Хмельницький національний університет
Факультет інформаційних технологій
Кафедра комп'ютерної інженерії та інформаційних систем

КВАЛІФІКАЦІЙНА РОБОТА

Інформаційна система збору та аналізу даних користувацької активності веб-сервісу
Назва теми

Рівень вищої освіти *перший (бакалаврський)*

Галузь знань *12 «Інформаційні технології»*
Шифр, назва

Спеціальність *126 «Інформаційні системи та технології»*
Шифр, назва

Освітня програма *«Інформаційні системи та технології»*
Назва

Шифр КвРІСТ 220184.22.01.13 ПЗ

Виконав здобувач IV курсу, група ІСТ-22-1 _____ Назарій ЯСТРЕМСЬКИЙ
Підпис Ініціали, прізвище

Керівник ДФ, доцент _____ Ольга ПАВЛОВА
Науковий ступінь, учене звання Підпис Ініціали, прізвище

Нормоконтролер канд.фіз - мат.наук., доц. _____ Тетяна КИСІЛЬ
Науковий ступінь, учене звання Підпис Ініціали, прізвище

До захисту допускаю:
завідувач кафедри КІС _____ Ольга ПАВЛОВА
«01» червня 2026 р. підпис Ініціали, прізвище
дата

Хмельницький 2026

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет Інформаційних технологій

Кафедра Комп'ютерної інженерії та інформаційних систем

Рівень вищої освіти перший (бакалаврський)

Галузь знань 12 Інформаційні технології

Спеціальність 126 Інформаційні системи та технології

Освітня програма «Інформаційні системи та технології»

ЗАТВЕРДЖУЮ
Завідувачка кафедри КІІС
_____ Ольга ПАВЛОВА

" 10 " 01 2026 р.

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

Ястремському Назарію Костянтиновичу
Прізвище, ім'я, по батькові студента

1. Тема проекту (роботи) Інформаційна система збору та аналізу даних користувацької активності веб-сервісу
Керівник проекту (роботи) Павлова Ольга Олександрівна, доктор філософії, доцент
Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

Затверджена наказом ректора університету від 20.01.2026 р. № 7

2. Термін подання здобувачем роботи на кафедру 01.06.2026 р.

3. Вихідні дані до роботи Завдання на кваліфікаційну роботу

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

Теоритичні основи побудови інформаційної системи збору та аналізу даних користувацької активності веб-сервісу

Проєктування інформаційної системи збору та аналізу користувацької активності веб-сервісу

Програмно-апаратна реалізація та тестування програмно-технічного засобу

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень)

Скріншот головної сторінки

ER-діаграма

Діаграма прецедентів Use Case

6. Консультанти розділів кваліфікаційної роботи

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

7. Дата видачі завдання «10» 01 2026 р.

КАЛЕНДАРНИЙ ПЛАН

№з/п	Назва етапів (розділів) дипломного проекту (роботи)	Термін виконання етапів проекту (роботи)	Примітк
1	Вибір напряму дослідження та узгодження тематики кваліфікаційної роботи з керівником	10.01.2026	виконано
2	Ознайомлення з предметною областю; формулювання мети та задач дослідження; визначення об'єкта та предмета дослідження	01.02.2026	виконано
3	Робота над розділом 1 – дослідження предметної області та постановка задачі	01.03.2026	виконано
4	Робота над розділом 2 – проєктування інформаційноїсистеми збору та аналізукористувацької активності веб-сервісу	01.04.2026	виконано
5	Робота над розділом 3 – програмно-апаратна реалізація та тестування програмно-технічного засобу	29.04.2026	виконано
6	Оформлення пояснювальної записки згідно вимог	25.05.2026	виконано
7	Попередній захист ВКР	26.05.2025	виконано
8	Захист ВКР на засіданні ЕК	Червень 2026 року	

Здобувач _____ Назарій ЯСТРЕМСЬКИЙ
Підпис Ім'я, ПРІЗВИЩЕ

Керівник кваліфікаційної роботи _____ Ольга ПАВЛОВА
Підпис Ім'я, ПРІЗВИЩЕ

№ рядка	формат	Позначення	Найменування	Кіл. листів	№ екз	Примітка
			Текстові документи			
1		КвРІСТ 220184.22.01.13 ПЗ	Пояснювальна записка	68		
			Графічні матеріали			
2		КвРІСТ 220184.22.01.13 Е8	Скріншот головної	1		
			сторінки			
3		КвРІСТ 220184.22.01.13Е8	ER-діаграма	1		
4		КвРІСТ 220184.22.01.13Е8	Діаграма прецедентів	1		
			Use Case			

Зм	Арк	№ докум	Підпис	Дата	КвРІСТ 220184.22.01.13 ВП		
Розробив		Ястремський			Відомість проєкту	Літера	Аркуш	Аркушів
Перевір.		Павлова		9.06		У	1	68
Н. контр.		Кисіль				ХНУ, ІСТ-22-1		
Затв.		Павлова		9.06				

АНОТАЦІЯ

Тема кваліфікаційної роботи: «Інформаційна система збору та аналізу даних користувацької активності веб-сервісу».

Автор роботи: Назарій ЯСТРЕМСЬКИЙ.

Керівник роботи: Ольга ПАВЛОВА.

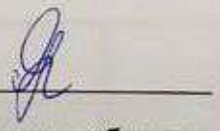
Пояснювальна записка: 68 с., 27 рис., 3 табл., 4 дод., 40 джерел.

Графічна частина: 3 креслення.

АГРЕГАЦІЯ ДАНИХ, БАЗА ДАНИХ, ВЕБ-АНАЛІТИКА, ВІЗУАЛІЗАЦІЯ, МОНІТОРИНГ.

Кваліфікаційна робота бакалавра присвячена розробці подієво-орієнтованої інформаційної системи моніторингу та аналізу поведінкових патернів користувачів вебсервісів. Актуальність теми зумовлена необхідністю забезпечення автономного володіння аналітичними даними згідно зі стандартами конфіденційності та потребою оптимізації вебінтерфейсів. Автоматизований контроль мікровзаємодій відвідувачів (перегляди сторінок, кліки, тривалість сесій) дозволяє оперативно виявляти логічні помилки в архітектурі сайтів, запобігати відтоку аудиторії та підвищувати конверсію цифрових продуктів.

Метою роботи є проєктування, реалізація та тестування програмно-технічного засобу для автоматизованого збору, персистентного зберігання, інтерактивної візуалізації та когнітивної інтерпретації телеметрії. У роботі виконано системний аналіз аналогів, обґрунтовано вибір хмарної NoSQL-архітектури, спроєктовано логічну структуру сховища Firestore. Практично реалізовано клієнтський скрипт-трекер, серверний шлюз на node.js/express та дашборд адміністратора на react/typeScript з інтеграцією моделі Gemini AI для автоматичної генерації експертних звітів природною мовою. Випробування підтвердили швидкодію системи в умовах інтенсивного навантаження.



Підпис здобувача

30.05.2026

Дата

ЗМІСТ

Вступ 4
1 Теоритичні основи побудови інформаційної системи збору та аналізу даних користувацької активності веб-сервісу 8
1.1 Сутність та призначення інформаіційних систем збору і аналізу даних 8
1.2 Особливості збору даних користувацької активності 9
1.3 Аналіз предметної області 11
1.4 Огляд існуючих аналогів 14
1.5 Проблематика розробки 20
1.6 Постановка задачі 22
1.7 Висновки до першого розділу 23
2 Проєктування інформаційної системи збору та аналізу користувацької активності веб-сервісу 25
2.1 Системний аналіз об'єкта проєктування та визначення технічних вимог 25
2.2 Архітектурне проєктування та розробка ескізного проєкту системи 27
2.3 Аналіз та вибір методів реалізації програмних підсистем 30
2.4 Проєктування логічної та фізичної структури бази даних 33
2.5 Розробка алгоритмів функціонування та математичного забезпечення системи 34
2.6 Проєктування людино-машинного інтерфейсу та UX-моделі 35
2.7 Проєктування модуля інтелектуального аналізу та когнітивної обробки метрик 37
2.8 Проєктування системи захисту інформації та розмежування прав доступу 39
2.9 Методологія інтеграції та життєвий цикл клієнтського модуля 40

2.10 Інфологічне проєктування та розробка ER-діаграми бази даних 41
2.11 Висновки до другого розділу 43
3 Програмно-апаратна реалізація та тестування програмно-технічного засобу 45
3.1 Опис середовища розробки та програмної реалізації модулів системи 45
3.2 Реалізація людино-машинного інтерфейсу та засобів візуалізації 46
3.3 Реалізація бази даних та підсистеми безпеки 50
3.4 Програмна реалізація модулів звітності та аналітичних досліджень 52
3.5 Реалізація сервісних функцій експорту та системи сповіщень 55
3.6 Реалізація механізмів локалізації та управління колірними темами 57
3.7 Тестування працездатності та аналіз результатів розробки 60
3.8 Висновки до третього розділу 63
Висновки 64
Перелік джерел посилань 65
Додаток А Скріншот головної сторінки 69
Додаток Б ER-діаграма 70
Додаток В Діаграма прецедентів Use Case 71
Додаток Г Лістинг коду 72

ВСТУП

Останнє є крайньою фазою формування глобального інформаційного суспільства, оскільки всі сфери людської діяльності майже повністю оцифровані, розподілені веб-сервіси та хмарні додатки слугують ключовим способом взаємодії між постачальниками послуг та споживачами. У контексті комп'ютерної інженерії та інформаційних технологій кожна з цих транзакцій розглядається як одинична подія інтерфейсу людина-машина, що призводить до безперервного високошвидкісного потоку неструктурованих телеметричних даних. Взаємодія користувача від перегляду сторінок та натискання інтерактивів до заповнення форми та виконання транзакцій утворює масив даних часових рядів, аналіз яких є важливим ресурсом для оцінки технічної стабільності, функціональної придатності та загальної ефективності веб-ресурсу. Загальна тенденція в цьому розвитку полягає в переході від агрегації логів у статичному сенсі до моніторингу в реальному часі на основі подій та прийняття підходу, орієнтованого на дані.

Огляд проблеми та актуальність теми. Хоча існують широко використовувані комерційні платформи веб-аналітики (наприклад, Google Analytics, Mixpanel, SimilarWeb), їх застосування несе кілька помітних інженерних та організаційних проблем. З цього боку більшість існуючих аналогів працюють на закритих моделях SaaS, що обмежує можливість адміністратора контролювати «сирі» дані, створює загрози вторгнення в конфіденційність (включаючи суворі європейські умови GDPR) та не дозволяє адаптувати внутрішні механізми обробки відповідно до вимог веб-ресурсу. Також застарілі аналітичні інструменти дають лише сухий підсумок статистики, і оператор повинен інтерпретувати метрики, шукати аномалії або виявляти причинно-наслідкові зв'язки лише для людей. Це стає значним інформаційним та когнітивним навантаженням на адміністратора. Актуальність теми кваліфікаційної роботи випливає з об'єктивної потреби у розробці автономно

розробленого програмного та технічного засобу для забезпечення повної незалежності від серверної інфраструктури, забезпечення конфіденційності телеметрії та автоматизації аналізу когнітивної інформації за допомогою інтегрованих сучасних моделей штучного інтелекту. Комерційний успіх розроблених результатів полягає в потенціалі локального використання, легкості інтеграції та негайній доступності експертних висновків природною мовою, що суттєво прискорює зменшення технічних та логічних помилок у веб-інтерфейсах.

Сфера застосування та мета розробки. Розроблений програмний та технічний засіб планується багатокомпонентною системою, призначеною для інтелектуального моніторингу та вимірювання поведінкових шаблонів, а також для збору та інтелектуальної інтерпретації. Сфера застосування розробки – це сфера веб-розробки, електронної комерції SaaS платформ та інших інформаційних сервісів, які надають послуги, що вимагають постійного відстеження дій користувачів. Система в основному спрямована на автоматичне прослуховування мікровзаємодій, без виснаження обчислювальної потужності клієнтської сторони, а також надає зручну інтерактивну панель для її адміністратора. Якщо вже існують деякі проблеми для вирішення, то аналіз існуючих проблем веде до мети кваліфікаційної роботи, яка полягає в розробці програмного та технічного засобу для збору та аналізу даних про активність користувачів веб-сервісу, що дозволяє оптимізувати управління веб-ресурсами шляхом візуалізації в реальному часі та забезпечення автоматизації інтерпретації статистичних показників. Кваліфікаційна робота, отже, повинна виконати наступні завдання для досягнення поставленої мети та вирішення науково-технічної проблеми, як зазначено вище:

1. Аналіз нових теоретичних принципів побудови інформаційних систем веб-аналітики; критичне спостереження за існуючими комерційними та відкритими аналогами.

2. Визначення характеристик теми, особливо HTTP-трафіку як безстанового середовища, та вивчення потокової та пакетної обробки подій активності.

3. Вивчення функціональних та нефункціональних вимог до системи, розгляд обмежень щодо масштабованості зберігання та захисту персональних даних відповідно до законодавства.

4. Виявлення основних взаємозв'язків взаємодії людина-машина, які потребують первинної фіксації, та розрахунок залежності сирих часових міток подій за допомогою часових міток подій на алгоритмах для проектування динамічних часових відер.

5. Проектування програмного забезпечення для інформаційних систем на основі подієво-орієнтованих парадигм (EDA), розробка інфологічної сутності ER-моделі та логічної та фізичної денормалізованої бази даних хмарного зберігання у всіх логічних аспектах.

6. Побудова алгоритмічної підтримки підсистеми вільного аналізу з високопродуктивними структурами даних на клієнтській стороні для усунення потреби у споживанні надлишкових серверних запитів.

7. Програмне та апаратне виконання: створення автономного клієнтського трекера скрипту, серверного шлюзу для прийому запитів від користувача та інтуїтивно зрозумілого інтерфейсу, призначеного для управління API з можливістю налаштування віджетів за допомогою функцій перетягування, підтримки багатомовності, зміни кольорової теми.

8. Реалізація інтелектуального аналізу програмного забезпечення шляхом інтеграції з Gemini API з Prompt Engineering (для генерації контекстних звітних даних природною мовою).

9. Проведення апробації та розширених модульних, інтегрованих та навантажувальних тестів на розробленому програмному та технічному засобі для забезпечення його функціональної відповідності очікуваному попиту та надійності при високому навантаженні на запис.

Отримані результати від виконання вищезазначених завдань повністю виконують та реалізують загальну мету проектування, а саме розробку сучасного інструменту який в цілому аналізує веб-ресурс, та визначає проблеми які потрібно виправити.

.

1 ТЕОРИТИЧНІ ОСНОВИ ПОБУДОВИ ІНФОРМАЦІЙНОЇ СИСТЕМИ ЗБОРУ ТА АНАЛІЗУ ДАНИХ КОРИСТУВАЦЬКОЇ АКТИВНОСТІ ВЕБ-СЕРВІСУ

1.1 Сутність та призначення інформаіційних систем збору і аналізу даних

З огляду на всесвітню цифровізацію економіки, швидкий розвиток інформаційного суспільства, веб-сервіси еволюціонували від простого інструменту для інформативного представлення інформації до взаємозв'язку, який також стає багатими інтерактивними системами через взаємодію між постачальником послуг та кінцевим користувачем. У комп'ютерній теорії кожна взаємодія відбувається у дискретній події в інтерфейсі людина-машина, телеметричні дані розглядаються як набір подій у цій галузі комп'ютерної інженерії [1]. Весь спектр дій користувача – від простого перегляду статичної сторінки до складних транзакційних дій, кліків на інтерактивні елементи або заповнення форми - становить постійний потік неструктурованих даних. Такі дані є важливим орієнтиром для оцінки технічної стабільності, функціональної корисності та продуктивності функціонування веб-ресурсу. Інформаційна система збору та аналізу даних про активність користувачів є багатопрофільним програмно-технічним інструментом з архітектурою, що вирішує завдання автоматичного накопичення, швидкої обробки, довгострокового зберігання та когнітивної інтерпретації статистичних показників. З точки зору системного підходу, цей комплекс служить зворотним зв'язком у кібернетичному механізмі управління веб-продуктом [2]. Основна мета полягає в перетворенні "сирих" даних, які самі по собі мають низьку інформативність, у структурований аналітичний інструмент, що надає об'єктивну основу для прийняття стратегічних управлінських та інженерних рішень через аналітичний інструмент. Функціональність таких систем базується на складних алгоритмах агрегації, які дозволяють не лише фіксувати кількісні показники, але й виявляти якісні особливості поведінки. Особливо, розгортання таких інструментів дозволяє

отримати рівень популярності окремих функціональних модулів веб-сервісу, глибокий аналіз конверсії на кожному етапі користувацького досвіду, моделювання складних поведінкових випадків. Крім того, аналітичні системи виступають як діагностичні інструменти, що дозволяють раннє виявлення основних недоліків у структурі інтерфейсу та "вузьких місць", які впливають на здатність користувача взаємодіяти з системою.

1.2 Особливості збору даних користувацької активності

Завдання збору телеметричних та поведінкових даних у сучасному розподіленому веб-середовищі є складною інженерною проблемою, яку потрібно вирішувати шляхом постійного запису дискретних подій, що відбуваються при безпосередній взаємодії з комплексом людина-машина. Як безстанні системи, що працюють під протоколом HTTP, веб-ресурси мають специфіку, що вимагає відповідних архітектурних рішень для підтримки контексту користувацьких сесій. Транспортний шар для передачі такої інформації може бути класифікований на основі двох загальних парадигм, хоча обидві вони мають обмеження та переваги для споживання серверних обчислювальних ресурсів [3]. Перша парадигма – це потокова, реального часу передача цих мікроподій, у якій, коли подія записується, вона передається до аналітичного шлюзу в браузері, безпосередньо серіалізується в пакет даних і відправляється до аналітичного шлюзу через асинхронні неблокуючі HTTP-запити або постійні з'єднання WebSocket. Друга парадигма використовує підхід пакетної обробки, у якому події спочатку збираються в локальному буфері на стороні клієнта і відправляються на сервер у вигляді одного масиву за запланованими подіями або коли створюються умови, що їх викликають. Вибір будь-якого з цих підходів визначає архітектуру всієї інформаційної системи та навантаження на комунікаційні шляхи в мережі. Як мінімум, аналіз на архітектурних рівнях веб-систем дозволить визначити основні три джерела

отримання первинних даних про активність. Перше джерело – це серверний журнал подій, у якому автоматично реєструється кожен HTTP-запит до системи, такі як IP-адреса, час доступу, запит ресурсів та значення відповіді тощо. Його високий рівень надійності (не блокує клієнта) та низький рівень дискретності (поведінка на сторінці, така як кліки на динамічні елементи або прокрутка, не відстежується) є двома основними перевагами. Друге, гнучке джерело даних – це трекерні скрипти, які безпосередньо встановлюються в DOM сторінки. Це означає, що вони мають вбудовані інструменти кешування, при цьому маючи можливість аналізувати поведінку користувача. Такі модулі збору працюють у окремому середовищі виконання браузера і можуть відстежувати будь-які події вводу-виводу в реальному часі. Третє джерело – це вбудовані API сторонніх сервісів, які можуть збагачувати поведінковий профіль, додаючи дані про транзакції, зовнішні переходи або стан авторизації. Крім обов'язків збору, існує також дуже важливе технічне завдання забезпечення цілісності, надійності та безпеки збереженої інформації. У умовах децентралізованого веб-середовища онлайн, реалізація захисту даних також передбачає впровадження суворих криптографічних механізмів та відповідність сучасним законам про конфіденційність, регульованим сучасним нормативним законодавством, таким як GDPR або CCPA. Нові інженерні методи вимагають повного усунення використання файлів ідентифікації третіх сторін на користь абсолютно нового способу кодування інформації і натомість покладаються на локальне зберігання сесій у браузері, що дозволяє ідентифікацію сесій та анонімну ідентифікацію сесій без збору будь-якої інформації про особистих користувачів. Крім того, для забезпечення надійного потоку інформації, можна також розробити валідацію та фільтрацію на рівні API-шлюзу для забезпечення стабільності потоку інформації, що автоматично блокує підроблені запити з мережі через дійсний API-шлюз, так що підроблені запити, зроблені пошуковими ботами або фальшивими автоматизаційними скриптами, автоматично припиняються, створені пошуковими ботами або зловмисними автоматизаційними скриптами. З

інженерної точки зору, проекти підсистем збору даних повинні враховувати дуже високу інтенсивність вхідного потоку, який слід розглядати як систему обробки великих даних, наскільки це не потрібно. Навантаження на сервер обробки даних значно асиметричне (наприклад, запис перевищує читання). Це передбачає високе навантаження на масштабованість та стійкість до збоїв серверної інфраструктури, яка повинна бути стійкою до піків трафіку, таких як лавинний трафік, без зниження продуктивності цільового веб-сервісу. Це питання вимагає оптимізації структур бази даних та спеціалізованих моделей зберігання NoSQL, які, порівняно з реляційними СУБД, пропонують лінійне горизонтальне масштабування та підтримку динамічних схем документів без тривалого блокування таблиць. Технічні властивості та характеристики збору даних формують вибір гнучких, легких та безпечних програмних інструментів, які будуть розгорнуті для ефективного вирішення невизначеності в мережевому середовищі.

