НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ПОГОДЖЕНО **Декан факультету**	**ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ** **Завідувач кафедри інформаційних систем і технологій**
_____ **Ігор БОЛБОТ** (підпис)	_____ **Михайло ШВИДЕНКО** (підпис)
«___» _____ 2026 р.	«___» _____ 2026 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Інформаційна система бізнес-аналітики для медіабаїнгу**»**

Спеціальність «126 Інформаційні системи та технології»

Освітньо-професійна програма «Інформаційні системи та технології»

Гарант освітньої програми
к.е.н., доцент _____ **Максим МОКРІЄВ**
(підпис)

Керівник бакалаврської кваліфікаційної роботи
доктор філософії, доцент _____ **Таїсія САЯПІНА**
(підпис)

Виконав _____ **Богдан ЧЕХІРКІН**
(підпис)

Київ-2026

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ЗАТВЕРДЖУЮ
Завідувач кафедри інформаційних систем і технологій

к.е.н., доцент _____ Михайло ШВИДЕНКО
(підпис)

«___» _____ 2026 р.

ЗАВДАННЯ ДО ВИКОНАННЯ БАКАЛАВРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ ЗДОБУВАЧУ

Чехіркіну Богдану Олексійовичу

Спеціальність «126 Інформаційні системи та технології»
ОП «Інформаційні системи та технології»
Тема бакалаврської кваліфікаційної роботи «Інформаційна система бізнес-аналітики для медіабаїнгу» затверджена наказом від «19» грудня 2025 р. № 3205 «С»
Термін подання завершеної роботи на кафедру «22» травня 2026 р.
Вихідні дані до бакалаврської кваліфікаційної роботи:

Нормативні та методичні матеріали щодо розроблення інформаційних систем; наукові джерела з питань бізнес-аналітики, медіабаїнгу, оброблення маркетингових даних та візуалізації KPI; документація Python, PyQt6, SQLite, Google Ads API, Meta Marketing API, TikTok Business API, Google Analytics Data API; матеріали аналізу існуючих програмних рішень у сфері рекламної аналітики; функціональні та нефункціональні вимоги до інформаційної системи бізнес-аналітики для медіабаїнгу.

Перелік питань, що підлягають дослідженню:

1. аналіз предметної області моніторингу технічного стану промислового обладнання та технологій predictive maintenance;
2. дослідження сучасних IoT-платформ і інформаційних систем моніторингу обладнання;
3. проєктування інформаційного та програмного забезпечення інтелектуальної системи моніторингу технічного стану обладнання;
4. розробка механізмів збору, збереження та оброблення телеметричних даних від IoT-пристроїв;
5. реалізація алгоритмів машинного навчання для виявлення аномалій і прогнозування можливих відмов обладнання.
6. розробка засобів візуалізації, аналітики та формування сповіщень про критичні стани обладнання.

7. тестування функціональних можливостей системи та оцінювання ефективності прогнозування технічного стану обладнання.

Дата видачі завдання «19» грудня 2025 р.

Керівник бакалаврської кваліфікаційної роботи _____ (підпис) **Таїсія САЯПІНА**

Завдання взяв до виконання _____ (підпис) **Богдан ЧЕХІРКІН**

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

1. API – Application Programming Interface, програмний інтерфейс взаємодії між системами.
2. BI – Business Intelligence, бізнес-аналітика.
3. CRM – Customer Relationship Management, система управління взаємодією з клієнтами.
4. БД – база даних.
5. DWH – Data Warehouse, сховище даних.
6. SQL – Structured Query Language, мова структурованих запитів до бази даних.
7. SQLite – вбудована реляційна система керування базами даних.
8. ETL – Extract, Transform, Load, процес вилучення, перетворення та завантаження даних.
9. ELT – Extract, Load, Transform, процес вилучення, завантаження та подальшого перетворення даних.
10. REST API – архітектурний підхід до організації програмного інтерфейсу обміну даними.
11. JSON – JavaScript Object Notation, текстовий формат обміну структурованими даними.
12. CSV – Comma-Separated Values, формат табличних даних із розділенням значень комами.
13. PDF – Portable Document Format, формат електронного документа.
14. XLSX – формат електронних таблиць Microsoft Excel.
15. UI – User Interface, користувацький інтерфейс.
16. UX – User Experience, користувацький досвід.
17. UML – Unified Modeling Language, уніфікована мова моделювання.
18. ER-діаграма – Entity-Relationship Diagram, діаграма «сутність-зв’язок».

