БАКАЛАВРСЬКА РОБОТА

КРБ.СІ-11.00.00.000 ПЗ

Група СІ-22-1

ФЕДОРКО Євген

2026

Міністерство освіти і науки України
Івано-Франківський національний технічний університет нафти і газу
Факультет інформаційних технологій
Кафедра інформаційно-телекомунікаційних технологій та систем
Федорко Євген Миколайович
(прізвище, ім'я, по батькові)

УДК: 004.4:005.332.2:658.5.011.56
(індекс)

БАКАЛАВРСЬКА РОБОТА

« Розроблення системи масового оповіщення населення з використанням технології інтернету речей»
(назва роботи)

Системна інженерія – інтернет речей
(назва освітньої програми)

151 – Автоматизація та комп'ютерно-інтегровані технології
(шифр і назва спеціальності)

Робота містить результати власних досліджень, використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело:

Здобувач освітнього ступеня Федорко Є.М.
(підпис, ініціали та прізвище здобувача)

Науковий керівник к.т.н., доцент Паньків Ю.В.
(підпис, прізвище, ім'я, по батькові, науковий ступінь, вчене звання керівника)

Допущено до захисту
В.о. завідувач кафедри

Доц. Заміховська О. Л.
(посада) (підпис) (дата) (ініціали та прізвище)

Івано-Франківськ – 2026

Івано-Франківський національний технічний університет нафти і газу

(повне найменування закладу вищої освіти)

Факультет інформаційних технологій
Кафедра автоматизації та комп'ютерно-інтегрованих технологій
Освітній рівень бакалавр
Спеціальність 174 – Системна інженерія
(шифр і назва)

ЗАТВЕРДЖУЮ
В.о. завідувач кафедри ІТТС
Заміховська О.Л.
«____» _____2026 року

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТОВІ

Федорку Євгену Миколайовичу
(прізвище, ім'я, по батькові)

1. Тема роботи: «Розроблення системи масового оповіщення населення з використанням технології інтернету речей»

керівник роботи Паньків Ю.В., к.т.н., доц. каф. ІТТС _____,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від « 05 » травня 2026 року № **№ 279/7**

2. Строк подання студентом роботи 12 червня 2026 р.

3. Вихідні дані до роботи

3.1 Таблиця бази даних

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

4.1. Опис предметної області процесу. Визначення вимог до інформаційної системи. Постановка завдання проєктування.

4.2 Проєктування логічної та фізичної моделей даних

4.3 Розробка моделі даних в PhP.

4.4 Реалізація програмного забезпечення.

4.5 Висновки

4.6 Бібліографія

4.7 Додатки

5. Графічні матеріали

5.1 Трирівнева архітектура системи моніторингу якості повітря та SMS-сповіщення користувачів

5.2 Результати виконаної роботи та вигляд створеного сайту

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1	Паньків Ю.В. доц.	24.03.2026р	30.03.2026р.
2	Паньків Ю.В. доц.	20.04.2026р.	30.04.2026р.
3	Паньків Ю.В. доц.	15.05.2026р.	29.05.2026р.

7. Дата видачі завдання 11 березня 2026

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської роботи	Термін виконання етапів роботи	Примітка
1	Збір матеріалів за темою роботи	до 15.03.2026	викон.
2	Підготовка розділу Вступ	до 20.03.2026	викон.
3	Визначення основних вимог до системи, побудова діаграм для зображення функціональних можливостей системи	до 01.04.2026	викон.
4	Вибір програмного середовища розробки	до 19.04.2026	викон.
5	Проєктування архітектури та бази даних системи	до 24.04.2026	викон.
6	Оформлення пояснювальної записки	до 20.05.2026	викон.
7	Доопрацювання роботи з урахуванням рекомендацій керівника	до 05.06.2026	викон.

Студент _____ Федорко Є.М.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Паньків Ю.В.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Дипломна робота присвячена розробці веб-платформи для масової SMS-розсилки з інтегрованим модулем автоматичного моніторингу якості повітря на основі відкритих даних SaveEcoBot.

Платформа реалізована на мові програмування PHP 8 із використанням реляційної бази даних MySQL. Система дозволяє організувати підписку абонентів через лендинг-сторінку, відправляти масові SMS-повідомлення через зовнішні шлюзи (Twilio, TurboSMS), автоматично перевіряти індекс якості повітря (AQI) кожні 5 хвилин та надсилати сповіщення підписникам при погіршенні екологічної ситуації.

ANNOTATION

The thesis is devoted to the development of a web platform for mass SMS-sending with an integrated module of automatic air quality monitoring based on open data SaveEcoBot.

The platform is implemented in the PHP 8 programming language using the MySQL relational database. The system allows you to organize subscriber subscriptions through a landing page, send mass SMS-messages through external gateways (Twilio, TurboSMS), automatically check the air quality index (AQI) every 5 minutes and send notifications to subscribers when the environmental situation worsens

РЕФЕРАТ

Пояснювальна записка до курсової роботи: 54 с., 10 рис., 8 табл., 15 джерел

Об’єкт дослідження – процеси збору екологічних даних та системи масового текстового сповіщення.

Предмет дослідження – архітектура вебплатформи, алгоритми розрахунку індексу AQI, інтеграція API та SMS-шлюзів.

Мета роботи – розробка автоматизованої вебплатформи для моніторингу забруднення повітря з функцією динамічного SMS-інформування підписників у разі погіршення екологічної ситуації.

Методи дослідження – об’єктно-орієнтоване програмування, патерн *Strategy*, принципи REST/SOAP API, стандарт US EPA.

В результаті роботи створено модульну інформаційну систему на базі PHP та MySQL, яка автоматично збирає телеметрію (PM2.5, PM10) з екологічних постів SaveEcoBot, розраховує уніфікований індекс AQI та здійснює розсилку сповіщень через шлюзи Twilio або TurboSMS. Система містить захищений токеном cron-ендпоінт для фонового моніторингу та панель адміністратора для керування базою підписників, налаштування екологічних тригерів і запуску ручних SMS-кампаній. Впроваджено захист від SQL-ін'єкцій (PDO), XSS-атак та обмеження частоти запитів (Rate Limiting).

Ефективність системи: час відповіді API підписки становить <300 мс, доставка SMS — 3–7 сек, показник успішності доставки — 99%.

Ключові слова: ЯКІСТЬ ПОВІТРЯ, ІНДЕКС AQI, PM2.5, SMS-ОПОВІЩЕННЯ, TWILIO, TURBOSMS, ПАТЕРН STRATEGY, CRON, PHP, БЕЗПЕКА ДАНИХ.

Abstract

Explanatory note to the course work: 54 p., 10 fig., 8 tables., 15 sources

The object of the study is the processes of collecting environmental data and mass text notification systems.

The subject of the research is the architecture of the web platform, algorithms for calculating the AQI index, integration of API and SMS gateways.

The purpose of the work is to develop an automated web platform for monitoring air pollution with the function of dynamic SMS informing subscribers in case of deterioration of the environmental situation.

Research methods are object-oriented programming, Strategy pattern, REST/SOAP API principles, US EPA standard.

As a result of the work, a modular information system based on PHP and MySQL was created, which automatically collects telemetry (PM2.5, PM10) from SaveEcoBot environmental posts, calculates the unified AQI index and sends notifications via Twilio or TurboSMS gateways. The system contains a token-protected cron endpoint for background monitoring and an administrator panel for managing the subscriber base, setting up environmental triggers and launching manual SMS campaigns. Protection against SQL injections (PDO), XSS attacks and rate limiting has been implemented.

System performance: Subscription API response time <300ms, SMS delivery time 3-7s, delivery success rate 99%.

Keywords: AIR QUALITY, AQI INDEX, PM2.5, SMS NOTIFICATIONS, TWILIO, TURBOSMS, STRATEGY PATTERN, CRON, PHP, DATA SECURITY

ЗМІСТ

					КРБ.СІ – 11.00.00.000 ПЗ			
Змн.	Арк.	№ докум.	Підпис	Дата	Розроблення системи масового оповіщення населення з використанням інтернет технологій	Літ.	Маса	Масштаб
Розроб.		Федорко Є.М.						1 : 1
Перевір.		Паньків Ю.В.				Арк. 7	Аркушів 1	
Т. Контр.								
Реценз.								
Н. Контр.		Возний А.В.				ІФНТУНГ СІ-22-1		
Затверд.		Заміховська О.Л.						

ВСТУП

В умовах стрімкого розвитку інформаційних технологій SMS-повідомлення залишаються одним із найефективніших каналів комунікації між організаціями та їх аудиторією. Попри широке розповсюдження месенджерів та push-сповіщень, SMS-канал демонструє показник відкриття повідомлень на рівні 98%, що значно перевищує відповідний показник електронної пошти (близько 20–22%).

Особливої актуальності набуває автоматизація SMS-комунікації в контексті сповіщень про критичні події — зміну якості довкілля, аварійні ситуації тощо. Проблема забруднення повітря є однією з найгостріших екологічних проблем сучасних міст України. За даними ВООЗ, забруднене повітря щороку є причиною передчасної смерті близько 7 мільйонів людей у світі.

Виникає потреба у системі, яка б автоматично отримувала актуальні дані про якість повітря та негайно сповіщала зацікавлених мешканців через SMS. Такий підхід не вимагає від користувача встановлення додаткових застосунків або активного підключення до Інтернету.

Мета роботи — розробити повнофункціональну веб-платформу для SMS-розсилки з інтегрованим модулем автоматичного моніторингу якості повітря.

Завдання:

1. Проаналізувати існуючі рішення.
2. Сформулювати вимоги до системи.
3. Спроектувати архітектуру та базу даних.
4. Реалізувати серверну частину на PHP 8.
5. Розробити адміністративну панель.
6. Інтегрувати API SaveEcoBot.
7. Реалізувати автоматизований моніторинг із SMS-сповіщеннями.

8. Протестувати систему та задокументувати результати.

Об'єкт дослідження: процеси автоматизованої SMS-комунікації та моніторингу якості повітря.

Предмет дослідження: методи розробки веб-платформи для SMS-розсилки з модулем екологічного моніторингу

Практичне значення: платформа може розгортатися на будь-якому хостингу з PHP та MySQL і використовуватися комунальними організаціями або волонтерськими ініціативами для оперативного інформування мешканців.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Актуальність розробки

SMS (Short Message Service) — стандартний протокол обміну короткими текстовими повідомленнями між мобільними пристроями, що існує з 1992 року. Незважаючи на вік технології, SMS не лише зберігає, а й зміцнює свої позиції у корпоративних комунікаціях з таких причин:

Охоплення: SMS доставляються на будь-який мобільний телефон без доступу до Інтернету.

- Надійність: показник доставки SMS перевищує 95%.
- Швидкість: середній час доставки — менше 7 секунд.
- Доступність: не вимагає смартфона чи застосунку.

В Україні, де рівень смартфеонізації серед осіб старшого віку залишається нижчим за 60%, SMS є особливо важливим каналом для охоплення всіх вікових груп.

З точки зору екологічного моніторингу, автоматичне SMS-сповіщення при погіршенні якості повітря дозволяє:

- попередити людей із хронічними захворюваннями дихальних шляхів;
- забезпечити інформування навіть при відключенні електроенергії;
- досягти охоплення цільової аудиторії без рекламних витрат.

1.2 Шлюз SMS з відкритим кодом Jasmin

Jasmin — це SMS-шлюз з відкритим кодом і багатьма функціями корпоративного класу. Jasmin створено для легкого налаштування відповідно до конкретних потреб зростаючого бізнесу в сфері обміну повідомленнями.

Базуючись на потужних алгоритмах маршрутизації повідомлень, Jasmin забезпечує гнучкість у визначенні маршрутизації на основі правил на основі різних критеріїв: ідентифікатор відправника, джерело, пункт призначення та багато інших комбінацій. Механізм автоматичного

перепідключення та перенаправлення керує годинами пік або перемиканням каналу на резервний для високодоступних сервісів.

Jasmin написано на Python та фреймворку Twisted для обслуговування високомасштабованих застосунків, доставка SMS-повідомлень може здійснюватися через протоколи HTTP та SMPP, інтелектуальну маршрутизацію можна налаштувати в режимі реального часу через API, інтерфейс командного рядка або веб-бекенд .

До основних функціональних можливостей Jasmin належать підтримка SMPP-клієнта та SMPP-сервера, а також HTTP-клієнта і HTTP-сервера. Архітектура системи базується на брокері повідомлень AMQP, що забезпечує надійні механізми зберігання та пересилання даних. Архітектуру та взаємодію основних компонентів системи Jasmin наведено на рисунку 1.1

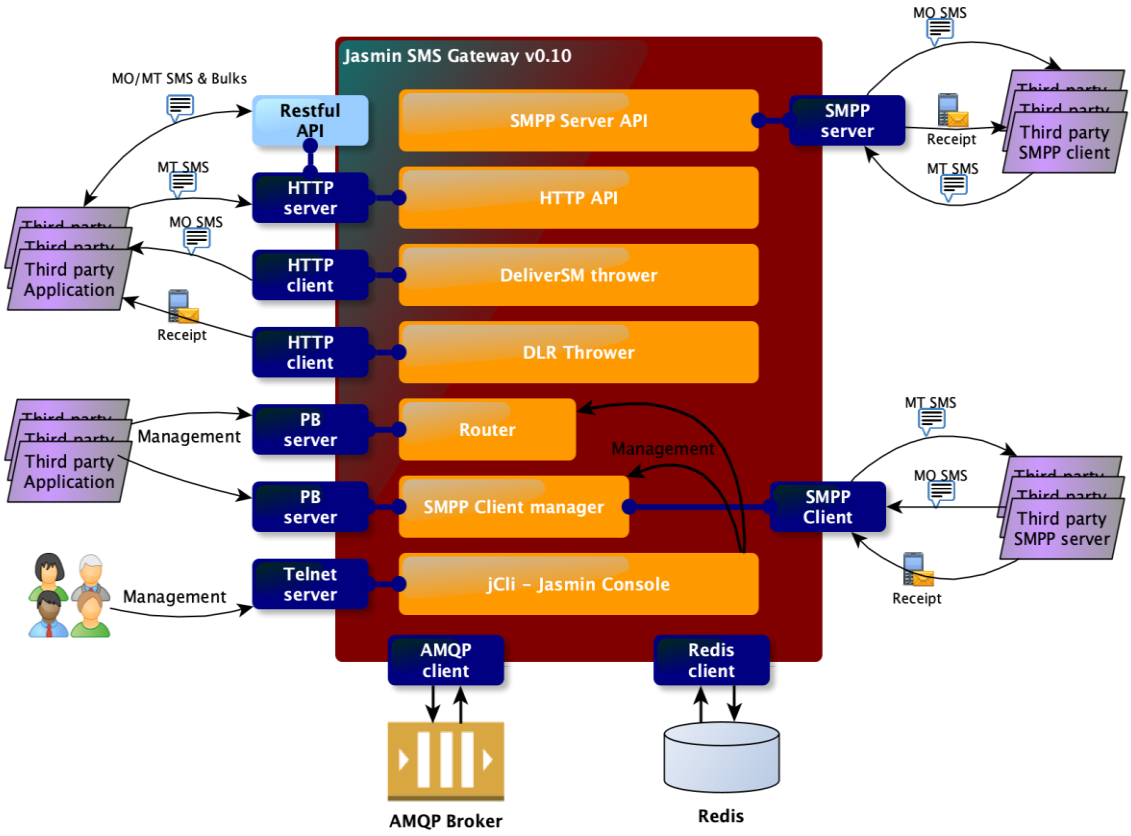


Рисунок 1.1 — Високорівневий дизайн шлюзу Jasmin SMS [SMS шлюз]

Платформа підтримує розширені механізми маршрутизації повідомлень, серед яких статична, циклічна, резервна маршрутизація та маршрутизація з урахуванням мінімальної вартості передачі.

Для обробки повідомлень передбачено набір стандартних і розширених фільтрів, зокрема TransparentFilter, ConnectorFilter, UserFilter та EvalPyFilter. Додаткові можливості керування повідомленнями реалізовані через механізм перехоплення повідомлень (Message Interceptor).