1.3 Аналіз предметної області

Темою цього дослідження є складний інформаційний потік для автоматизованого збору дискретних подій з розподілених веб-сервісів, їх збирання для високопродуктивного зберігання з часом у добре спроектованих базах даних та інтелектуальних операцій для створення змістовних аналітичних звітів, а також для обчислення ключових показників ефективності. У комп'ютерній інженерії цей механізм розглядається як побудова архітектурно-специфічного конвеєра для вилучення даних з вхідних даних та їх обробки, так що кожна процедура – від фіксації вводу користувача до візуалізації на панелі адміністратора – повинна бути оптимізована з урахуванням швидкості, стійкості до збоїв та цілісності даних. Аналіз предметної області дозволяє формалізувати взаємодію користувацького інтерфейсу як потік телеметричних станів, потім розглянутих з різних точок зору. Реалізація такого складного процесу вимагає

взаємодії цілого ряду автономних, але тісно пов'язаних компонентів, які разом утворюють єдину програмно-апаратну екосистему. Першим ланцюгом у цій мережі є джерело прямих подій, яке зазвичай є клієнтським середовищем – веб-браузер, мобільний додаток або зовнішній інтерфейс програмування додатків. Це джерело обробляється через збір даних, і це контрольна функція модуля збору, програмний процес, який веде низькорівневий запис кожної значущої події системи. Другим ключовим вузлом є сервер прийому, який є високошвидкісним вузлом для прийому вхідних пакетів даних, перевірки пакетів даних до цього моменту, що забезпечить маршрутизацію до сховища даних. Також саме сховище, засноване на сучасних системах управління базами даних, є основою для зберігання минулої інформації на довгий термін. Модулі обробки та розрахунку метрик на верхніх рівнях архітектури перетворюють сирі записи за допомогою цих спеціалізованих алгоритмів, використовуючи математичний підхід, у структуровані метрики, які, нарешті, передаються з цих точок даних до підсистеми представлення результатів, де будуються графічні панелі, інтерактивні діаграми та табличні звіти. Обробка зібраної інформації також включає різноманітні критерії, використовуючи методи статистичного аналізу та фільтрації даних. Основна увага на цьому етапі приділяється об'єднанню не систематизованих подій у значущі аналітичні об'єкти, тим самим відфільтровуючи інформаційний шум і виявляючи корисні повідомлення. Алгоритм системи може бути запрограмований для досягнення оптимізації статистики роботи сервісу з найважливіших експлуатаційних характеристик користувача (середня тривалість сесії користувача, конверсія, частота повернення тощо) за допомогою алгоритмічної допомоги. Особливо важливо, щоб ви аналізували метрики збоїв і розробляли логічні маршрути, щоб інженерна команда могла виявити логічні прогалини в інтерфейсі або технічні проблеми, що перешкоджають правильному виконанню цільових дій. Для забезпечення правильного побудови та функціонування систем такого типу використовуються ряд інструментальних технік комп'ютерної інженерії. Логіка клієнта будується з

клієнтських неблокуючих скриптів браузера, які захоплюють потік подій у реальному часі без погіршення продуктивності клієнтської сторони. На серверному рівні ми проектуємо масштабовані API інтерфейси, які дозволяють безпечно та рівномірно приймати дані від тисяч клієнтів одночасно. Ми обираємо зберігання інформації в інфраструктурах баз даних, розроблених для забезпечення важкого запису та швидкого отримання великих масивів часових рядів. Остання фаза процесу аналізу забезпечується функціями візуалізації, які перетворюють цифрові ряди у візуальні представлення, які можуть бути прочитані з високим рівнем огляду, що дозволяє оператору приймати рішення на основі об'єктивної інформації моніторингу.

На рисунку 1.1 – зображена схема алгоритму роботи таких сервісів як Google Analytics.

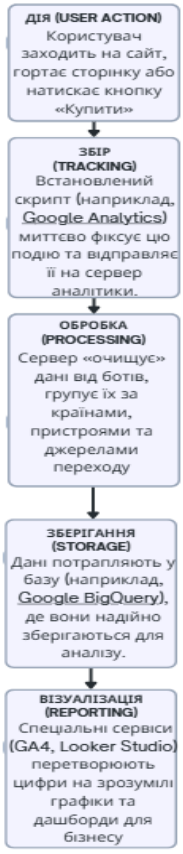


Рисунок 1.1 – Алгоритм роботи сервісу Google Analytics [4]

1.4 Огляд існуючих аналогів

Огляд вже доступних аналогів. Вже існує багато систем для отримання даних про активність користувачів та аналізу з використанням відкритого коду (та комерційних) у вигляді систем. Вони відрізняються за функціональністю, робочими моделями, ліцензійними угодами, стилями адаптації та вимогами до впровадження нового елемента. Найпоширенішим інструментом веб-аналітики є Google Analytics [5]. Ця система діє як хмарний сервіс і додає код відстеження до веб-ресурсу для додавання користувацької інформації для нього. Після встановлення сервіс автоматично збирає інформацію про відвідування сторінок, джерела трафіку, поведінку користувачів, тривалість сесії та інші метрики. Аналітична обробка здійснюється на стороні сервера постачальника послуг, а результати надаються у вигляді структурованих звітів. Перевагою цього рішення є швидкість впровадження та велика різноманітність готових метрик, але воно не забезпечує повного контролю над сирими даними, а також їх розташуванням. Для компаній з підвищеною конфіденційністю та нестандартними вимогами до обробки інформації або для компаній, які потребують обробки інформації на нестандартній інфраструктурі, ця залежність є особливо критичною.

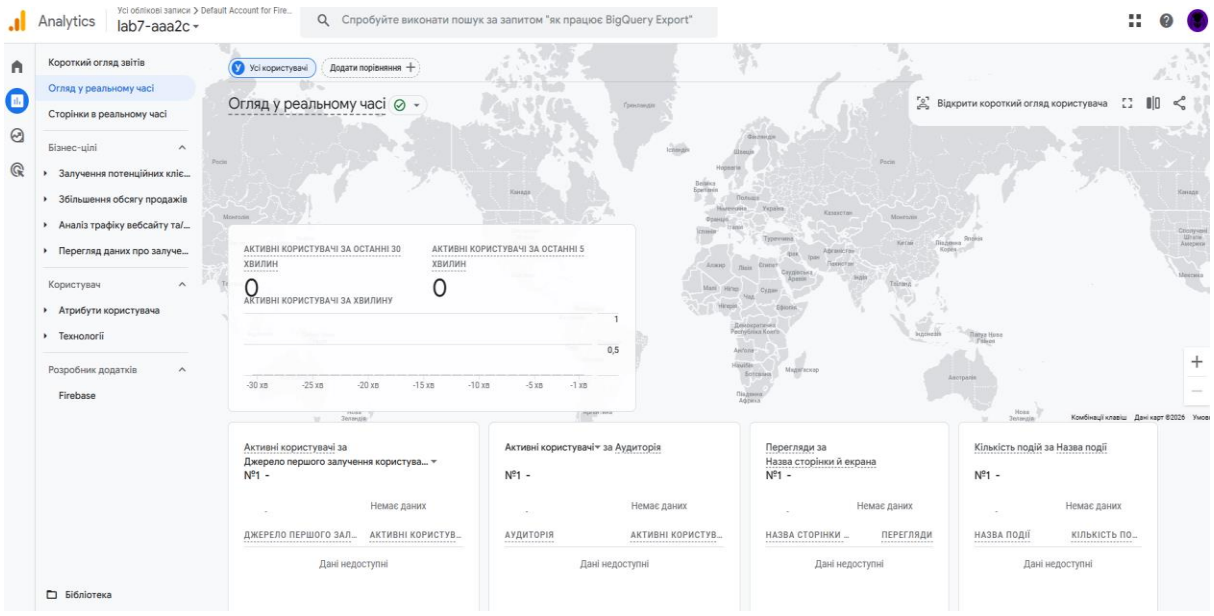


Рисунок 1.2 – Google Analytics [5]

Конкурентом комерційного SaaS рішення є Matomo [6], платформа з відкритим кодом. Matomo може використовуватися на сервері підприємства, на відміну від згаданого вище, надаючи повний контроль над усім процесом збору, зберігання та аналізу даних. Модель SaaS, розміщена на сервері, поки що не має жодних з цих вимог, незважаючи на існування хмарних рішень, доступних для хмарного навантаження. Вона дозволяє відстеження подій, встановлення цілей, сегментацію аудиторії та створення звітів. Завдяки модульній архітектурі функціональність може бути розширена за допомогою додаткових плагінів. Впровадження такого рішення одночасно передбачає серверну інфраструктуру, технічну підтримку та адміністрування, що робить експлуатацію набагато складнішою. Matomo має більше можливостей для налаштування, хоча також потребує більше ресурсів.

Головне меню панелі інструментів зображено на рисунку 1.3.

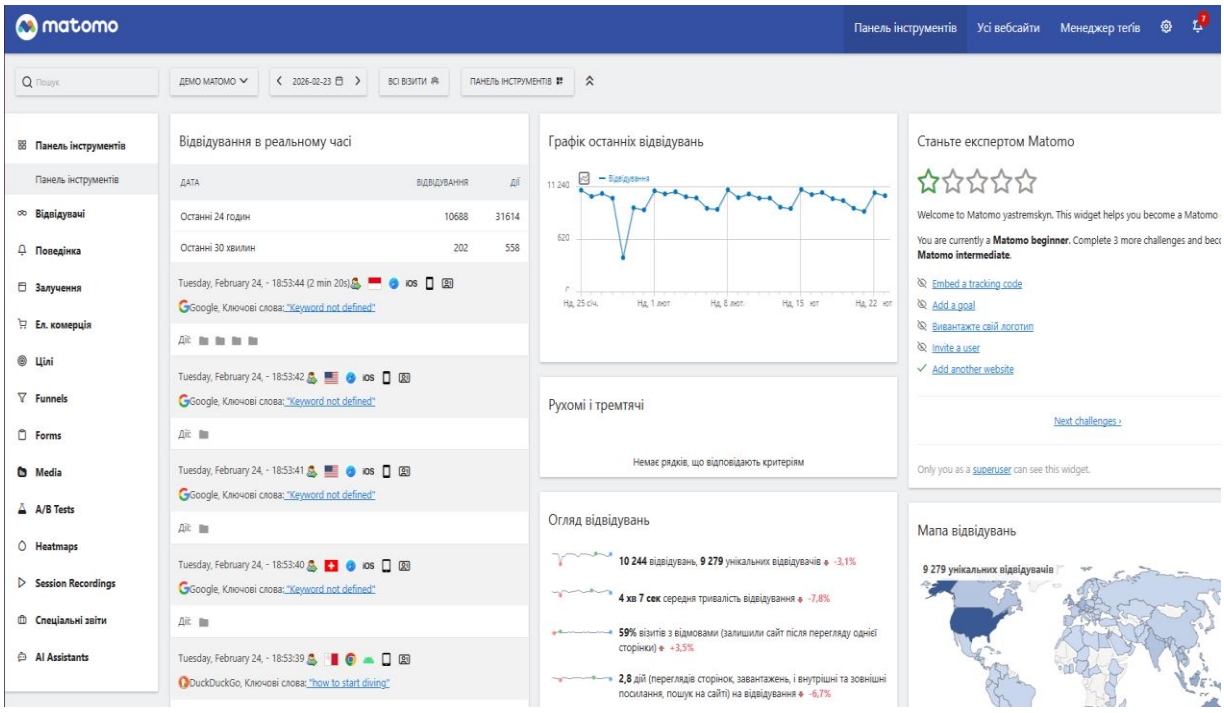


Рисунок 1.3 – Панель інструментів сервісу Matomo [6]

Mixpanel [7] є ще одним прикладом сучасної системи аналітики, яка особливо зосереджена на аналізі та дослідженні поведінки користувачів у

сценаріях. Тут увага приділяється не лише кількісним метрикам відвідувань, але й аналізу дій, послідовностей взаємодії та груп користувачів. Mixpanel зазвичай використовується в продуктах SaaS та мобільних додатках для тестування використання функціональних можливостей, пов'язаних із сервісом. Окрім надання гнучких форм формування звітів та воронок конверсії, система також діє як хмарне рішення з відповідними обмеженнями на зберігання та контроль даних, а повна функціональність здебільшого на платній основі.

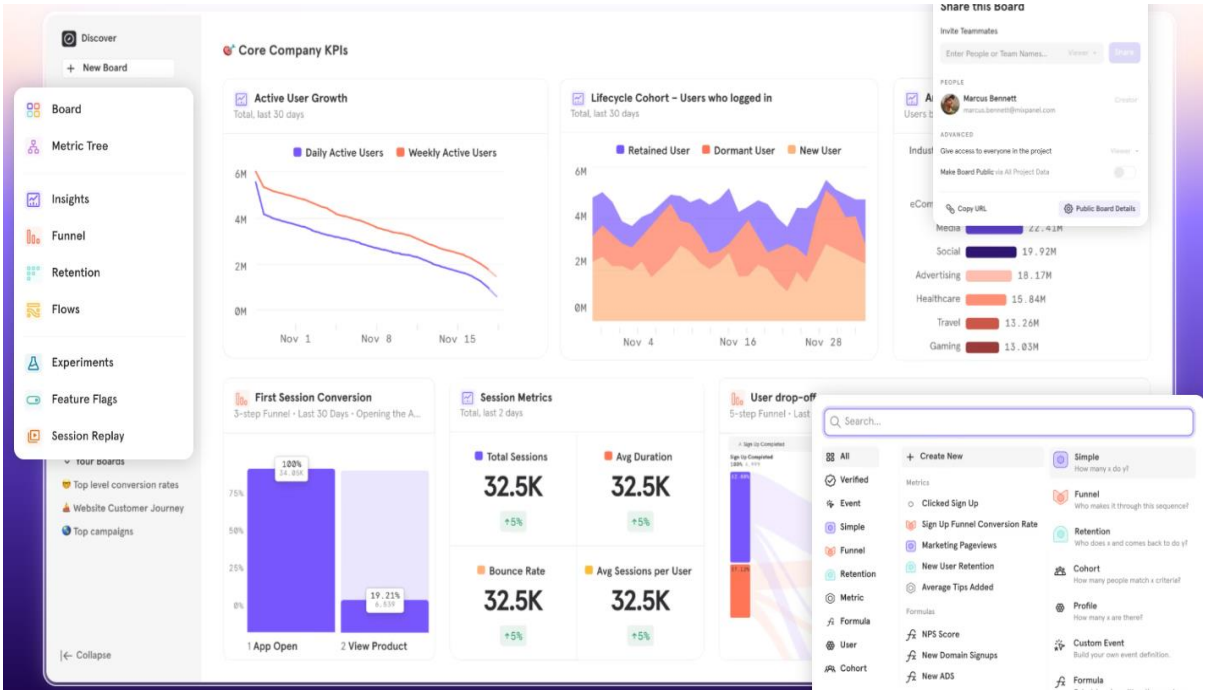


Рисунок 1.4 – Mixpanel [7]

У корпоративному сегменті вагоме місце посідає Adobe Analytics, що функціонує в межах комплексної маркетингової екосистеми Adobe. Ця платформа вирізняється розширеними можливостями інтеграції, здатністю обробляти значні масиви даних (Big Data) та вбудованими інструментами предиктивного аналізу. Водночас суттєві фінансові витрати на впровадження, а також високий поріг входження під час налаштування обмежують її дистрибуцію переважно масштабами великих підприємств та транснаціональних компаній.

Окремої уваги заслуговує платформа SimilarWeb [8], яка є одним із галузевих стандартів у сфері цифрової аналітики. На відміну від систем прямого трекінгу (як-от Google Analytics або проєктувальне у цій роботі рішення), методологія SimilarWeb базується на зовнішньому моніторингу та оцінюванні мережевого трафіку без безпосередньої інтеграції програмних скриптів у вихідний код цільового ресурсу. Такий підхід робить її ефективним інструментом для конкурентного аналізу та стратегічного планування, хоча й накладає певні обмеження на верифікацію мікровзаємодій користувачів із веб-інтерфейсом.

Технологічний стек SimilarWeb опирається на багатоканальний збір відомостей із чотирьох ключових джерел: панельних даних (user panels), даних інтернет-провайдерів (ISP data), прямих вимірювань та індексації власними пошуковими роботами. Оперування подібною архітектурою потребує значних обчислювальних потужностей для щоденної обробки терабайтів неструктурованої інформації. Для синтезу цих потоків і побудови прогностичних моделей відвідуваності застосовуються алгоритми машинного навчання. У контексті комп'ютерної інженерії SimilarWeb демонструє класичний приклад системи обробки великих даних, де архітектурна цінність зміщена з інтерфейсної частини на інтелектуальний аналіз "сирих" інформаційних потоків задля досягнення статистичної значущості результатів.

Функціонал платформи дозволяє аналізувати вектори залучення аудиторії за ключовими каналами (direct, referrals, search, social, mail, display). Це забезпечує розуміння загального контексту поведінки користувачів, проте виявляє обмеження в аспекті деталізації мікропатернів. Оскільки SimilarWeb оперує апроксимованими (наближеними) показниками, вона не здатна реєструвати поодинокі події, як-от кліки на конкретні елементи інтерфейсу чи динамічну трансформацію DOM-структури сторінки в реальному часі, що є критичним для UX/UI-досліджень. Відтак, SimilarWeb позиціонується як макроаналітичний інструмент, який доповнює, але не замінює локальні системи

внутрішнього моніторингу, зорієнтовані на подієво-керований аналіз (event-driven analysis).

Головною перевагою аналізованої платформи є функція порівняльного аналізу, що дозволяє зіставляти показники відмов, середню тривалість сесій та глибину перегляду сторінок із метриками конкурентів. Незважаючи на це, висока вартість комерційних ліцензій та помітне зниження точності розрахунків для веб-ресурсів із низьким або середнім обсягом трафіку (до 50 000 візитів на місяць) роблять її використання недоцільним для локальних проєктів чи стартапів на ранніх етапах розвитку. До того ж закритий характер алгоритмів обробки даних унеможливлює інтеграцію специфічних модулів інтелектуального аналізу на базі власних нейромережевих моделей. Це додатково підтверджує актуальність розробки кастомних інформаційних систем, адаптованих під завдання збору та аналізу активності користувачів конкретного веб-сервісу.

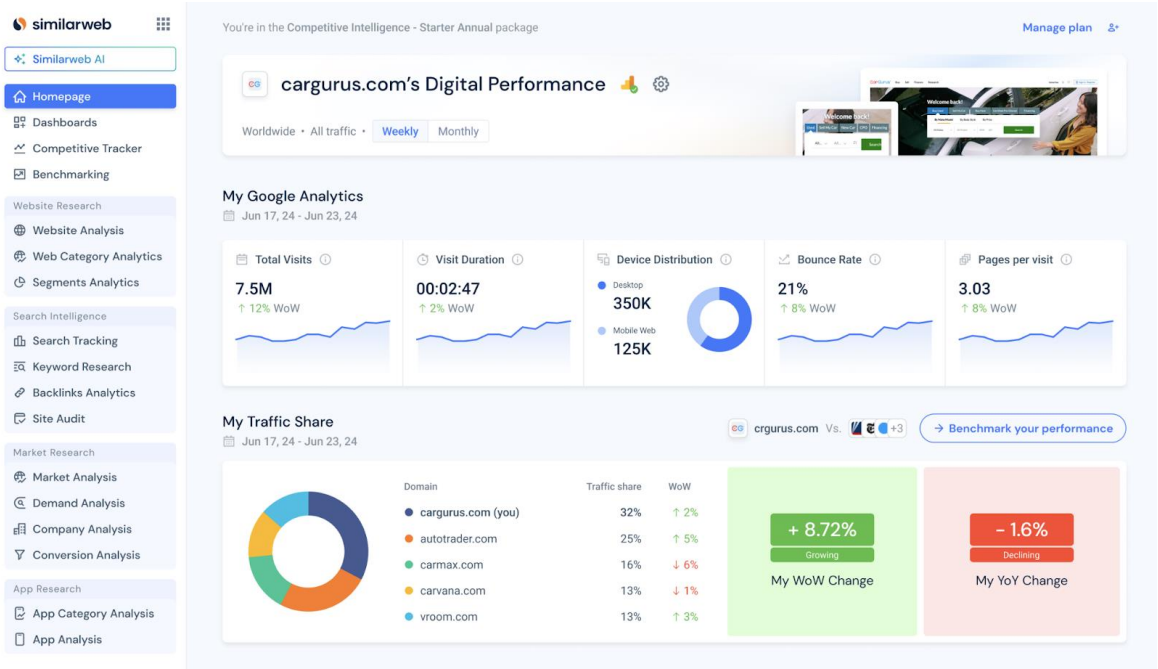


Рисунок 1.5 – SimilarWeb [8]

Критичний огляд вищезгаданих аналогів показує, що їх можна розділити на 2 основні напрямки. Існують механізми закритих хмарних сервісів та обробки

інформації, що базуються на хмарі, та відкриті (з відкритим вихідним кодом) платформи, які в основному спираються на локальну інфраструктуру користувача. Перша категорія забезпечує низький поріг входу, можливість швидкого розгортання та розширені інструменти для візуалізації даних, але обмежує користувача в можливості доступу та контролю критично важливих даних. На противагу цьому, автономні архітектурні рішення надають більше можливостей вибору та масштабування, але вимагають від персоналу володіння передовими, чіткими та спеціалізованими знаннями в галузі IT, а також початкових витрат на підтримку серверної інфраструктури. Той факт, що існує такий широкий вибір інструментів та рішень, не зменшує обґрунтованості розробки індивідуальних рішень для збору та аналізу активності користувачів. Розробку описаного рішення слід вважати життєво важливою, коли існує висока потреба в триманні інформації під локальним контролем, створенні власних аналітичних або прогнозних процедур та безпосередньому вбудовуванні функції відстеження в загальну структуру веб-сервісу. Завдяки цій техніці можна розробити комп'ютеризоване рішення, яке було б глибоко модифіковане на основі численних вимог, що існують на конкретному веб-сайті, і яке також відповідає академічній та прикладній методології «Інформаційних систем і технологій».

Порівняння наведено в таблиці 1.1.

