id: 12,
    name: "Ямпіль",
    territory: "lower",
    region: "Вінницька",
    river: "Велика Вись",
    area: 2820,
    image: "/images/jampil.jpg",
  },
];

const handleImageClick = (id) => {
  navigate(`/map?post_id=${id}`);
};

return (
  <div className={styled.container}>
    <h1 className={styled.title}>Басейн Південного Бугу</h1>
    <div className={styled.gallery}>
      {basins.map((basin) => (
        <div
          key={basin.id}
          className={styled.imageCard}
          onClick={() => handleImageClick(basin.id)}
        >
          <img src={basin.image} alt={basin.name} className={styled.image} />
          <div className={styled.overlay}>
```

```
            <div className={styled.info}>
              <h3>{basin.name}</h3>
              <p>
                Територія:{" "}
                {basin.territory === "upper"
                  ? "Верхня течія"
                  : basin.territory === "middle"
                  ? "Середня течія"
                  : "Нижня течія"}
              </p>
              <p>Область: {basin.region}</p>
              <p>Річка: {basin.river}</p>
              <p>Площа: {basin.area} км²</p>
            </div>
          </div>
        </div>
      ))}
    </div>
  </div>
 );
}
```

Фрагмент коду Map:

```
import { useEffect, useRef } from "react";
import L from "leaflet";
import "leaflet/dist/leaflet.css";
import { useNavigate, useSearchParams } from "react-router-dom";
import styled from "./map.module.scss";

export default function Map() {
 const mapRef = useRef(null);
 const navigate = useNavigate();
 const [searchParams] = useSearchParams();
 const postId = searchParams.get("post_id");
```

```
const locations = [
 {
  id: 1,
  name: "Олександрівка (Південний Буг)",
  lat: 49.284426,
  lng: 26.330327,
  area: 46200,
  color: "#87CEEB",
 },
 {
  id: 2,
  name: "Лелітка (Південний Буг)",
  lat: 48.9,
  lng: 30.2,
  area: 4000,
  color: "#87CEEB",
 },
 {
  id: 3,
  name: "Тростянчик (Південний Буг)",
  lat: 48.1,
  lng: 29.1,
  area: 17400,
  color: "#87CEEB",
 },
 {
  id: 4,
  name: "Підгір’я (Південний Буг)",
  lat: 48.6,
  lng: 28.4,
  area: 24600,
  color: "#87CEEB",
 },
 {
```

```
  id: 5,
  name: "Первомайськ (Південний Буг)",
  lat: 47.8,
  lng: 31.8,
  area: 44000,
  color: "#87CEEB",
},
{
  id: 6,
  name: "Кіровоград (Інгулець)",
  lat: 48.5,
  lng: 32.25,
  area: 840,
  color: "#87CEEB",
},
{
  id: 7,
  name: "Седнівка (Інгулець)",
  lat: 48.7,
  lng: 32.5,
  area: 4770,
  color: "#87CEEB",
},
{
  id: 8,
  name: "Новогорожене (Інгулець)",
  lat: 48.1,
  lng: 30.1,
  area: 6670,
  color: "#87CEEB",
},
{
  id: 9,
  name: "Синюхін Брід (Синюха)",
  lat: 48.2328,
```

```
    lng: 30.4682,
    area: 16700,
    color: "#87CEEB",
  },
  {
    id: 10,
    name: "Ново-Архангельська ГЕС (Синюха)",
    lat: 48.6833,
    lng: 30.8167,
    area: 9600,
    color: "#87CEEB",
  },
  {
    id: 11,
    name: "Ямпіль (Велика Вись)",
    lat: 48.2414,
    lng: 28.2819,
    area: 2820,
    color: "#87CEEB",
  },
];

useEffect(() => {
  if (!mapRef.current) {
    mapRef.current = L.map("map");

    L.tileLayer("https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png", {
      attribution:
        '© <a href="https://www.openstreetmap.org/copyright">OpenStreetMap</a>',
    }).addTo(mapRef.current);

    const maxArea = Math.max(...locations.map((loc) => loc.area));
    const minRadius = 1000;
    const maxRadius = 15000;
```