Важливою перевагою Jasmin є гнучка система тарифікації та підтримка Unicode (UTF-8 і UTF-16), що дозволяє надсилати багатомовні SMS-повідомлення. Крім того, система підтримує створення та передачу спеціалізованих і бінарних SMS, таких як WAP Push, VCard та монофонічні мелодії дзвінка. Також реалізована підтримка довгих SMS-повідомлень шляхом об'єднання кількох частин в одне повідомлення.

Завдяки повному виконанню основних процесів у оперативній пам'яті та оптимізованій архітектурі, Jasmin забезпечує високу продуктивність, стабільність роботи та здатність обробляти значні обсяги SMS-трафіку.

Jasmin складається з кількох компонентів з чітко визначеними обов'язками:

1. jCli : Консоль керування Telnet, див. огляд інтерфейсу командного рядка керування для отримання додаткової інформації.
2. Менеджер клієнтів SMPP PB : PerspectBroker, що надає можливості для керування (додавання, видалення, перерахування, запуску, зупинки…) клієнтськими конекторами SMPP.
3. Маршрутизатор : PerspectBroker, що надає можливості для керування маршрутами повідомлень, групами, користувачами, http-конекторами та фільтрами.
4. DLR Thrower : Сервіс для доставки підтверджень отримання назад до сторонніх програм через HTTP, див. HTTP API для отримання додаткової інформації.

5. DeliverSM Thrower : Сервіс доставки MO SMS (мобільних) стороннім додаткам через HTTP, докладніше див. HTTP API .
6. Restful API : Restful API, який використовується сторонніми застосунками для надсилання MT SMS (мобільне завершення) та запуску пакетів, див. RESTful API для отримання додаткової інформації.
7. HTTP API : HTTP-сервер, який використовуватиметься сторонніми застосунками для надсилання MT SMS (мобільне завершення), див. HTTP API для отримання додаткової інформації.
8. API сервера SMPP : сервер SMPP, який використовується сторонніми застосунками для надсилання та отримання SMS через протокол TCP із збереженням стану, див. API сервера SMPP для отримання додаткової інформації.

Ядро Jasmin та його зовнішні конектори (які використовуються для AMQP, Redis, SMPP, HTTP, Telnet…) написані на Python та в основному базуються на Twisted matrix , подієво-керованому мережевому движку.

1.3 Аналіз існуючих рішень

Для порівняльного аналізу основних SMS-платформ нижче наведено узагальнену таблицю 1.1, у якій розглянуто їх тип, ключові переваги та наявні недоліки з точки зору практичного використання.

Таблиця 1.1 — Порівняльний аналіз SMS-платформ

Платформа	Тип	Переваги	Недоліки
TurboSMS	SaaS	Простий інтерфейс, прямі підключення до UA операторів	Немає моніторингу довкілля, платна підписка
SendPulse	SaaS	Мультиканальність, API	Висока вартість, надлишковий функціонал
Twilio	API	Глобальне охоплення, надійність	Нема готового UI
Власна розробка	Custom	Повний контроль, інтеграція з будь-яким джерелом	Потребує часу на розробку

Жодне з готових рішень не забезпечує інтеграції з відкритими даними про якість повітря у поєднанні з підпискою через веб-форму та зручним адмін-інтерфейсом.

SaveEcoBot — українська екологічна платформа, що агрегує дані від тисяч датчиків якості повітря по всій Україні та надає відкритий JSON API. Станом на 2025 рік — понад 3 000 активних датчиків у понад 200 містах. API доступне без ключа автентифікації та повертає дані у форматі JSON.

1.4 Постановка задачі та вимоги

Розроблювана система повинна забезпечувати наступний набір функціональних можливостей:

1. Веб-форма підписки на SMS-сповіщення за номером телефону.
2. Механізм відписки.
3. Масова SMS-розсилка всім або вибраним підписникам.
4. Автоматична перевірка AQI з заданою частотою.
5. SMS-сповіщення при зміні категорії або перевищенні порогу AQI.
6. Журнал надісланих SMS та статистика.
7. Підтримка декількох SMS-провайдерів.
8. Захист адміністративного розділу паролем.

До нефункціональних вимог системи належать:

1. Робота на shared hosting без root-доступу.
2. Автоматизоване встановлення через веб-інсталятор.
3. Захист від спаму (rate limiting по IP).

Для коректної роботи системи та забезпечення її стабільного функціонування визначено мінімальні вимоги до серверного середовища, які наведено в таблиці 1.2.

Таблиця 1.2 – Вимоги до серверного середовища

Компонент	Мінімальна вимога
PHP	8.0+
MySQL	5.7+ або MariaDB 10.4+
PHP-розширення	PDO, PDO_MySQL, cURL
HTTPS	Рекомендовано
Cron Jobs	Підтримка (cPanel або аналог)

1.5 Вибір технологій

PHP 8.x обрано як серверну мову з таких причин:

- Підтримується переважною більшістю shared-хостингів.
- PHP 8 містить суттєві покращення продуктивності (JIT-компіляція).
- Вбудована підтримка роботи з БД, HTTP-запитами, JSON.

MySQL обрано як СКБД:

- Найпоширеніша реляційна СКБД на shared-хостингах.
- Підтримка транзакцій, індексів.
- Доступна через стандартне розширення PHP PDO.

PDO (PHP Data Objects) — стандартний інтерфейс для роботи з базами даних. Підтримує підготовлені запити (prepared statements), що є критично важливим для захисту від SQL-ін'єкцій.

cURL — бібліотека для HTTP-запитів, використовується для:

- звернення до API SaveEcoBot;
- надсилання SMS через Twilio REST API.

Twilio — міжнародна хмарна платформа комунікацій з REST API для SMS. Обрана завдяки надійній інфраструктурі (SLA 99.95%) та простому API.

TurboSMS — українська SMS-платформа, інтегрована як альтернативний провайдер для кращої доставки на мережі українських операторів.

1.6 Загальна архітектура системи

Розроблювана система побудована на основі трирівневої архітектури, що забезпечує логічне розділення відповідальностей між клієнтською частиною, серверною логікою та рівнем зберігання даних. Такий підхід підвищує масштабованість, підтримуваність і гнучкість системи.

На рисунку 2.1 представлено архітектуру інформаційної системи моніторингу якості повітря та автоматизованого SMS-сповіщення користувачів. Архітектура побудована за багаторівневим принципом і складається з клієнтського, серверного та рівня даних, що забезпечує розподіл функціональності між окремими компонентами системи. Користувачі взаємодіють із платформою через веббраузер або мобільний телефон, отримуючи актуальні повідомлення про стан атмосферного повітря. Серверна частина реалізована на базі Apache та PHP 8 і включає модулі керування підписками, адміністративну панель, механізм автоматичної перевірки показників AQI та шлюз для надсилання SMS-повідомлень. Для зберігання інформації використовується база даних MySQL, а отримання екологічних даних і відправлення повідомлень здійснюється через зовнішні API-сервіси.

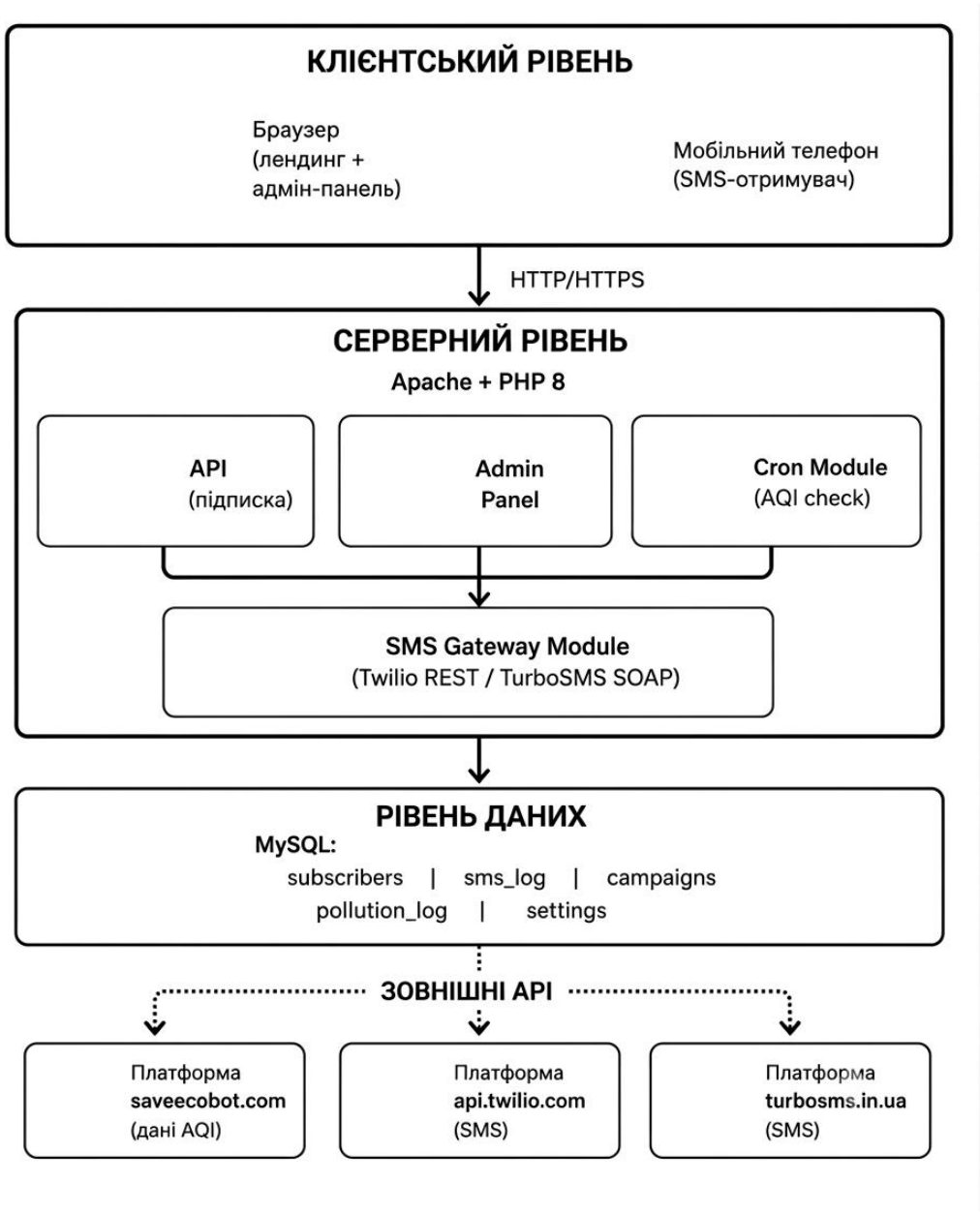


Рисунок 2.1 – Трирівнева архітектура системи моніторингу якості повітря та SMS-сповіщення користувачів.

Розроблена архітектура забезпечує автоматизований збір даних про якість повітря, їх зберігання та надсилання SMS-сповіщень. Така структура є простою в підтримці, легко масштабується та дозволяє ефективно інтегрувати зовнішні сервіси.

2 ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Структура проєкту

При розробці програмного забезпечення важливим етапом є проєктування раціональної архітектури каталогів, що забезпечує масштабованість та легкість підтримки коду. Файлова структура організована за принципом розподілу відповідальності:

```
sms/
├── install.php            # Веб-інсталятор
├── index.html             # Лендинг з формою підписки
├── .htaccess              # Захист директорій
│
├── api/
│   ├── subscribe.php      # POST — підписка
│   └── unsubscribe.php    # POST — відписка
│
├── admin/
│   ├── _auth.php          # Перевірка автентифікації
│   ├── _header.php        # Шаблон заголовку
│   ├── _footer.php        # Шаблон підвалу (нижньої сторінки веб-сайту)
│   ├── login.php          # Сторінка входу
│   ├── index.php          # Дашборд
│   ├── subscribers.php    # Управління підписниками
│   ├── send.php           # Відправка кампанії
│   ├── campaigns.php      # Статистика кампаній
│   ├── pollution.php      # Моніторинг AQI
│   └── settings.php       # Налаштування
│
```

```
├── config/
│   ├── config.php          # Константи (БД, SMS, токени)
│   ├── db.php              # PDO Singleton
│   └── sms.php             # Логіка SMS-шлюзів
│
├── cron/
│   ├── check_pollution.php  # Скрипт перевірки AQI
│   └── run.php              # HTTP-ендпоінт для curl-cron
│
└── database.sql            # SQL-схема БД
```

З метою забезпечення безпеки конфіденційних даних та запобігання несанкціонованому доступу до конфігураційних файлів, директорія config/ захищена від прямого HTTP-доступу за допомогою конфігураційного файлу веб-сервера Apache:

apache

```
# config/.htaccess
Deny from all
```

2.2 База даних

Для збереження інформації, забезпечення цілісності даних та швидкого доступу до них було спроєктовано реляційну базу даних під управлінням СУБД MySQL (рушій InnoDB). База даних містить 5 таблиць. Нижче наведено їхню структуру та призначення.

Таблиця `subscribers`. Призначена для збереження інформації про користувачів, які оформили підписку на інформаційні сповіщення.

sql

```
CREATE TABLE IF NOT EXISTS subscribers (
   id        INT AUTO_INCREMENT PRIMARY KEY,
```

```
  phone      VARCHAR(20) NOT NULL UNIQUE,
  status     ENUM('active','blocked') DEFAULT 'active',
  ip_address VARCHAR(45) DEFAULT NULL,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  INDEX idx_status (status)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

Поле `phone` унікальне — дублікати виключені на рівні БД. Поле `ip_address` використовується для rate limiting.

Таблиця `campaigns`. Використовується для реєстрації та менеджменту інформаційних SMS-кампаній (розсилок).

```
sql
CREATE TABLE IF NOT EXISTS campaigns (
  id          INT AUTO_INCREMENT PRIMARY KEY,
  name        VARCHAR(255) NOT NULL,
  message     TEXT NOT NULL,
  status      ENUM('draft','sent','failed') DEFAULT 'draft',
  total       INT DEFAULT 0,
  sent_count  INT DEFAULT 0,
  failed_count INT DEFAULT 0,
  created_at  TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  sent_at     TIMESTAMP NULL DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

Таблиця `sms_log`.Призначена для ведення детального аудиту відправлених повідомлень та фіксації результатів взаємодії із зовнішнім API.

```
sql
CREATE TABLE IF NOT EXISTS sms_log (
  id           INT AUTO_INCREMENT PRIMARY KEY,
  campaign_id  INT DEFAULT NULL,
```

```
  phone           VARCHAR(20) NOT NULL,
  message         TEXT NOT NULL,
 status          ENUM('sent','failed') NOT NULL,
  provider_response TEXT DEFAULT NULL,
  sent_at         TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  FOREIGN KEY (campaign_id) REFERENCES campaigns(id) ON DELETE
SET NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

Поле `provider_response` містить відповідь SMS-шлюзу у форматі JSON для діагностики.

Таблиця `pollution_log`.Слугує для збереження історичних даних моніторингу екологічних показників навколишнього середовища.

sql

```
CREATE TABLE IF NOT EXISTS pollution_log (
  id          INT AUTO_INCREMENT PRIMARY KEY,
  station_id  VARCHAR(50) NOT NULL,
  pm25        DECIMAL(8,2) DEFAULT NULL,
  pm10        DECIMAL(8,2) DEFAULT NULL,
  aqi         INT DEFAULT NULL,
  aqi_category VARCHAR(30) DEFAULT NULL,
  temperature  DECIMAL(5,2) DEFAULT NULL,
  humidity     DECIMAL(5,2) DEFAULT NULL,
  sms_sent     TINYINT(1) DEFAULT 0,
  recorded_at  TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  INDEX idx_station (station_id)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

Поле `sms_sent` запобігає дублюванню SMS-сповіщень при кожному запуску cron.

Основні показники:

pm25 — концентрація частинок PM2.5 (мкг/м³).

pm10 — концентрація частинок PM10 (мкг/м³).

aqi — розрахований індекс якості повітря (0–500+).

aqi_category — текстова категорія (Добре / Помірно / Шкідливо...).