Таблиця 1.1 – Порівняння популярних систем аналізу користувацької активності

Характеристика	Google Analytics	Matomo	Mixpanel	Розроблена система (очікувано)
Ліцензійна модель	SaaS	Відкритий код	SaaS	Відкритий код

Кінець таблиці 1.1

Контроль даних	Обмежений	Повний	Обмежений	Повний
Підтримка подієвого аналізу	Так	Так	Так	Так
Кастомні звіти	Обмежено	Так	Так	Так
Масштабованість	Висока	Середня	Висока	В залежності від архітектури
Вартість впровадження	Низька/безкоштовно	Безкоштовно	Платно	Власний
Візуалізація	Вбудована	Вбудована	Вбудована	За реалізацією
Конфіденційність даних	Залежить від 3-ї сторони	Повна	Залежить від 3-ї сторони	Повна

1.5 Проблематика розробки

Розробка власної інформаційної системи збору та аналізу даних користувацької активності пов’язана з низкою проблемних аспектів:

1. Обсяг даних

Веб-сервіси генерують великий потік подій при активній взаємодії користувачів. Система повинна бути здатною обробляти великі обсяги інформації, оптимізувати запити до бази даних та забезпечувати доступність даних для аналізу. Особливої уваги потребує організація зберігання подій

користувацької активності, таких як перегляд сторінок, кліки, авторизації, виконання дій у системі. Накопичення великої кількості записів може призводити до зниження продуктивності, тому необхідно передбачити механізми архівації, агрегування даних та використання ефективних структур бази даних для забезпечення стабільної роботи системи.

2. Масштабованість

З ростом кількості користувачів зростає й навантаження на систему. Необхідно передбачити масштабовану архітектуру, яка дозволить забезпечити ефективну роботу навіть при значному зростанні трафіку. Це може включати розподіл навантаження між серверами, використання мікросервісного підходу або горизонтальне масштабування бази даних. Однак важливо, щоб система могла адаптуватися до змін у кількості користувачів без суттєвої модифікації програмного коду чи повної перебудови архітектури.

3. Точність та валідність даних

Помилки в зборі або обробці даних можуть призвести до хибних висновків. Важливо забезпечити механізми перевірки цілісності, уникнення дублювання подій, обробки неповних даних. Для цього необхідно реалізувати перевірку вхідних параметрів, контроль унікальності подій, логування помилок та механізми повторної обробки у випадку збоїв. Також, потрібно враховувати вплив технічних факторів, таких як блокування скриптів у браузері або нестабільне інтернет-з’єднання, що можуть впливати на повноту зібраної статистики.

4. Конфіденційність і захист персональних даних

Відповідно до вимог законодавства (зокрема GDPR, локальних норм щодо персональних даних), система має забезпечувати захист від несанкціонованого доступу, шифрування та безпечне зберігання інформації. Під час проєктування необхідно мінімізувати обсяг персональних даних, які зберігаються, використовувати механізми анонімізації або псевдонімізації, а також реалізувати розмежування прав доступу до аналітичної інформації. Захист каналів

передавання даних і резервне копіювання також є важливими складовими забезпечення безпеки системи.

5. Візуалізація та інтерпретація результатів

Аналітична інформація повинна бути представлена у зручному для користувача вигляді: графіки, діаграми, дашборди, що вимагає вибору відповідних бібліотек чи побудови власних компонентів. Важливо не лише коректно розрахувати показники, але й забезпечити їх логічну структуру, можливість фільтрації, групування та порівняння даних за різними параметрами. Зрозуміла візуалізація сприяє швидкому аналізу ситуації та прийняттю обґрунтованих управлінських рішень на основі отриманої статистики.

Підсумовуючи, проектування інформаційної системи збору та аналізу даних користувацької активності слід розглядати як комплексне інженерне завдання, що інтегрує технічний, архітектурний та організаційний зрізи. Синергія таких факторів, як імператив обробки високоінтенсивних потоків інформації, забезпечення лінійного масштабування в умовах пікових навантажень, верифікація точності й валідності метрик, а також суворе дотримання регуляторних вимог у сфері захисту персональних даних, формує детермінований пул системних викликів. Додаткове ускладнення архітектури зумовлене необхідністю розробки ергономічних та інтуїтивно зрозумілих інтерфейсів візуалізації. Відтак, системне вирішення окреслених проблем є фундаментальною передумовою для розгортання відмовостійкого, високопродуктивного та релевантного програмного комплексу, здатного забезпечити повноцінну аналітичну підтримку сучасного веб-сервісу.

1.6 Постановка задачі

Основними завданнями роботи є:

1. Провести аналіз предметної області та існуючих систем веб-аналітики.
2. Визначити функціональні та нефункціональні вимоги до системи.

3. Розробити архітектуру інформаційної системи збору та аналізу даних.
4. Реалізувати модуль збору даних про дії користувачів (перегляди сторінок, кліки, сесії тощо).
5. Реалізувати базу даних для зберігання статистичної інформації.
6. Реалізувати механізм формування аналітичних показників і звітів.
7. Провести тестування та оцінити ефективність розробленої системи.

1.7 Висновки до першого розділу

Результати аналітичного дослідження, проведеного в цьому першому розділі, свідчать про те, що розробка спеціалізованої програмно-апаратної системи для моніторингу та аналізу взаємодії користувачів стала критично важливим вектором для прогресу сучасної комп'ютерної інженерії. Дослідження предметної області підтвердило, що ефективне функціонування та стратегічне масштабування веб-сервісів тісно пов'язане з якістю інфраструктури зворотного зв'язку, яка може перетворювати високонапружені канали сирої телеметрії у структуроване інформаційне забезпечення для управлінських рішень. Глибоке розуміння технічних вимог до захоплення мікровзаємодій у розподілених середовищах виявило, що оскільки протокол HTTP є безстанним, необхідно розробляти гнучкі архітектурні шаблони на основі клієнтських трекер-скриптів та безсерверних обчислень у хмарі. Початковий аналіз глобальних версій - включаючи Google Analytics, Matomo, Mixpanel та SimilarWeb - підкреслив системну дисонансність між функціональними можливостями пропрієтарних систем та ступенем реального контролю над конфіденційною інформацією. Більшість закритих SaaS-рішень обмежують доступ до первинних («сирих») даних, тоді як впровадження систем з відкритим кодом супроводжується значними інженерними (наприклад, розробка програмного забезпечення, тестування) та економічними (наприклад, витрати на підтримку власної інфраструктури) витратами. Ця проблема демонструє необхідність розробки

можливостей пропрієтарного програмного продукту для гарантування ізоляції даних користувачів, високої дискретності реєстрації подій, а також впровадження високорівневих передових методик для автоматичної інтерпретації статистичних рядів через моделі нейронних мереж. Основним результатом аналізу стало виявлення ключових технологічних проблем щодо горизонтального масштабування, обробки величезних обсягів Big Data та проектування ергономічних інтерфейсів людина-машина. Систематизація цих елементів завдання полегшила формалізацію завдання проектування та розробки системи, яка відповідає стандартам продуктивності, відмовостійкості та безпеки. Результати були інтерпретовані для надання детального теоретико-методологічного та технічного підґрунтя для подальшої розробки структури логічної підтримки даних (логічної підтримки та алгоритмічного управління) інформаційної системи, що стане основою для подальшої роботи.

2 ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ЗБОРУ ТА АНАЛІЗУ КОРИСТУВАЦЬКОЇ АКТИВНОСТІ ВЕБ-СЕРВІСУ

2.1 Системний аналіз об’єкта проєктування та визначення технічних вимог

Розробка сучасного програмно-технічного комплексу, який працює з великими обсягами даних, вимагає ретельного аналізу того, над чим ми працюємо. У нашому випадку мова йде про інформаційне середовище, де користувач взаємодіє з веб-інтерфейсом. Одна з основних складностей тут полягає в тому, щоб створити модель збору даних, яка дозволить вести детальне логування подій без значних втрат у продуктивності на стороні клієнта (frontend). Під час дослідження стало зрозуміло, що традиційні методи збору статистики часто затримують передачу даних або вимагають занадто багато налаштувань від розробників. Тому ми визначили пріоритетним завданням створення автономного програмного рішення, яке б автоматично адаптувалося до структури веб-ресурсу без потреби у додаткових конфігураціях. На етапі аналітики головне завдання – це розбити абстрактні цілі на конкретні архітектурні патерни реалізації. Щоб забезпечити горизонтальне масштабування системи, розробили механізм ідентифікації сесій, який не залежить від серверних cookies, оскільки сучасні браузери накладають серйозні обмеження на їх використання. Найкращим рішенням у цій ситуації стало застосування локального сховища браузера (Web Storage API) разом із генерацією універсальних унікальних ідентифікаторів (UUID). Це дає змогу стабільно відстежувати поведінку користувача в межах одного проекту, гарантує анонімність даних та швидкість операцій читання/запису. Крім того, під час специфікації вимог ми зрозуміли, що потрібно впровадити інтелектуальний компонент для перетворення сирих подій у структуровані аналітичні метрики. Не менш важливою частиною проектування є забезпечення високої доступності системи. Ми стабілізуємо роботу аналітичного шлюзу під час пікових навантажень завдяки безсерверній архітектурі. Тут кожна транзакція

обробляється в окремому обчислювальному середовищі, що усуває потребу в складних балансувальниках навантаження на ранніх стадіях розробки. Це дозволяє системі динамічно виділяти ресурси пам'яті та процесора залежно від інтенсивності вхідного HTTP-трафіку. В результаті комплексного аналізу ми змогли сформувати набір жорстких вимог та проектних рішень, які стали основою для подальшого архітектурного моделювання системи.

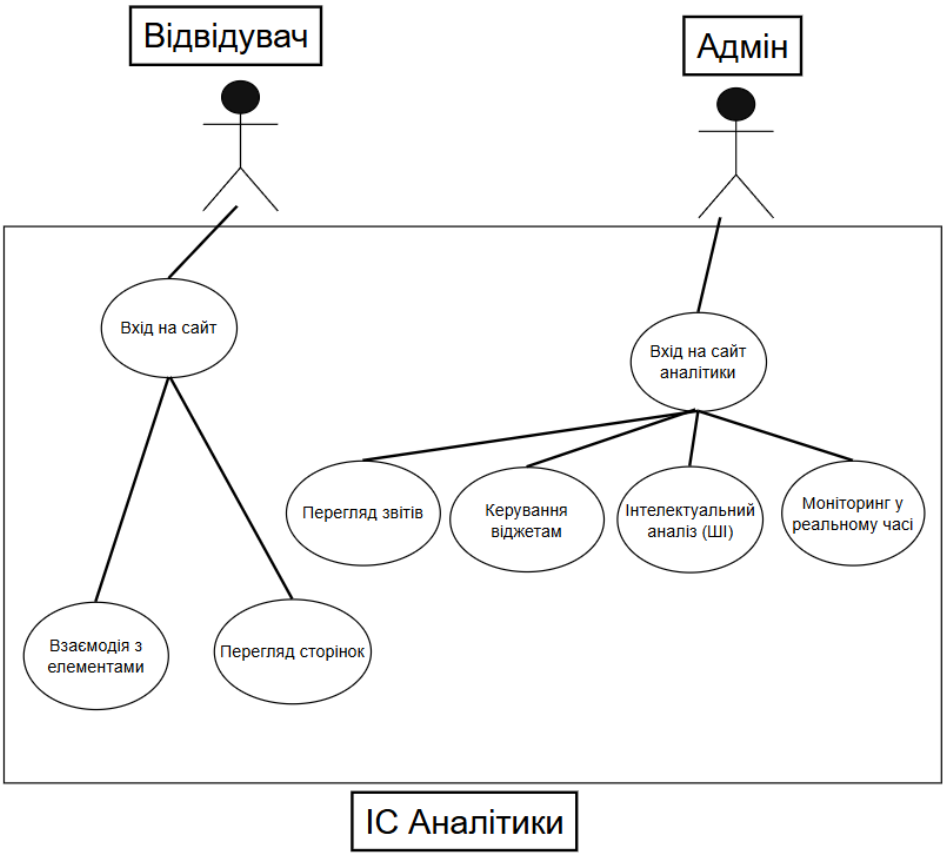


Схема 2.1 – Діаграма прецедентів Use Case

На схемі 2.1 зображено дві ключові ролі. Актор «Відвідувач» ініціює прецеденти збору даних, такі як «Перегляд сторінок» та «Взаємодія з елементами», які виконуються автоматично за допомогою вбудованого JavaScript-трекера.

Актор «Адмін» є основним користувачем панелі управління. Його взаємодія з системою починається з прецеденту «Вхід на сайт аналітики», мається на увазі вхід на сайт як адміністратор, який аналізує якийсь веб- сервіс,

це обов’язкова умова. Після цього йому відкривається можливість до різних аналітичних функцій, такі як «Перегляд звітів», «Керування віджетами», «ШІ аналітика» та «Моніторинг у реальному часі».

2.2 Архітектурне проєктування та розробка ескізного проєкту системи

При розробці архітектури програмно-технічного засобу я вирішив використати трирівневу модель для організації інформаційних потоків. Одним із ключових моментів стало питання, чому я обрав подієво-орієнтовану архітектуру (EDA). А як же налаштувати систему так, щоб вона миттєво реагувала на дії користувачів? У цій моделі кожна дія сприймається не просто як запис у базі даних, а як окрема подія, що запускає ланцюгову реакцію всередині системи. Перший рівень – це рівень збору. Тут я реалізував легкий скрипт-ін’єкцію, який просто фіксує контекст події: URL, сесія, елемент взаємодії і миттю передає цю інформацію на наступний рівень. Завдяки такому підходу, між сайтом-джерелом та аналітичною платформою досягається «слабке зчеплення», що дуже зручно. Другий рівень – це серверний шлюз обробки. Я спирався на концепцію Edge Computing, щоб обробка даних відбувалася якомога ближче до користувача. На цьому етапі система автоматично аналізує мережеві заголовки для визначення геолокації та характеристик пристрою, що дозволяє клієнтській частині уникнути важких обчислень. А ось третій рівень – це вже рівень персистентності та візуалізації. Тут я проєктую механізм синхронізації стану в реальному часі. Щоб відповісти на запитання «як оновлювати графіки без перезавантаження?», ми обрали модель WebSocket-підписок, яка інтегрується безпосередньо в логіку сховища даних. Архітектурна схема системи зображена на рисунку 2.2.

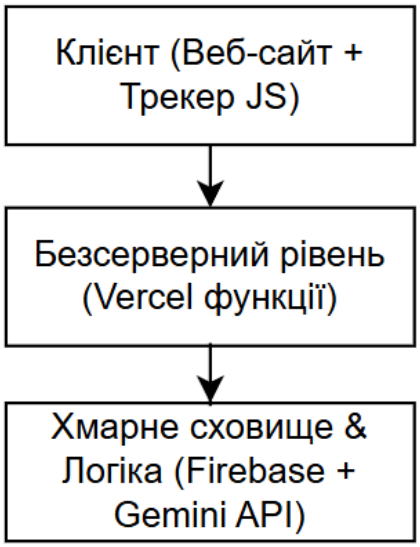


Рисунок 2.2 – Архітектурна схема системи

Ескізний проект інформаційної системи передбачає інтеграцію комплексного механізму безпечного доступу та розмежування прав користувачів. Процедура автентифікації й авторизації адміністратора реалізується на базі сучасного галузевого протоколу OAuth 2.0 [9]. Делегування цих функцій довіреним провайдерам ідентифікації (Identity Providers) уможливлює розгортання надійної системи контролю доступу без необхідності проектування та підтримки власної локальної інфраструктури збереження й шифрування паролів. Таке архітектурне рішення суттєво нівелює ризики компрометації облікових даних і підвищує загальний рівень кібербезпеки програмно-техничного комплексу.

Для гарантування конфіденційності та автентичності інформаційних потоків усі компоненти розроблюваної системи взаємодіють виключно через захищені інтерфейси прикладного програмування (Secure APIs) з обов'язковим шифруванням трафіку за допомогою протоколу TLS. Це забезпечує наскрізну цілісність відомостей на всьому життєвому циклі їхнього руху – від моменту генерації події в клієнтському браузері (інтерфейсний рівень) до агрегації та фінальної візуалізації на аналітичній платформі.

Основними перевагами EDA для систем аналізу активності є:

1. Слабке зчеплення: Модуль збору даних нічого не знає про те, як дані будуть оброблятися на сервері. Це дозволяє змінювати логіку аналітики, не торкаючись скрипта-трекера.

2. Масштабованість: Оскільки події обробляються асинхронно, система може витримувати величезні піки трафіку, чергуючи події для подальшого запису в базу даних.

3. Реактивність у реальному часі: EDA дозволяє миттєво реагувати на критичні події (наприклад, падіння трафіку), ініціюючи сповіщення або оновлюючи дашборди без затримки.

Однак існують і недоліки, такі як складність відстеження ланцюжка подій та потенційна проблема узгодженості даних у кінцевому підсумку, що вимагає ретельного проєктування черг та механізмів підтвердження доставки.

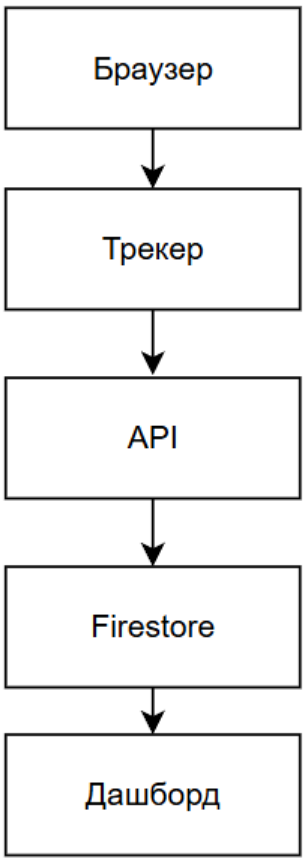


Схема 2.2 – Схема послідовності

2.3 Аналіз та вибір методів реалізації програмних підсистем

Обґрунтування та вибір технологічного інструментарію є важливим етапом проектування, що безпосередньо впливає на швидкість обробки інформаційних потоків. У процесі аналізу методів комп'ютерної інженерії та системного оцінювання моделей збереження даних проведено компаративний аналіз традиційних реляційних систем керування базами даних (СКБД) та рішень класу NoSQL. Результати цього зіставлення систематизовано в таблиці 2.1.

Таблиця 2.1 - Порівняння NoSQL та SQL.

Параметр	SQL	NOSQL
Структура даних	Сувора. Дані зберігаються у таблицях із заздалегідь визначеними колонками та типами.	Гнучка. Можуть зберігатися документи (JSON), пари ключ-значення, графи або колонки.
Схема	Фіксована. Будь-які зміни в структурі потребують модифікації всієї бази	Динамічна. Можна додавати нові поля "на льоту" без зупинки системи.
Масштабованість	Вертикальна. Потрібно нарощувати потужність одного сервера (CPU, RAM).	Горизонтальна. Можна легко додавати нові дешеві сервери в кластер.

Кінець таблиці 2.1

Цілісність даних	Максимальна (ACID). Пріоритет на точність, несуперечність та фінансову надійність.	Висока швидкість (BASE). Пріоритет на доступність та швидкість роботи під великим навантаженням.
Запити та зв'язки	Складні. Ідеально для вибірок із багатьма зв’язками (JOIN) та глибокої аналітики.	Прості. Оптимізовані для швидкого отримання конкретних об’єктів за ключем.
Обробка запитів	Стандартна мова SQL. Універсальна та зрозуміла для більшості систем.	Кожна БД має свій унікальний API або синтаксис для роботи з даними.

Для розробки людино-машинного інтерфейсу я вирішив використати компонентний підхід на основі бібліотеки React [10]. Це важливо, бо потрібно забезпечити високу швидкість відгуку аналітичного дашборду, особливо коли багато інтерактивних віджетів працюють одночасно. Тут нам на допомогу приходить технологія Virtual DOM та алгоритм реконсиляції. Цей механізм дозволяє оновлювати лише ті частини інтерфейсу, які дійсно зазнали змін. Це критично для стабільної роботи теплових карт у реальному часі та динамічних графіків активності користувачів. Також я впровадив TypeScript [11] для реалізації клієнтської і серверної логіки. Це, безумовно, допомагає уникнути багатьох помилок під час виконання програми і гарантує, що дані завжди будуть консистентними при комунікації між серверами. А ще це спрощує рефакторинг коду в подальшому. Окремо звернули увагу на інтеграцію методів інтелектуального аналізу даних. Для автоматичної інтерпретації та генерації текстових висновків на основі зібраних метрик я підключив API великих мовних

моделей, зокрема моделі Gemini [12]. Замість того, щоб витрачати час і ресурси на розробку власної нейромережевої архітектури, ми можемо реалізувати складний когнітивний аналіз часових рядів через методологію конструювання промптів. Архітектура нашого інтелектуального модуля ґрунтується на динамічному формуванні контекстних вікон. Система автоматично агрегує релевантні статистичні дані та паттерни поведінки користувачів перед тим, як передати запит до LLM-шлюзу. Завдяки цьому підходу ми забезпечуємо високу точність і контекстуальну валідність висновків штучного інтелекту.

Таблиця 2.2 наглядно показує вибір рішень для різних підсистем та їхні переваги.

Таблиця 2.2 – Вибір рішень для різних підсистем

Категорія підсистем	Проєктне рішення	Аргументація вибору
Обробка та візуалізація	React(TypeScript)	Висока реактивність та строга типізація моделей даних
Серверне середовище	Node.js / Express	Асинхронна обробка великої кількості HTTP-з'єднань
Сховище даних	NoSQL(Firestore)	Можливість динамічної зміни схеми даних без простоїв
Візуальний аналіз	Recharts	Оптимізація рендерингу складних часових рядів
Когнітивний аналіз	Gemini AI API	Автоматизація інтерпритації статистичних показників

2.4 Проєктування логічної та фізичної структури бази даних

Розробка ефективної стратегії для зберігання аналітичних даних – це, безумовно, одне з найскладніших завдань у сфері архітектури. Коли ми працюємо над логічною структурою сховища, зрозуміло, що традиційна нормалізація реляційних моделей не завжди підходить. Тому ми вирішили перейти до денормалізованої структури з "пласкими" документами.

У нашій системі логічний контур даних базується на трьох основних сутностях: подія користувацької активності (AnalyticsEvent), журнал аудиту дій адміністратора (AuditLog) та конфігурація тригерів сповіщень (AlertConfig). Кожна подія тут виступає як самостійний об’єкт, який містить увесь контекст стану системи на момент, коли користувач взаємодіє з веб-інтерфейсом.