19. DFD – Data Flow Diagram, діаграма потоків даних.

20. BPMN – Business Process Model and Notation, нотація моделювання бізнес-процесів.

21. KPI – Key Performance Indicator, ключовий показник ефективності.

22. CTR – Click-Through Rate, показник клікабельності рекламного оголошення.

23. CPC – Cost Per Click, вартість одного кліка.

24. CPA – Cost Per Action, вартість цільової дії.

25. CPL – Cost Per Lead, вартість залучення одного ліда.

26. ROAS – Return on Ad Spend, окупність витрат на рекламу.

27. ROI – Return on Investment, рентабельність інвестицій.

28. CVR – Conversion Rate, коефіцієнт конверсії.

29. Lead / лід – потенційний клієнт, отриманий у результаті рекламної взаємодії.

30. Conversion / конверсія – виконання користувачем цільової дії.

31. CRUD – Create, Read, Update, Delete, базові операції створення, читання, оновлення та видалення даних.

32. OLAP – Online Analytical Processing, технологія багатовимірної аналітичної обробки даних.

33. RBAC – Role-Based Access Control, модель керування доступом на основі ролей.

34. JWT – JSON Web Token, токен для безпечної передачі даних між клієнтом і сервером.

35. OAuth2 – протокол авторизації для безпечного доступу до ресурсів.

36. HTTPS – HyperText Transfer Protocol Secure, захищений протокол передавання даних.

37. TLS – Transport Layer Security, криптографічний протокол захисту передавання даних.

ВСТУП

Мільйони рекламних взаємодій щоденно генеруються у цифровому середовищі, формуючи великі обсяги різнорідних даних у вигляді кліків, показів, конверсій, витрат, поведінкових метрик користувачів та атрибуційних подій. У сфері медіабаїнгу ці дані надходять із численних джерел - рекламних платформ, систем веб-аналітики та CRM, що створює складність їх узгодженого аналізу та інтерпретації. Основною проблемою для фахівців із медіабаїнгу є не лише отримання доступу до цих даних, а й їх ефективна консолідація, оброблення та перетворення у релевантну аналітичну інформацію для прийняття управлінських рішень.

Традиційні інструменти аналітики часто обмежуються базовими показниками ефективності (CTR, CPC, CPA, ROAS), не забезпечуючи комплексного аналізу взаємозв’язків між каналами трафіку, креативами, аудиторіями та часовими факторами. Відсутність єдиної інтегрованої системи бізнес-аналітики призводить до втрати частини інформаційної цінності даних, ускладнює виявлення закономірностей та знижує ефективність оптимізації рекламних кампаній.

Це обумовлює необхідність розроблення інформаційних систем бізнес-аналітики для медіабаїнгу, здатних автоматизовано інтегрувати дані з різних джерел, виконувати їх оброблення в реальному часі, здійснювати багатовимірний аналіз та формувати аналітичні висновки із застосуванням сучасних методів оброблення даних, машинного навчання та статистичного аналізу. Такі системи створюють основу для підвищення точності прийняття рішень, оптимізації рекламних бюджетів і зростання ефективності маркетингових кампаній.

Метою кваліфікаційної роботи є розроблення та наукове обґрунтування інформаційної системи бізнес-аналітики для медіабаїнгу, яка забезпечує автоматизований збір, консолідацію, аналіз і візуалізацію даних рекламних

кампаній з метою підвищення ефективності управління маркетинговими процесами.

Для досягнення поставленої мети необхідно вирішити такі основні **завдання:**

1. виконати системний аналіз предметної області бізнес-аналітики у сфері медіабаїнгу, визначити сучасні підходи, їх переваги та недоліки;
2. розробити концептуальну архітектуру системи збору, інтеграції та оброблення даних із рекламних платформ, CRM та систем веб-аналітики;
3. створити модель консолідації даних із різнорідних джерел із використанням ETL/ELT та event-driven підходів;
4. розробити методи розрахунку ключових показників ефективності (KPI) та їх аналітичної інтерпретації;
5. реалізувати механізми виявлення відхилень, трендів та закономірностей у даних рекламних кампаній;
6. розробити вебінтерфейс для візуалізації аналітики, формування звітів та підтримки прийняття рішень;
7. провести оцінювання ефективності системи на основі реальних або наближених до реальних даних.