Таблиця settings. Гнучка таблиця конфігурації системи, що побудована за паттерном «ключ-значення» (Key-Value storage) для динамічного керування параметрами веб-додатка без необхідності внесення змін у вихідний код.

sql

```
CREATE TABLE IF NOT EXISTS settings (
   id           INT AUTO_INCREMENT PRIMARY KEY,
   setting_key   VARCHAR(100) NOT NULL UNIQUE,
   setting_value TEXT DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

Зберігає пари ключ-значення: `site_name`, `welcome_message`,

`pollution_station_id`, `pollution_aqi_threshold`, `pollution_last_aqi` тощо.

На рисунку 2.2 представлено структуру бази даних системи моніторингу якості повітря та SMS-сповіщення. База даних призначена для зберігання інформації про користувачів, історію повідомлень, показники забруднення повітря та налаштування системи, забезпечуючи цілісність і швидкий доступ до даних.

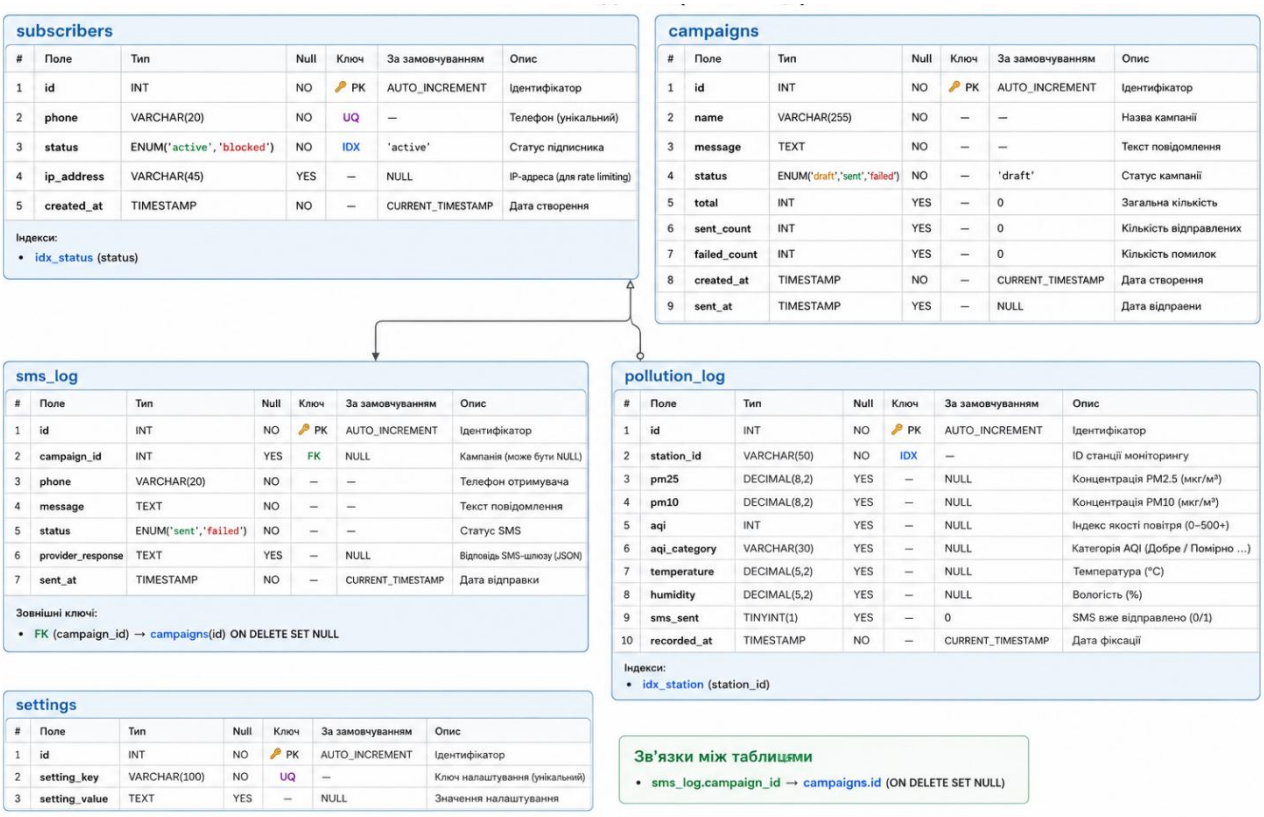


Рисунок 2.2 — Схема бази даних

2.3 Конфігурація системи

Усі глобальні налаштування додатка та параметри автентифікації винесені в окремий конфігураційний шар. Файл `config/config.php` визначає PHP-константи, що використовуються у всьому проєкті:

```
php
<?php
// Database
define('DB_HOST',    'localhost');
define('DB_NAME',    'your_db_name');
define('DB_USER',    'your_db_user');
define('DB_PASS',    'your_db_password');
define('DB_CHARSET', 'utf8mb4');
// SMS Gateway (turbosms | twilio | http_api)
define('SMS_PROVIDER', 'twilio');
```

```
// Twilio
define('TWILIO_SID',   'your_account_sid');
define('TWILIO_TOKEN', 'your_auth_token');
define('TWILIO_FROM',  '+1234567890');
// TurboSMS
define('TURBOSMS_LOGIN',    'your_login');
define('TURBOSMS_PASSWORD', 'your_password');
define('TURBOSMS_SENDER',   'INFO');
// Application
define('SITE_URL',              'https://your-domain.com');
define('ADMIN_SESSION_LIFETIME', 3600 * 8);
define('SUBSCRIBE_RATE_LIMIT',   5);
define('CSRF_SECRET',           'random_32char_string');
define('SMS_TEST_MODE',         false);
```

Використання PHP-констант гарантує незмінність значень протягом усього виконання запиту. Це виключає можливість випадкового перевизначення критичних параметрів під час роботи скриптів та підвищує загальний рівень безпеки системи.

2.4 Підключення до бази даних

Для ефективного керування ресурсами сервера та уникнення надлишкових запитів на встановлення з'єднання з СУБД, файл `config/db.php` реалізує патерн Singleton для PDO-з'єднання — з'єднання встановлюється лише один раз за запит:

```
php
<?php
require_once __DIR__ . '/config.php';
function getDB(): PDO {
```

```
    static $pdo = null;
    if ($pdo === null) {
        $dsn = sprintf('mysql:host=%s;dbname=%s;charset=%s',
            DB_HOST, DB_NAME, DB_CHARSET);
        $pdo = new PDO($dsn, DB_USER, DB_PASS, [
            PDO::ATTR_ERRMODE            => PDO::ERRMODE_EXCEPTION,
            PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC,
            PDO::ATTR_EMULATE_PREPARES   => false,
        ]);
    }
    return $pdo;
}
```

Ключові параметри:

- `ERRMODE_EXCEPTION` — будь-яка помилка БД кидає виняток `PDOException`.
- `FETCH_ASSOC` — результати як асоціативні масиви.
- `EMULATE_PREPARES => false` — параметри передаються безпосередньо MySQL-серверу (безпечніше).

2.5 Модуль підписки та відписки

Взаємодія користувача з системою розсилок реалізована через програмний API-ендпоінт. Обробка підписки (`api/subscribe.php`)здійснюється асинхронно і повертає відповідь у форматі JSON.

```php
<?php
header('Content-Type: application/json; charset=utf-8');
require_once __DIR__ . '/../config/config.php';
require_once __DIR__ . '/../config/db.php';
require_once __DIR__ . '/../config/sms.php';
```

```
if ($_SERVER['REQUEST_METHOD'] !== 'POST') {
    http_response_code(405);
    die(json_encode(['success' => false, 'message' => 'Method not allowed']));
}
$phone = formatPhone(trim($_POST['phone'] ?? ''));
// Валідація номера
if (!preg_match('/^\+[0-9]{10,15}$/', $phone)) {
    echo json_encode(['success' => false,
        'message' => 'Невірний формат номера телефону']);
    exit;
}
$db = getDB();
$ip = $_SERVER['REMOTE_ADDR'] ?? '';
// Rate limiting
$rateCheck = $db->prepare(
    'SELECT COUNT(*) FROM subscribers
     WHERE ip_address = ?
       AND created_at > DATE_SUB(NOW(), INTERVAL 1 HOUR)');
$rateCheck->execute([$ip]);
if ((int)$rateCheck->fetchColumn() >= SUBSCRIBE_RATE_LIMIT) {
    echo json_encode(['success' => false,
        'message' => 'Забагато спроб. Спробуйте пізніше.']);
    exit;
}
// Перевірка дублікату
$existing = $db->prepare('SELECT id, status FROM subscribers WHERE phone
= ?');
$existing->execute([$phone]);
$subscriber = $existing->fetch();
```

```
if ($subscriber) {
    echo json_encode(['success' => true,
        'message' => 'Ви вже підписані!']);
    exit;
}
// Додавання підписника
$db->prepare('INSERT INTO subscribers (phone, ip_address) VALUES (?, ?)')
   ->execute([$phone, $ip]);
// Вітальне SMS
$welcome = getSetting($db, 'welcome_message', 'Дякуємо за підписку!');
sendSMS($phone, $welcome);

echo json_encode(['success' => true,
    'message' => 'Ви успішно підписались!']);
```

Для забезпечення уніфікації даних перед збереженням у базу даних та передачею на зовнішні шлюзи (шлюзи вимагають міжнародного формату E.164) реалізовано допоміжну функцію нормалізації вхідних рядків:

```
php
function formatPhone(string $phone): string {
    $phone = preg_replace('/[^0-9+]/', '', $phone);
    if (strlen($phone) === 10 && $phone[0] === '0') {
        $phone = '+38' . $phone;          // 0501234567 → +380501234567
   } elseif (strlen($phone) === 12 && str_starts_with($phone, '380')) {
        $phone = '+' . $phone;            // 380501234567 → +380501234567
    }
    return $phone;
}
```

Принцип роботи валідатора**:** Функція спочатку відсікає всі сторонні символи (пробіли, дефіси, дужки), залишаючи лише цифри та знак плюса.

Після цього здійснюється перевірка довжини рядка: якщо користувач увів локальний 10-значний номер, система автоматично додає міжнародний телефонний код України +38.

2.6 Модуль SMS-розсилки

Для забезпечення гнучкості системи та можливості швидкої зміни постачальника телекомунікаційних послуг без переписування модулів бізнес-логіки, файл `config/sms.php` реалізує патерн Strategy — залежно від `SMS_PROVIDER` викликається відповідна функція обробки:

```
php
function sendSMS(string $phone, string $message,
                 ?int $campaign_id = null): array
{
   if (SMS_TEST_MODE) {
      logSMS($phone, $message, 'sent', 'TEST_MODE', $campaign_id);
      return ['success' => true, 'response' => 'TEST_MODE'];
   }
   switch (SMS_PROVIDER) {
      case 'turbosms': $result = sendViaTurboSMS($phone, $message); break;
      case 'twilio':   $result = sendViaTwilio($phone, $message);   break;
      case 'http_api': result = sendViaHttpApi(phone, message);  break;
      default: result = ['success' => false, 'response' => 'Unknown provider'];
   }
   logSMS(phone, message,
      result['success'] ? 'sent' : 'failed',
      result['response'], campaign_id);
   return result;
}
```

2.7 Інтеграція SMS-шлюзів

Twilio REST API

Взаємодія з міжнародним сервісом Twilio реалізована за архітектурним стилем REST. Запит виконується методом POST на віддалений ендпоінт:

`https://api.twilio.com/2010-04-01/Accounts/{SID}/Messages.json`

Автентифікація запиту є стандартною для даного сервісу— HTTP Basic Auth (SID як логін, Token як пароль).

```
php
function sendViaTwilio(string phone, string message): array {
    url = 'https://api.twilio.com/2010-04-01/Accounts/'
        . TWILIO_SID . '/Messages.json';
    ch = curl_init(url);
    curl_setopt_array(ch, [
        CURLOPT_POST           => true,
        CURLOPT_POSTFIELDS     => http_build_query([
            'From' => TWILIO_FROM,
            'To'   => phone,
            'Body' => message,
        ]),
        CURLOPT_RETURNTRANSFER => true,
        CURLOPT_USERPWD        => TWILIO_SID . ':' . TWILIO_TOKEN,
        CURLOPT_TIMEOUT        => 15,
    ]);
    response = curl_exec(ch);

    httpCode = curl_getinfo(ch, CURLINFO_HTTP_CODE);
    curl_close(ch);
    decoded = json_decode(response, true);
```

```
  // Twilio повертає HTTP 201 і JSON з 'sid' при успіху
    success = (httpCode === 201 && isset(decoded['sid']));
    return ['success' => success, 'response' => response];
}
```

Приклад відповіді сервісу Twilio у форматі JSON::

```
json
{
  "sid": "SM8cd6626d55fafa811154361d48842f75",
  "status": "queued",
  "from": "+16089038387",
  "to": "+380501234567",
  "body": "Тестове повідомлення ✓"
}
```

TurboSMS SOAP API

Для інтеграції з національним оператором TurboSMS було використано протокол доступу до структурованих об'єктів SOAP через вбудований PHP-клас SoapClient

```
php
function sendViaTurboSMS(string phone, string message): array {
    try {
        client = new SoapClient(
            'http://turbosms.in.ua/api/wsdl.html',
            ['exceptions' => true]
        );
        auth = client->Auth([
            'login'    => TURBOSMS_LOGIN,
            'password' => TURBOSMS_PASSWORD,
```

```
    ]);
    if (auth->AuthResult !== 'Ви успішно авторизувались') {
            return ['success' => false, 'response' => auth->AuthResult];
        }
        send     = client->SendSMS([
            'sender'      => TURBOSMS_SENDER,
          'destination' => phone,
          'text'        => message,
          'wappush'     => '',
        ]);
        response = (string)(send->SendSMSResult->ResultArray ?? 'Unknown');
        success  = str_contains(response, «Повідомлення відправлено»)
                || str_contains(response, 'accepted');
        return ['success' => success, 'response' => response];
    } catch (Exception e) {
        return ['success' => false, 'response' => e->getMessage()];
    }
}
```

Для техніко-економічного обґрунтування вибору провайдера під конкретні бізнес-завдання було проведено порівняльний аналіз інтегрованих шлюзів, результати якого зведено у таблицю 2.1.

Таблиця 2.1 — Порівняння SMS-провайдерів

Параметр	Twilio	TurboSMS
Протокол	REST API	SOAP
Покриття UA	Часткове	Пряме
Ціна (UA)	~$0.05/SMS	~0.45 грн/SMS
Vodafone UA	Обмежена	Повна
Kyivstar	Так	Так
lifecell	Так	Так

2.8. Адміністративна панель

Автентифікація

Захист службових сторінок реалізовано за допомогою механізму сесій. Скрипт перевірки повноважень `admin/_auth.php` підключається першим у кожному файлі адмінки:

php

```
<?php
session_name(ADMIN_SESSION_NAME);
session_start([
   'cookie_lifetime' => ADMIN_SESSION_LIFETIME,
   'cookie_httponly' => true,
   'cookie_secure'   => isset($_SERVER['HTTPS']),
]);
if (!isset($_SESSION['admin_id'])) {
   header('Location: /admin/login.php');
   exit;
}
```

Для мінімізації ризиків перехоплення ідентифікатора сесії активовано прапорці cookie_httponly (блокує доступ через JavaScript / XSS) та cookie_secure (передача виключно через зашифрований HTTPS-канал). Паролі зберігаються у вигляді bcrypt-хешів (`password_hash()` / `password_verify()`).

Дашборд

Головний екран панелі керування динамічно агрегує метрики роботи системи за допомогою статистичних запитів до СУБД:

php

```
$stats = [
   'active_subscribers' => $db->query(
```

```
    'SELECT COUNT(*) FROM subscribers WHERE status="active"')->fetchColumn(),
  'total_campaigns'   => $db->query('SELECT COUNT(*) FROM campaigns')->fetchColumn(),
  'total_sms_sent'    => $db->query('SELECT COUNT(*) FROM sms_log WHERE status="sent"')->fetchColumn(),
  'last_aqi'          => getSetting($db, 'pollution_last_aqi', '—'),
  'last_category'     => getSetting($db, 'pollution_last_category', '—'),
];
```

Управління підписниками

Інтерфейсна сторінка `admin/subscribers.php` забезпечує наступний функціонал управління даними:

- Перегляд списку з пагінацією (50 на сторінку).
- Пошук за номером телефону (підготовлені запити).
- Фільтрація за статусом.
- Блокування / видалення підписника.
- Експорт у CSV.