Коли справа доходить до фізичної реалізації бази даних на платформі Google Cloud Firestore [13], ми використовуємо ієрархічну модель колекцій і підколекцій. Основна колекція events оптимізована для великої кількості записів. Кожен документ має набір обов'язкових атрибутів: унікальний проектний ідентифікатор (projectId), маркер типу події (клік чи перегляд сторінки), відносний шлях ресурсу (path), повну URL-адресу та рядок технічних характеристик клієнтського оглядача (userAgent).

Аналіз географічного розподілу аудиторії – це ще одна важлива задача. Щоб зберегти конфіденційність і відповідати вимогам, ми додаємо опціональні поля для географічних координат (lat, lng) та ISO-коду країни в структуру документа. Ці атрибути заповнюються лише на серверному рівні після перевірки мережевих заголовків HTTP-запиту. Завдяки цьому ми можемо уникнути збереження особистих IP-адрес користувачів у сховищі.

Не менш важливим етапом є розробка стратегії індексації. Оскільки аналітичні вибірки часто вимагають агрегації та фільтрації за часом і метриками, наша база даних підтримує складені індекси. Наприклад, індекс за полями projectId, eventType та timestamp дозволяє виконувати складні запити для

побудови графіків активності в лічені секунди, незалежно від обсягу даних. Цей підхід не лише забезпечує високу продуктивність, а й дає нам гнучкість у розширенні схеми даних і додаванні нових типів подій без якихось перерв чи складних міграцій.

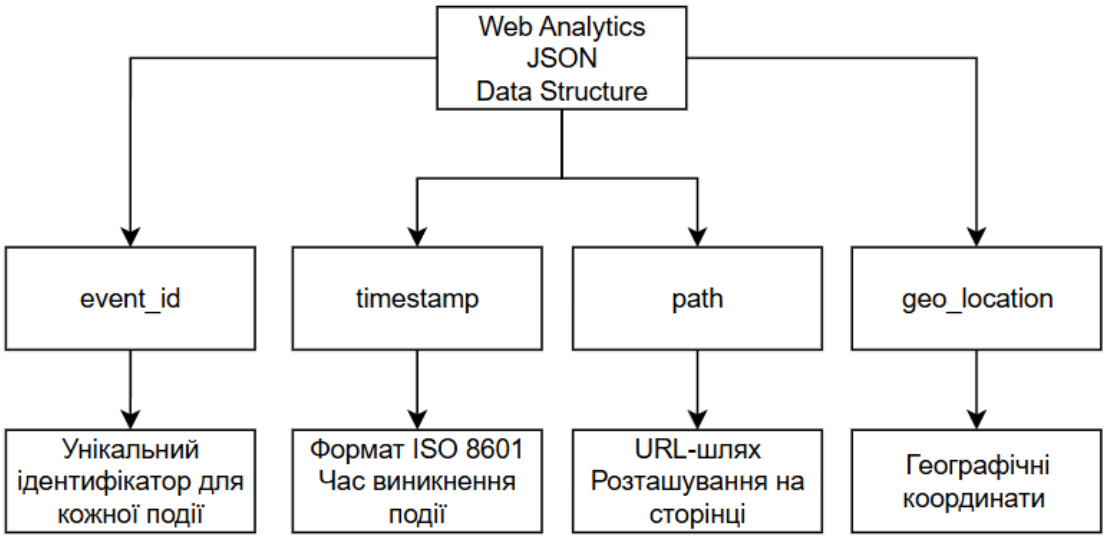


Рисунок 2.3 – Схема структури бази даних

2.5 Розробка алгоритмів функціонування та математичного забезпечення системи

Центральним завданням етапу проєктування є опис алгоритмічної логіки. Як реалізувати трансформацію сирих даних у візуальні звіти? Основний алгоритм обробки часових рядів базується на методі динамічної агрегації (Time-series aggregation). Для реалізації графіків активності за різні періоди (24 години, 7 днів, 30 днів) спроєктовано алгоритм ключової ініціалізації масивів.

Процес реалізації полягає у створенні об’єкта-карти, де ключами є формовані мітки часу (наприклад, години або дні), а значеннями – лічильники подій. Це дозволяє реалізувати лінійну складність алгоритму, що є критично важливим для обробки великих масивів даних безпосередньо у браузері адміністратора.

Особливої уваги заслуговує алгоритм моніторингу в реальному часі (Real-time tracking). Як реалізувати плавне оновлення графіка активності за останні 2 хвилини? Для цього спроєктовано алгоритм часових бакетів (Bucketing algorithm). Система розбиває 120-секундне вікно на 12 рівних інтервалів по 10 секунд кожен. При отриманні масиву подій алгоритм виконує ітерацію, порівнюючи мітку часу кожної події з межами кожного бакета.

Математично цей процес можна описати як операцію квантування сигналу, де кожна подія e з часом t_e потрапляє в бакет B_i, якщо виконується умова:

$$T_{\{current\}} - (i + 1) \cdot \Delta t < t_e \leq T_{\{current\}} - i \cdot \Delta t,$$

де Δt – крок дискретизації (10 с).

Така реалізація дозволяє забезпечити ефект «живого» графіка, який зміщується разом із плином часу.

Окремим алгоритмічним модулем є конструктор вільних звітів. Це завдання вирішується через алгоритм динамічного групування. Проєктування передбачає використання структури Map для накопичення унікальних значень обраних вимірів та підрахунку відповідних метрик. У випадку підрахунку унікальних користувачів алгоритм використовує структуру Set для зберігання ідентифікаторів сесій, що гарантує математичну точність показника U(t) (потужність множини унікальних сесій), про який йшлося у математичному обґрунтуванні системи.

2.6 Проєктування людино-машинного інтерфейсу та UX-моделі

Проектування інтерфейсу користувача (рисунок 2.4) в аналітичних системах спрямоване на розв'язання проблеми інформаційного перевантаження оператора. Ця задача вирішується шляхом впровадження модульної архітектури дашборду, де проектне рішення базується на концепції «віджетної сітки» (grid layout), а кожен аналітичний показник ізольований у межах автономного графічного компонента.

Реалізація такої моделі передбачає використання стану activeTab, що дозволяє логічно розмежувати ключові функції системи за тематичними напрямами: загальний огляд, детальні звіти, теплова карта реального часу та модуль інтелектуального аналізу.

Важливим інженерним кроком для підвищення ергономічності робочого простору адміністратора є механізм кастомізації звітів за допомогою технології перетягування (Drag-and-Drop). Це забезпечує гнучкість інтерфейсу, дозволяючи оператору самостійно визначати пріоритетність та розташування метрик. Окреслена архітектура повністю відповідає сучасним стандартам адаптивності (Responsive Web Design): сітка дашборду проектується з урахуванням автоматичної зміни кількості колонок залежно від роздільної здатності екрана та типу пристрою користувача.

Ефективна людино-машинна взаємодія також підтримується за рахунок інтеграції системи візуальних сповіщень та інтерактивної чат-панелі. Проектування інтерфейсу ШІ-асистента передбачає потокове виведення тексту (streaming розв'язання) та підтримку розмітки Markdown для покращення сприйняття складних аналітичних висновків.

Окрім цього, архітектура інтерфейсу закладає підтримку двохколірних схем (світлої та темної). Таке рішення диктується вимогами до ергономіки, оскільки тривала робота з моніторингом даних за різного рівня освітлення робочого місця вимагає мінімізації зорового навантаження на оператора системи.

Для забезпечення стабільної роботи такого інтерфейсу, реалізовано механізм реактивного оновлення стану компонентів. Це дозволяє миттєво відображати нові показники в реальному часі без повного перезавантаження сторінки. Такий підхід гарантує плавність роботи системи, і її швидку реакцію на дії користувача. Це відчутно покращує загальну роботу системи і оптимізовує її для комфортного функцінування.

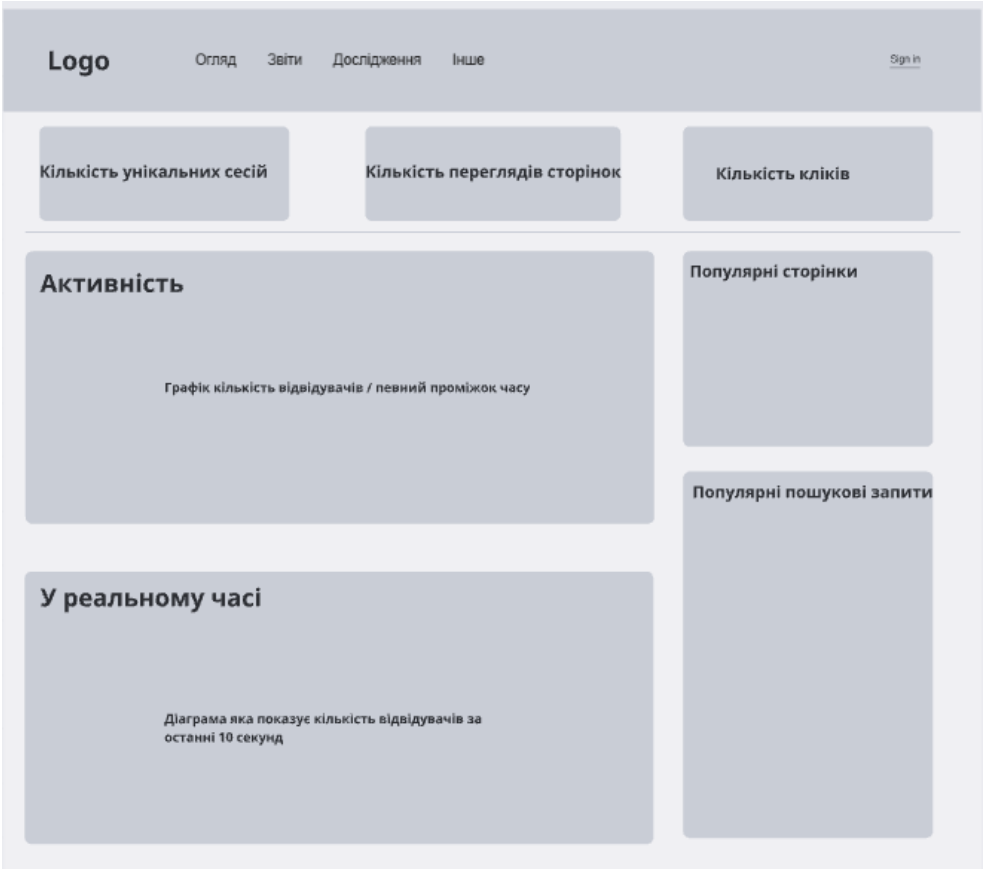


Рисунок 2.4 – Wireframe приблизного вигляду головної сторінки аналітики

2.7 Проєктування модуля інтелектуального аналізу та когнітивної обробки метрик

Архітектурне проектування переходу від описової статистики до предиктивної та когнітивної аналітики спрямоване на автоматизацію інтерпретації великих масивів інформації. Проектне рішення базується на імплементації інтелектуального агента, що забезпечує трансформацію числових метрик у структуровані висновки природною мовою. Реалізація цього модуля передбачає побудову дворівневої системи обробки контексту. На першому (інфраструктурному) рівні здійснюється агрегація сирих подій телеметрії у валідні JSON-об’єкти, які містять ключові показники ефективності (KPI): сумарну кількість сесій, глибину переглядів, домінантні поведінкові маршрути (user flows) та верифіковані аномальні сплески активності аудиторії.

Другий рівень інтелектуального аналізу охоплює проектування механізму динамічної ін’єкції контексту (Context Injection) у структуру запиту до великої мовної моделі. Задля забезпечення високої точності й релевантності висновків штучного інтелекту розроблено методологію формування системних інструкцій (System Prompts). Архітектура системи автоматично аппендить до кожного інформаційного пакету рольову специфікацію експерта з веб-аналітики та дата-саєнтиста, що містить жорсткі предикати фільтрації та інтерпретації даних. Таке інженерне рішення дозволяє моделі Gemini здійснювати комплексний компаративний аналіз, детермінувати причинно-наслідкові зв’язки між мікроконтекстом дій користувачів та макропоказниками ефективності веб-сервісу. Окремим архітектурним апектом є проектування сесійної пам’яті (Chat History Buffer), що уможливлює ведення послідовного контекстно-залежного діалогу без надлишкової ретрансляції всього масиву історичних даних при кожному новому запиті.

Інтерфейс взаємодії з когнітивним модулем проектується за принципом інтерактивного асистента (AI Assistant). Для оптимізації сприйняття результатів аналізу текстові звіти рендеряться з використанням розмітки Markdown, що дозволяє реалізувати чітке візуальне структурування: акцентування на критичних системних проблемах, маркування позитивних трендів та динамічних змін. Це суттєво знижує когнітивне навантаження на адміністратора й прискорює прийняття управлінських рішень на основі даних, зібраних у реальному часі.

Загальний алгоритм функціонування інтелектуальної системи аналітики (рисунок 2.5) реалізує закритий цикл обробки: ініціація запиту активує процедуру селекції та агрегації релевантних метрик із бази даних, після чого сформований датасет передається до аналітичного шлюзу, де на основі зумовлених системних інструкцій бізнес-аналітика LLM-модель генерує фінальний людино-читаємий експертний звіт і транслює його на клієнтський інтерфейс користувача.

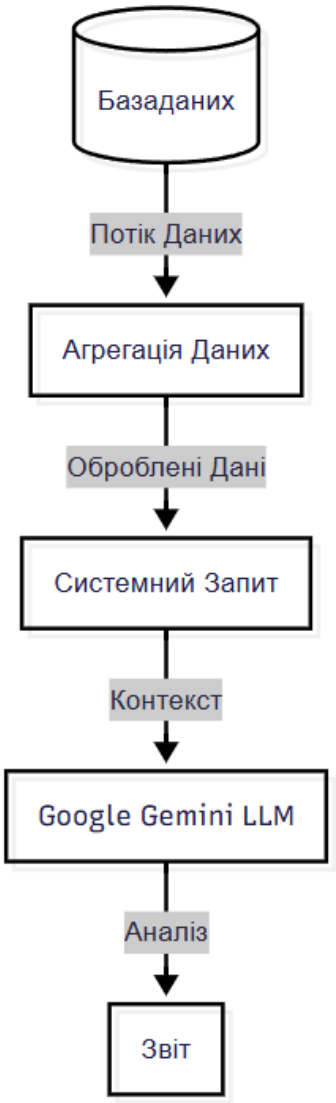


Рисунок 2.5 – Загальний алгоритм функціонування інтелектуальної системи аналітики

2.8 Проєктування системи захисту інформації та розмежування прав доступу

Розробка системи аналітики, яка працює з даними про поведінку користувачів, потребує першочергової уваги до інформаційної безпеки та захисту приватності. Тут важливо дотримуватися принципу ешелонованої оборони. Перший рівень захисту реалізується на транспортному рівні через обов'язкове використання криптографічних протоколів шифрування TLS 1.3. Це фактично виключає можливість перехоплення пакетів і атак типу «людина

посередині» під час передачі даних від клієнтського трекера до аналітичного шлюзу. Контроль доступу до адміністративної панелі побудований на рольовій моделі розмежування повноважень (RBAC). Для аунтифікації я обрав стандарт OpenID Connect (OIDC), який дозволяє легко інтегрувати систему з надійними корпоративними постачальниками ідентифікації. Проєкт передбачає обов'язкову верифікацію кожного запиту до бази даних за ідентифікатором проекту. Захист інформації на рівні бази даних забезпечується через впровадження декларативних правил безпеки Firestore. Ці правила блокують будь-які спроби несанкціонованого доступу, зміни або видалення записів прямо на ядрі системи. Кожен документ у колекції подій містить атрибут власника, що дає можливість реалізувати строгий розподіл даних між різними клієнтами у спільній фізичній інфраструктурі хмарного сховища. Не менш важливим елементом нашої системи є проектування механізму журналювання аудиту. Він забезпечує прозорість дій та незаперечність операцій у системі. Усі критичні транзакції – від автентифікації оператора до запиту на запуск модуля аналізу – фіксуються в незмінному журналі подій. Логи включають електронну пошту ініціатора, точний час та детальний опис операції. Такий підхід допомагає не лише зменшити ризики внутрішніх загроз кібербезпеки, але й забезпечити необхідні інструменти для розслідування інцидентів і перевірки відповідності стандартам. Отже, ця комплексна архітектура захисту охоплює всі рівні - від клієнтського скрипта до хмарного сховища та інтелектуального інтерфейсу, що робить систему максимально безпечною та ефективною.

2.9 Методологія інтеграції та життєвий цикл клієнтського модуля

Останній етап проектування – це розробка методології для впровадження інформаційної системи на різні веб-ресурси. Ми плануємо створити автономний інструмент для розробників, або SDK. Архітектурно ми вирішили зробити асинхронний завантажувач (Loader). Його ініціалізація відбувається через

простий рядок коду в HTML цільового сайту. Цей модуль працює на основі селекторів data-атрибутів, що дозволяє динамічно передавати унікальний ідентифікатор проекту (projectId). Завдяки цьому, ми уникаємо хардкодингу констант у скрипті, що значно підвищує гнучкість і спрощує інтеграцію клієнтського трекера через системи управління тегами. Наш клієнтський SDK базується на подієво-орієнтованій моделі. Як тільки скрипт завантажується, він переходить у стан очікування ініціалізації об'єктної моделі документа.

Щоб оптимізувати обчислювальну складність і зменшити навантаження на процесор, ми використовуємо архітектурний патерн делегування подій на глобальному об'єкті window. Це означає, що нам не потрібно створювати тисячі окремих слухачів подій для кожного інтерактивного елемента інтерфейсу. Натомість, є один глобальний обробник, який аналізує, куди поширюється подія, і визначає цільові DOM-вузли. Таке рішення забезпечує надійність та стабільну роботу системи навіть під навантаженням з динамічно генерованим контентом. Щоб система витримувала мережеві збої та гарантувала консистентність даних, ми розробили механізм локальної буферизації. Якщо виникають проблеми зі зв'язком з аналітичним шлюзом, наш модуль накопичує події в черзі локального сховища та повторно надсилає пакети після відновлення мережі. Це допомагає зменшити ризики втрати даних і забезпечити безперервний моніторинг активності навіть за нестабільних умов.

Фінальним акордом у проектуванні життєвого циклу SDK є формалізація процедури деактивації трекера. На цьому етапі важливо правильно звільнити ресурси оперативної пам'яті, відписатися від глобальних подій і коректно очистити тимчасові ідентифікатори після закриття сесії користувачем.

2.10 Інфологічне проєктування та розробка ER-діаграми бази даних

Коли ми працюємо над логічною структурою сховища даних для аналітичної системи, важливо створити інфологічну модель, що чітко відображає

основні сутності та їх зв'язки. Якщо говорити про документо-орієнтовані NoSQL-бази, як-от Firebase Firestore, потрібно адаптувати модель «сутність-зв'язок» до особливостей денормалізації і гнучкості схем документів. Основні елементи системи включають кілька сутностей: Адміністратор (той, хто користується аналітичною панеллю), подія (конкретна взаємодія відвідувача на сайті), журнал аудиту (незмінний запис про дії в системі) і сповіщення (налаштування повідомлень). Кожна з цих сутностей має свій набір атрибутів, які допомагають зберігати повний контекст інформаційних взаємодій. Сутність адміністратор – це ключовий елемент для ідентифікації в нашій системі. Вона включає атрибути, такі як унікальний ідентифікатор користувача (uid), підтверджена електронна пошта (email) та ім'я профілю. Зв'язок між адміністратором і подією є логічним і має тип «один до багатьох» (1:M), який реалізується через ідентифікатор проекту (projectId). Це означає, що один зареєстрований адміністратор може володіти унікальним цифровим токеном проекту, на який надходять мільйони окремих аналітичних подій від відвідувачів сторонніх вебресурсів. Кожна подія, у свою чергу, містить атрибути типу дії (eventType), відносного шляху сторінки (path), часової мітки (timestamp), технічних метаданих пристрою (userAgent) та опціональних геопросторових координат (lat, lng). Таке архітектурне рішення дозволяє проводити високоефективну агрегацію часових рядів без виконання ресурсомістких реляційних операцій об'єднання (JOIN).

Аналогічно, зв'язки типу «один-до-багатьох» (1:M) закладаються між сутністю адміністратор та сутностями журнал аудиту й сповіщення. Конфігурація сповіщень прив'язується до облікового запису через projectId, що уможливлює динамічну валідацію активності тригерів у реальному часі. Журнал аудиту пов'язується з адміністратором через атрибут електронної пошти, що гарантує суворе логування кожної критичної операції (як-от генерація когнітивного звіту за допомогою ШІ або зміна конфігурації інтерфейсної сітки дашборду) за конкретною відповідальною особою. Проект розробленої

інфологічної моделі забезпечує консистентність і цілісність даних на логічному рівні, повністю нівелюючи ризики надлишкового навантаження на хмарну інфраструктуру в умовах інтенсивного запису потокових даних телеметрії.