Об’єктом дослідження є процес збору, оброблення та аналітичної інтерпритації даних рекламних кампаній у сфері медіабаїнгу.

Предметом дослідження є методи, моделі та програмні засоби побудови інформаційних систем бізнес-аналітики, що забезпечують інтеграцію, аналіз і візуалізацію маркетингових даних.

Методи дослідження включають застосування системного аналізу, математичного моделювання, статистичного аналізу та оброблення даних. Для оброблення інформації використано підходи ETL/ELT, методи агрегації та нормалізації даних, а також алгоритми аналізу часових рядів і виявлення аномалій. Для розрахунку ефективності застосовано метрики CTR, CPC, CPA,

CPL, ROAS та інші показники маркетингової аналітики. Проєктування системи виконано з використанням UML-діаграм і моделей даних ER.

Практична значущість одержаних результатів полягає у розробленні інтегрованої інформаційної системи бізнес-аналітики для медіабаїнгу, яка поєднує механізми збору та консолідації даних із різних джерел із аналітичними інструментами виявлення закономірностей і відхилень у режимі, наближеному до реального часу. Запропоновано підхід до динамічного аналізу ефективності рекламних кампаній із урахуванням багатоканальної взаємодії користувачів та часових факторів. Розроблено модель автоматизованого формування аналітичних звітів і візуалізацій, що підвищує швидкість і точність прийняття управлінських рішень. Відмінністю запропонованого підходу є інтеграція потокової обробки подій, аналітики KPI та механізмів контролю відхилень у єдиному програмному середовищі.

1 СИСТЕМНИЙ АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Опис предметної області

Предметна область інформаційної системи бізнес-аналітики для медіабаїнгу охоплює процеси збору, інтеграції, оброблення та аналітичної інтерпретації даних рекламних кампаній з метою підвищення ефективності управління маркетинговою діяльністю. Основними об’єктами є користувачі аналітичного контуру (медіабаєр, аналітик, керівник), зовнішні джерела даних (рекламні платформи, CRM, системи веб-аналітики) та функціональні модулі системи.

На рисунку 1.1 подано функціональну модель предметної області у вигляді DFD-діаграми, яка відображає структуру основних процесів, сховищ даних і інформаційних потоків.