```
const markers = locations.map((loc) => {

  const radius =
    minRadius + (loc.area / maxArea) * (maxRadius - minRadius);
  const marker = L.circle([loc.lat, loc.lng], {
    color: loc.color,
    fillColor: loc.color,
    fillOpacity: 0.5,
    radius: radius, // Радіус залежить від площі
  })
    .addTo(mapRef.current)
    .bindTooltip(`${loc.name} (${loc.area} км²)`, {
      permanent: false,
      direction: "top",
    })
    .bindPopup(
      `<b>${loc.name}</b><br>` +
        `Площа: ${loc.area} км²<br>`
    )
    .on("click", () => {
      navigate(`/graphs?post_id=${loc.id}`);
    });
  return { id: loc.id, marker };
});

if (postId) {
  const selectedLocation = locations.find(
    (loc) => loc.id === parseInt(postId)
  );
  if (selectedLocation) {
    mapRef.current.setView(
      [selectedLocation.lat, selectedLocation.lng],
      10
    );
    const marker = markers.find((m) => m.id === parseInt(postId))?.marker;
```

```
        if (marker) marker.openPopup();
      } else {
        mapRef.current.setView([48.5, 30.5], 7);
      }
    } else {
      mapRef.current.setView([48.5, 30.5], 7);
    }
  }

  return () => {
    if (mapRef.current) {
      mapRef.current.remove();
      mapRef.current = null;
    }
  };
 }, [navigate, postId]);

 return (
  <div className={styled.container}>
    <h1 className={styled.title}>
      Карта гідрологічних постів Південного Бугу
    </h1>
    <div className={styled.mapContainer}>
      <div id="map" style={{ height: "500px" }}></div>
    </div>
  </div>
 );
}
```

Фрагмент коду Graphs:

```
import { useState, useEffect, useRef } from "react";
import { useSearchParams } from "react-router-dom";
import Chart from "chart.js/auto";
import * as XLSX from "xlsx";
```

```
import styled from "./graphs.module.scss";

export default function Graphs() {
  const [searchParams] = useSearchParams();
  const [data, setData] = useState([]);
  const [years, setYears] = useState([]);
  const [error, setError] = useState(null);
  const [filters, setFilters] = useState({
    post_id: searchParams.get("post_id") || "all",
    year: "all",
    month: "all",
    chartType: "bar",
  });
  const chartRef = useRef(null);
  const canvasRef = useRef(null);

  const SUPABASE_URL = "https://kldszfebdkeyvxzgkpct.supabase.co";
  const SUPABASE_KEY =
    "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJzdXBhYmFzZSIsInJlZiI6ImtsZHN6ZmViZGtleXZ4emdrcGN0Iiwicm
9sZSI6InNlcnZpY2Vfcm9sZSIsImlhdCI6MTc0NzA1OTYyNSwiZXhwIjoyMDYyNjM1NjI1fQ.t2ved9VjzsmETI6u8cBZZZEbtQE
oaXS2bR6ZoqseOsI";

  const months = [
    { value: "all", label: "Усі місяці" },
    { value: "january", label: "Січень" },
    { value: "february", label: "Лютий" },
    { value: "march", label: "Березень" },
    { value: "april", label: "Квітень" },
    { value: "may", label: "Травень" },
    { value: "june", label: "Червень" },
    { value: "july", label: "Липень" },
    { value: "august", label: "Серпень" },
    { value: "september", label: "Вересень" },
    { value: "october", label: "Жовтень" },
    { value: "november", label: "Листопад" },
    { value: "december", label: "Грудень" },
```

```
];

const prepareChartData = () => {
  let labels = [];
  let chartData = [];

  if (filters.post_id === "all") {
    labels = locations.map((loc) => loc.name);
    chartData = locations.map((loc) => {
      const locData = data.filter((d) => d.id_hydrological_post === loc.id);
      const sum = locData.reduce((acc, d) => acc + (d.yearly_avg || 0), 0);
      return locData.length ? sum / locData.length : 0;
    });
  } else if (filters.year === "all") {
    labels = years;
    chartData = years.map((y) => {
      const yearData = data.filter((d) => d.year === y);
      const sum = yearData.reduce((acc, d) => acc + (d.yearly_avg || 0), 0);
      return yearData.length ? sum / yearData.length : 0;
    });
  } else if (filters.month === "all") {
    labels = months.slice(1).map((m) => m.label);
    chartData = months.slice(1).map((m) => {
      const monthData = data.map((d) => d[m.value] || 0);
      const sum = monthData.reduce((acc, val) => acc + val, 0);
      return monthData.length ? sum / monthData.length : 0;
    });
  } else {
    labels = data.map((d) => d.year);
    chartData = data.map((d) => d[filters.month] || 0);