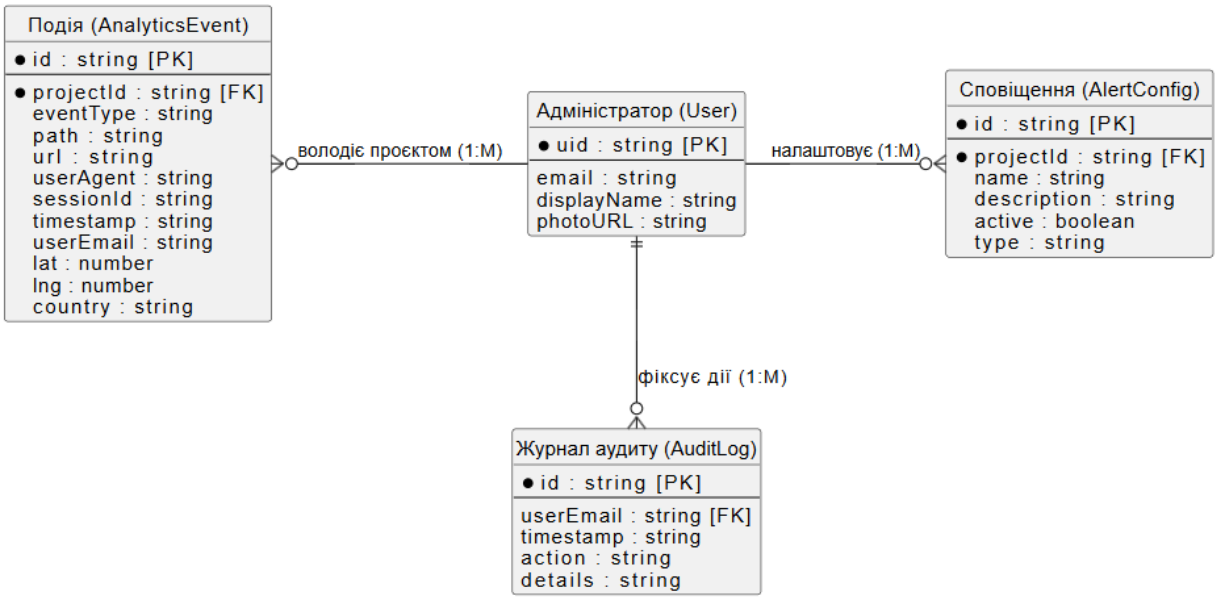


Рисунок 2.6 – ER-діаграма

2.11 Висновки до другого розділу

У другому розділі виконано повний цикл архітектурного та системного проектування програмно-технічного комплексу для моніторингу й аналізу користувацької активності веб-сервісів. На основі комплексного аналізу об’єкта розробки формалізовано технічні вимоги та детерміновано оптимальні інженерні патерни реалізації системи. Обґрунтовано доцільність впровадження подієво-орієнтованої архітектури (EDA) та хмарних безсерверних технологій (Serverless), які демонструють найвищу ефективність за критеріями горизонтального масштабування, відмовостійкості та мінімізації затримок при високій інтенсивності записів. В межах проектування даних розроблено логічну та фізичну структури денормалізованого NoSQL-сховища, а також формалізовано алгоритмічне забезпечення для потокової обробки часових рядів у реальному часі на базі методу динамічних часових бакетів.

Проектування людино-машинного інтерфейсу та когнітивного модуля дозволило сформувати цілісну концепцію програмного продукту, який не лише агрегує сиру телеметрію, а й забезпечує її автоматичну експертну інтерпретацію природною мовою. Окремий вектор роботи присвячено побудові багаторівневого контуру безпеки, що включає шифрування транспортного рівня, рольове розмежування доступу та декларативні правила ізоляції даних. Запропоновані проектні рішення, що спираються на сучасний технологічний стек (React, TypeScript, Firebase Firestore, Gemini API), формують стабільну теоретико-методологічну й технічну базу для наступного етапу роботи – безпосередньої програмної реалізації, верифікації та тестування інформаційної системи.

3 ПРОГРАМНО-АПАРАТНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ПРОГРАМНО-ТЕХНІЧНОГО ЗАСОБУ

3.1 Опис середовища розробки та програмної реалізації модулів системи

Розробка комплексу «UWebAnalytics» виконувалась на базі сучасного технологічного стеку, який ідеально підходить для високонавантажених веб-застосунків і хмарних обчислень. Для створення клієнтського інтерфейсу та адміністративної панелі ми обрали TypeScript у поєднанні з бібліотекою React 18. Використання TypeScript дозволило нам забезпечити строгість типізації всіх об’єктів і даних, які обмінюються між різними підсистемами. Це дуже важливо, адже допомагає уникнути логічних помилок під час десеріалізації вхідних JSON-пакетів. Серверна частина системи працює на Node.js [14], і ми використовуємо безсерверні функції, що забезпечують автоматичне масштабування під час пікових навантажень. Що стосується архітектури програми, то вона модульна: кожен компонент виконує свою чітко визначену роль. Наприклад, модуль для інтерактивного захоплення подій – це файл client-script.js, який займається перехопленням транзакцій у DOM цільового вебресурсу. Його ініціалізація відбувається за допомогою унікального токена проєкту, а асинхронне неблокуюче відправлення телеметрії на серверний шлюз – це ще одна його перевага. Центральним елементом управління станом програми є модуль App.tsx. Він координує процеси автентифікації користувачів, менеджмент сесій і інтеграцію графічних компонентів для візуалізації аналітичних даних. А серверний модуль відповідає за валідацію вхідних HTTP POST-запитів на базі Express [15], фільтрацію аномалій та трансляцію структурованих записів у хмарну NoSQL-базу даних Firestore за допомогою офіційного SDK.

Окремим вагомим інженерним компонентом є модуль даних chat.ts, який забезпечує програмний інтерфейс (шлюз) до генеративних моделей штучного інтелекту Google Gemini. Реалізація цього модуля передбачає динамічне конструювання складних системних інструкцій, які безпосередньо в момент

запиту збагачуються актуальними статистичними масивами. Процес передачі контексту оптимізований за принципом мінімальної надлишковості: архітектура відсікає нерелевантні сирі логи, транслюючи до моделі лише агреговані метрики й часові зрізи. Такий підхід суттєво оптимізує споживання лімітів контекстного вікна та мінімізує час очікування під час генерації когнітивних звітів.

Усі розроблені модулі глибоко інтегровані в єдину систему за допомогою асинхронних викликів та реактивних механізмів управління станом на базі кастомних і вбудованих хуків React (React Hooks). Це гарантує наскрізну консистентність даних у процесі взаємодії підсистем та забезпечує високу швидкість відгуку інтерфейсу користувача.

3.2 Реалізація людино-машинного інтерфейсу та засобів візуалізації

Людино-машинний інтерфейс (рис. 3.1) нашого засобу створено з акцентом на те, щоб він був максимально інформативним і інтерактивним. Дашборд реалізовано на основі модульної сітки (Grid Layout), де кожен елемент візуалізації працює як окремий і самостійний компонент. Для створення лінійних і стовпчикових діаграм ми використовуємо спеціальну бібліотеку Recharts [16]. Цікаво, що однією з архітектурних особливостей цієї реалізації є застосування обгортки ResponsiveContainer, що дозволяє графікам адаптуватися до зміни роздільної здатності екрана, а це дуже зручно для операторів. На клієнтському рівні ця функція займається обробкою масивів телеметрії, вона групує та агрегує дані за визначеними часовими інтервалами, такими як 24 години або 7 днів, спираючись на парсинг і нормалізацію міток часу у форматі ISO 8601.

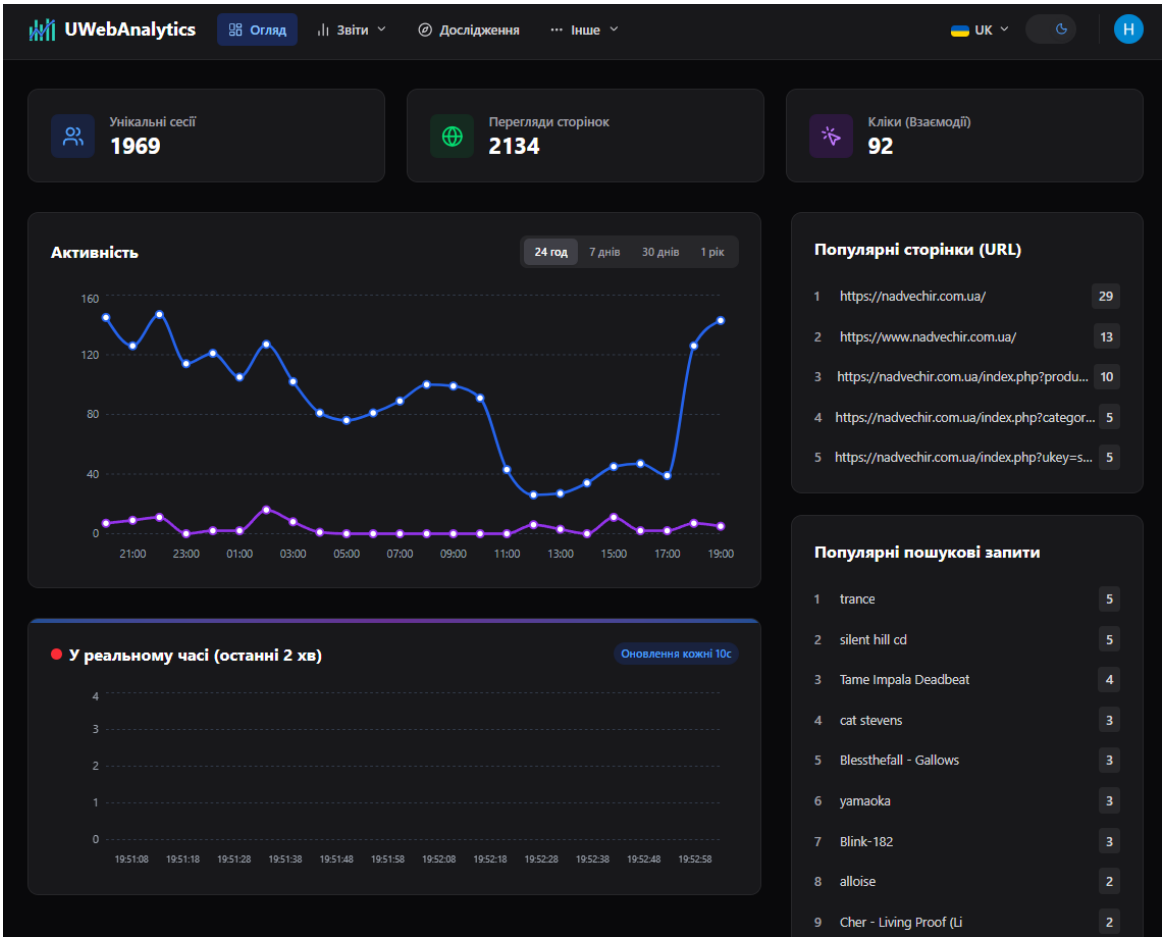


Рисунок 3.1 – Головна сторінка

Важливим компонентом є карта яка працює у реальному часі (рис 3.2), реалізована за допомогою бібліотек react-simple-maps та date-fns. Програмна логіка карти передбачає рендеринг двох типів маркерів: статичних для подій за останні 24 години та анімованих для активних сесій, що виникли протягом останніх 30 хвилин. Реалізація механізму ZoomableGroup дозволяє оператору детально вивчати активність у конкретних регіонах світу. Для підвищення зручності користування впроваджено підтримку багатомовності через пакет react-i18next. Реалізація функції getFlagEmoji дозволяє системі динамічно підвантажувати прапори країн на основі кодів ISO 3166-1 alpha-2, що значно покращує візуальне сприйняття географічної статистики.

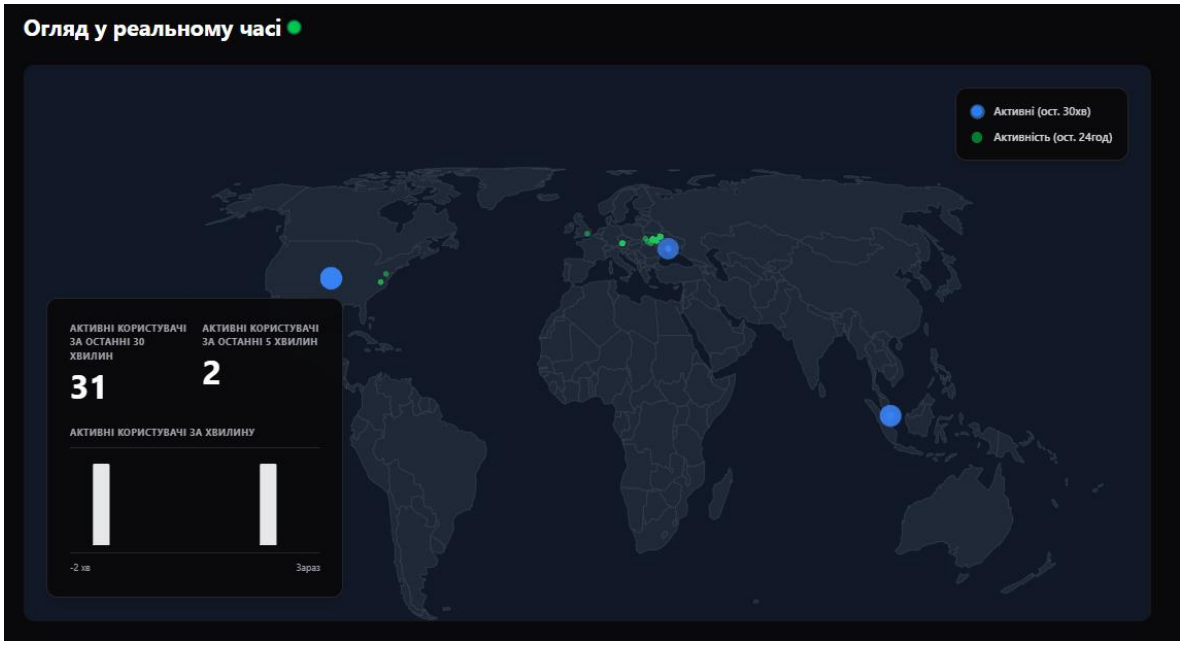


Рисунок 3.2 – Огляд у реальному часі на інтерактивній карті

Фрагмент псевдо коду з впровадженням функціх processRealTimeData, яка готує дані для гістограми реального часу:

```
ALGORITHM processRealTimeData(events, now)
    // Ініціалізація констант та змінних
    bucketCount ← 12          // кількість інтервалів
    bucketSize ← 10000      // розмір інтервалів в мілісекундах
(10 секунд)
    currentTimestamp ← getTimestampInMs(now)
    buckets ← createEmptyList()
    // Крок 1: Генерація часових інтервалів
    FOR i FROM (bucketCount - 1) DOWNTO 0 DO
        bucketEnd ← currentTimestamp - (i * bucketSize)
        bucketStart ← bucketEnd - bucketSize
        timeLabel ← formatTime(bucketEnd, "HH:mm:ss")

        // Створення структури інтервалу
        newBucket ← Structure(
            timeLabel: timeLabel,
            start: bucketStart,
```

```
                end: bucketEnd,
                eventsCount: 0
            )
            appendToList(buckets, newBucket)
        END FOR
        // Крок 2: Агрегація даних
        FOR EACH event IN events DO
            IF event.timestamp IS NULL OR EMPTY THEN
                CONTINUE
            END IF
            eventTime ← parseToTimestampInMs(event.timestamp)
            // Пошук часового проміжку
            FOR EACH bucket IN buckets DO
                IF  eventTime  >  bucket.start  AND  eventTime  ≤
bucket.end THEN
                    bucket.eventsCount ← bucket.eventsCount + 1
                    BREAK // Подію знайдено, перехід до наступної
                END IF
            END FOR
        END FOR
        RETURN buckets
    END ALGORITHM
```

Для реалізації механізму кастомізації інтерфейсу з використанням Drag-and-Drop ми взяли за основу спеціалізований інструмент dnd-kit [17]. Цей підхід дає змогу користувачеві легко переставляти аналітичні віджети в робочому просторі в реальному часі. У системі управління станом ми описали конфігурацію відображення через масив об’єктів widgets. Кожен з цих об’єктів має свій унікальний ідентифікатор, індекс позиції та параметр видимості. Коли завершуються перетягування, спрацьовує функція handleDragEnd, яка рахує нові індекси елементів у масиві. Вона оновлює відображення сітки інтерфейсу, а нові координати віджетів автоматично записуються в журнал аудиту, щоб зафіксувати поточну сесію. Це рішення для фронтенд-архітектури забезпечує

чудову гнучкість у керуванні візуальними компонентами й дозволяє швидко налаштувати інформаційну панель відповідно до конкретних аналітичних потреб.

3.3 Реалізація бази даних та підсистеми безпеки

Розгортання та налаштування бази даних у хмарному середовищі Google Cloud Firestore стало справжньою основою для зберігання аналітичних даних. Тут ми створили колекцію подій, таку собі «events», яка ідеально підходить для інтенсивних потоків записів – це найнавантаженіший елемент нашої системи. Щоб швидко фільтрувати й агрегувати метрики в реальному часі, ми налаштували складені індекси. Кожен документ події має свій унікальний ідентифікатор сесії та токен проекту (projectId). Це дозволяє нам забезпечити надійну ізоляцію даних для різних клієнтів прямо на рівні сховища. Щодо програмної взаємодії з базою даних, то ми використовуємо реактивні методи onSnapshot з офіційного SDK. Це дозволяє отримувати оновлення даних у реальному часі, а також миттєво синхронізувати стан інтерфейсу користувача без потреби постійно надсилати HTTP-запити чи перезавантажувати дашборд.

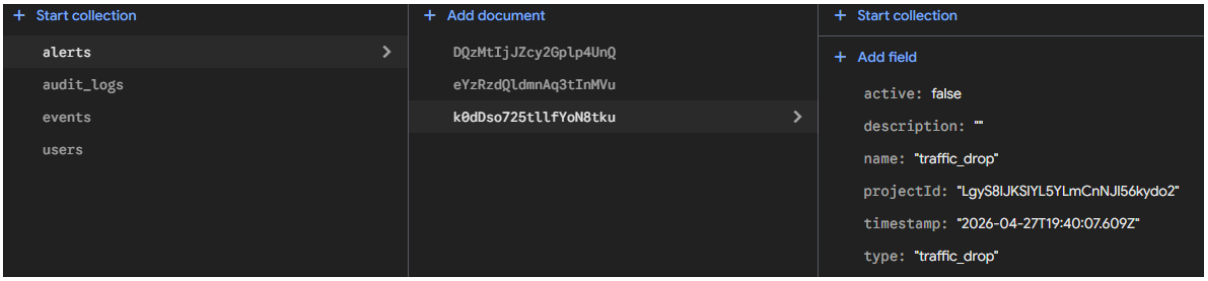


Рисунок 3.3 – Колекції FireStore

Система безпеки нашого програмно-технічного комплексу працює завдяки класному поєднанню Google OAuth 2.0 для автентифікації та Firebase Security Rules [18] для контролю доступу. Коли оператор запускає процедуру входу через функцію loginWithGoogle, після успішної перевірки його даних, система віддає

об'єкт користувача з унікальним маркером доступу. На основі цього ідентифікатора (uid) відбувається динамічна фільтрація інформаційних потоків, так що адміністратор бачить тільки ті аналітичні метрики, які стосуються його проектів. Але це ще не все! Безпека API також на висоті завдяки механізмам перевірки запитів, таким як CORS-політики та верифікація заголовків Origin/Referer. Це допомагає заблокувати несанкціоновану передачу фальшивих або скомпрометованих пакетів телеметрії від сторонніх скриптів або зловмисних джерел, гарантуючи валідність і цілісність аналітичного сховища.

Додатковим рівнем безпеки та проактивного моніторингу слугує журнал аудиту, який я реалізував у хмарному сховищі в окремій колекції audit_logs. У нас є спеціальна функція logAuditAction, що автоматично фіксує всі важливі транзакції та дії, які виконує оператор. Це може бути автентифікація в системі, зміна шаблонів звітів, активація тригерів сповіщень або надсилання запитів до інтелектуального модуля ШІ. Кожен запис у цій колекції містить чітко визначений набір атрибутів: точну мітку часу, електронну пошту адміністратора та детальний опис виконаної операції. Це рішення забезпечує абсолютну прозорість роботи програмно-технічного комплексу. Воно гарантує швидкий аудит дій користувачів із високими привілеями та надає необхідні інструменти для розслідування підозрілих активностей чи інцидентів в нашій інформаційній системі аналітики.

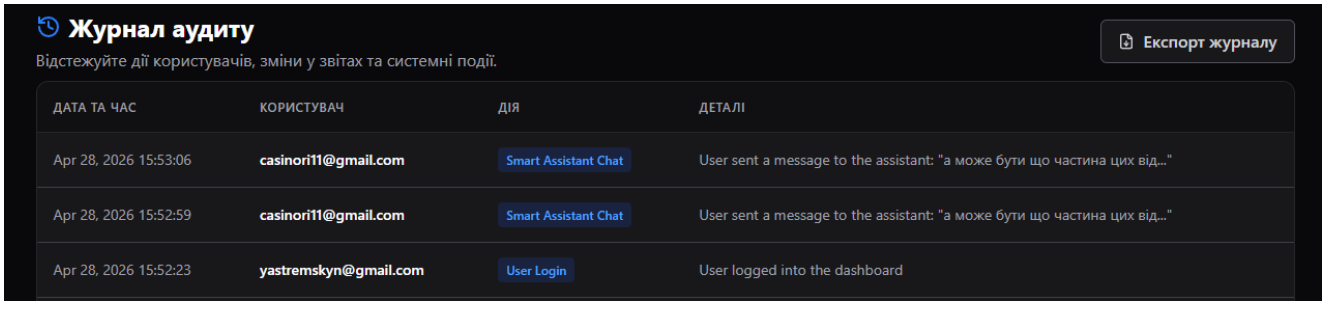


Рисунок 3.4 - Журнал аудиту

3.4 Програмна реалізація модулів звітності та аналітичних досліджень

Реалізація модуля інтелектуального аналізу зосереджена на інтеграції з API Google Gemini через захищений серверний проксі. Одне з головних завдань тут – оптимізувати контекстне вікно, яке передається генеративній моделі. На рівні клієнта функція handleSendMessage займається попередньою агрегацією та фільтрацією даних: вона рахує загальну кількість зафіксованих подій, знаходить п’ять найбільш відвідуваних сторінок і формує статистику щодо клікабельності різних елементів. Зібрані дані перетворюються на структурований текстовий блок і динамічно додаються до системної інструкції, яка визначає, як штучний інтелект має себе вести – немов досвідчений аналітик. Завдяки цій архітектурі ми уникаємо абстрактних відповідей і отримуємо глибокий, контекстно-орієнтований аналіз конкретних показників ефективності вебресурсу.