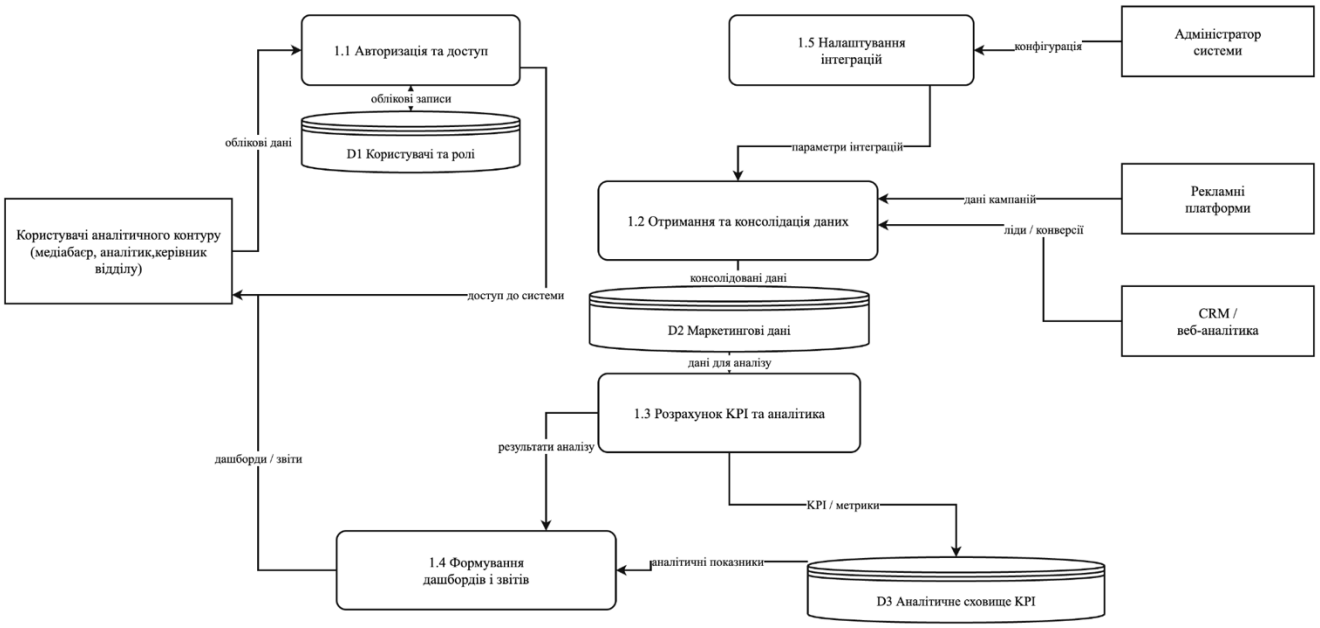

Рисунок 1.1 – DFD-модель предметної області інформаційної системи бізнес-аналітики для медіабаїнгу.

Вхідними даними системи є показники рекламних кампаній (покази, кліки, витрати), а також дані про ліди та конверсії, що надходять із зовнішніх джерел. У межах системи ці дані проходять етапи інтеграції та консолідації, після чого зберігаються у централізованому сховищі маркетингових даних. Подальша

обробка передбачає розрахунок ключових показників ефективності (CTR, CPC, CPA, CPL, ROAS) та формування аналітичних метрик, які акумулюються в аналітичному сховищі KPI.

Результати оброблення використовуються для формування дашбордів і звітів, що забезпечують візуалізацію аналітичної інформації та підтримку прийняття управлінських рішень. Доступ до функціоналу системи здійснюється через механізми авторизації з урахуванням ролей користувачів, а налаштування підключення до зовнішніх джерел реалізується через підсистему конфігурації інтеграцій.

Логіка функціонування системи формалізована за допомогою BPMN-діаграми, наведеної на рисунку 1.2, яка відображає послідовність виконання операцій аналізу рекламних кампаній.