  return { labels, chartData };
};

useEffect(() => {
```

```
  const fetchYears = async () => {
   try {
    const response = await fetch(
     `${SUPABASE_URL}/rest/v1/hydrological_data?select=year&order=year.asc`,
     {
      headers: {
       apikey: SUPABASE_KEY,
       Authorization: `Bearer ${SUPABASE_KEY}`,
      },
     }
    );
    if (!response.ok) {
     throw new Error(`HTTP помилка: ${response.status}`);
    }
    const result = await response.json();
    const uniqueYears = [...new Set(result.map((item) => item.year))].sort(
     (a, b) => b - a
    );
    setYears(uniqueYears);
   } catch (error) {
    console.error("Помилка при отриманні років:", error);
    setError(`Помилка при завантаженні років: ${error.message}`);
   }
  };

  fetchYears();
 }, []);

 useEffect(() => {
  const fetchData = async () => {
   setError(null);
   let query = `${SUPABASE_URL}/rest/v1/hydrological_data?select=*`;
   if (filters.post_id !== "all") {
    query += `&id_hydrological_post=eq.${filters.post_id}`;
    console.log(filters.post_id, filters.year);
```

```
    }
    if (filters.year !== "all") {
      query += `&year=eq.${filters.year}`;
    }
    if (filters.month !== "all") {
      query += `&select=year,${filters.month},yearly_avg`;
    }

    try {
      const response = await fetch(query, {
        headers: {
          apikey: SUPABASE_KEY,
          Authorization: `Bearer ${SUPABASE_KEY}`,
        },
      });
      if (!response.ok) {
        throw new Error(`HTTP помилка: ${response.status}`);
      }
      const result = await response.json();
      if (result.length === 0) {
        setError("Дані відсутні для вибраних фільтрів");
      }
      setData(result);
    } catch (error) {
      console.error("Помилка при отриманні даних:", error);
      setError(`Помилка: ${error.message}`);
    }
  };

  fetchData();
}, [filters]);
useEffect(() => {
  if (data.length === 0 && !error) return;
  const { labels, chartData } = prepareChartData();
```

```
if (chartRef.current) chartRef.current.destroy();
chartRef.current = new Chart(canvasRef.current, {
  type: filters.chartType,
  data: {
    labels,
    datasets: [
      {
        label: "Витрата води (м³/с)",
        data: chartData,
        backgroundColor:
          filters.chartType === "pie"
            ? locations.map((loc) => loc.color)
            : "rgba(54, 162, 235, 0.5)",
        borderColor:
          filters.chartType === "pie"
            ? locations.map((loc) => loc.color)
            : "rgba(54, 162, 235, 1)",
        borderWidth: 1,
      },
    ],
  },
  options: {
    responsive: true,
    scales:
      filters.chartType !== "pie"
        ? {
            y: {
              beginAtZero: true,
              title: { display: true, text: "Витрата води (м³/с)" },
            },
            x: {
              title: {
                display: true,
                text:
                  filters.post_id === "all"
```

```
                ? "Басейн"
                : filters.year === "all"
                ? "Рік"
                : "Місяць",
            },
          },
        }
      : {},
  },
});
```

Фрагмент коду Report:

```
  return () => {
    if (chartRef.current) chartRef.current.destroy();
  };
}, [data, error, filters, years]);

const handleFilterChange = (e) => {
  setFilters({ ...filters, [e.target.name]: e.target.value });
};

const generateReport = () => {
  if (!data.length || error) {
    alert("Немає даних для генерації звіту!");
    return;
  }

  const wb = XLSX.utils.book_new();
  const wsData = [];
  const selectedLocation =
    filters.post_id !== "all"
      ? locations.find((loc) => loc.id === parseInt(filters.post_id))?.name ||
        "Усі басейни"
```