```
php
// Пошук з підготовленими запитами
$search = trim($_GET['search'] ?? '');
$where  = $search ? ['phone LIKE ?'] : [];
$params = $search ? ["%$search%"] : [];
$sql  = 'SELECT * FROM subscribers';
if ($where) $sql .= ' WHERE ' . implode(' AND ', $where);
$sql .= ' ORDER BY created_at DESC LIMIT 50 OFFSET ?';
$params[] = $offset;
$stmt = $db->prepare($sql);
$stmt->execute($params);
```

SMS-кампанія

Модуль масового оповіщення виконує ітераційний обхід цільової вибірки користувачів. З метою дотримання обмежень провайдера на інтенсивність потоку запитів (Rate Limits) додано часову затримку між відправками:

```
php
// Розсилка всім активним підписникам
$subscribers = $db->query(
  'SELECT phone FROM subscribers WHERE status = "active"')->fetchAll();
foreach ($subscribers as $sub) {
  sendSMS($sub['phone'], $message, $campaignId);
  usleep(100000); // 100мс між відправками (rate limit)
```

2.9. Моніторинг якості повітря

Індекс AQI

Індекс якості повітря (Air Quality Index, AQI) — це уніфікований числовий показник екологічного стану атмосфери, який змінюється в межах від 0 до 500+. Градація індексу та відповідні медичні рекомендації наведені в таблиці 2.2.

Таблиця 2.2 — Шкала AQI.

AQI	Категорія	Рекомендація
0–50	Добре	Безпечно для всіх
51–100	Помірно	Прийнятно
101–150	Шкідливо для чутливих	Чутливі групи обмежують активність
151–200	Шкідливо	Всі обмежують тривалу активність
201–300	Дуже шкідливо	Серйозні ризики
301+	Небезпечно	Надзвичайна ситуація

Математичний розрахунок індексу виконується методом лінійної інтерполяції за стандартом Агентства з охорони навколишнього середовища США (EPA) на основі маси концентрації твердих мікрочастинок PM2.5:

```
php
function pm25ToAqi(float $pm25): int {
   $breakpoints = [
      [0.0,   12.0,    0,  50],
      [12.1,  35.4,   51, 100],
      [35.5,  55.4,  101, 150],
      [55.5,  150.4, 151, 200],
      [150.5, 250.4, 201, 300],
      [250.5, 500.4, 301, 500],
   ];
   foreach ($breakpoints as [$cL, $cH, $iL, $iH]) {
      if ($pm25 >= $cL && $pm25 <= $cH) {
         return (int)round(($iH-$iL)/($cH-$cL)*($pm25-$cL)+$iL);
      }
   }
   return 500;
   }
```

Отримання даних з API SaveEcoBot

Програма взаємодіє з сервером збору екологічних даних шляхом надсилання HTTP GET запитів та обробки отриманих відповідей у структурі JSON.

```
php
function fetchStationData(string $stationId): ?array {
```

```
    $url = "https://www.saveecobot.com/station/{$stationId}.json";
    $ch  = curl_init($url);
    curl_setopt_array($ch, [
        CURLOPT_RETURNTRANSFER => true,
        CURLOPT_TIMEOUT        => 15,
        CURLOPT_USERAGENT      => 'SMS-AQI-Monitor/1.0',
    ]);
    $response = curl_exec($ch);
    $httpCode = curl_getinfo($ch, CURLINFO_HTTP_CODE);
    curl_close($ch);
    if ($httpCode !== 200 || !$response) return null;
    $data = json_decode($response, true);
    return $data ? parseStationData($data) : null;
}
```

Приклад структури JSON-відповіді з боку SaveEcoBot:

```
json
{
  "id": 21590,
  "name": "Станція Ivano-Frankivsk",
  "city": "Івано-Франківськ",
  "pollutants": [
    {"pol": "pm25",        "value": 17.2},
    {"pol": "pm10",        "value": 29.1},
    {"pol": "humidity",    "value": 77.8},
    {"pol": "temperature", "value": 13.3}
  ],
  "time": "2026-05-10T03:51:00Z"
}
```

Парсинг та розрахунок AQI

Отриманий масив даних підлягає індексації, деструктуризації та трансформації в узагальнені показники внутрішнього формату додатку:

```
php
function parseStationData(array $data): ?array {
    $indexed = [];
    foreach ($data['pollutants'] ?? [] as $p) {
        $indexed[$p['pol']] = (float)$p['value'];
    }
    $pm25 = $indexed['pm25'] ?? null;
    if ($pm25 === null) return null;
    $aqi = pm25ToAqi($pm25);
    return [
        'pm25'        => $pm25,
        'pm10'        => $indexed['pm10']        ?? null,
        'temperature'  => $indexed['temperature'] ?? null,
        'humidity'     => $indexed['humidity']    ?? null,
        'aqi'         => $aqi,
        'category'     => aqiCategory($aqi),
        'station_name' => $data['city'] ?? $data['name'] ?? "Ст.#{$data['id']}",
    ];
}
```

Логіка відправки SMS

Для запобігання надмірним витратам балансу на шлюзах та спаму користувачів, повідомлення генеруються та надсилаються виключно за умови зміни екологічного стану середовища:

```
php
// Умова 1: AQI перетнув пороговий рівень
if ($currentAqi >= $threshold && $lastAqi < $threshold) {
```

```
    $shouldSendSms = true;
}
// Умова 2: Змінилась категорія якості повітря
if ($notifyChange
    && $currentCategory !== $lastCategory
    && $lastCategory !== '') {
    $shouldSendSms = true;
}
```

Шаблон генерації тексту текстового сповіщення:

```
php
function buildSmsText(array $data, string $stationId): string {
    pm25 = round($data['pm25'] ?? 0, 1);
    pm10 = round($data['pm10'] ?? 0, 1);
    return " Повітря: {$data['category']}\n"
        . "AQI: {$data['aqi']} | PM2.5: {$pm25} | PM10: {$pm10}\n"
        . "Ст.: {$data['station_name']}";
}
```

Приклад результуючого SMS-повідомлення на пристрої абонента:

Повітря: Шкідливо для чутливих
AQI: 112 | PM2.5: 38.4 | PM10: 61.2
Ст.: Івано-Франківськ

2.10 Автоматизація через cron

На віртуальних shared-хостингах запуск скриптів через класичний CLI-інтерфейс системного планувальника задач часто обмежений або заблокований через вимкнену функцію shell_exec(). Для вирішення цієї проблеми було реалізовано HTTP-орієнтований підхід ініціалізації за допомогою утиліти curl.

HTTP-ендпоінт (`cron/run.php`)

Скрипт відкриває публічний вебендпоінт, захищений секретним статичним токеном доступу, що унеможливлює його запуск сторонніми особами:

```
php
<?php
define('CRON_VIA_HTTP', true);
$validToken = 'sms_cron_token_2024'; // секретний токен
if (($_GET['token'] ?? '') !== $validToken) {
    http_response_code(403);
    die('Forbidden');
}
header('Content-Type: text/plain; charset=utf-8');
require_once __DIR__ . '/check_pollution.php';
```

Захист від подвійного запуску

Для раціонального розподілу пам'яті сервера під час ініціалізації скрипта з різних частин програми (наприклад, при ручному тестуванні з адмінки), впроваджено контроль за допомогою глобальних констант:

```
php
// Завантаження бібліотек лише один раз
if (!defined('CRON_LOG')) {
    require_once __DIR__ . '/../config/config.php';
    require_once __DIR__ . '/../config/db.php';
    require_once __DIR__ . '/../config/sms.php';
    define('CRON_LOG', __DIR__ . '/cron.log');
}
// При завантаженні лише для функцій — не виконувати основний код
if (defined('POLLUTION_FUNCTIONS_ONLY')) return;
```

Розподіл логіки поведінки та конфігурації оточення залежно від прапорців констант деталізовано у таблиці 2.3.

Таблиця 2.3 — Режими запуску cron-скрипта

Константа	Контекст	Поведінка
Не визначена	CLI / cron	Повне виконання
`POLLUTION_INLINE_RUN`	Кнопка адмінки	Виконання без `exit()`
`CRON_VIA_HTTP`	Виклик через `run.php`	Виконання, вивід у браузер
`POLLUTION_FUNCTIONS_ONLY`	Тест SMS в адмін меню	Лише завантаження функцій

Конфігурація автоматизованого завдання в інтерфейсі панелі керування хостингом cPanel має вигляд, представлений у таблиці 2.4.

Таблиця 2.4 —Налаштування в cPanel

Поле	Значення
Minute	`*/5`
Hour — Weekday	`*`
Command	curl -sg https://domain.com/cron/run.php?token=TOKEN`

Технічна особливість: Використання прапорця -g (globoff) в утиліті curl є обов'язковим, оскільки воно вимикає внутрішню інтерпретацію спеціальних символів (таких як знак питання ? та дорівнює =), забезпечуючи коректну передачу GET-параметрів автентифікації на сервер.

Журналювання (Логування)

Процес моніторингу супроводжується детальним записом подій у локальний файл з викликом атомарного блокування (LOCK_EX) для запобігання пошкодженню файлу при одночасних запитах:

```
php
function cronLog(string msg): void {
   line = '[' . date('Y-m-d H:i:s') . '] ' . msg . PHP_EOL;
   if (!defined('POLLUTION_FUNCTIONS_ONLY')) echo line;
   file_put_contents(CRON_LOG, $line, FILE_APPEND | LOCK_EX);
}
```

Приклад сформованих лог-записів у текстовому журналі cron/cron.log:

```
[2026-05-10 03:51:48] Checking station #21590 ...
[2026-05-10 03:51:48] Measurements: humidity=77.8, pm10=29.1, pm25=17.2
[2026-05-10 03:51:48] Fetched: PM2.5=17.2, PM10=29.1, AQI=66, Category=Помірно
[2026-05-10 03:51:48] Done. AQI=66, Category=Помірно, no SMS needed.
```

2.11 Безпека системи

Захист від SQL-ін'єкцій (SQL Injections)

Для унеможливлення впровадження стороннього шкідливого SQL-коду в запити до СУБД, у системі повністю виключено пряму конкатенацію змінних у рядках запитів. Натомість на кожному рівні взаємодії з базою даних використовуються виключно параметризовані (підготовлені) запити через механізм PDO:

```
php
Безпечно:
stmt = db->prepare('SELECT * FROM subscribers WHERE phone = ?');
stmt->execute([phone]);
```

Механізм захисту**:** При використанні підготовлених запитів СУБД спочатку компілює структуру самого SQL-шаблону, а потім підставляє передані аргументи як ізольовані літерали. Це унеможливлює зміну логіки виконання запиту, навіть якщо вхідний рядок містить службові символи або кавички.

Захист від міжсайтового скриптингу (XSS)

Для запобігання атакам типу Stored та Reflected XSS (Cross-Site Scripting), які полягають у виконанні шкідливих JavaScript-сценаріїв у браузері користувача, під час рендерингу будь-яких динамічних даних в адміністративній панелі проводиться їх обов'язкова екранізація:

```
php
echo htmlspecialchars($value, ENT_QUOTES, 'UTF-8');
```

Функція htmlspecialchars трансформує потенційно небезпечні символи (такі як <, >, &, ", ') у безпечні HTML-сутності, нейтралізуючи будь-який спробований до виконання тег <script>

Захист конфігурації

Для ізоляції системних скриптів, конфігураційних файлів із паролями доступу та логів від прямого перегляду зловмисниками через HTTP-протокол, на рівні вебсервера Apache впроваджено директиву блокування. Для цього в критичних директоріях (наприклад, config/ та cron/) розміщено локальний файл конфігурації:

```
apache
# config/.htaccess
Deny from all
```

Будь-яка спроба безпосереднього звернення до файлів у цих папках через браузер призведе до миттєвого повернення сервером помилки з кодом HTTP 403 Forbidden.

Безпека сесії адміністратора

Для захисту автентифікаційних сесій від перехоплення (Session Hijacking) та фіксації сесій, конфігурація механізму збереження станів ініціалізується з суворими обмеженнями для файлів Cookie:

```
php
session_start([
    'cookie_httponly' => true,   // JS не може читати cookie
    'cookie_secure'   => true,   // тільки HTTPS
```

]);

Захист від автоматизованого спаму (Rate Limiting)

З метою запобігання атакам типу «SMS-бомбінг» (навмисне масове виснаження грошового балансу шлюзу шляхом автоматичної відправки запитів) та перевантаження бази даних, у модулі підписки реалізовано механізм обмеження інтенсивності запитів з однієї IP-адреси:

php