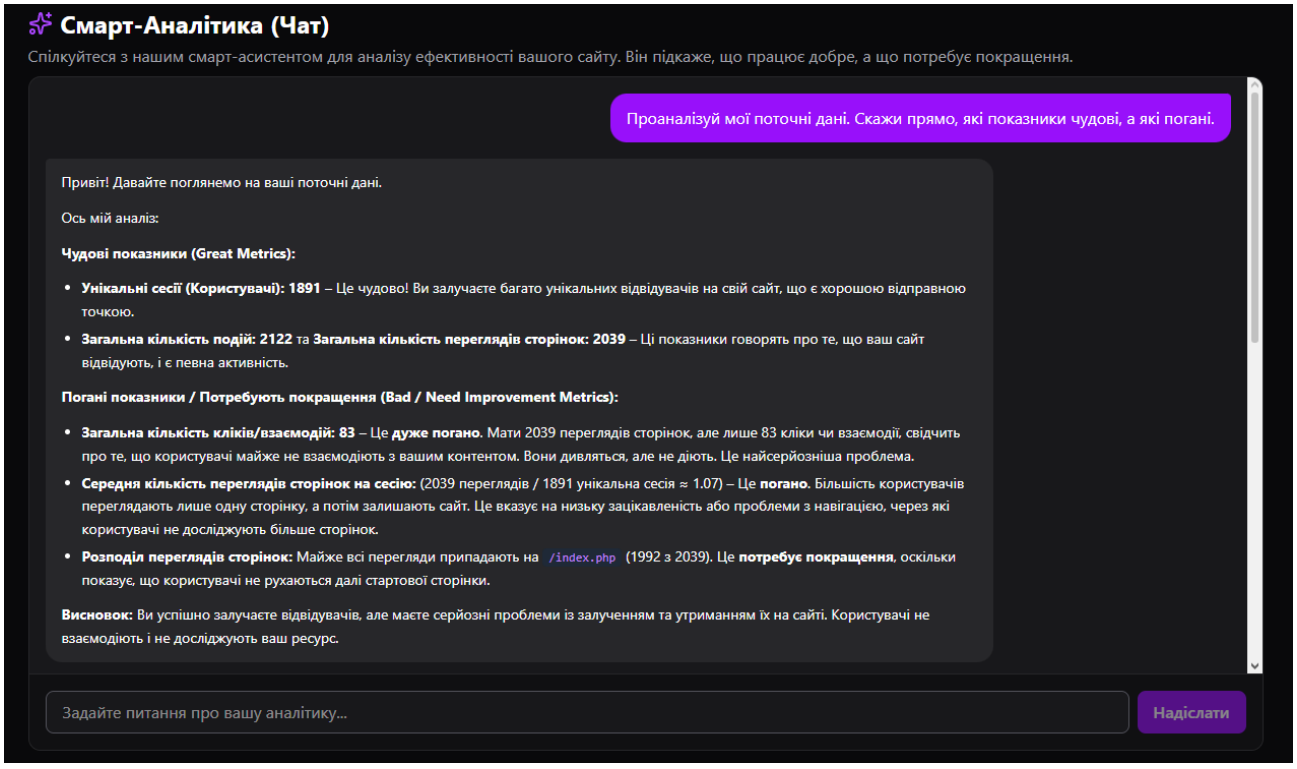


Рисунок 3.5 – Чат з ШІ асистентом (Смарт-Аналітика)

Процес створення системи звітності у програмно-технічному комплексі базується на принципах шаблонного проектування інтерфейсів. Програмна реалізація аналітичного підмодуля передбачає інтеграцію трьох базових конфігурацій (пресетів): поведінкової, маркетингової та комерційної (рис. 3.6). Кожен із цих шаблонів описується як зумовлений набір налаштувань для масиву візуальних компонентів, що дозволяє оператору миттєво змінювати вектор аналізу без необхідності ручної конфігурації окремих діаграм.

На рівні фронтенд-архітектури це реалізовано за допомогою спеціалізованих функцій ініціалізації стану, таких як getInitialBehaviorWidgets, які динамічно формують і повертають структуру сітки дашборду. Кожен віджет у структурі звіту функціонує як незалежний ієрархічний об’єкт, що інкапсулює властивості унікального ідентифікатора, логічного прапорця видимості та відносної ширини в межах адаптивної сітки (Grid Layout). Окреслене інженерне рішення забезпечує високу гнучкість візуалізації, стабільність рендерингу та оптимізацію обчислювальних ресурсів під час опрацювання великих обсягів вхідної інформації (рис. 3.7).

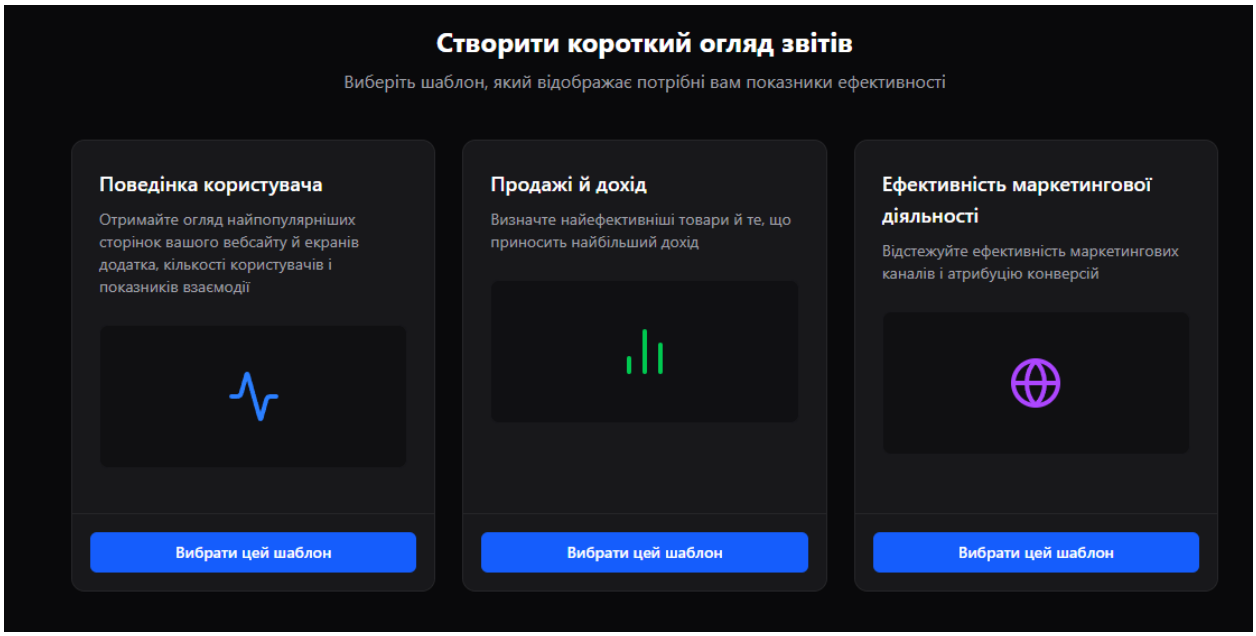


Рисунок 3.6 – Вибір готових шаблонів звіту

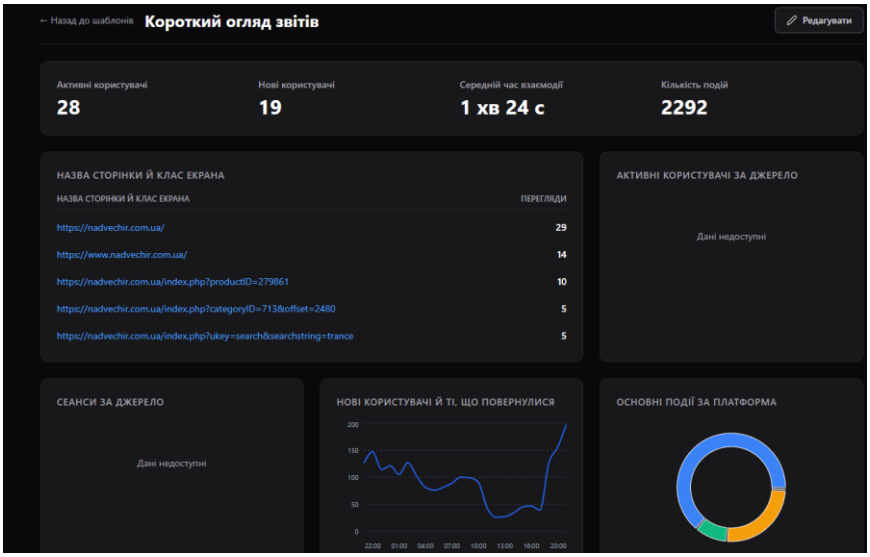


Рисунок 3.7 – Структура звіту

Одним із важливих компонентів нашої системи є модуль аналітичних досліджень, який дозволяє проводити кастомний аналіз даних. Для його реалізації нам довелося розробити оптимізований алгоритм багатовимірної агрегації. Це дає можливість оператору самостійно виявляти кореляції між різними вимірами та метриками. Основна логіка обробки закладена у функції generateExplorationData, яка виконує ітеративний перебір і фільтрує накопичені масиви телеметрії в реальному часі.

Щоб швидко групувати дані за різними полями, ми вирішили використати структуру даних Map. Алгоритм просто проходить через кожен об’єкт події, витягує значення потрібного параметра – наприклад, шлях сторінки чи тип події і пов’язує його з відповідною метрикою. Коли мова йде про підрахунок унікальних користувачів, ми залучаємо структуру Set для кожного ключа. Це дуже зручно, адже вона автоматично відсікає дублікати сесійних ідентифікаторів. Результати досліджень можна візуалізувати через динамічний компонент. Залежно від того, що ви виберете – таблицю, кругову діаграму чи лінійний графік – дані перетворюються в зручний для сприйняття вигляд. Це дозволяє операторам глибше аналізувати поведінку користувачів, знаходити проблеми в веб-інтерфейсах і генерувати нові ідеї для оптимізації сервісів. Увесь

процес обробки даних відбувається на стороні клієнта після завантаження потрібних даних із бази, тому інтерфейс миттєво реагує на зміни фільтрів без повторних запитів до сервера.

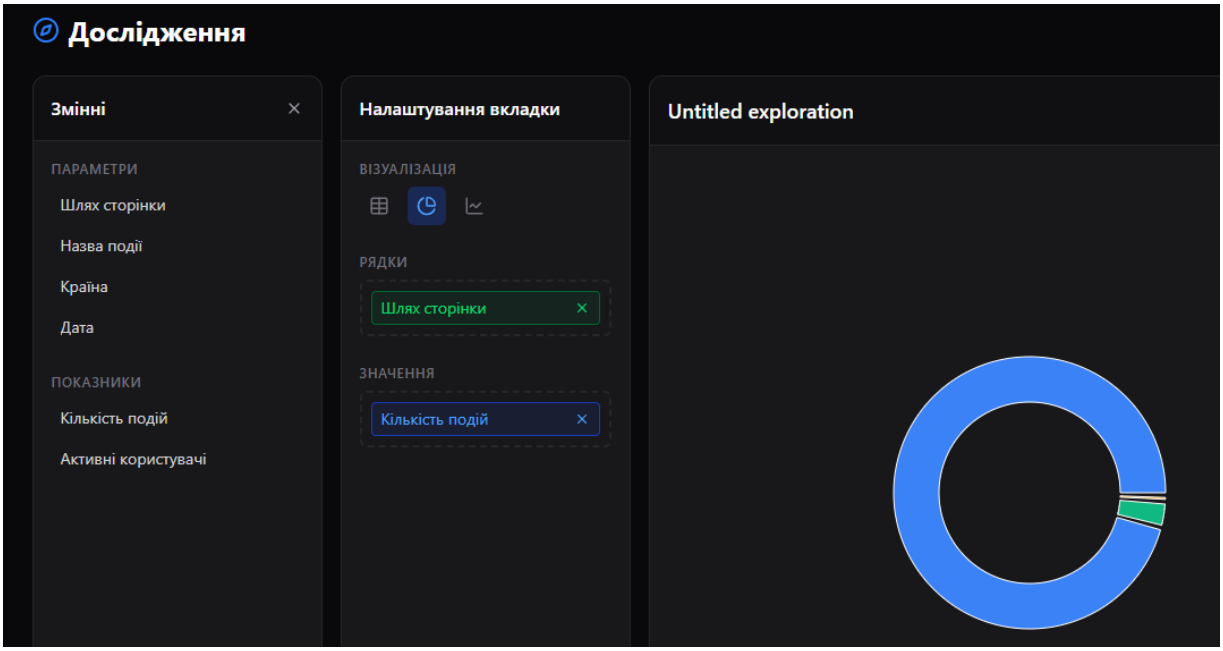


Рис 3.6 – Кругова діаграма

3.5 Реалізація сервісних функцій експорту та системи сповіщень

Для забезпечення повноцінного циклу роботи з аналітичною інформацією у програмно-технічному засобі було реалізовано комплекс додаткових сервісних функцій. Важливим завданням була реалізація механізму експорту звітів у формат PDF, що дозволяє адміністратору зберігати результати аналізу для подальшого використання поза межами системи. Програмна реалізація цієї функції базується на використанні системних викликів друку браузера у поєднанні з адаптивною CSS-верстkою. При ініціації експорту через інтерфейс, система активує специфічні медіа-запити (@media print), які приховують елементи навігації, кнопки управління та чат-панель ШІ, залишаючи лише змістовну частину з графіками та таблицями. Це дозволяє отримати професійно

оформлений документ, що повністю відповідає візуальному стану дашборду на момент експорту (рис 3.7).

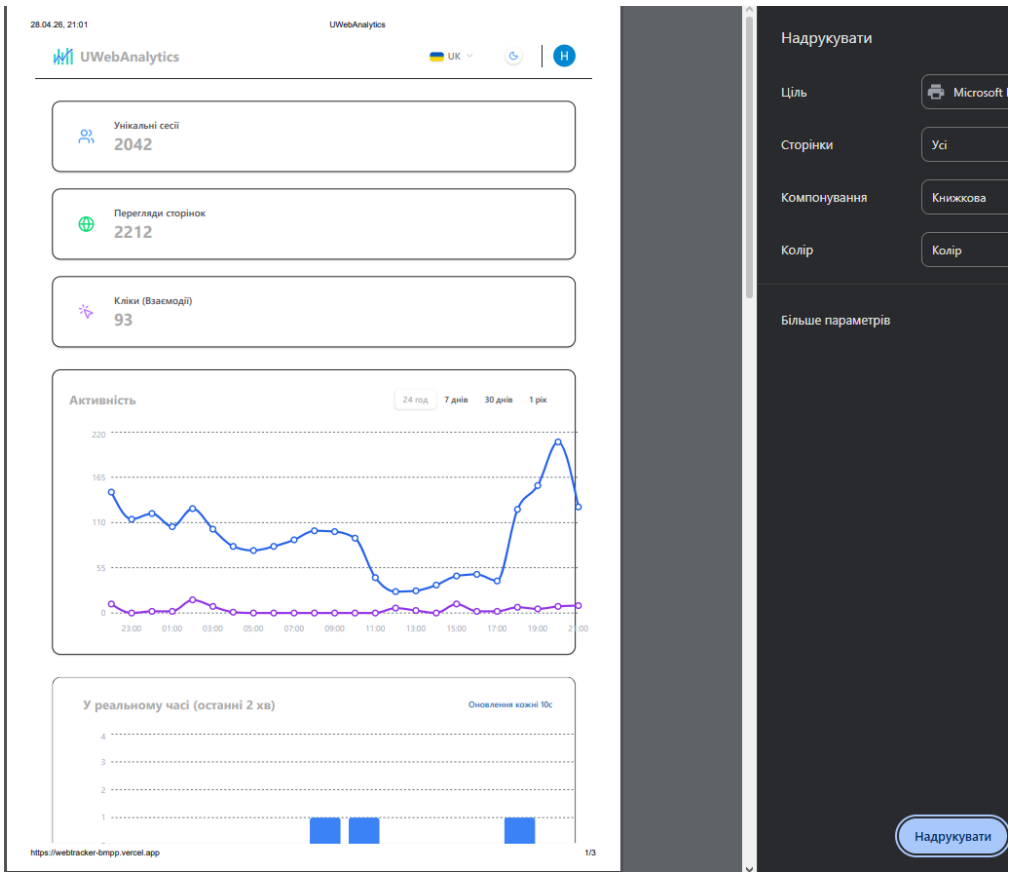


Рисунок 3.7 – Приклад готової до друку сторінки

Іншим критичним компонентом сервісного рівня є система сповіщень та Email-інформування. Питання про те, як реалізувати оперативне інформування адміністратора про критичні зміни в трафіку, було вирішено шляхом створення підсистеми «Alerts» (рис 3.8). Програмна реалізація сповіщень базується на взаємодії з колекцією alerts у хмарному сховищі та серверними обробниками. Користувач має можливість активувати стандартні типи сповіщень, такі як падіння трафіку або отримання щотижневого звіту. Реалізація функції handleToggleStandardAlert забезпечує асинхронне оновлення стану алертів у базі даних, що автоматично тригерить відповідні серверні функції при досягненні встановлених порогів активності.

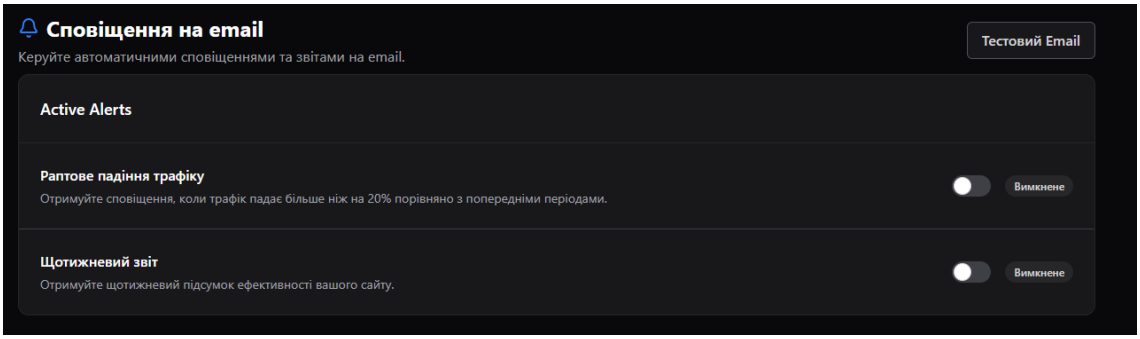


Рисунок 3.8 – Меню налаштувань сповіщень

Щоб перевірити, як працюют канали зв'язку, ми реалізували функцію тестового відправлення повідомлень. Це просто – потрібно надіслати POST-запит до серверного ендпоінту /api/alerts/test, і це запускає поштовий шлюз. Завдяки цій можливості адміністратори можуть швидко перевірити, чи все налаштовано правильно і чи сервіс доступний, не чекаючи справжніх подій. Поєднання функцій експорту та багатоканальних сповіщень робить нашу аналітичну систему справжнім інструментом моніторингу. Вона не лише збирає дані, а й активно взаємодіє з оператором, що дозволяє миттєво реагувати на будь-які зміни в роботі веб-сервісу.

3.6 Реалізація механізмів локалізації та управління колірними темами

Проєктуючи сучасний вебінтерфейс для аналітичної системи, важливо врахувати зручність використання в різних умовах освітлення і для користувачів, які говорять різними мовами. Щоб вирішити задачу глобалізації програмного забезпечення, ми впровадили систему локалізації на основі бібліотеки i18next[19]. Мовна адаптація реалізована через декларативні файли перекладів у форматі JSON. Це дозволяє нам відокремити текстовий контент від програмної логіки компонентів, що, насправді, спрощує всю цю історію. Механізм перемикання мов працює за допомогою хука useTranslation; він реагує на зміни і оновлює текстові константи у всьому дереві компонентів React без необхідності перезавантажувати сторінку. Ще одним цікавим моментом у локалізації є

візуальна ідентифікація обраної мови. Ми інтегрували логіку вибору мов з хедером системи, де реалізували випадаюче меню. Як тільки користувач змінює мову, система автоматично оновлює стан бібліотеки i18n-iso-countries, що гарантує правильний переклад назв країн на інтерактивній карті світу та в аналітичних звітах. Це все робить наш продукт більш зрозумілим і доступним для користувачів.

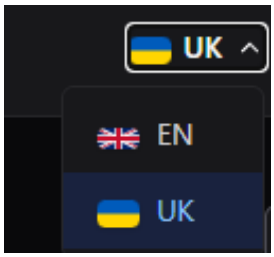


Рисунок 3.9 – Випадаюче меню вибору мови

Питання комфорту було вирішено шляхом реалізації динамічного управління колірними темами. Процес програмної реалізації темного та світлого режимів базується на використанні можливостей утилітарного css-фреймворку Tailwind [20]. Для забезпечення безперервності досвіду користувача було спроєктовано механізм персистентності стану теми. Програма використовує комбінований алгоритм ініціалізації: при першому завантаженні система аналізує системні налаштування операційної системи через медіа-запит prefers-color-scheme, а в подальшому пріоритет надається вибору користувача, який зберігається у локальному сховищі браузера.

Реалізація перемикання тем здійснюється через централізований ефект у кореневому компоненті App.tsx. Програмний код маніпулює класом dark на рівні кореневого елемента document.documentElement, що ініціює каскадне оновлення кольорів усіх дочірніх компонентів. Для візуалізації стану теми використано бібліотеку іконок lucide-react, де компоненти Sun та Moon виконують роль інтерактивних індикаторів. Така реалізація не лише підвищує естетичну привабливість системи, а й має практичне значення для зменшення зорової втоми

оператора при роботі в нічний час, що є критично важливим для систем безперервного моніторингу активності.