```
    : "Усі басейни";
const selectedYear = filters.year !== "all" ? filters.year : "Усі роки";

wsData.push(["Звіт про витрату води"]);
wsData.push([`Басейн: ${selectedLocation}`, `Рік: ${selectedYear}`]);
wsData.push([]);

let headers = ["Параметр"];
const { labels, chartData } = prepareChartData();
if (filters.post_id === "all") headers.push("Середня витрата (м³/с)");
else if (filters.year === "all") headers.push("Середня витрата (м³/с)");
else if (filters.month === "all") headers.push("Середня витрата (м³/с)");
else headers.push(filters.month, "Річний середній показник (м³/с)");
wsData.push(headers);

if (filters.post_id === "all") {
  locations.forEach((loc, index) => {
    wsData.push([loc.name, chartData[index]]);
  });
} else if (filters.year === "all") {
  years.forEach((y, index) => {
    wsData.push([y, chartData[index]]);
  });
} else if (filters.month === "all") {
  months.slice(1).forEach((m, index) => {
    wsData.push([m.label, chartData[index]]);
  });
} else {
  data.forEach((d, index) => {
    wsData.push([d.year, chartData[index], d.yearly_avg]);
  });
}

const ws = XLSX.utils.aoa_to_sheet(wsData);
XLSX.utils.book_append_sheet(wb, ws, "Звіт");
```

```
  const canvas = canvasRef.current;
  const chartImage = canvas.toDataURL("image/png");
  const imgBlob = dataURLtoBlob(chartImage);
  const imgFile = new File([imgBlob], "chart.png", { type: "image/png" });

  XLSX.writeFile(
    wb,
    `water_flow_report_${selectedLocation}_${selectedYear}.xlsx`
  );

  const link = document.createElement("a");
  link.href = chartImage;
  link.download = `chart_${selectedLocation}_${selectedYear}.png`;
  link.click();
};

const dataURLtoBlob = (dataURL) => {
  const arr = dataURL.split(",");
  const mime = arr[0].match(/:(.*?);/)[1];
  const bstr = atob(arr[1]);
  let n = bstr.length;
  const u8arr = new Uint8Array(n);
  while (n--) u8arr[n] = bstr.charCodeAt(n);
  return new Blob([u8arr], { type: mime });
};
return (
  <div className={styled.container}>
    <h1 className={styled.title}>Кількість води у басейні Південного Бугу</h1>
    {error && <div className={styled.error}>{error}</div>}
    <div className={styled.filters}>
      <div className={styled.filter}>
        <label className={styled.label}>Басейн</label>
        <select
          name="post_id"
```

```
    value={filters.post_id}
    onChange={handleFilterChange}
    className={styled.select}
   >
    <option value="all">Усі басейни</option>
    {locations.map((loc) => (
     <option key={loc.id} value={loc.id}>
      {loc.name}
     </option>
    ))}
   </select>
  </div>
  <div className={styled.filter}>
   <label className={styled.label}>Рік</label>
   <select
    name="year"
    value={filters.year}
    onChange={handleFilterChange}
    className={styled.select}
   >
    <option value="all">Усі роки</option>
    {years.map((year) => (
     <option key={year} value={year}>
      {year}
     </option>
    ))}
   </select>
  </div>
  <div className={styled.filter}>
   <label className={styled.label}>Місяць</label>
   <select
    name="month"
    value={filters.month}
    onChange={handleFilterChange}
    className={styled.select}
```