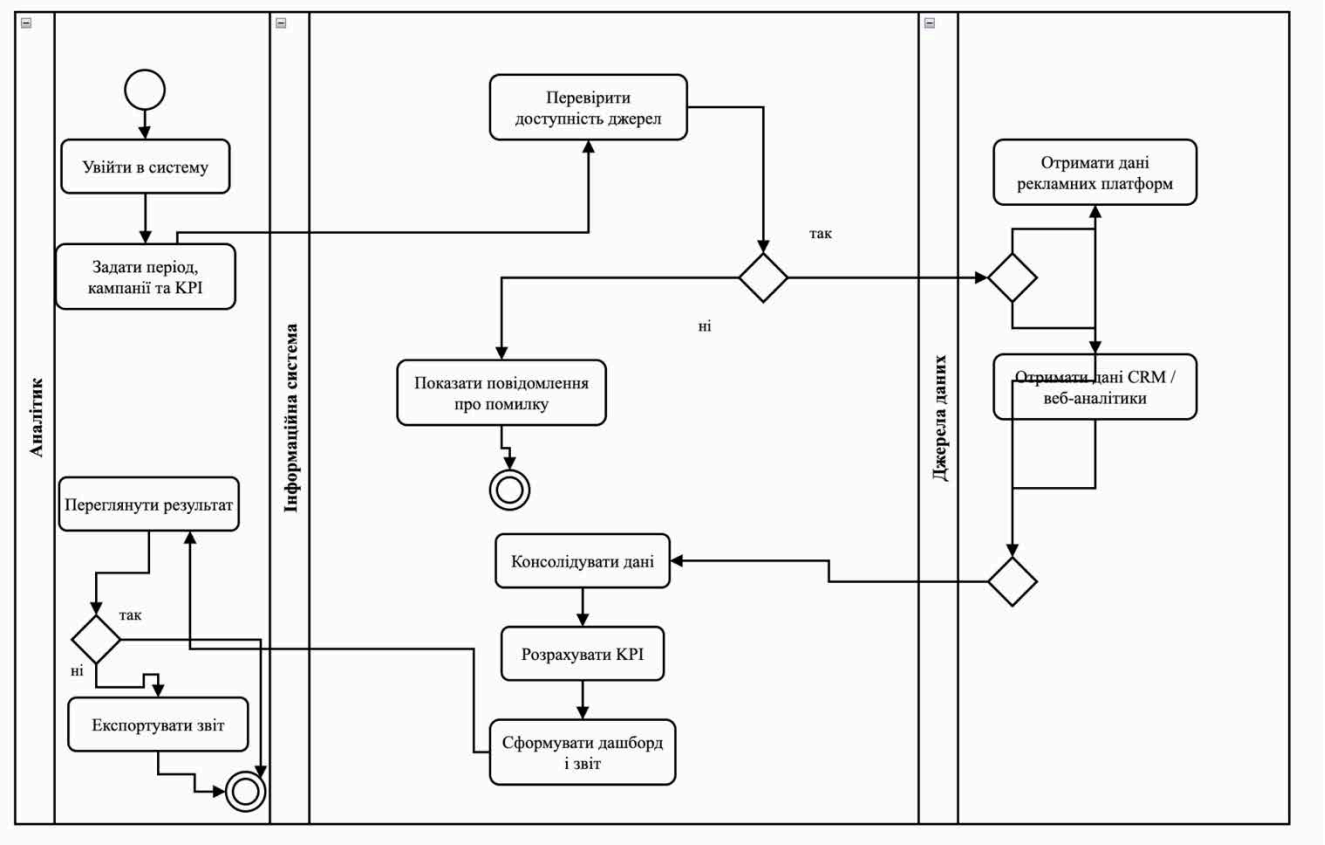

Рисунок 1.2 – BPMN-діаграма бізнес-процесу аналізу рекламних кампаній в інформаційній системі бізнес-аналітики для медіабаїнгу.

Процес включає авторизацію користувача, задання параметрів аналізу, перевірку доступності джерел даних, отримання та консолідацію інформації,

розрахунок KPI та формування аналітичних результатів із подальшою візуалізацією або експортом звітів. У разі відсутності доступу до джерел або помилок оброблення система формує відповідні повідомлення.

Для узагальнення структури предметної області основні компоненти та інформаційні потоки системи наведено у таблиці 1.1.

Таблиця 1.1 – Основні компоненти та інформаційні потоки предметної області

№	Компонент системи	Вхідні дані	Вихідні дані	Основні функції
1	Авторизація та доступ	Облікові дані	Сесія користувача	Аутентифікація, контроль доступу
2	Інтеграція даних	Дані рекламних платформ, CRM	Консолідовані дані	Збір, уніфікація
3	Сховище маркетингових даних	Потоки даних	Структуровані дані	Зберігання
4	Розрахунок KPI	Дані кампаній	KPI	Обчислення метрик
5	Сховище KPI	KPI	Агреговані дані	Аналітичне зберігання
6	Дашборди і звіти	KPI	Візуалізація	Представлення
7	Налаштування інтеграцій	Параметри API	Конфігурації	Управління підключеннями

Як видно з таблиці 1.1, предметна область включає взаємопов’язані модулі, що забезпечують повний цикл оброблення маркетингових даних – від їх отримання до формування аналітичних результатів. Така структура дозволяє інтегрувати різнорідні джерела даних, забезпечити їх узгоджену обробку та підвищити ефективність прийняття управлінських рішень у сфері медіабаїнгу.

1.2 Аналіз існуючих рішень

У процесі дослідження предметної області було проведено порівняльний аналіз сучасних програмних рішень, які застосовуються для керування рекламними кампаніями, збору маркетингових даних, розрахунку KPI та побудови аналітичних звітів. До таких рішень належать Google Ads, Meta Ads Manager, TikTok Ads Manager, Google Analytics 4 та Microsoft Power BI. На рисунках 1.3–1.7

подано приклади інтерфейсів зазначених платформ, що відображають їхні функціональні можливості у сфері медіабаїнгу та бізнес-аналітики.

На рисунку 1.3 показано приклад аналітичного звіту Google Ads, у якому відображаються базові показники рекламних кампаній: кількість показів, кліків, витрати та конверсії. Платформа Google Ads є одним із основних інструментів медіабаїнгу, оскільки дозволяє створювати пошукові, медійні, відео- та торгові рекламні кампанії. Її перевагою є детальна статистика за ключовими словами, оголошеннями, аудиторіями та витратами. Водночас система орієнтована переважно на власну рекламну екосистему Google і не забезпечує повної консолідації даних із Meta Ads, TikTok Ads, CRM та інших джерел без додаткових інтеграцій.