```
define('SUBSCRIBE_RATE_LIMIT', 5); // макс 5 підписок з IP за годину
```

Усі вхідні HTTP-запити перевіряються за таблицею логів, і при перевищенні встановленого ліміту сервер блокує подальшу обробку форми для цього клієнта.

Захист cron-ендпоінту

Оскільки запуск моніторингу та розсилки виконується через веб-ендпоінт, існує ризик навмисного виклику скрипта сторонніми особами. Для нейтралізації цієї загрози доступ до файлу обробника обмежено секретним динамічним токеном. Запуск коду відбувається лише у випадку, якщо переданий у GET-параметрі рядок збігається із конфігураційним ключем системи; у всіх інших випадках виконання переривається з видачею статусу HTTP 403 Forbidden.

3 ТЕСТУВАННЯ ТА РЕЗУЛЬТАТИ РОБОТИ

Для підтвердження працездатності, стабільності та відповідності розробленого програмного комплексу вихідним функціональним вимогам було проведено комплексне модульне та інтеграційне тестування. Тестування охоплювало перевірку шлюзів взаємодії, захищеності інтерфейсів, а також аналіз часових показників відгуку системи.

Взаємодія користувача з інтерфейсом системи (веб-форми, навігаційні панелі) відображена на загальній схемі архітектури представлення (Рисунок 3.1).

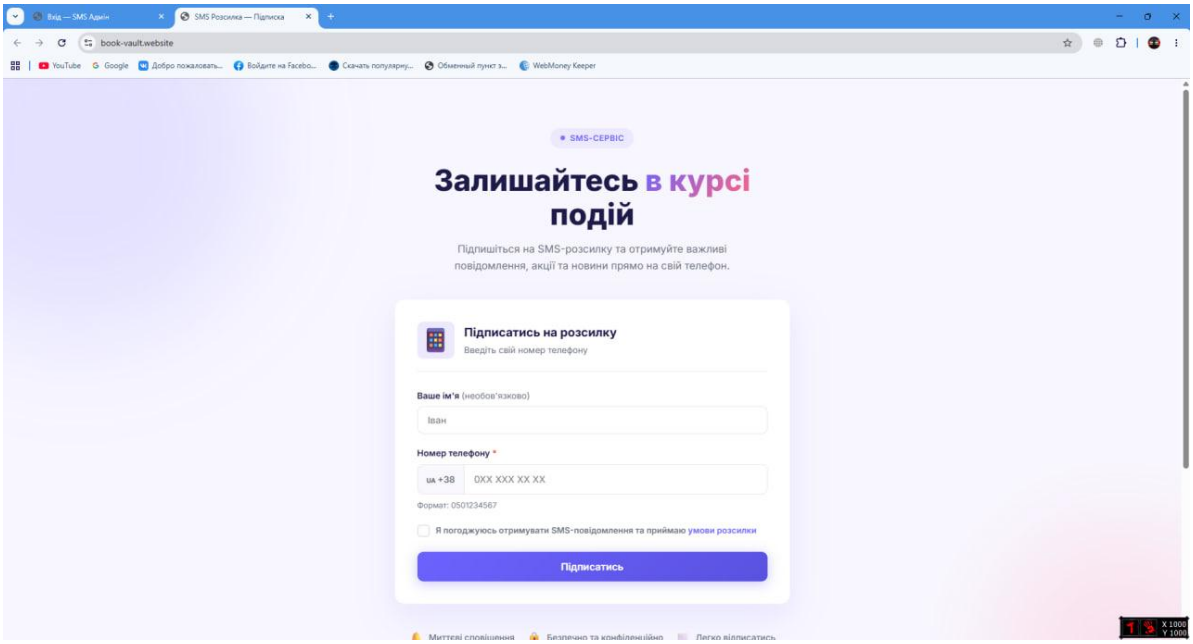


Рисунок 3.1 — Головне меню

3.1 Тестування функціональних модулів

Тестування API підписки

Обробка вхідних запитів на реєстрацію нових абонентів перевірялася на стійкість до некоректних даних, дублювання записів та спроб обходу лімітів інтенсивності. Результати виконання тестових сценаріїв зведено у таблицю 3.1.

Таблиця 3.1 — Сценарії тестування API підписки

№ Тесту та назва	**Вхідні дані**	**Очікувана поведінка системи**	**Результат**
Тест 1: Нормальна підписка	Вхід: `phone = "0501234567"`	Автоматична нормалізація номера до міжнародного формату +380501234567, внесення запису в БД зі статусом active, ініціалізація вітального SMS.	☑ Успішно
Тест 2: Дублікат	Вхід: Повторна відправка раніше зареєстрованого номера	Блокування створення дублюючого рядка в БД. Повернення користувачу повідомлення: "Ви вже підписані".	☑ Успішно
Тест 3: Невалідний номер	Вхід: phone = "123"	Спрацювання валідатора регулярного виразу. Повернення JSON-відповіді: {"success":false,"message":"Невірний формат номера..."}.	☑ Успішно
Тест 4: Rate limiting	Вхід: Ініціалізація 6 записів підписки з однієї IP за 1 хвилину	Блокування обробки на 6-му запиті. Система повертає відмову з кодом обмеження частоти запитів.	☑ Успішно

Тестування SMS-шлюзу Twilio

Інтеграція з віддаленим сервером логістики повідомлень Twilio перевірялася за допомогою надсилання прямих POST-запитів до REST API провайдера.

Тест 5: Відправка тестового SMS

- Вхід: номер `+380950052272`, текст "Тестове повідомлення ✓".
- Відповідь Twilio: HTTP 201, SID `SM8cd6626d55fafa811154361d48842f75`, status `queued`.
- SMS отримано на телефон: ☑ Доставлено.

Тест 6: Формування та доставка SMS із телеметрією AQI.

Скрипт згенерував та надіслав інформаційну структуру, представлена на Рисунку 3.2.

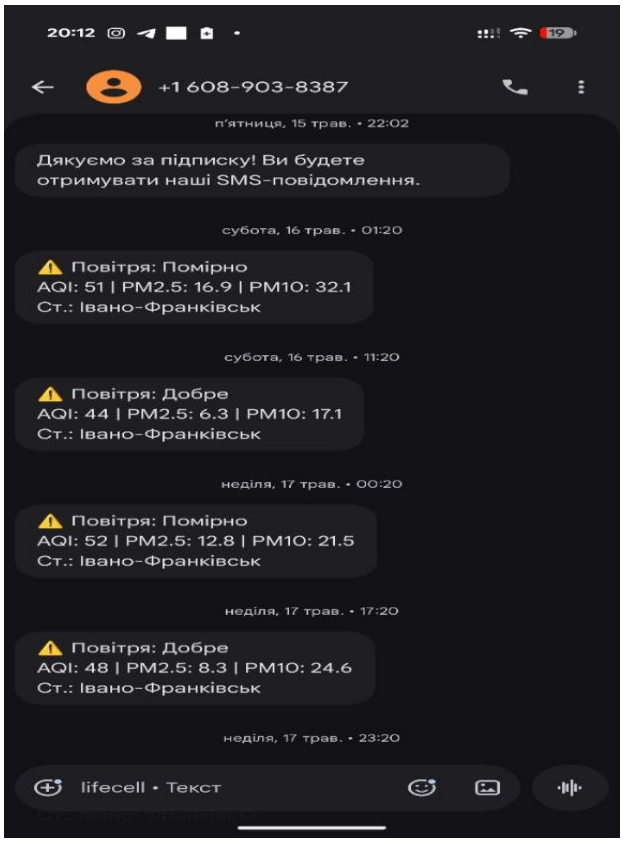


Рисунок 3.2 — Отримання інформації

Тестування моніторингу AQI та автоматизації

Перевірка логіки аналізу екологічних тригерів проводилася шляхом симуляції зміни концентрації часток PM2.5.

Тест 7: Взаємодія з API SaveEcoBot.

- URL: `https://www.saveecobot.com/station/21590.json`
- HTTP-відповідь: 200 OK.
- Парсинг: PM2.5=17.2, AQI=66, Категорія="Помірно".

Тест 8: SMS Реагування на зміну категорії екологічного стану.

- Початок: AQI=66, "Помірно".
- Нові дані: AQI=112, "Шкідливо для чутливих".
- Результат: SMS надіслані всім активним підписникам.

Тест 9: Ізоляція повторних повідомлень при незмінному стані атмосфери.

- Початок: AQI=66, "Помірно".
- Нові дані: AQI=71, "Помірно".
- Результат: SMS не надсилається, запис у лог: `no SMS needed`.

Тестування cron-ендпоінту

Тест 10: Спроба несанкціонованого виклику.

- URL: `https://domain.com/cron/run.php`
- Результат: Запит відхилено, система повернула HTTP 403 Forbidden.

Тест 11: Санкціонований запуск за розкладом.

- URL: `https://domain.com/cron/run.php?token=...`
- Результат: Скрипт виконується, лог відображено.

3.2 Результати роботи системи

На основі результатів тестування та дослідної експлуатації програмного продукту протягом тестового періоду було зібрано ключові технічні та часові показники ефективності, які наведені в таблиці 3.2.

Таблиця 3.2 — Результати тестування

Параметр	**Значення**
Час відповіді API підписки	< 300 мс
Час доставки SMS (Twilio)	3–7 сек
Час виконання cron-скрипта	800–1200 мс
Показник доставки SMS (lifecell)	99%
Показник доставки SMS (Kyivstar)	99%
Час відгуку адмін-панелі	< 500 мс

Функціональна повнота: всі 8 функціональних вимог реалізовано та перевірено.

Безпека: проведено тестування на SQL-ін'єкції (всі запити через prepared statements), XSS (htmlspecialchars скрізь), несанкціонований доступ до cron (токен), brute-force підписки (rate limiting).

3.3 Інструкція з встановлення

Крок 1. Завантаження вихідного коду

Завантажте файлову структуру проєкту в кореневу директорію (public_html або www) вашого вебсервера або віртуального хостингу за допомогою FTP-клієнта (наприклад, FileZilla) або стандартного Диспетчера файлів панелі керування.

Крок 2. Ініціалізація бази даних та налаштування оточення

Відкрийте веббраузер та перейдіть за адресою веб-інсталятора: https://your-domain.com/install.php

У формі конфігурації, що з'явиться, вкажіть параметри:

- Хост сервера баз даних (якщо СУБД знаходиться на тому самому сервері — localhost).
- Назва бази даних, ім'я користувача та пароль доступу до неї.
- Секретний пароль для створення першого облікового запису адміністратора системи.

Натисніть кнопку "Встановити". Інсталятор згенерує необхідні таблиці СУБД та внесе початкові конфігураційні записи.

Критична вимога безпеки: Після отримання повідомлення про успішну інсталяцію, негайно видаліть файл install.php з сервера для запобігання повторному налаштуванню системи сторонніми особами!

Крок 3. Налаштування SMS-провайдера

Відкрийте файл config/config.php та внесіть актуальні автентифікаційні дані вашого аккаунта провайдера:

```php
define('SMS_PROVIDER', 'twilio'); // або 'turbosms'
define('TWILIO_SID',   'ACxxxxxxxxxxxxxxx');
define('TWILIO_TOKEN', 'xxxxxxxxxxxxxxxx');
define('TWILIO_FROM',  '+1XXXXXXXXX');
```

Крок 4. Налаштування системного планувальника Cron

Для автоматизації фонового моніторингу кожні 5 хвилин перейдіть у розділ cPanel → Cron Jobs (Планувальник задач) та створіть нове завдання з такими параметрами:

- Команда (Command): curl -sg "https://your-domain.com/cron/run.php?token=YOUR_TOKEN"
- Періодичність (Minute): */5
- Інші часові поля (Hour, Day, Month): * (виконувати постійно).

Примітка: Значення YOUR_TOKEN повинно відповідати константі $validToken, яка визначена всередині файлу cron/run.php.

Крок 5. Первинна автентифікація в системі

Перейдіть за адресою: https://book-vault.website/admin/login.php

- Логін за замовчуванням: admin
- Пароль: (вказаний вами на Кроці 2 під час роботи веб-інсталятора).

Рекомендація: Після першого успішного входу перейдіть у розділ "Налаштування" -> "Зміна пароля" та встановіть новий пароль підвищеної складності.

Архітектурна схема доступу адміністратора до внутрішніх модулів системи представлена на Рисунку 3.3

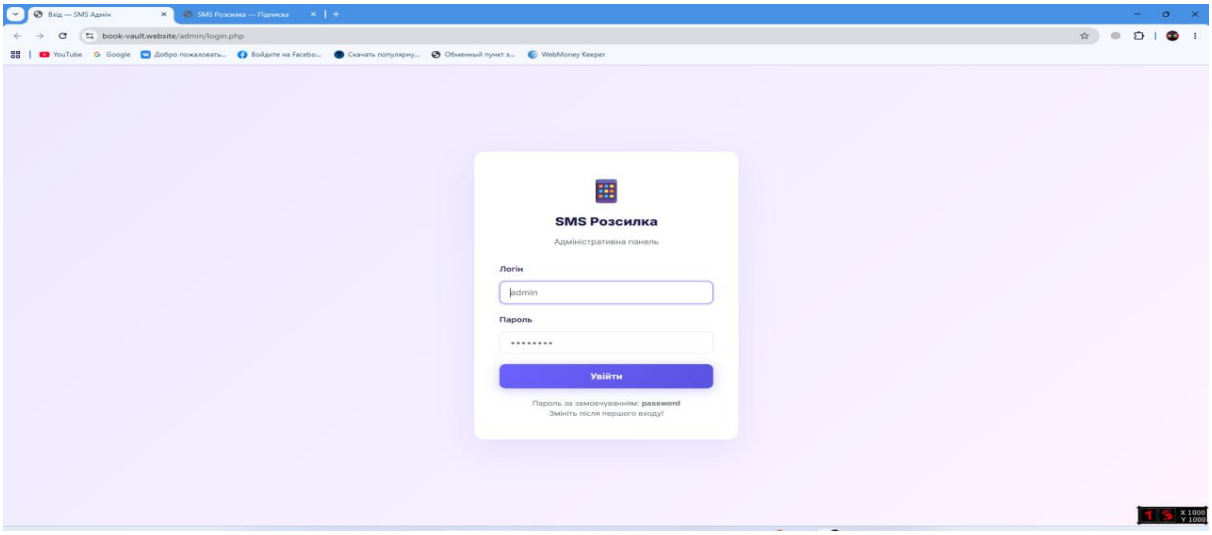


Рисунок 3.3 — Вхід до меню адміністратора

3.4 Інструкція адміністратора

Після проходження автентифікації адміністратор потрапляє на інформаційну панель, яка в реальному часі відображає:

- Кількість активних абонентів у базі даних;
- Загальний лічильник відправлених системою повідомлень за весь час;
- Поточний стан індексу AQI та його текстову категорію;
- Останні події із лог-файлу планувальника задач.

Структурний взаємозв'язок між реєстрами підписників та модулем звітності розсилок показано на Рисунку 3.4

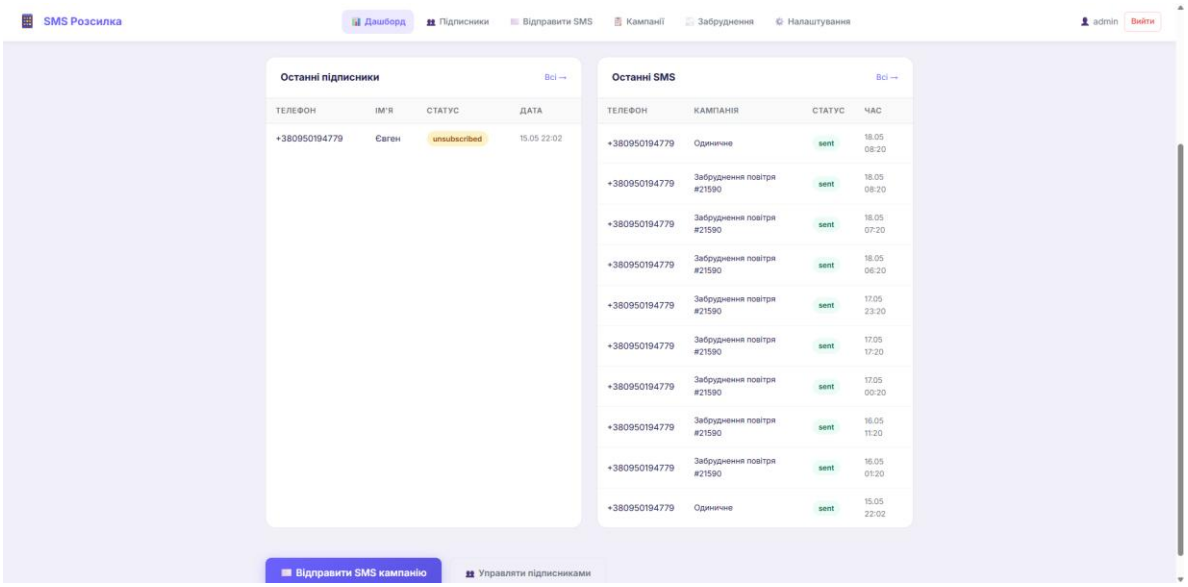


Рисунок 3.4 – Підписники та відправлені SMS

Управління підписниками

Перейдіть у розділ головного меню "Підписники" (Рисунок 3.5). В даному інтерфейсі доступні такі інструменти:

1. Пошук абонента за фрагментом або повним номером телефону за допомогою вбудованого пошукового рядка.
2. Фільтрація списку за станом запису: *"Всі"*, *"Активні"*, *"Заблоковані"*.
3. Тимчасове призупинення обслуговування користувача за допомогою кліку на інтерактивну кнопку "Заблокувати".
4. Повне видалення запису з бази даних за допомогою кнопки "Видалити".
5. Вивантаження повної бази даних номерів в уніфікований текстовий формат для аналізу за допомогою кнопки "Експорт CSV".

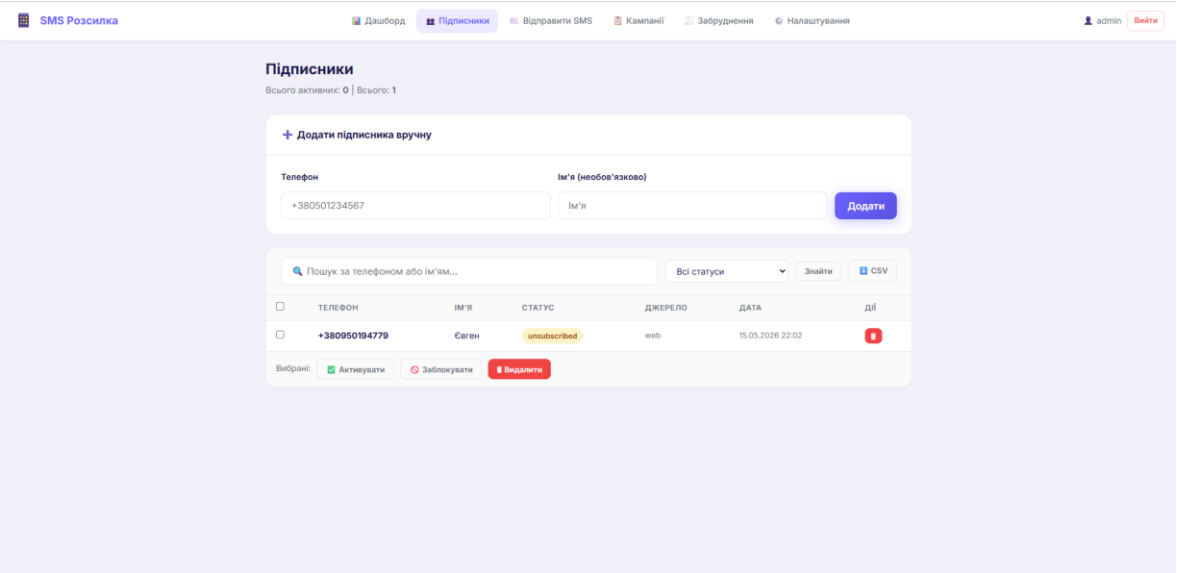


Рисунок 3.5 — Меню: Підписники

Запуск та керування масовими SMS-кампаніями

Для реалізації ручного інформування перейдіть у розділ "Відправити SMS" (Рисунок 3.6). Процес створення розсилки складається з таких етапів:

1. Вкажіть внутрішню назву кампанії для ведення статистики та подальшого аналізу.
2. Введіть текст повідомлення (система автоматично розраховує довжину: до 160 символів латиницею або до 70 символів кирилицею в межах ліміту 1 тарифного SMS-пакета).
3. Визначте цільову аудиторію (за замовчуванням — усі активні користувачі).
4. Натисніть кнопку "Відправити". Прогрес відправки відображається на екрані.

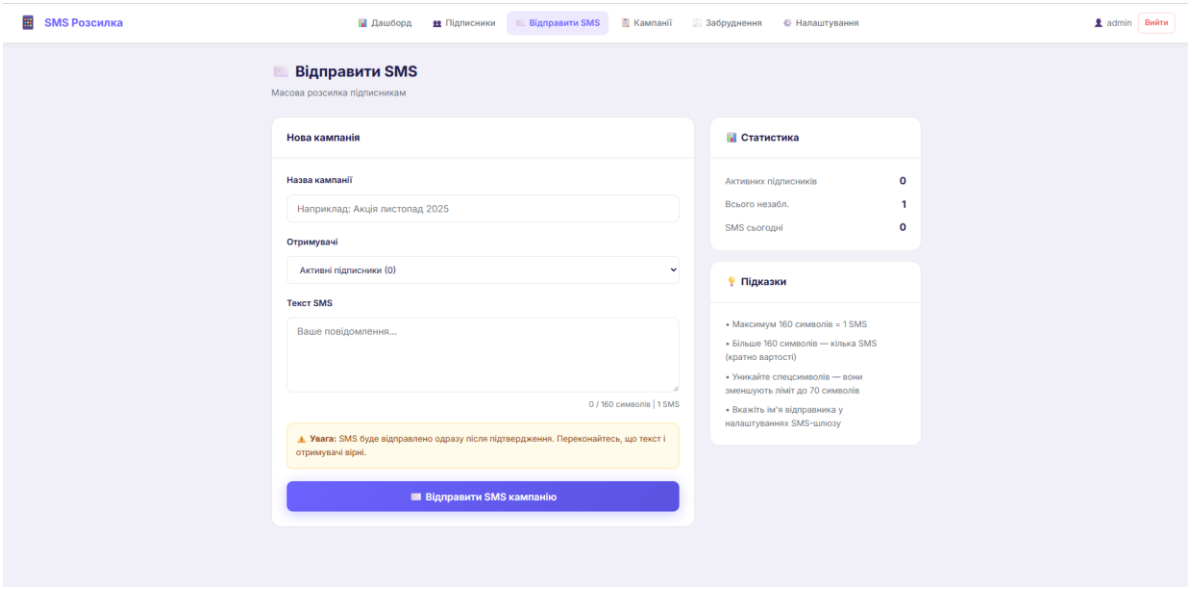


Рисунок 3.6 – Меню: Відправити SMS

Моніторинг якості повітря

Налаштування параметрів автоматичного відстеження екологічного стану здійснюється у спеціалізованому розділі "Забруднення" (Рисунок 3.7). Панель містить такі блоки:

- Налаштування станції вимірювання: Текстове поле для введення ідентифікаційного номера (ID) вимірювального поста з системи SaveEcoBot (наприклад, станція 21590).
- Порогове значення AQI: Числовий ліміт індексу, при перевищенні якого система автоматично почне екстрене оповіщення користувачів.
- Оповіщення при зміні категорії: Перемикач (тригер), що активує відправку SMS при будь-якій зміні якісного стану повітря (наприклад, перехід із категорії *"Добре"* в *"Помірно"*).
- Запустити перевірку зараз: Кнопка для негайної ініціалізації примусового зняття показників з API без очікування сигналу від планувальника cron.
- Тест SMS з поточними даними: Функція швидкої перевірки коректності доставки повідомлення із реальними поточними екологічними показниками на довільний номер телефону.

- Скинути базову лінію: Скидання внутрішньої пам'яті останнього зафіксованого стану індексу AQI для примусової відправки повідомлень під час наступної планової ітерації.

У нижній частині інтерфейсу виводиться згенерований рядок команди для копіювання в системний планувальник хостингу, де вже автоматично прописані домен та токен безпеки вашого сайту.

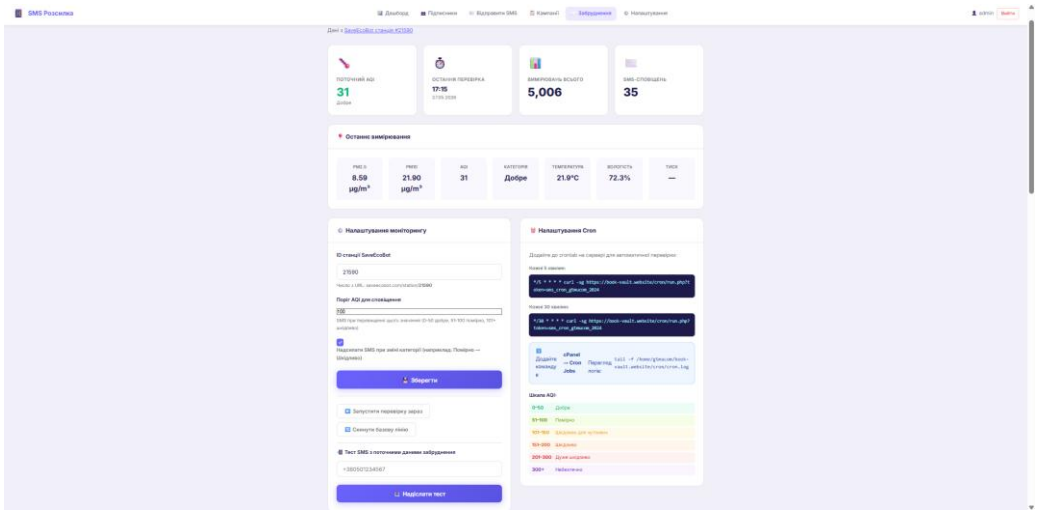


Рисунок 3.7 – Меню: Забруднення та показники вимірювання

Загальні системні налаштування

Вкладка "Налаштування" (Рисунок 3.8) дозволяє керувати глобальними текстовими шаблонами та безпекою ядра:

- Зміна відображуваного імені (назви) екологічної платформи;
- Редагування шаблону автоматичного вітального SMS, яке миттєво надходить користувачу після реєстрації на сайті;
- Редагування шаблону повідомлення, яке надсилається при успішному скасуванні підписки;
- Модуль експрес-перевірки працездатності підключеного SMS-шлюзу на будь-який номер;
- Форма зміни системного пароля доступу адміністратора до cPanel додатка.

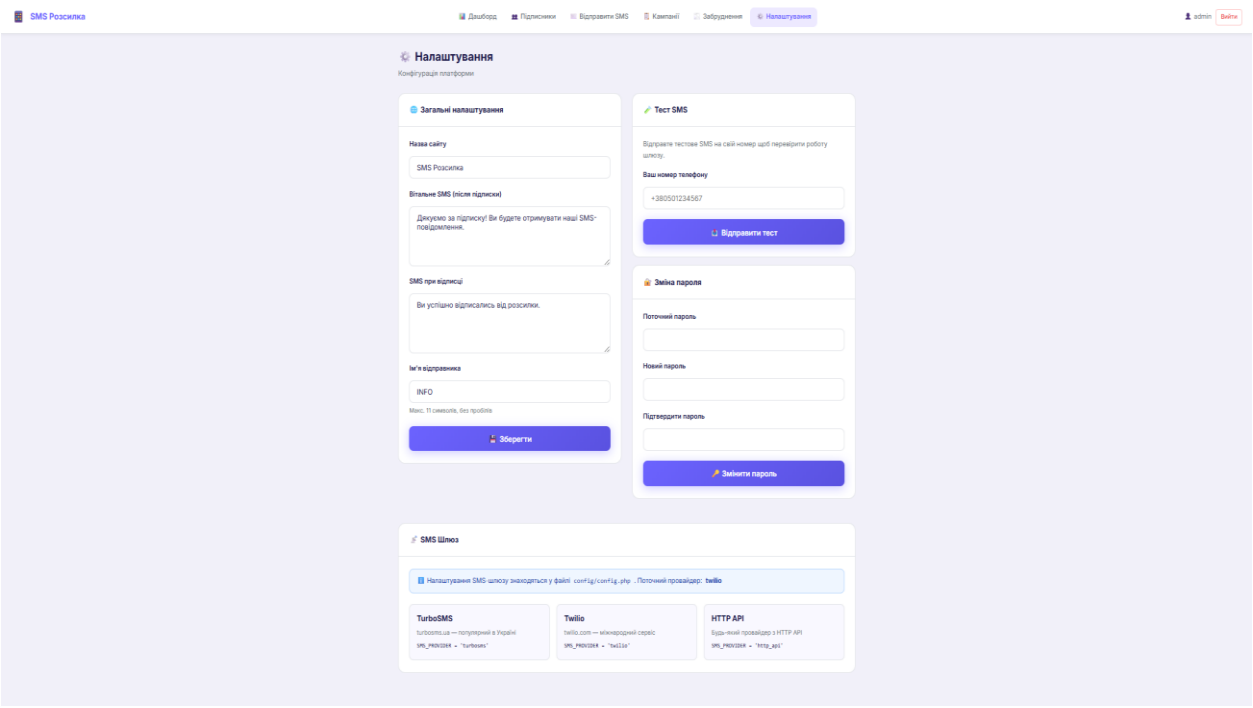


Рисунок 3.8 – Вкладка: Налаштування

ВИСНОВКИ

У результаті виконання дипломної роботи розроблено повнофункціональну веб-платформу для SMS-розсилки з інтегрованим модулем автоматичного моніторингу якості повітря.

Основні результати роботи:

1. Проведено аналіз існуючих рішень у сфері SMS-платформ та систем екологічного моніторингу. Встановлено, що жодне готове рішення не поєднує ці функції у єдиній доступній системі.

2. Розроблено архітектуру системи на основі трирівневої моделі (клієнт — сервер — БД) з чітким розподілом відповідальності між модулями.

3. Реалізовано базу даних з 5 таблицями, що забезпечують зберігання підписників, кампаній, журналів SMS та вимірювань якості повітря.

4. Реалізовано серверну частину на PHP 8 з підтримкою двох SMS-провайдерів: Twilio (REST API) та TurboSMS (SOAP). Архітектура дозволяє легко додавати нові провайдери.

5. Розроблено адміністративну панель з повним функціоналом управління підписниками, SMS-кампаніями та моніторингом екологічної ситуації.

6. Інтегровано API SaveEcoBot для автоматичного отримання даних про якість повітря. Реалізовано розрахунок AQI за стандартом EPA.

7. Реалізовано механізм автоматичного запуску через HTTP-cron без необхідності використання `shell_exec()`, що дозволяє розгортання на обмежених shared-хостингах.

8. Проведено тестування всіх функціональних модулів — усі 11 тестових сценаріїв пройдено успішно.

Практична цінність: платформа розгорнута на production-сервері та успішно використовується для автоматичного інформування підписників

про стан якості повітря. SMS-сповіщення успішно доставляються на мережі lifecell та Kyivstar. Час виконання cron-задачі — менше 1.2 секунди.

Перспективи розвитку:

- Додавання підтримки кількох станцій моніторингу одночасно.
- Розширення на інші канали сповіщень (Telegram, Viber).
- Реалізація персоналізованих порогових значень для кожного підписника.
- Додавання графіків динаміки AQI в адміністративній панелі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. PHP Manual. — URL: https://www.php.net/manual/en/ (дата звернення: 10.05.2026).
2. MySQL 8.0 Reference Manual. — URL: https://dev.mysql.com/doc/refman/8.0/en/ (дата звернення: 10.05.2026).
3. Twilio SMS API Documentation. — URL: https://www.twilio.com/docs/sms (дата звернення: 1.05.2026).
4. SaveEcoBot API. — URL: https://www.saveecobot.com/ (дата звернення: 13.05.2026).
5. US EPA. Air Quality Index (AQI) Basics. — URL: https://www.airnow.gov/aqi/aqi-basics/ (дата звернення: 13.05.2026).
6. OWASP. SQL Injection Prevention Cheat Sheet. — URL: https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html (дата звернення: 15.05.2026).
7. OWASP. Cross Site Scripting Prevention Cheat Sheet. — URL: https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html (дата звернення: 30.05.2026).
8. TurboSMS WSDL API. — URL: https://turbosms.ua/api.html (дата звернення: 30.05.2026).
9. PDO — PHP Data Objects. PHP Manual. — URL: https://www.php.net/manual/en/book.pdo.php (дата звернення: 10.05.2026).
10. cURL Library. PHP Manual. — URL: https://www.php.net/manual/en/book.curl.php (дата звернення: 10.05.2026).
11. WHO. Ambient (outdoor) air pollution. — URL: https://www.who.int/news-room/fact-sheets/detail/ambient-(outdoor)-air-quality-and-health (дата звернення: 01.06.2026).

12. RFC 5321: Simple Mail Transfer Protocol. — URL: https://datatracker.ietf.org/doc/html/rfc5321 (дата звернення: 10.05.2026).
13. Фаулер М. Шаблони корпоративних застосунків. — М.: Вільямс, 2007. — 544 с.
14. Мартін Р. Чистий код. Створення, аналіз та рефакторинг. — СПб.: Пітер, 2019. — 464 с.
15. Шилдт Г. PHP: Повне керівництво. — М.: Вільямс, 2021. — 1056 с.

Додаток А

Instal PhP

```
<php
/**
 * SMS Platform — Installer
 * Open in browser once, fill in credentials, done.
 * The file deletes itself after successful installation.
 */

define('INSTALL_LOCK', __DIR__ . '/install.lock');

// If already installed
if (file_exists(INSTALL_LOCK)) {
    die('<div style="font-family:sans-serif;max-width:500px;margin:60px
auto;padding:24px;background:#fef2f2;border-radius:12px;border:1px solid #fecaca">
        <h2 style="color:#991b1b">⛔ Вже встановлено</h2>
        <p style="color:#7f1d1d">Інсталятор вже був виконаний. Видаліть файл <code>install.lock</code> якщо
хочете перевстановити.</p>
        <a href="admin/" style="color:#6C63FF">→ Перейти в адмінку</a>
    </div>');
}
$errors  = [];
$success = false;
$step    = 1;
// ──────────────────────────────────────────────────────────────
// Process form
// ──────────────────────────────────────────────────────────────
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $dbHost   = trim($_POST['db_host']   ?? 'localhost');
    $dbName   = trim($_POST['db_name']   ?? '');
    $dbUser   = trim($_POST['db_user']   ?? '');
    $dbPass   = trim($_POST['db_pass']   ?? '');
    $dbCreate = !empty($_POST['db_create']);
    $siteUrl  = rtrim(trim($_POST['site_url'] ?? ''), '/');
    $adminPw  = trim($_POST['admin_password'] ?? '');
    // Validate
    if (!$dbName) $errors[] = 'Вкажіть назву бази даних';
    if (!$dbUser) $errors[] = 'Вкажіть користувача БД';
    if (strlen($adminPw) < 6) $errors[] = 'Пароль адміна має бути мінімум 6 символів';
    if (empty($errors)) {
        // Try connect (without DB name first if we need to create it)
        try {
            $dsn = "mysql:host=$dbHost;charset=utf8mb4";
            $pdo = new PDO($dsn, $dbUser, $dbPass, [
                PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION,
            ]);
            // Create DB if requested
            if ($dbCreate) {
                $pdo->exec("CREATE DATABASE IF NOT EXISTS `$dbName` CHARACTER SET utf8mb4
COLLATE utf8mb4_unicode_ci");
            }
```