```
          >
            {months.map((month) => (
              <option key={month.value} value={month.value}>
                {month.label}
              </option>
            ))}
          </select>
        </div>
        <div className={styled.filter}>
          <label className={styled.label}>Тип графіку</label>
          <select
            name="chartType"
            value={filters.chartType}
            onChange={handleFilterChange}
            className={styled.select}
          >
            <option value="bar">Стовпці</option>
            <option value="line">Лінійний</option>
            <option value="pie">Кругова</option>
          </select>
        </div>
        <button onClick={generateReport} className={styled.button}>
          Згенерувати звіт
        </button>
      </div>
      <div className={styled.chart}>
        <canvas ref={canvasRef}></canvas>
      </div>
    </div>
  );
}
```

Додаток Г

(обов’язковий)

ІЛЮСТРАТИВНА ЧАСТИНА

ІНФОРМАЦІЙНА СИСТЕМА ПРОСТОРОВО-ЧАСОВОГО АНАЛІЗУ КІЛЬКОСТІ ВОДИ У РІЧКОВОМУ БАСЕЙНІ ПІВДЕННОГО БУГУ

Нормоконтроль: к.т.н., доцент

_____ Сергій ЖУКОВ

«___» _____ 2025 р.

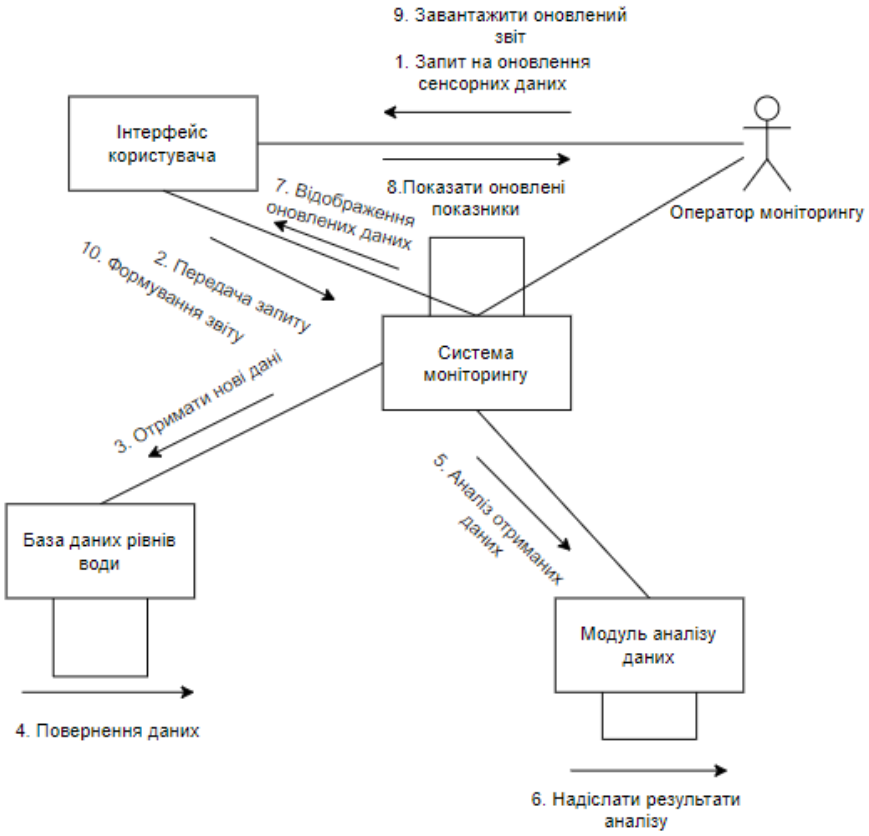