Рисунок 1.3 – Приклад аналітичного звіту Google Ads із показниками ефективності рекламних кампаній

На рисунку 1.4 наведено інтерфейс аналітики Meta Ads, що використовується для аналізу рекламних кампаній у Facebook та Instagram. Система надає дані про витрати, покази, кліки, CTR, CPC, CPA, покупки та інші показники ефективності. Meta Ads Manager є зручним інструментом для керування рекламними кампаніями всередині екосистеми Meta, однак має обмеження щодо комплексного аналізу багатоканальних кампаній. Дані з інших

рекламних платформ або CRM не консолідуються автоматично, що ускладнює побудову єдиного аналітичного контуру для медіабаїнгу.

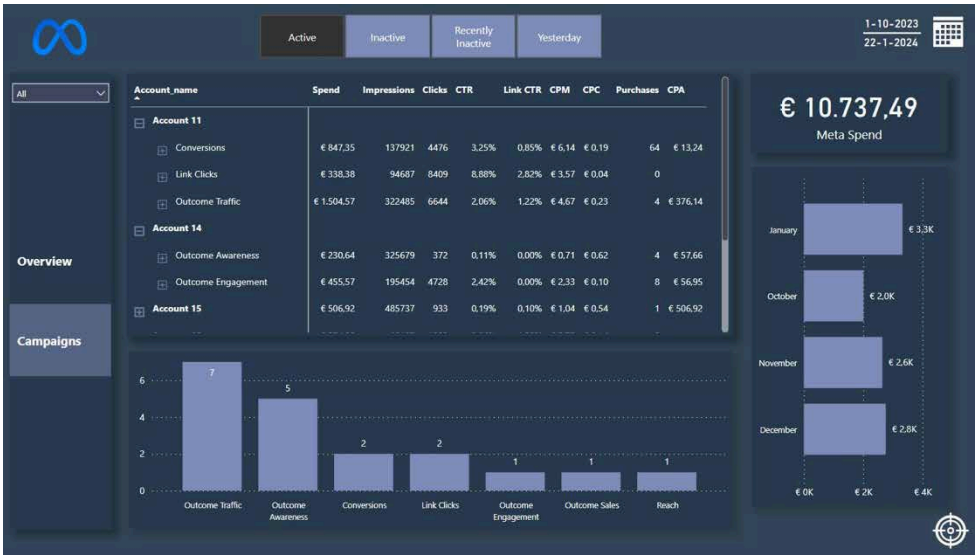

Рисунок 1.4 – Інтерфейс аналітики Meta Ads із показниками витрат, кліків, CTR, CPC та CPA

На рисунку 1.5 подано фрагмент інтерфейсу TikTok Ads Manager, який забезпечує створення та моніторинг рекламних кампаній у TikTok. Платформа дозволяє відстежувати витрати, покази, кліки, статуси груп оголошень та загальну динаміку кампаній. Основною перевагою TikTok Ads Manager є орієнтація на відеоконтент і молоду аудиторію, що робить його важливим каналом у сучасному медіабаїнгу. Проте система має обмежені можливості для наскрізної бізнес-аналітики та потребує додаткового об’єднання з даними CRM, веб-аналітики й інших рекламних платформ.