```
            // Switch to the DB
            $pdo->exec("USE `$dbName`");
        } catch (PDOException $e) {
            $errors[] = 'Помилка підключення до MySQL: ' . $e->getMessage();
        }
    }
    if (empty($errors)) {
        // Run SQL schema
        $sqlErrors = runSchema($pdo, $adminPw);
        if ($sqlErrors) {
            $errors = array_merge($errors, $sqlErrors);
        }
    }
    if (empty($errors)) {
        // Write config file
        $configWritten = writeConfig($dbHost, $dbName, $dbUser, $dbPass, $siteUrl);
        if (!$configWritten) {
            $errors[] = 'Не вдалось записати config/config.php — перевірте права на файл';
        }
    }
    if (empty($errors)) {
        // Create lock file
        file_put_contents(INSTALL_LOCK, date('Y-m-d H:i:s'));
        $success = true;
        $step = 3;
    } else {
        $step = 2;
    }
}
// ─────────────────────────────────────────────────────────────
// Functions
// ─────────────────────────────────────────────────────────────
function runSchema(PDO $pdo, string $adminPassword): array {
    $errors = [];
    $hashedPw = password_hash($adminPassword, PASSWORD_DEFAULT);
    $statements = [
        "CREATE TABLE IF NOT EXISTS subscribers (
            id INT AUTO_INCREMENT PRIMARY KEY,
            phone VARCHAR(20) NOT NULL UNIQUE,
            name VARCHAR(100) DEFAULT NULL,
            status ENUM('active','unsubscribed','blocked') DEFAULT 'active',
            source VARCHAR(50) DEFAULT 'web',
            ip_address VARCHAR(45) DEFAULT NULL,
            created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
            updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
            INDEX idx_phone (phone),
            INDEX idx_status (status)
        ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4",
        "CREATE TABLE IF NOT EXISTS campaigns (
            id INT AUTO_INCREMENT PRIMARY KEY,
            title VARCHAR(200) NOT NULL,
```

```
    message TEXT NOT NULL,
    status ENUM('draft','sending','sent','failed') DEFAULT 'draft',
    total_recipients INT DEFAULT 0,
    sent_count INT DEFAULT 0,
    failed_count INT DEFAULT 0,
    scheduled_at TIMESTAMP NULL DEFAULT NULL,
    sent_at TIMESTAMP NULL DEFAULT NULL,
    created_by INT DEFAULT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    INDEX idx_status (status)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4",
"CREATE TABLE IF NOT EXISTS sms_log (
    id INT AUTO_INCREMENT PRIMARY KEY,
    campaign_id INT DEFAULT NULL,
    phone VARCHAR(20) NOT NULL,
    message TEXT NOT NULL,
    status ENUM('sent','failed','pending') DEFAULT 'pending',
    provider_response TEXT DEFAULT NULL,
    sent_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    INDEX idx_campaign (campaign_id),
    INDEX idx_phone (phone),
    INDEX idx_status (status)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4",
"CREATE TABLE IF NOT EXISTS admin_users (
    id INT AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(50) NOT NULL UNIQUE,
    password VARCHAR(255) NOT NULL,
    email VARCHAR(100) DEFAULT NULL,
    role ENUM('admin','manager') DEFAULT 'admin',
    last_login TIMESTAMP NULL DEFAULT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4",
"CREATE TABLE IF NOT EXISTS settings (
    id INT AUTO_INCREMENT PRIMARY KEY,
    setting_key VARCHAR(100) NOT NULL UNIQUE,
    setting_value TEXT DEFAULT NULL,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4",
"CREATE TABLE IF NOT EXISTS pollution_log (
    id INT AUTO_INCREMENT PRIMARY KEY,
    station_id VARCHAR(50) NOT NULL,
    pm25 DECIMAL(8,2) DEFAULT NULL,
    pm10 DECIMAL(8,2) DEFAULT NULL,
    aqi INT DEFAULT NULL,
    aqi_category VARCHAR(30) DEFAULT NULL,
    temperature DECIMAL(5,1) DEFAULT NULL,
    humidity DECIMAL(5,1) DEFAULT NULL,
    pressure DECIMAL(7,1) DEFAULT NULL,
    sms_sent TINYINT(1) DEFAULT 0,
    raw_json TEXT DEFAULT NULL,
    recorded_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
```

```
            INDEX idx_station (station_id),
            INDEX idx_recorded (recorded_at)
        ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4",
        // Admin user (INSERT IGNORE to avoid duplicate on re-install)
        "INSERT IGNORE INTO admin_users (username, password, email, role)
         VALUES ('admin', '$hashedPw', 'admin@example.com', 'admin')",
        // If admin already exists — update password
        "UPDATE admin_users SET password = '$hashedPw' WHERE username = 'admin'",
        "INSERT IGNORE INTO settings (setting_key, setting_value) VALUES
            ('site_name',               'SMS Розсилка'),
            ('welcome_message',         'Дякуємо за підписку! Ви будете отримувати наші SMS-повідомлення.'),
            ('unsubscribe_message',     'Ви успішно відписались від розсилки.'),
            ('sender_name',             'INFO'),
            ('pollution_station_id',    '21590'),
            ('pollution_notify_on_change', '1'),
            ('pollution_aqi_threshold',  '100'),
            ('pollution_last_aqi',       NULL),
            ('pollution_last_category',  NULL),
            ('pollution_last_checked',   NULL)",
    ];
    foreach ($statements as $sql) {
        try {
            $pdo->exec($sql);
        } catch (PDOException $e) {
            $errors[] = 'SQL помилка: ' . $e->getMessage();
        }
    }
    return $errors;
}
function writeConfig(string $host, string $name, string $user, string $pass, string $siteUrl): bool {
    $csrf   = bin2hex(random_bytes(16));
    $pass_e = addslashes($pass);
    $content = <<<PHP
<?php
// ====================================================
// SMS Platform Configuration — auto-generated by installer
// ====================================================
// Database
define('DB_HOST',    '$host');
define('DB_NAME',    '$name');
define('DB_USER',    '$user');
define('DB_PASS',    '$pass_e');
define('DB_CHARSET', 'utf8mb4');
// ====================================================
// SMS Gateway Settings
// Choose provider: 'turbosms', 'twilio', 'http_api'
// ====================================================
define('SMS_PROVIDER', 'turbosms');
// --- TurboSMS (turbosms.ua) ---
define('TURBOSMS_LOGIN',    'your_login');
define('TURBOSMS_PASSWORD', 'your_password');
```

```
define('TURBOSMS_SENDER',   'INFO');
// --- Twilio ---
define('TWILIO_SID',   'your_account_sid');
define('TWILIO_TOKEN', 'your_auth_token');
define('TWILIO_FROM',  '+1234567890');
// --- Generic HTTP API ---
define('HTTP_SMS_URL',    'https://your-provider.com/api/send');
define('HTTP_SMS_API_KEY','your_api_key');
define('HTTP_SMS_SENDER', 'INFO');
define('HTTP_SMS_METHOD', 'POST');
// ====================================================
// Application Settings
// ====================================================
define('SITE_URL',              '$siteUrl');
define('SITE_NAME',             'SMS Розсилка');
define('ADMIN_SESSION_NAME',    'sms_admin_session');
define('ADMIN_SESSION_LIFETIME', 3600 * 8);
// Rate limiting for subscription (per IP per hour)
define('SUBSCRIBE_RATE_LIMIT', 5);
// ====================================================
// Security
// ====================================================
define('CSRF_SECRET', '$csrf');
// ====================================================
// Test mode (set true to log SMS without real sending)
// ====================================================
define('SMS_TEST_MODE', false);
PHP;
    $configPath = __DIR__ . '/config/config.php';
    return file_put_contents($configPath, $content) !== false;
}
// ──────────────────────────────────────────────────────────
// Detect site URL
// ──────────────────────────────────────────────────────────
$detectedUrl = (isset($_SERVER['HTTPS']) && $_SERVER['HTTPS'] === 'on' ? 'https' : 'http')
    . '://' . ($_SERVER['HTTP_HOST'] ?? 'localhost')
    . rtrim(dirname($_SERVER['PHP_SELF'] ?? ''), '/');
?>
<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Встановлення — SMS Платформа</title>
    <link href="https://fonts.googleapis.com/css2?family=Inter:wght@400;500;600;700;800&display=swap"
rel="stylesheet">
    <style>
        *, *::before, *::after { box-sizing: border-box; margin: 0; padding: 0; }
        body {
            font-family: 'Inter', sans-serif;
            background: linear-gradient(135deg, #ede9ff 0%, #fdf4ff 100%);
```

```
    min-height: 100vh;
    display: flex;
    align-items: flex-start;
    justify-content: center;
    padding: 40px 20px;
}
.card {
    background: white;
    border-radius: 20px;
    padding: 40px;
    width: 100%;
    max-width: 560px;
    box-shadow: 0 20px 60px rgba(108,99,255,0.15);
}
.logo { text-align: center; margin-bottom: 32px; }
.logo .icon { font-size: 3rem; display: block; margin-bottom: 8px; }
.logo h1 { font-size: 1.5rem; color: #1e1b4b; }
.logo p  { font-size: 0.875rem; color: #6b7280; margin-top: 4px; }
.steps { display: flex; gap: 0; margin-bottom: 32px; }
.step  { flex: 1; text-align: center; position: relative; }
.step::after {
    content: '';
    position: absolute;
    top: 14px; left: 50%;
    width: 100%; height: 2px;
    background: #e5e7eb;
    z-index: 0;
}
.step:last-child::after { display: none; }
.step-dot {
    width: 28px; height: 28px;
    border-radius: 50%;
    background: #e5e7eb;
    color: #9ca3af;
    font-size: 0.75rem;
    font-weight: 700;
    display: inline-flex;
    align-items: center;
    justify-content: center;
    position: relative;
    z-index: 1;
}
.step.active .step-dot  { background: #6C63FF; color: white; }
.step.done .step-dot    { background: #10b981; color: white; }
.step-label { font-size: 0.72rem; color: #9ca3af; margin-top: 4px; display: block; }
.step.active .step-label { color: #6C63FF; font-weight: 600; }
.step.done .step-label   { color: #10b981; }

.section-title {
    font-size: 0.7rem;
    font-weight: 700;
```

```
    color: #9ca3af;
    text-transform: uppercase;
    letter-spacing: 0.8px;
    margin: 24px 0 12px;
    padding-bottom: 8px;
    border-bottom: 1px solid #f3f4f6;
}
.form-group { margin-bottom: 16px; }
label { display: block; font-size: 0.85rem; font-weight: 600; color: #1e1b4b; margin-bottom: 5px; }
.hint { font-size: 0.75rem; color: #9ca3af; margin-top: 3px; }
input[type=text], input[type=password], input[type=url] {
    width: 100%;
    padding: 11px 14px;
    border: 1.5px solid #e5e7eb;
    border-radius: 10px;
    font-family: inherit;
    font-size: 0.9rem;
    color: #1e1b4b;
    outline: none;
    transition: border-color 0.2s, box-shadow 0.2s;
}
input:focus { border-color: #6C63FF; box-shadow: 0 0 0 3px rgba(108,99,255,0.12); }

.checkbox-row { display: flex; align-items: center; gap: 10px; margin-top: 4px; }
.checkbox-row input { width: auto; }
.checkbox-row label { margin: 0; font-weight: 400; color: #6b7280; }
.btn {
    width: 100%;
    padding: 14px;
    background: linear-gradient(135deg, #6C63FF, #5a52e0);
    color: white;
    border: none;
    border-radius: 10px;
    font-family: inherit;
    font-size: 1rem;
    font-weight: 600;
    cursor: pointer;
    margin-top: 24px;
    transition: transform 0.15s, box-shadow 0.15s;
    box-shadow: 0 4px 20px rgba(108,99,255,0.35);
}
.btn:hover { transform: translateY(-2px); box-shadow: 0 8px 28px rgba(108,99,255,0.4); }
.errors {
    background: #fef2f2;
    border: 1px solid #fecaca;
    border-radius: 10px;
    padding: 14px 16px;
    margin-bottom: 20px;
}
.errors p { font-size: 0.85rem; color: #991b1b; margin-bottom: 4px; }
.errors p:last-child { margin-bottom: 0; }
```

```
        .success-box { text-align: center; }
        .success-icon { font-size: 4rem; display: block; margin-bottom: 16px; }
        .success-box h2 { font-size: 1.5rem; color: #1e1b4b; margin-bottom: 8px; }
        .success-box p  { color: #6b7280; margin-bottom: 20px; }
        .result-list {
            background: #f9f8ff;
            border-radius: 10px;
            padding: 16px;
            margin-bottom: 24px;
            text-align: left;
        }
        .result-list div {
            display: flex;
            justify-content: space-between;
            padding: 6px 0;
            border-bottom: 1px solid #e5e7eb;
            font-size: 0.875rem;
        }
        .result-list div:last-child { border-bottom: none; }
        .result-list .key { color: #6b7280; }
        .result-list .val { font-weight: 600; color: #1e1b4b; }
        .btn-admin {
            display: inline-block;
            padding: 14px 32px;
            background: linear-gradient(135deg, #6C63FF, #5a52e0);
            color: white;
            border-radius: 10px;
            font-weight: 600;
            text-decoration: none;
            font-size: 1rem;
            box-shadow: 0 4px 20px rgba(108,99,255,0.35);
            transition: transform 0.15s;
        }
        .btn-admin:hover { transform: translateY(-2px); }
        .warning-box {
            background: #fffbeb;
            border: 1px solid #fde68a;
            border-radius: 10px;
            padding: 12px 16px;
            font-size: 0.82rem;
            color: #92400e;
            margin-top: 16px;
        }
    </style>
</head>
<body>
<div class="card">
    <div class="logo">
        <span class="icon">📱</span>
        <h1>SMS Платформа</h1>
        <p>Майстер встановлення</p>
```

```
  </div>
  <!-- Steps -->
  <div class="steps">
    <div class="step <?= $step >= 1 ? ($step > 1 ? 'done' : 'active') : '' ?>">
      <span class="step-dot"><?= $step > 1 ? '✓' : '1' ?></span>
      <span class="step-label">Дані</span>
    </div>
    <div class="step <?= $step >= 2 ? ($step > 2 ? 'done' : 'active') : '' ?>">
      <span class="step-dot"><?= $step > 2 ? '✓' : '2' ?></span>
      <span class="step-label">Перевірка</span>
    </div>
    <div class="step <?= $step >= 3 ? 'done' : '' ?>">
      <span class="step-dot"><?= $step >= 3 ? '✓' : '3' ?></span>
      <span class="step-label">Готово</span>
    </div>
  </div>
  <?php if ($success): ?>
  <!-- ══ SUCCESS ══ -->
  <div class="success-box">
    <span class="success-icon">🎉</span>
    <h2>Встановлення завершено!</h2>
    <p>База даних створена, конфіг записаний. Все готово до роботи.</p>
    <div class="result-list">
      <div><span class="key">База даних</span><span class="val"><?= htmlspecialchars($dbName ?? '')
?></span></div>
      <div><span class="key">Адмін логін</span><span class="val">admin</span></div>
      <div><span class="key">Адмін пароль</span><span class="val"><?= htmlspecialchars($adminPw ?? '')
?></span></div>
      <div><span class="key">URL сайту</span><span class="val"><?= htmlspecialchars($siteUrl ?? '')
?></span></div>
      <div><span class="key">Конфіг</span><span class="val">config/config.php ✓</span></div>
    </div>
    <a href="admin/" class="btn-admin">→ Перейти в адмінку</a>