Рисунок Г.1 – Діаграма оновлення даних

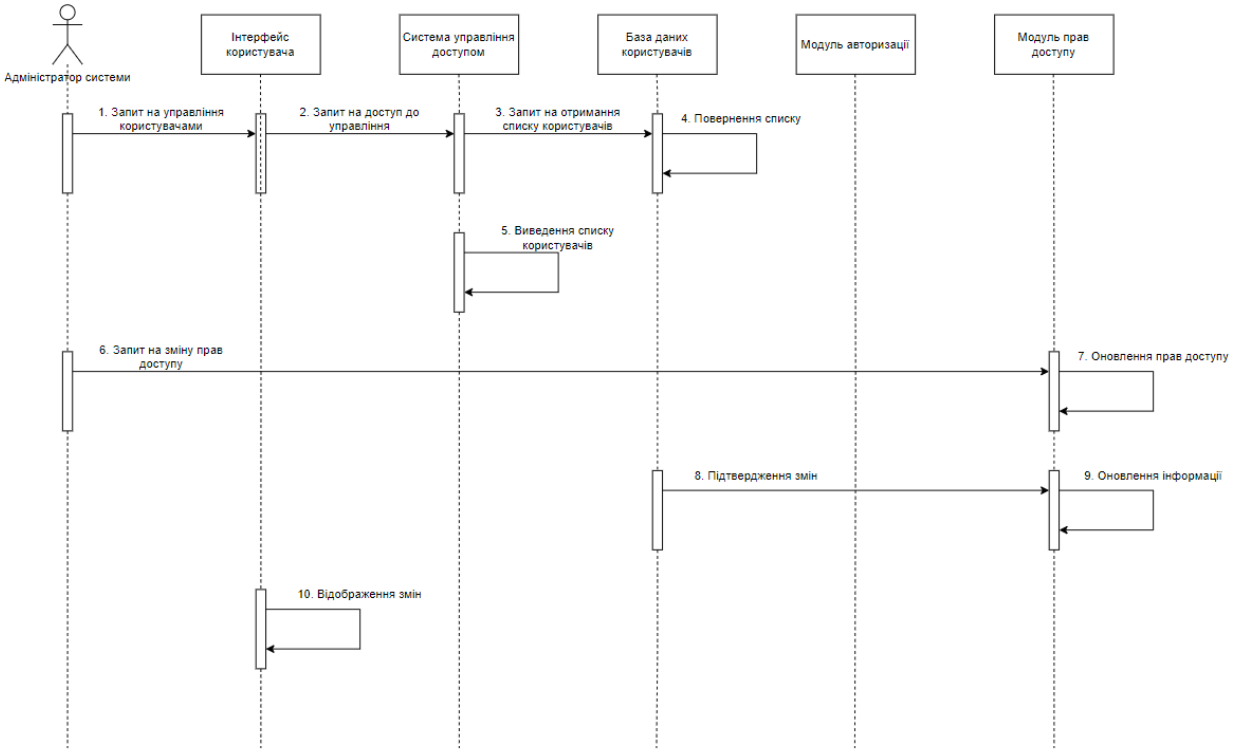

Рисунок Г.2 – Діаграма послідовності для керування користувачами

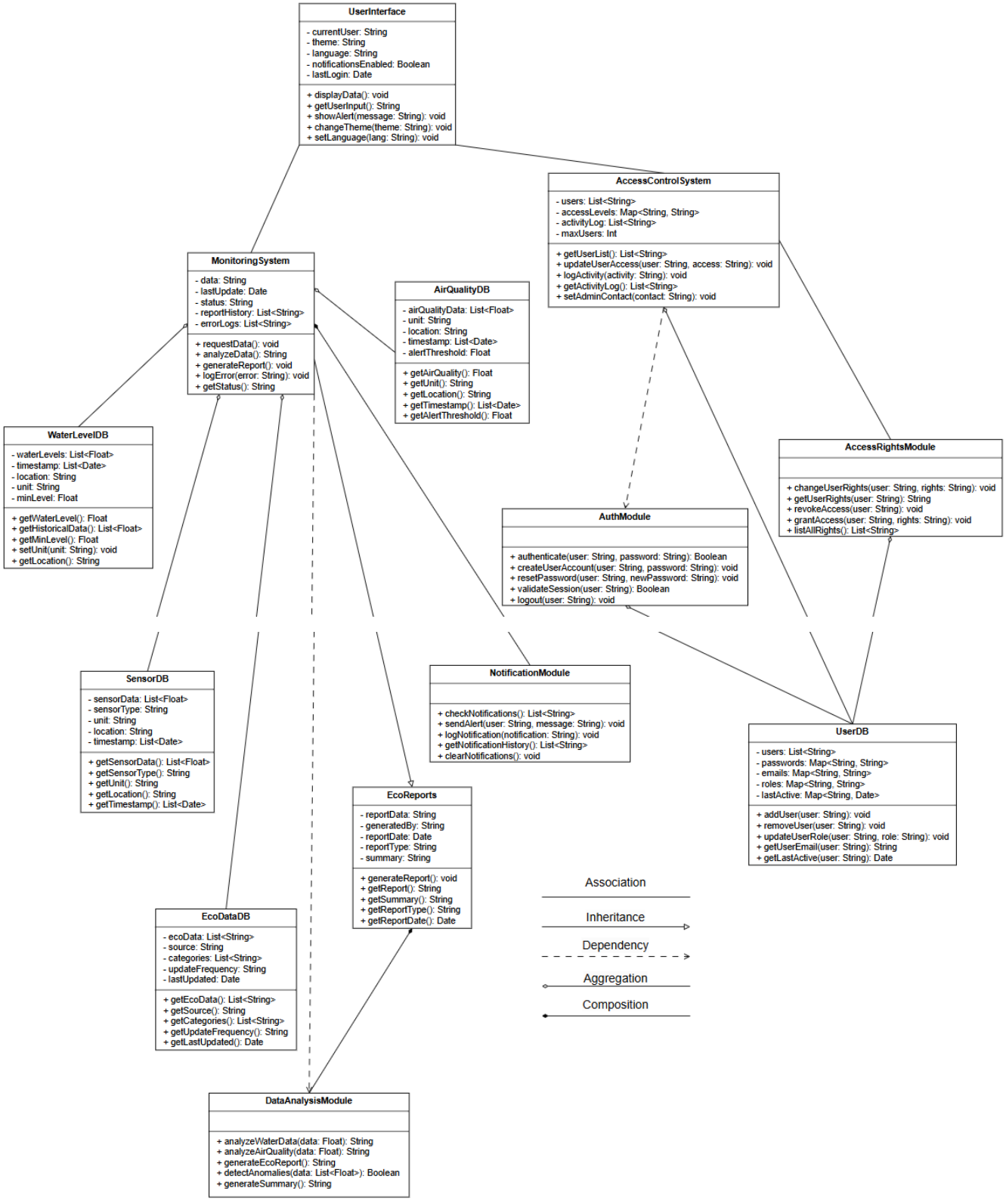

Рисунок Г.3 – Діаграма класів

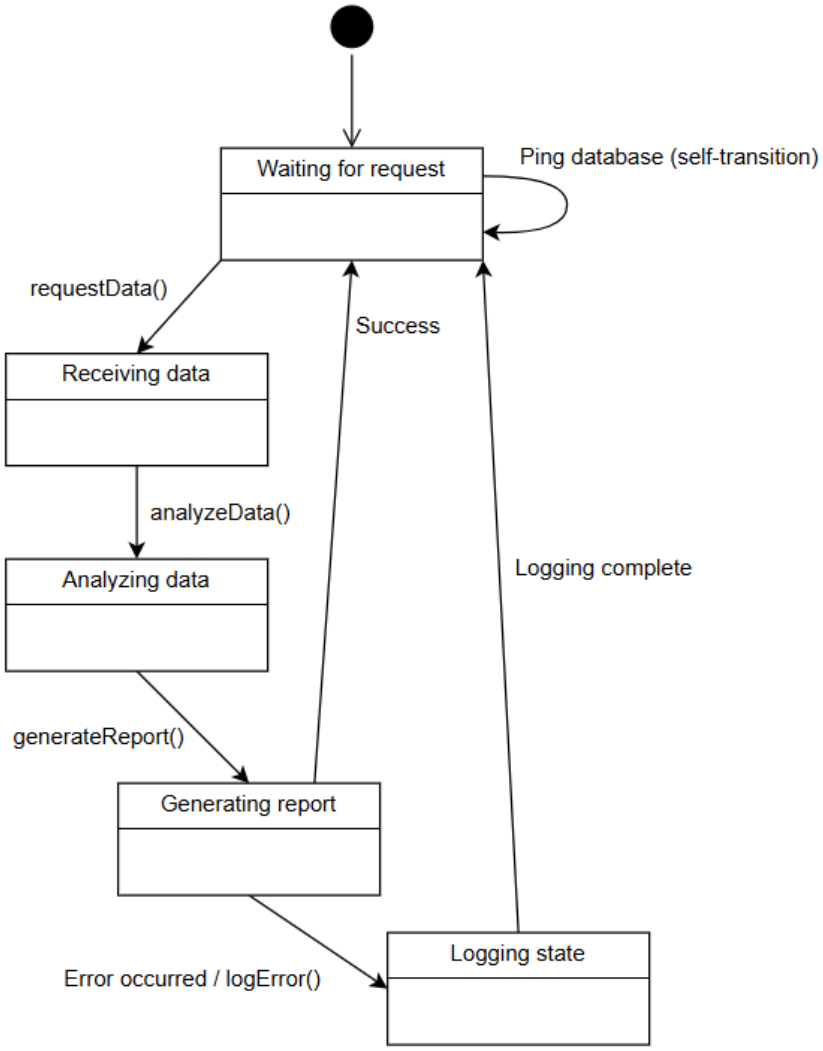

Рисунок Г.4 – Діаграма станів для система моніторингу

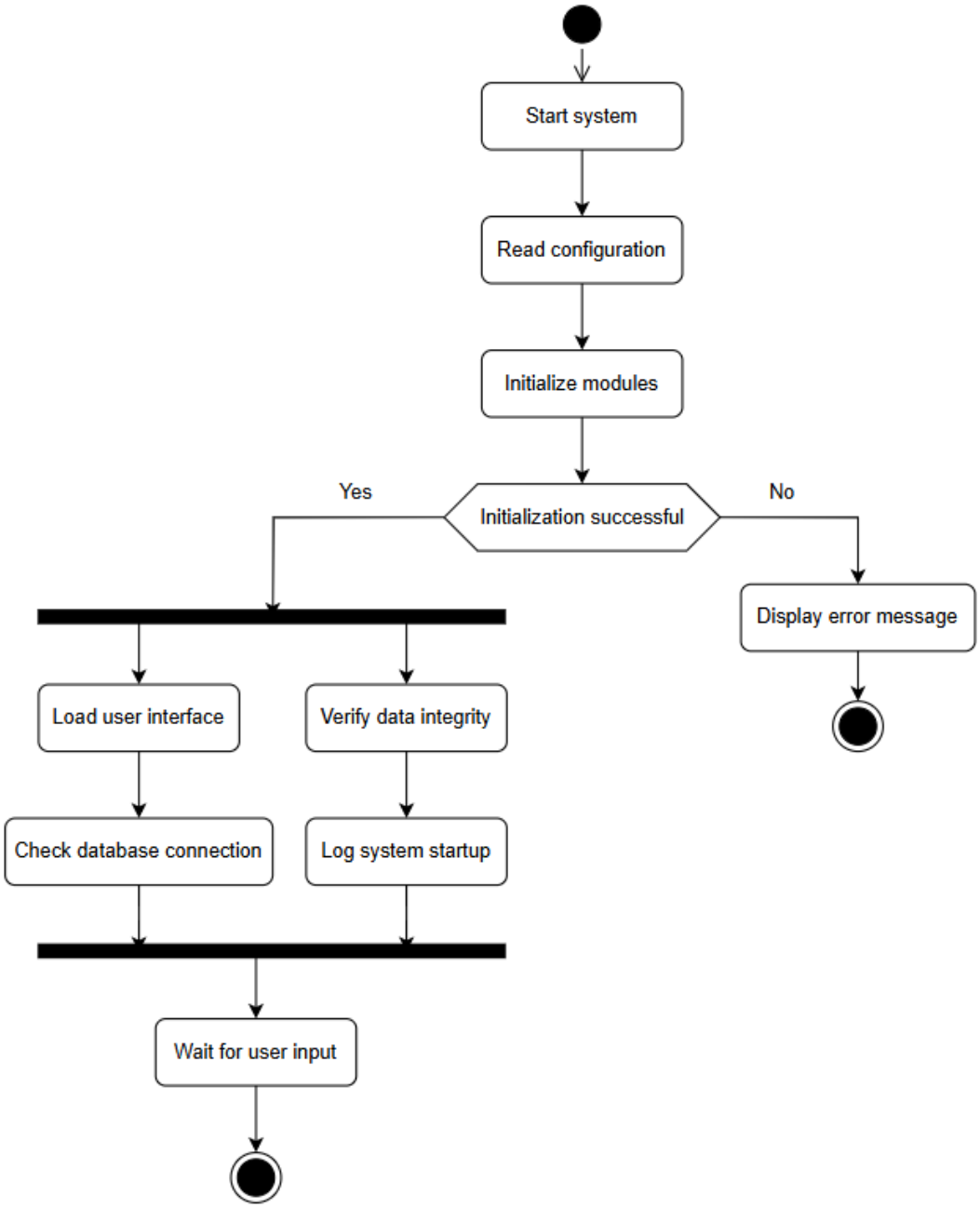

Рисунок Г.5 – Діаграма запуску та ініціалізації системи