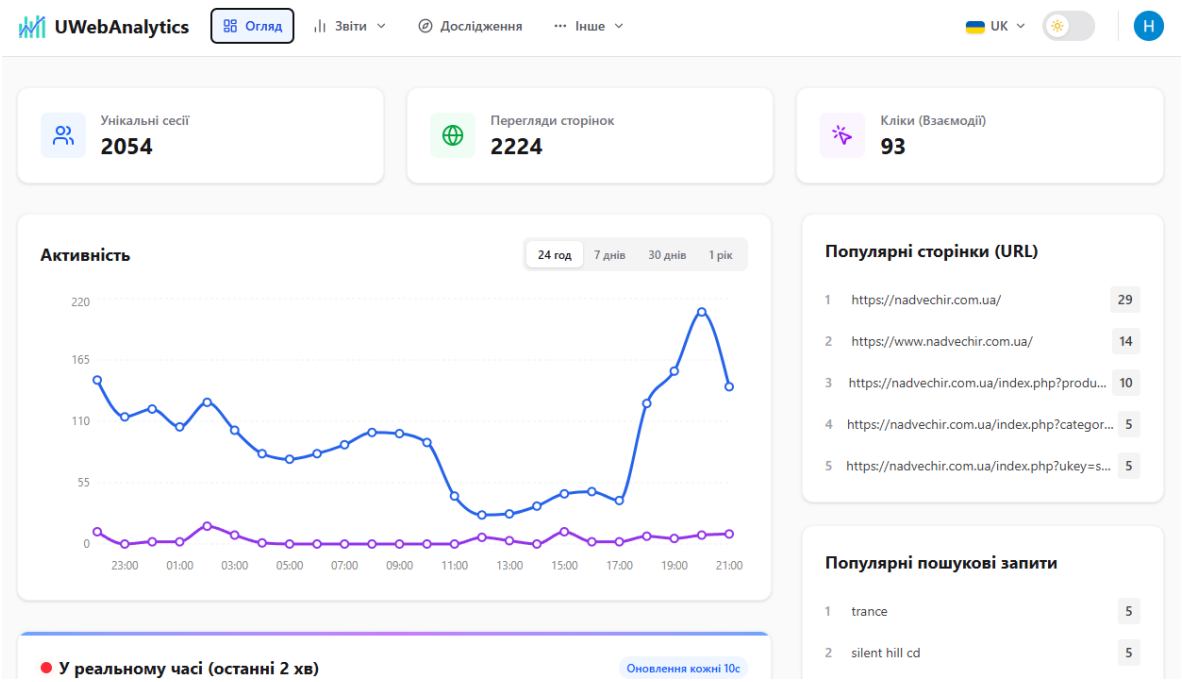


Рисунок 3.10 – Вигляд системи з білою темою

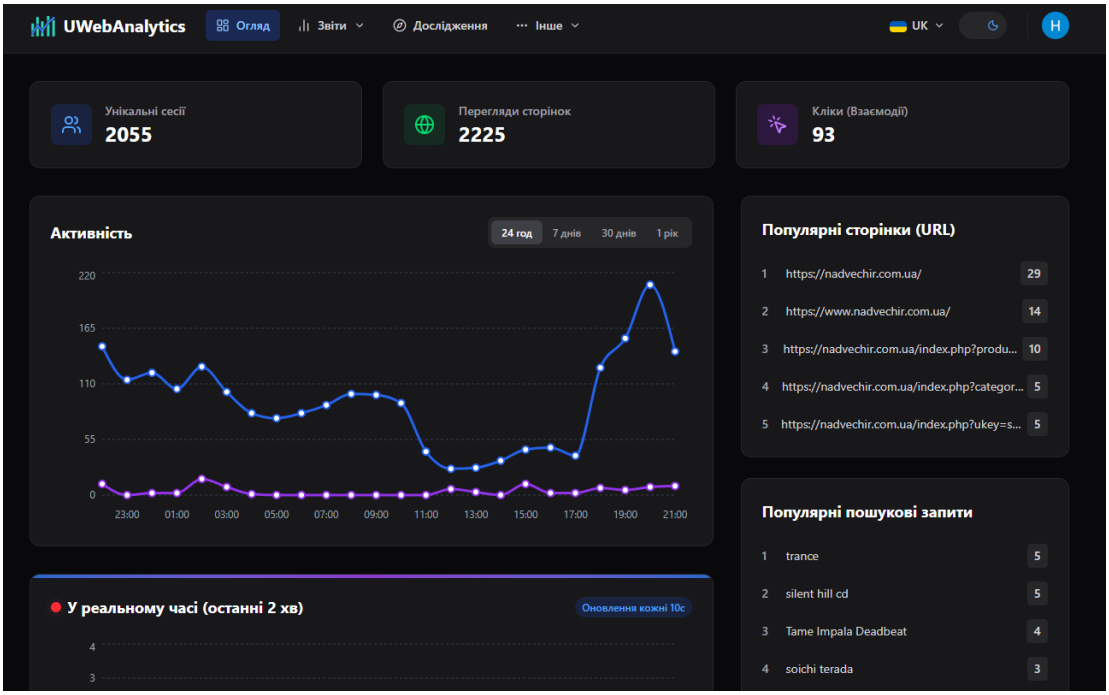


Рисунок 3.11 – Вигляд системи з темною темою

3.7 Тестування працездатності та аналіз результатів розробки

Процес верифікації та валідації розробленого програмно-технічного засобу аналітики є критичним етапом, що дозволяє підтвердити відповідність реалізованого функціоналу вимогам технічного завдання та забезпечити стабільність роботи системи в умовах реальної експлуатації. Методологія дослідження працездатності комплексу базувалася на багаторівневому підході, що включав модульне тестування окремих компонентів логіки, інтеграційну перевірку взаємодії між хмарними сервісами та клієнтським SDK, а також системне тестування всього середовища під навантаженням. Основним завданням етапу тестування було виявлення потенційних дефектів у алгоритмах агрегації даних, перевірка стійкості до відмов мережі та оцінка швидкодії людино-машинного інтерфейсу при обробці великих масивів подій.

Модульне тестування було зосереджене на перевірці чистоти виконання функцій обробки даних, таких як алгоритм бакетізації часових рядів та методи групування для модуля досліджень. Для цього використовувалися набори фіктивних вхідних даних, що імітували різні часові позначки та типи подій. Особлива увага приділялася крайовим випадкам, наприклад, обробці подій, що надходять на межі десятисекундних інтервалів або при зміні дати. Тестування підтвердило математичну точність алгоритмів та коректність роботи фільтрів, що забезпечує достовірність аналітичних показників, які виводяться на дашборд адміністратора. Паралельно проводилося тестування компонентів React, де перевірялася коректність реактивного оновлення стейту при отриманні нових документів із бази даних Firestore через WebSocket-з’єднання.

Інтеграційний етап тестування мав на меті перевірку цілісності конвеєра даних від моменту ініціації події на цільовому сайті до її фінальної візуалізації. Для реалізації цього етапу було створено ізольоване середовище з тестовим веб-ресурсом, де за допомогою скриптів автоматизації імітувався трафік різної інтенсивності. Було перевірено коректність формування пакетів даних,

правильність передачі заголовків ідентифікації проєкту та стійкість системи до блокування сторонніх записів. Результати показали, що система успішно ідентифікує унікальні сесії та коректно збагачує їх геопросторовою інформацією на рівні серверного проксі. Окремо тестувалася підсистема безпеки: проводилися спроби несанкціонованого доступу до API без валідного ідентифікатора проєкту та спроби зчитування даних іншого користувача, що були успішно заблоковані правилами безпеки хмарного сховища.

Навантажувальне тестування було спрямоване на визначення межі пропускної здатності системи та оцінку стабільності безсерверної архітектури. Під час стрес-тестів імітувалося одночасне надходження великої кількості подій від декількох сотень віртуальних клієнтів. Аналіз журналів аудиту та метрик продуктивності Google Cloud підтвердив, що архітектура на базі хмарних функцій ефективно масштабується, підтримуючи мінімальну затримку обробки навіть при пікових сплесках активності. Разом із цим проводилося тестування зручності користування інтерфейсом, де оцінювалася швидкість рендерингу складних графіків Recharts. Було встановлено, що використання механізмів мемоізації та оптимізації оновлень у React дозволяє підтримувати плавність роботи інтерфейсу навіть при відображенні тисяч точок даних на лінійних діаграмах.

Завершальна стадія тестування включала перевірку сервісних функцій, зокрема системи інтелектуального аналізу на базі ШІ та модулів сповіщень. Перевірка інтелектуального асистента підтвердила його здатність коректно інтерпретувати переданий контекст та генерувати логічні висновки щодо динаміки трафіку.

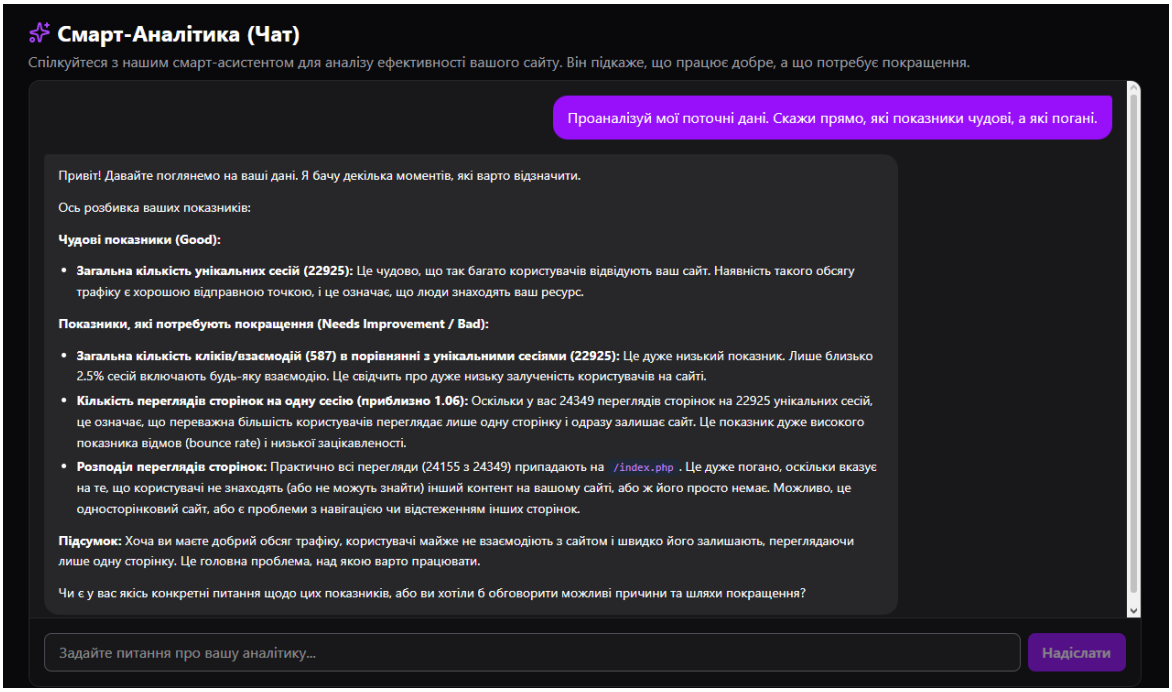


Рисунок 3.12 – Тестування смарт-аналітики

Тестування системи email-інформування через тригерні запити продемонструвало стабільну доставку повідомлень адміністратору.

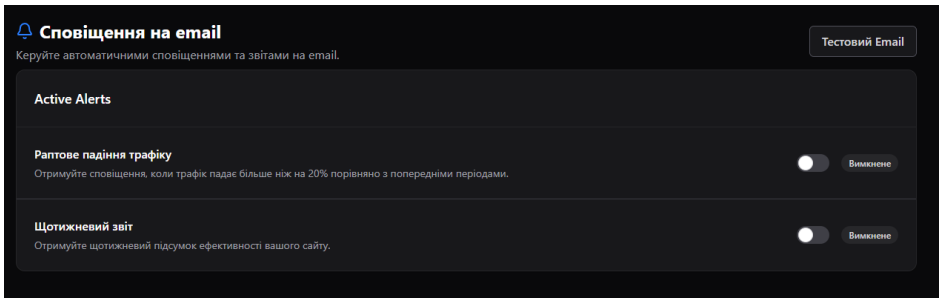


Рисунок 3.13 – Меню сповіщень на email

Підсумковий аналіз результатів тестування дозволяє зробити висновок про повну функціональну придатність програмно-технічного засобу та його готовність до практичного впровадження. Усі виявлені під час розробки недоліки були оперативно усунуті, що гарантує високу надійність та достовірність аналітичної підтримки для веб-сервісів будь-якої складності.

3.8 Висновки до третього розділу

У третьому розділі ми успішно завершили повний цикл реалізації та тестування нашого програмно-технічного засобу «UWebAnalytics». Завдяки вибору сучасного технологічного стеку на базі TypeScript, React і безсерверних функцій Node.js, ми змогли створити надзвичайно ефективну систему, яка впорається з інтенсивними потоками даних у реальному часі. Особливо зосередилися на розробці зручного для користувача інтерфейсу. Тут модульна архітектура, динамічна візуалізація через бібліотеку Recharts і інтерактивна карта світу допомогли зробити аналітичну інформацію максимально доступною. Крім того, адаптивна верстка, підтримка кількох мов та різних кольорових тем дійсно відповідають сучасним вимогам щодо ергономіки та інклюзивності. Одним із важливих технічних досягнень стало впровадження інтелектуального модуля на основі API Google Gemini. Це дало нам можливість перетворити систему з простого інструмента збору статистики на справжній помічник для прийняття рішень. Реалізація складних алгоритмів для динамічної агрегації та групування даних у модулі «Explorations» довела, що ми можемо проводити глибокий ad-hoc аналіз прямо на стороні клієнта. Наше рішення щодо безпеки, яке побудоване на OAuth 2.0 і декларативних правилах Firestore, забезпечує надійну ізоляцію даних і захист від несанкціонованого доступу – це надзвичайно важливо для систем, що працюють з поведінковою телеметрією. Комплексне тестування підтвердило, що наша система повністю відповідає технічним вимогам. Результати модульних, інтеграційних і навантажувальних випробувань показали стабільність архітектури, швидкий відгук інтерфейсу та точність математичних розрахунків активності. Додатково реалізувавши сервіси як Email-сповіщення та експорт звітів у PDF, ми завершили формування функціонально повного продукту, готового до впровадження у реальні веб-сервіси для моніторингу та аналізу взаємодії користувачів.

ВИСНОВКИ

У цій роботі ми розглянули результати теоретичних і практичних досліджень, в рамках яких створили програмно-технічний засіб для збору та аналізу даних про активність користувачів веб-сервісу. Цей інструмент забезпечує автоматизований моніторинг, інтелектуальну інтерпретацію та візуалізацію поведінкових патернів у реальному часі. У першому розділі ми провели аналіз предметної області і вже наявних систем вебаналітики, таких як Google Analytics, Matomo, Mixpanel і SimilarWeb. З'ясували технічні особливості збору даних у розподіленому веб-середовищі та встановили, що велика кількість подій потребує застосування масштабованих NoSQL архітектур. Важливо зазначити, що розробка власного засобу є актуальною для повного контролю над конфіденційними даними й реалізації когнітивного аналізу на основі штучного інтелекту. Другий розділ присвячений системному та архітектурному проєктуванню інформаційної системи. Тут ми обґрунтували вибір подієво-орієнтованої архітектури (EDA) і трирівневої моделі обробки даних. Розробили логічну та фізичну структури бази даних Firestore, спроєктували інформаційну ER-модель сутностей і зв'язків. Також сформували алгоритмічне забезпечення системи, зокрема математичні моделі часової бакетизації для роботи в реальному часі та методи динамічного групування для конструктора звітів. Не забули й про wireframe-проєкт людино-машинного інтерфейсу та стратегію безпеки на основі протоколу OAuth 2.0. У третьому розділі ми реалізували програмні модулі системи з використанням TypeScript, React та хмарних функцій. Створили інтерактивний дашборд із візуалізацією за допомогою Recharts, інтегрували інтелектуального асистента на базі Gemini AI, а також додали механізми кастомізації інтерфейсу через Drag-and-Drop. Впровадили систему локалізації, управління кольоровими темами й сервіси сповіщень і експорту даних. Тестування показало, що система працює справно, масштабується добре і швидко обробляє дані – це доводить її функціональну придатність для практичного використання у комп'ютерній інженерії та веб-технологіях.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Кваліфікаційна робота : методичні вказівки щодо її виконання для студентів освітнього рівня «бакалавр» спеціальності 123 «Комп'ютерна інженерія» / Т. О. Говорущенко, О. С. Савенко, С. М. Лисенко, А. В. Горошко, Є. Г. Гнатчук. Хмельницький : ХНУ, 2021. 54 с.
2. Ігнатьєва, І.В. & Коваленко, І.О. Аналіз поведінкової активності користувачів веб-застосувань для поліпшення досвіду користувача. *Вісник НТУУ "КПІ". Інформатика, управління та обчислювальні системи*. 2023. С. 29(1), 29-38.
3. Caushik A. Web Analytics 2.0: The Art of Online Accountability & Science of Customer Centricity. 2017. С. 23-38. URL: https://dokumen.pub/qdownload/web-analytics-20-the-art-of-online-accountability-science-of-customer-centricity.html (дата звернення: 17.03.2026).
4. Електроний ресурс Google Analytics. URL: https://developers.google.com/analytics (дата звернення: 17.03.2026).
5. Google Analytics 4 Documentation : Google Developers resource. URL: https://developers.google.com/analytics/devguides/collection/ga4 (дата звернення: 17.03.2026).
6. Електроний ресурс Matomo. URL: https://matomo.org (дата звернення: 17.03.2026).
7. Електроний ресурс Mixpanel. URL: https://mixpanel.com/home/ (дата звернення: 17.03.2026).
8. Електроний ресурс Similar Web. URL: similarweb.com (дата звернення: 17.03.2026).
9. OAuth 2.0 Framework : Internet Engineering Task Force (IETF) RFC 6749. URL: https://oauth.net/2/ (дата звернення: 16.03.2026).
10. React : a JavaScript library for building user interfaces : official documentation. URL: https://react.dev (дата звернення: 17.03.2026).

11. TypeScript : JavaScript with syntax for types : official documentation. URL: https://www.typescriptlang.org (дата звернення: 17.03.2026).

12. Gemini API Overview : Google AI Studio Documentation. URL: https://ai.google.dev/gemini-api/docs (дата звернення: 19.03.2026).

13. Cloud Firestore : Google Firebase Documentation. URL: https://firebase.google.com/docs/firestore (дата звернення: 09.03.2026).

14. Node.js JavaScript runtime : official website documentation. URL: https://nodejs.org (дата звернення 10.04.2026).

15. Express : Fast, unopinionated, minimalist web framework for Node.js. URL: https://expressjs.com (дата звернення 10.04.2026).

16. Recharts : A composable charting library built on React components. URL: https://recharts.org (дата звернення: 16.04.2026).

17. dnd-kit : A lightweight, modular, extensible drag and drop toolkit for React. URL: https://dndkit.com (дата звернення 10.04.2026).

18. Firebase Security Rules Reference : Google Firebase Documentation. URL: https://firebase.google.com/docs/rules (дата звернення 10.04.2026).

19. i18next : internationalization-framework for JavaScript. URL: https://www.i18next.com (дата звернення 10.04.2026).

20. Tailwind CSS : Rapidly build modern websites without ever leaving your HTML : documentation. URL: https://tailwindcss.com (дата звернення 10.04.2026).

21. What Is a Chatbot? | IBM. URL: https://www.ibm.com/topics/chatbots (дата звернення 11.04.2026).

22. Що таке нейронна мережа та як вона працює. URL: https://mc.today/uk/shho-take-nejronna-merezha/ (дата звернення 10.04.2026).

23. MySQL. URL: https://www.mysql.com/ (дата звернення 10.04.2026).

24. SQL vs NoSQL. 5 Critical Differences. URL: https://www.integrate.io/blog/the-sql-vs-nosql-difference/ (дата звернення 10.04.2026).

25. The pros and cons of Firebase. URL: https://www.ascensor.com/post/the-pros-and-cons-of-firebase (дата звернення 10.04.2026).

26. What makes a good website. URL: https://www.wix.com/blog/what-makes-a-good-website (дата звернення 10.04.2026).

27. How to track user activity on website. URL: https://uxcam.com/blog/how-to-track-user-activity-on-website/ (дата звернення 10.04.2026).

28. A begginer's guide to building an AI chat bot with Google Gemini API and Next.js. URL: https://tutorialsdojo.com/hello-user-a-beginners-guide-to-building-an-ai-chatbot-with-google-gemini-api-and-next-js/ (дата звернення 18.04.2026).

29. Gemini AI: How to Create Chatbots, Content, and More with Natural Language Processing AI. URL: https://medium.com/@Avillalonv/gemini-ai-how-to-create-chatbots-content-and-more-with-natural-language-processing-ai-9c58bf10e9d2 (дата звернення 18.04.2026).

30. Website Push Notifications: notifications from your website. URL: https://adrenalead.com/en/blog/notification-from-website/ (дата звернення 18.04.2026).

31. Firebase vs MYSQL. URL: https://www.integrate.io/blog/firebase-vs-mysql/ (дата звернення 22.04.2026).

32. What is Web Analytics. URL: https://wp-statistics.com/2025/06/what-is-web-analytics/ (дата звернення 24.04.2026).

33. 10 best web analytics tools for 2026. URL: https://wpmailsmtp.com/best-web-analytics-tools/ (дата звернення 24.04.2026).

34. Вразливості сучасних JavaScript-додатків: як виявляти та нейтралізувати. URL: https://careers.epam.ua/blog/vulnerabilities-of-modern-js-apps-how-to-detect-and-mitigate-them (дата звернення 25.04.2026).

35. Visual Studio Code - Code Editing. Redefined. URL: https://code.visualstudio.com/ (дата звернення 25.04.2026).

36. Online Digram Mermaid. URL: https://mermaid.ai/open-source/?utm_medium=hero&utm_campaign=variant_a&utm_source=mermaid_js (дата звернення 25.04.2026).