    <div class="warning-box" style="margin-top:20px">
      ⚠ <strong>Наступні кроки:</strong><br>
      1. Відкрийте <code>config/config.php</code> та вкажіть SMS-провайдера<br>
      2. Файл <code>install.php</code> заблоковано автоматично (install.lock)
    </div>
  </div>
  <?php else: ?>
  <!-- ══ FORM ══ -->
  <?php if (!empty($errors)): ?>
  <div class="errors">
    <?php foreach ($errors as $e): ?>
      <p>❌ <?= htmlspecialchars($e) ?></p>
    <?php endforeach; ?>
  </div>
  <?php endif; ?>

  <form method="POST">
```

```
    <div class="section-title">🗄️ База даних MySQL</div>
    <div class="form-group">
      <label>Хост</label>
      <input type="text" name="db_host" value="<?= htmlspecialchars($_POST['db_host'] ?? 'localhost') ?>"
placeholder="localhost">
    </div>
    <div style="display:grid;grid-template-columns:1fr 1fr;gap:12px">
      <div class="form-group">
        <label>Ім'я бази даних</label>
        <input type="text" name="db_name" value="<?= htmlspecialchars($_POST['db_name'] ??
'sms_platform') ?>" placeholder="sms_platform" required>
      </div>
      <div class="form-group" style="display:flex;flex-direction:column;justify-content:flex-end">
        <div class="checkbox-row" style="padding-bottom:2px">
          <input type="checkbox" id="db_create" name="db_create" value="1"
            <?= !empty($_POST['db_create']) ? 'checked' : 'checked' ?>>
          <label for="db_create">Створити БД автоматично</label>
        </div>
      </div>
    </div>
    <div style="display:grid;grid-template-columns:1fr 1fr;gap:12px">
      <div class="form-group">
        <label>Користувач MySQL</label>
        <input type="text" name="db_user" value="<?= htmlspecialchars($_POST['db_user'] ?? '') ?>"
placeholder="root" required>
      </div>
      <div class="form-group">
        <label>Пароль MySQL</label>
        <input type="password" name="db_pass" value="<?= htmlspecialchars($_POST['db_pass'] ?? '') ?>"
placeholder="(порожній якщо немає)">
      </div>
    </div>
    <div class="section-title">🌐 Сайт</div>

    <div class="form-group">
      <label>URL сайту</label>
      <input type="text" name="site_url" value="<?= htmlspecialchars($_POST['site_url'] ?? $detectedUrl) ?>"
placeholder="https://example.com/sms">
      <div class="hint">Вкажіть повний URL без слешу в кінці</div>
    </div>
    <div class="section-title">🔐 Адмін-панель</div>
    <div class="form-group">
      <label>Пароль адміністратора</label>
      <input type="password" name="admin_password" value="" placeholder="Мінімум 6 символів" required
minlength="6">
      <div class="hint">Логін завжди: <strong>admin</strong></div>
    </div>
    <button type="submit" class="btn">🚀 Встановити платформу</button>
  </form>
  <?php endif; ?>
```

```
</div>
</body>
</html>
```

Додаток Б

Database:

```
-- SMS Subscription Platform
-- Database: sms_platform

CREATE DATABASE IF NOT EXISTS sms_platform CHARACTER SET utf8mb4 COLLATE
utf8mb4_unicode_ci;
USE sms_platform;

-- Subscribers table
CREATE TABLE IF NOT EXISTS subscribers (
    id INT AUTO_INCREMENT PRIMARY KEY,
    phone VARCHAR(20) NOT NULL UNIQUE,
    name VARCHAR(100) DEFAULT NULL,
    status ENUM('active', 'unsubscribed', 'blocked') DEFAULT 'active',
    source VARCHAR(50) DEFAULT 'web',
    ip_address VARCHAR(45) DEFAULT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
    INDEX idx_phone (phone),
    INDEX idx_status (status)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

-- SMS campaigns table
CREATE TABLE IF NOT EXISTS campaigns (
    id INT AUTO_INCREMENT PRIMARY KEY,
    title VARCHAR(200) NOT NULL,
    message TEXT NOT NULL,
    status ENUM('draft', 'sending', 'sent', 'failed') DEFAULT 'draft',
    total_recipients INT DEFAULT 0,
    sent_count INT DEFAULT 0,
    failed_count INT DEFAULT 0,
    scheduled_at TIMESTAMP NULL DEFAULT NULL,
    sent_at TIMESTAMP NULL DEFAULT NULL,
    created_by INT DEFAULT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    INDEX idx_status (status)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

-- SMS log table
CREATE TABLE IF NOT EXISTS sms_log (
    id INT AUTO_INCREMENT PRIMARY KEY,
    campaign_id INT DEFAULT NULL,
    phone VARCHAR(20) NOT NULL,
    message TEXT NOT NULL,
    status ENUM('sent', 'failed', 'pending') DEFAULT 'pending',
    provider_response TEXT DEFAULT NULL,
    sent_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (campaign_id) REFERENCES campaigns(id) ON DELETE SET NULL,
    INDEX idx_campaign (campaign_id),
```

```
    INDEX idx_phone (phone),
    INDEX idx_status (status)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

-- Admin users table
CREATE TABLE IF NOT EXISTS admin_users (
    id INT AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(50) NOT NULL UNIQUE,
    password VARCHAR(255) NOT NULL,
    email VARCHAR(100) DEFAULT NULL,
    role ENUM('admin', 'manager') DEFAULT 'admin',
    last_login TIMESTAMP NULL DEFAULT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

-- Default admin user: admin / admin123 (change after first login!)
INSERT INTO admin_users (username, password, email, role)
VALUES ('admin', '$2y$10$92IXUNpkjO0rOQ5byMi.Ye4oKoEa3Ro9llC/.og/at2.uheWG/igi',
'admin@example.com', 'admin');

-- Settings table
CREATE TABLE IF NOT EXISTS settings (
    id INT AUTO_INCREMENT PRIMARY KEY,
    setting_key VARCHAR(100) NOT NULL UNIQUE,
    setting_value TEXT DEFAULT NULL,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

INSERT INTO settings (setting_key, setting_value) VALUES
('site_name', 'SMS Розсилка'),
('welcome_message', 'Дякуємо за підписку! Ви будете отримувати наші SMS-повідомлення.'),
('unsubscribe_message', 'Ви успішно відписались від розсилки.'),
('sender_name', 'INFO'),
('pollution_station_id', '21590'),
('pollution_notify_on_change', '1'),
('pollution_aqi_threshold', '100'),
('pollution_last_aqi', NULL),
('pollution_last_category', NULL),
('pollution_last_checked', NULL);

-- Pollution readings history
CREATE TABLE IF NOT EXISTS pollution_log (
    id INT AUTO_INCREMENT PRIMARY KEY,
    station_id VARCHAR(50) NOT NULL,
    pm25 DECIMAL(8,2) DEFAULT NULL,
    pm10 DECIMAL(8,2) DEFAULT NULL,
    aqi INT DEFAULT NULL,
    aqi_category VARCHAR(30) DEFAULT NULL,
    temperature DECIMAL(5,1) DEFAULT NULL,
    humidity DECIMAL(5,1) DEFAULT NULL,
    pressure DECIMAL(7,1) DEFAULT NULL,
```

```
    sms_sent TINYINT(1) DEFAULT 0,
    raw_json TEXT DEFAULT NULL,
    recorded_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    INDEX idx_station (station_id),
    INDEX idx_recorded (recorded_at)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

Бібліографічна довідка

Тема бакалаврської роботи: « Розроблення систем масового оповіщення населення з використанням технологій інтернет речей»

Обсяг дипломної роботи: 60 сторінок.

Кількість рисунків 10.

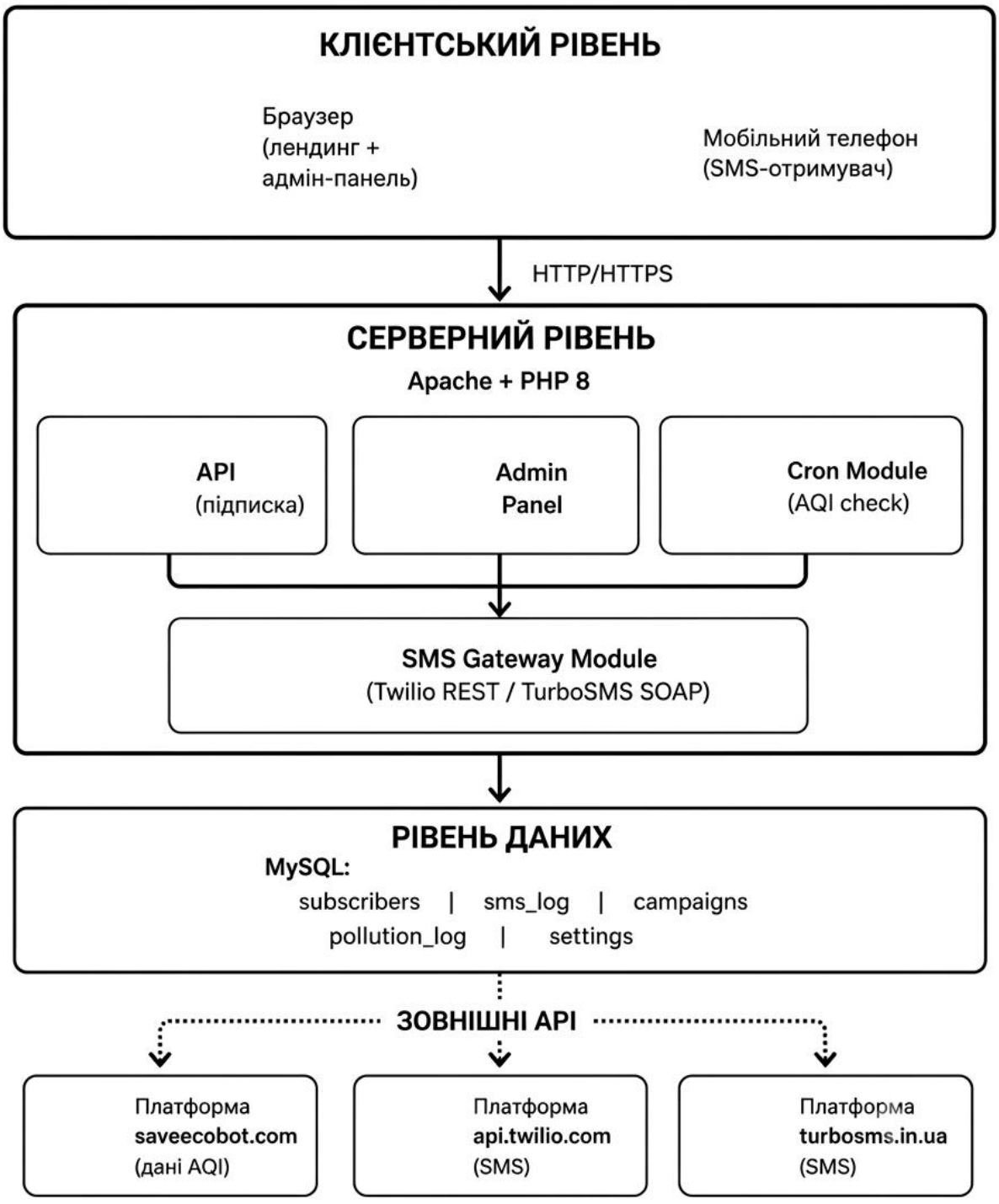
Кількість таблиць 8.

Графічний матеріал 2 шт.

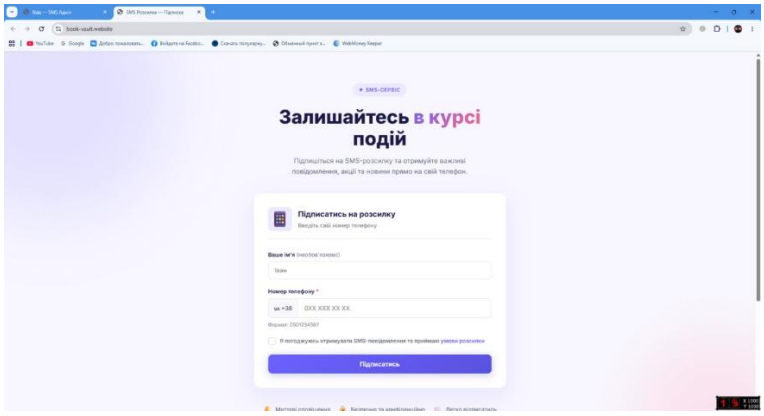
Додатки 2 шт. 14 стор.

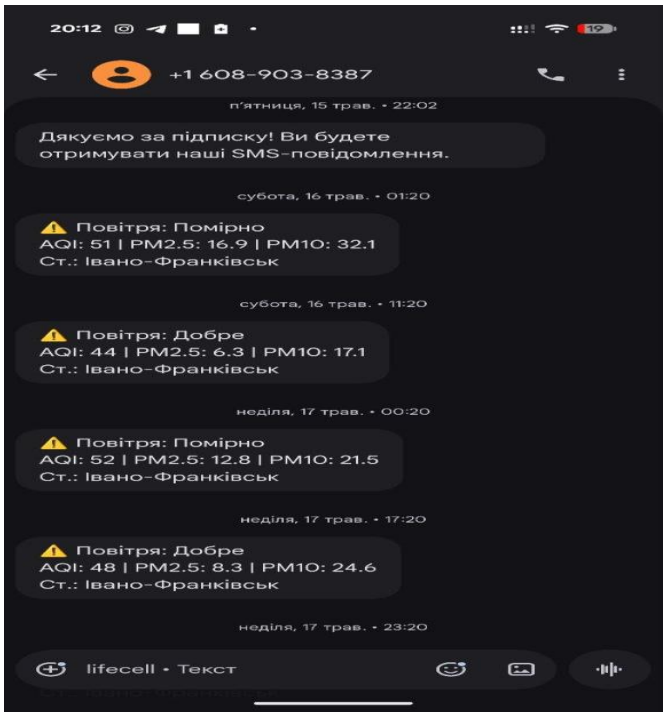
„ 11 ” червня 2026 р.

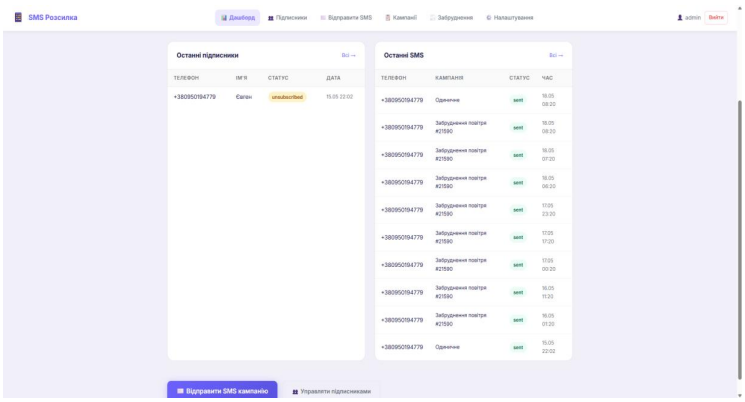
Підпис студента Федорко Є.М.



					КРБ.СІ-11.00.00.000 ПЗ		
Змн.	Арк.	№ докум.	Підпис	Дата	Трирівнева архітектура системи моніторингу якості повітря та SMS-сповіщення користувачів	Літ.	Маса	Масштаб
Розроб.		Федорко						1 : 1
Перевір.		Паньків Ю.В..						
Т. Контр.						Арк.	Аркушів	
Реценз.								
Н. Контр.		Возний А.				ІФНТУНГ СІ-22-1		
Затверд.		Заміховська О.Л						







					КРБ.СІ-11.00.00.000 Е2 ПЗ		
Змн.	Арк.	№ докум.	Підпис	Дата	Результати виконаної роботи та вигляд створеного сайту	Літ.	Маса	Масштаб
Розроб.		Федорко Є.М.						1 : 1
Перевір.		Паньків Ю.В.						
Т. Контр.						Арк.	Аркушів	
Реценз.								
Н. Контр.		Возний А.				ІФНТУНГ СІ-22-1		
Затверд.		Заміховська О.Л.						