37. Online Digrams. URL: https://app.diagrams.net (дата звернення 09.04.2026).

38. Темна тема у веб-дизайні: коли використовувати та як реалізувати. URL: https://webscraft.org/blog/temna-tema-u-vebdizayni-koli-vikoristovuvati-ta-yak-realizuvati (дата звернення 08.04.2026).

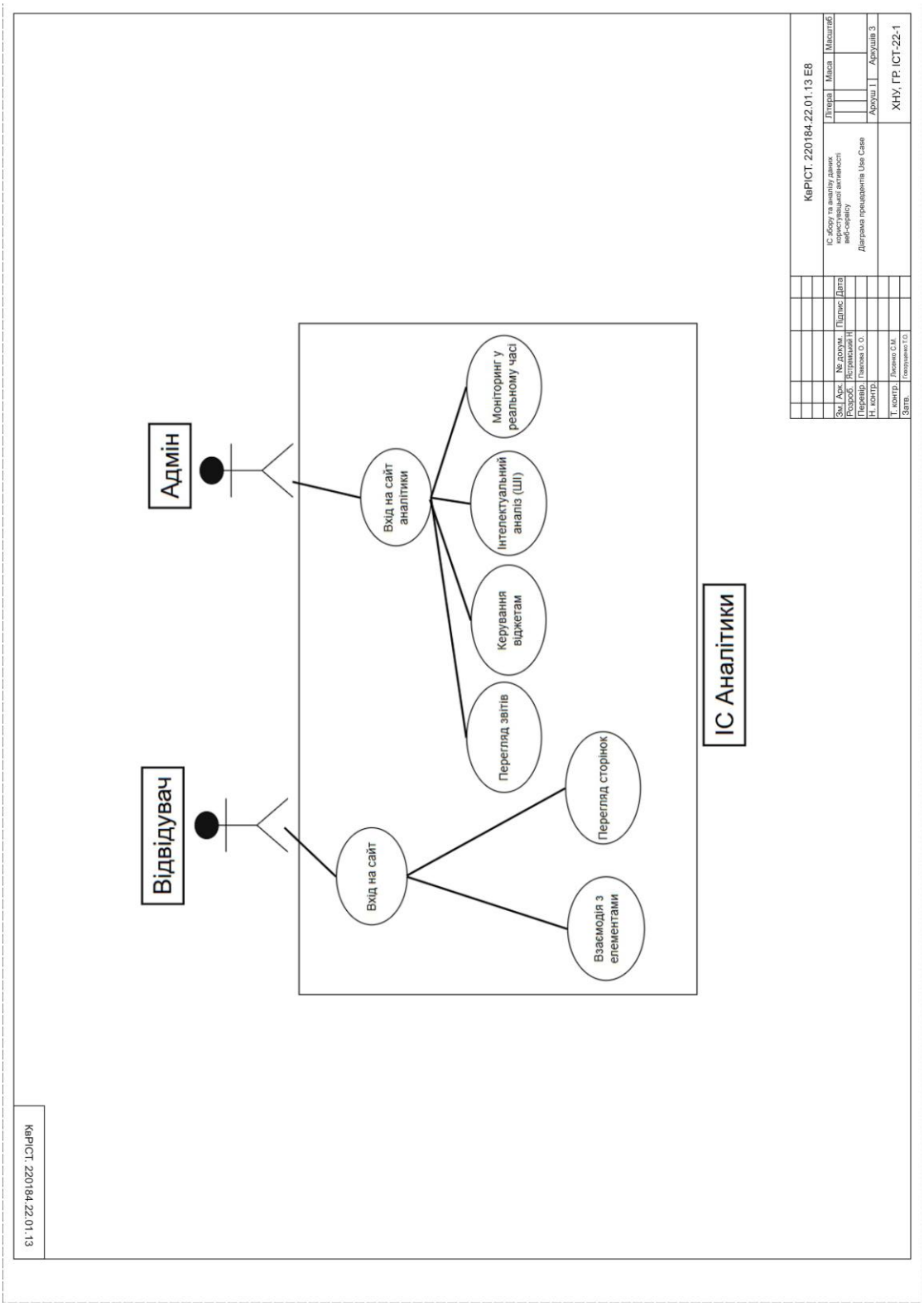
39. Getting started with Vercel. URL: https://vercel.com/docs/getting-started-with-vercel (дата звернення 09.04.2026).

40. How it works and how to create Vercel. URL: https://medium.com/@mayank2803sharma/demystifying-vercel-how-it-works-and-how-to-create-vercel-763586070478 (дата звернення 09.04.2026).

ДОДАТОК А

(обов’язковий)

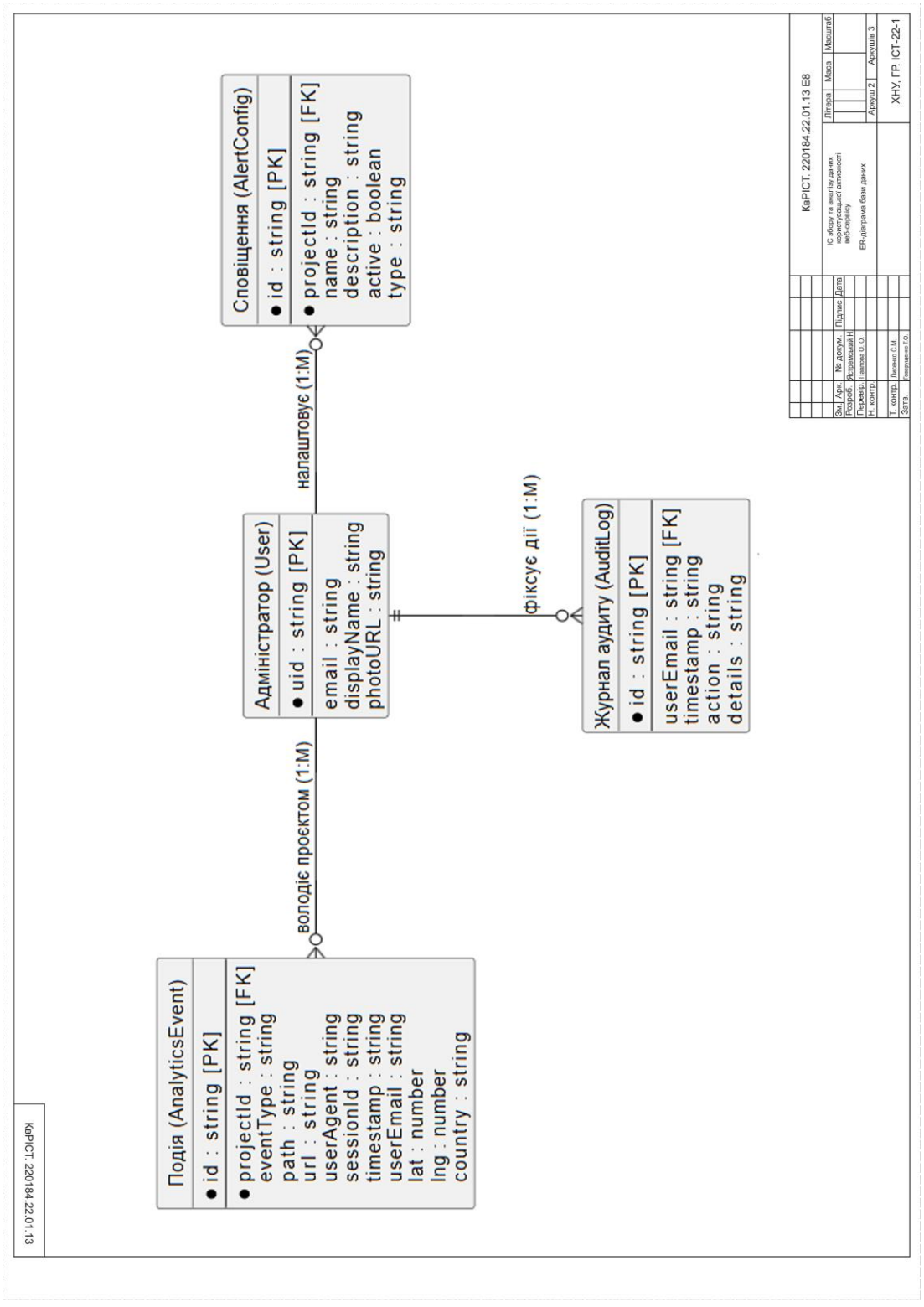
Копія креслення «Діаграма прецедентів Use Case»



ДОДАТОК Б

(обов’язковий)

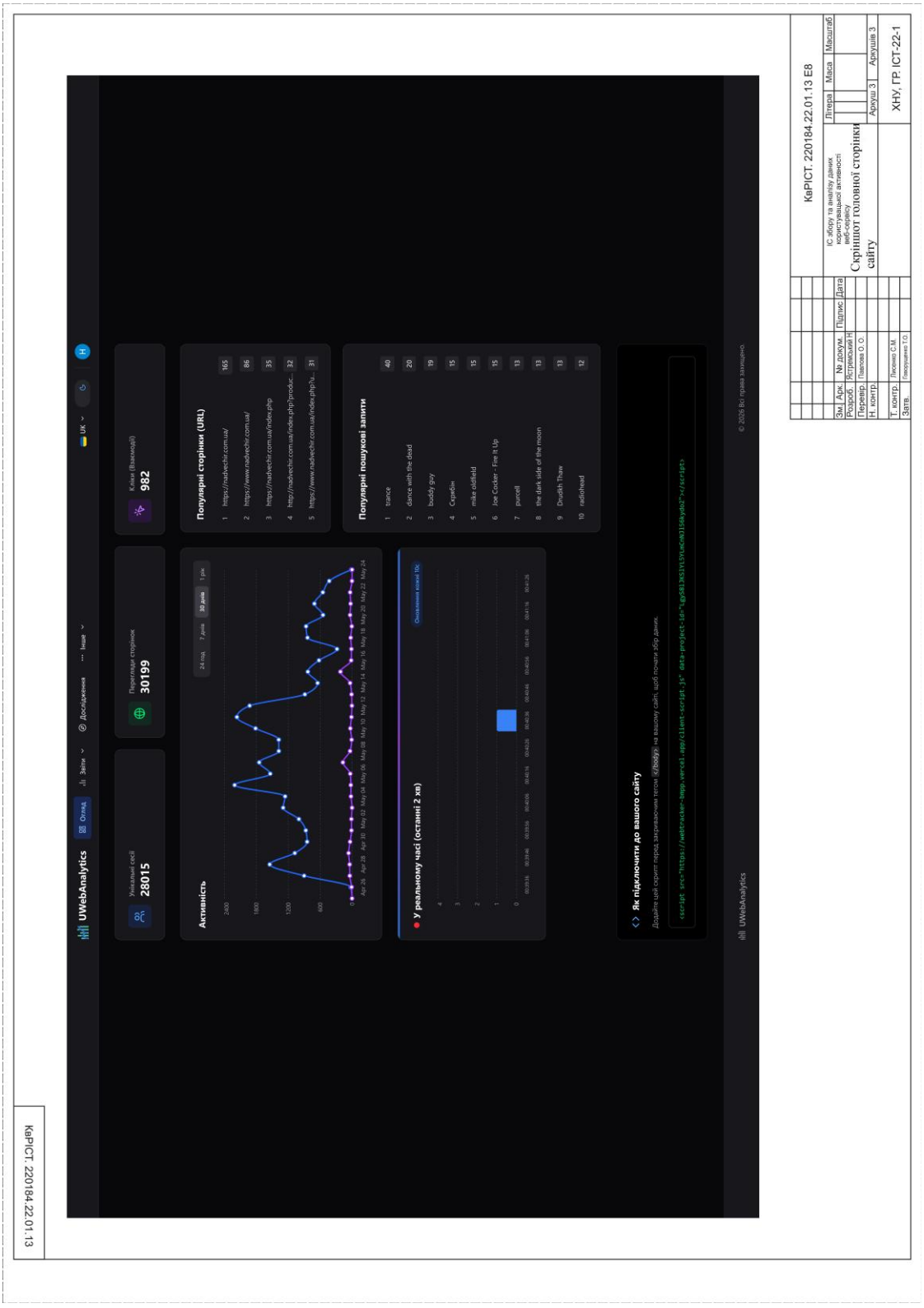
Копія креслення «ER-діграма»



ДОДАТОК В

(обов’язковий)

Копія креслення «Скріншот головної сторінки»



ДОДАТОК Г

(обов’язковий)

Лістинг коду «Підключення потоку даних з Firebase у реальному часі»

```
  useEffect(() => {
  if (!user) return; // Чекаємо на авторизацію користувача

  // Створюємо запит: останні 1000 подій, відсортовані за часом
  const eventsQuery = query(
    collection(db, 'events'),
    where('projectId', '==', user.uid),
    orderBy('timestamp', 'desc'),
    limit(1000)
  );

  // Встановлюємо слухача змін у базі даних (onSnapshot)
  const unsubscribeEvents = onSnapshot(eventsQuery, (snapshot) => {
    const eventsData = snapshot.docs.map(doc => ({
      id: doc.id,
      ...doc.data()
    })) as AnalyticsEvent[];

    setEvents(eventsData);
  }, (error) => {
    console.error('Помилка завантаження подій:', error);
  });

  // Відписуємось при розмонтуванні компонента
  return () => unsubscribeEvents();
}, [user]);
```

Протокол аналізу звіту подібності експертом

Заявляю, що я ознайомився (-лась) з Повним звітом подібності, який був згенерований Системою виявлення і запобігання плагіату щодо роботи:

Автор: Назарій ЯСТРЕМСЬКИЙ

Співавтор:

Назва: Інформаційна система збору та аналізу даних користувацької активності веб-сервісу

Експерт: Ольга ПАВЛОВА

Підрозділ: Кафедра комп'ютерної інженерії та інформаційних систем

Коефіцієнт подібності 1:1.51%

Коефіцієнт подібності 2:0.37%

Мікропробіли: 3

Заміна букв: 3

Інтервали: 0

Білі знаки: 0

Дата створення звіту: 2026-05-26 20:40:10.0

Після аналізу Звіту подібності констатую наступне:

☒ **Запозичення, виявлені в роботі є законними і не є плагіатом. Рівень подібності не перевищує допустимої межі. Таким чином робота незалежна і приймається.**

☐ **Запозичення не є плагіатом, але перевищено граничне значення рівня подібностей. Таким чином робота повертається на доопрацювання.**

☐ **Виявлено запозичення і плагіат або навмисні текстові спотворення (маніпуляції), як передбачувані спроби укриття плагіату, які роблять роботу невідповідною вимогам законодавства (Ст. 32. ЗУ Про вищу освіту, пункт 3.1, Ст. 42. ЗУ Про освіту) та вимог НАЗЯВО (Критерій 5), а також кодексу етики і процедурам. Таким чином робота не приймається.**

Обгрунтування:

2026-05-27 Доцент Андрій Нічепорук

Дата експерт

Anti-Plagiarism (http://ap.km.ua) v-15.701

Максимальне співпадіння з одним документом 1.0%

Словники перевірки: en_US, ru_RU, ua_UA. **Помилок в документах: 12%**

ID: 272404 Назва: БКР Інформаційна система збору та аналізу даних користувацької активності веб-сервісу Додано в БД: 2026-05-27 Автора: Назарій ЯСТРЕМСЬКИЙ Керівники: Ольга ПАВЛОВА Консультанти: Опоненти:	Документ		Сумарний збіг по Базі Даних	
	Символи	Лексеми	Символи	Лексеми
	86725	617	1471 (2%)	22 (4%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

Дипломник: Ястремський Назарій Костянтинович

Тема: Інформаційна система збору та аналізу даних користувацької активності веб-сервісу

Спеціальність: 126 «Інформаційні системи та технології»

Обсяг кваліфікаційної роботи:

Кількість листів креслень 3 Кількість сторінок записки 68

1. Короткий зміст роботи та прийнятих рішень: Метою кваліфікаційної роботи є проєктування та реалізація програмно-технічного засобу для моніторингу дій відвідувачів веб-сервісу.

2. Висновок про відповідність роботи дипломному завданню: Робота повністю відповідає поставленому завданню.

3. Характеристика виконання кожного розділу, ступінь використання останніх досягнень науки і техніки і передових методів роботи: В першому розділі кваліфікаційної роботи проведено дослідження предметної області (виконано детальний аналіз існуючих аналогів, виявлено їхні недоліки та переваги та сформовано технічні вимоги для розробки системи) та виконано постановку задачі дослідження. В другому розділі кваліфікаційної роботи проведено системне проєктування архітектури проєкту. Сформовано подієво-орієнтовану модель, розроблено структуру NoSQL бази даних для високої інтенсивності запису подій, математично алгоритмізовано метод часової бекетізації. Особливу увагу приділено Promt Engineering – розроблено методологію інтегрування JSON зрізів даних у системі для інтеграції великої мовної моделі. В третьому розділі кваліфікаційної роботи виконано апаратну реалізацію проєкту на базі сучасного технологічного стеку: розроблено клієнтський JavaScript-трекер, серверний шлюз обробки на Edge-функціях таких як Node.js. Інтегровано аналітичну панель яка формує звіти, та велику мовну модель Gemini через API яка аналізує вхідні дані і формує звіт із порадами.

4. Позитивні сторони роботи: висока практична цінність роботи.

5. Негативні сторони роботи: недостатня увага пропускної здатності хмарного сховища, обмежена і невелика кількість функцій для формування звітів.

6. Оцінка графічного оформлення та пояснювальної записки роботи: Пояснювальна записка оформлена коректно, згідно діючих стандартів оформлення документації. Графічні матеріали наочно ілюструють архітектурні рішення.

7. Відгук про роботу в цілому: Робота виконана на належному науково-технічному рівні.

8. Інші зауваження: _____

9. Оцінка дипломної роботи: добре

Рецензент (прізвище, ім'я, по батькові, посада, місце роботи) _____

Педосенко І. П., зав. каф. ІПЗ, ХНУ

_____

"___"_____2026 р. _____(підпис)

Зав. кафедри КІІС
д-р. філософії Ользі ПАВЛОВІЙ

Назарій ЯСТРЕМСЬКИЙ
ПІБ здобувача вищої освіти

ФІТ, 4 курсу, групи ІСТ-22-1

ЗАЯВА

З правилами чинного Положення про систему забезпечення академічної доброчесності у Хмельницькому національному університеті, згідно з яким виявлення академічного плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту і застосування заходів академічної відповідальності, ознайомлений (а). Про використання спеціалізованих програмних засобів (СПЗ) StrikePlagiarism та Anti-Plagiarism для перевірки кваліфікаційних робіт здобувачів вищої освіти на наявність академічного плагіату оповіщений (а). Надаю університету право на передачу моєї роботи для обробки та збереження в базах даних СПЗ і використання роботи для виявлення академічного плагіату в інших роботах, які перевіряються СПЗ.

Також надаю свою згоду на обробку й збереження університетом моєї роботи в Інституційному репозитарії Хмельницького національного університету.

Робота надається для перевірки в електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

_____ 2026 року

РІШЕННЯ ЕКСПЕРТНОЇ КОМІСІЇ

КАФЕДРИ КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА ІНФОРМАЦІЙНИХ СИСТЕМ
ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Назва кваліфікаційної роботи Інформаційна система збору та аналізу даних користувацької активності веб-сервісу
Автор Назарій ЯСТРЕМСЬКИЙ
Освітня програма Інформаційні системи та технології
Рівень вищої освіти перший (бакалаврський)
Спеціальність 126 Інформаційні системи та технології
Науковий керівник: ДФ, доцент Ольга ПАВЛОВА

На основі аналізу кваліфікаційної роботи на дотримання вимог академічної доброчесності (у т.ч. відсутності ознак академічного плагіату) з урахуванням результатів перевірки роботи спеціалізованим програмним засобом(ами) комісія зробила такий висновок:

№	Висновок	Позначка про відповідність
1	Ознаки академічного плагіату	
1.1	Запозичення, виявлені в роботі, є законними і не є академічним плагіатом (далі – зазначаються підстави віднесення запозичень до правомірних, якщо потрібно). Робота приймається до захисту.	відповідає
1.2	Виявлені запозичення не є академічним плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи (далі – зазначаються детальні та аргументовані підстави віднесення запозичень до правомірних). Робота приймається до захисту, але має бути відкоригована.	
1.3	Виявлені запозичення не є академічним плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. Робота може бути допущена до захисту після того як буде відкоригована та доопрацьована і успішно пройде повторну перевірку на академічний плагіат.	
1.4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття текстових запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	
2	Інші види порушень академічної доброчесності	

Підтвердження:

Запозичення, виявлені в роботі, є законними і не є плагіатом, оскільки:

1) усі запозичення фрагментарні, або мають належним чином оформленні посилання;

2) окремі виявлені збіги є загальновживаними фразами або виразами, про що свідчить посилання системи на збіг з джерелами на один фрагмент речення;

3) всі зафіксовані системою ознаки модифікації тексту відносяться до комбінування латинських символів зі україномовними скороченнями індексів в формулах, що не є модифікацією тексту.

4) значна частина знайденого плагіату відноситься до списку використаних джерел

Сумарний обсяг всіх запозичень, визначений системою виявлення збігів/ ідентичності/схожості StrikePlagiarism, складає 1,51% і адресується до 10 першоджерела; та системою Anti-Plagiarism складає 1%, що, з урахуванням наведених обгрунтувань, відповідає характеру наукового дослідження і свідчить на користь кваліфікаційної роботи.

Завідувач кафедри Підпис Ольга ПАВЛОВА
Ім'я, ПРІЗВИЩЕ

Гарант освітньої програми Підпис Єлизавета ГНАТЧУК
Ім'я, ПРІЗВИЩЕ

Керівник кваліфікаційної роботи Підпис Ольга ПАВЛОВА
Ім'я, ПРІЗВИЩЕ