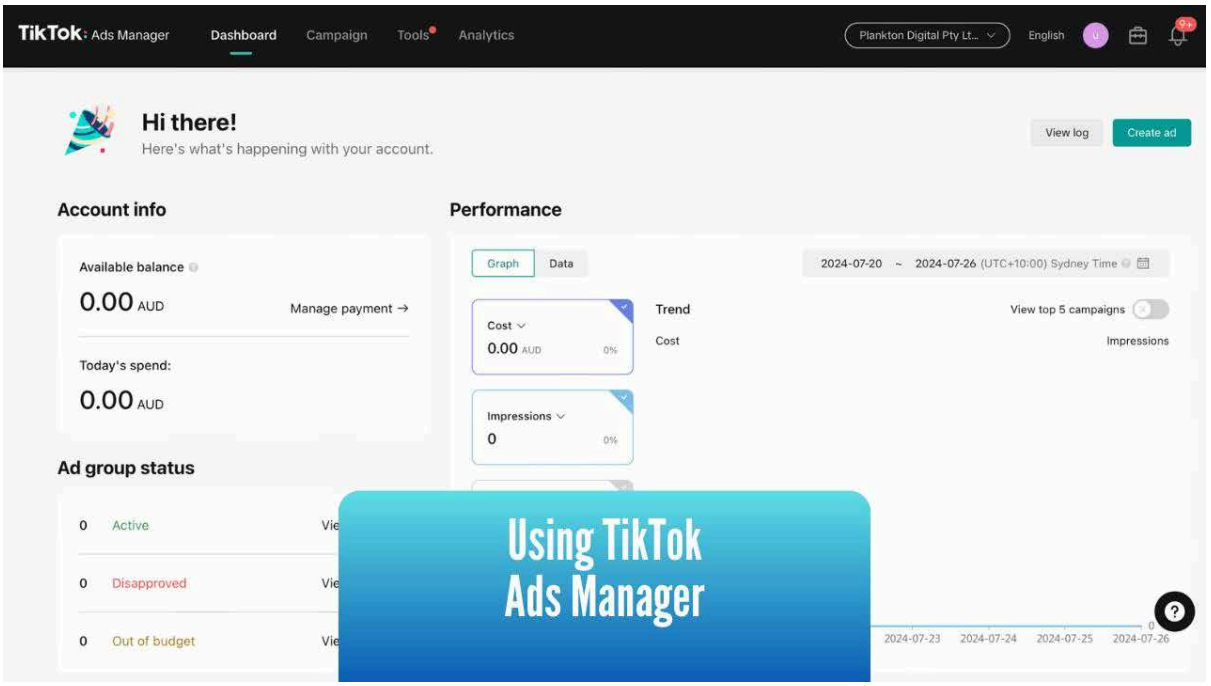

Рисунок 1.5 – Інтерфейс TikTok Ads Manager для моніторингу стану рекламного акаунта та кампаній

На рисунку 1.6 представлено приклад рекламної аналітики в середовищі Google Analytics 4 або суміжного PPC-дашборду, де відображаються витрати та вартість кліка за різними джерелами трафіку. Google Analytics 4 забезпечує аналіз поведінки користувачів на сайті, подій, конверсій і джерел залучення. Це рішення є важливим для оцінювання післяклікової поведінки користувачів, однак воно не завжди надає повну картину витрат, ставок і рекламних налаштувань без інтеграції з рекламними кабінетами. Тому GA4 доцільно розглядати як джерело аналітичних даних, а не як повноцінну систему бізнес-аналітики для медіабаїнгу.

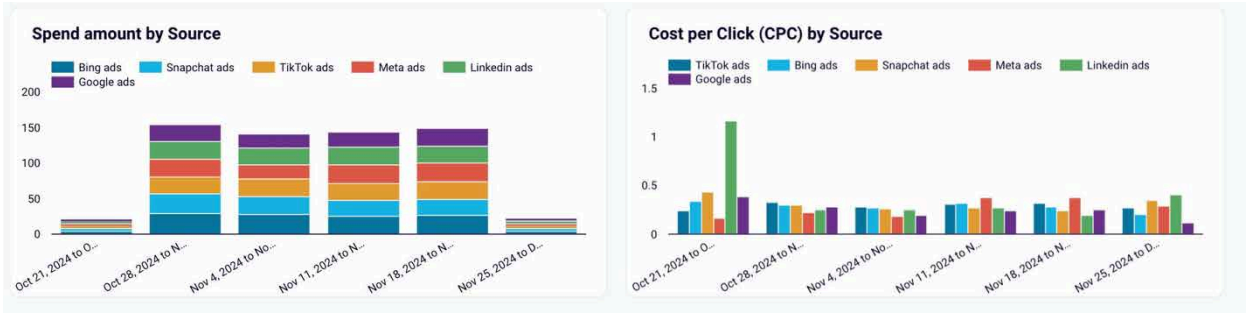

Рисунок 1.6 – Приклад аналітики рекламних джерел і вартості кліка в середовищі Google Analytics 4 / PPC-дашборду

На рисунку 1.7 зображено приклад Digital Ads Dashboard у Microsoft Power BI. Це рішення дає змогу створювати інтерактивні дашборди, об’єднувати дані з різних джерел, будувати графіки, таблиці, теплові карти та аналітичні зрізи. Microsoft Power BI має широкі можливості для візуалізації та бізнес-аналітики, але не є спеціалізованою системою саме для медіабаїнгу. Для його ефективного використання необхідно окремо налаштовувати моделі даних, інтеграції, формули KPI та правила оновлення інформації.

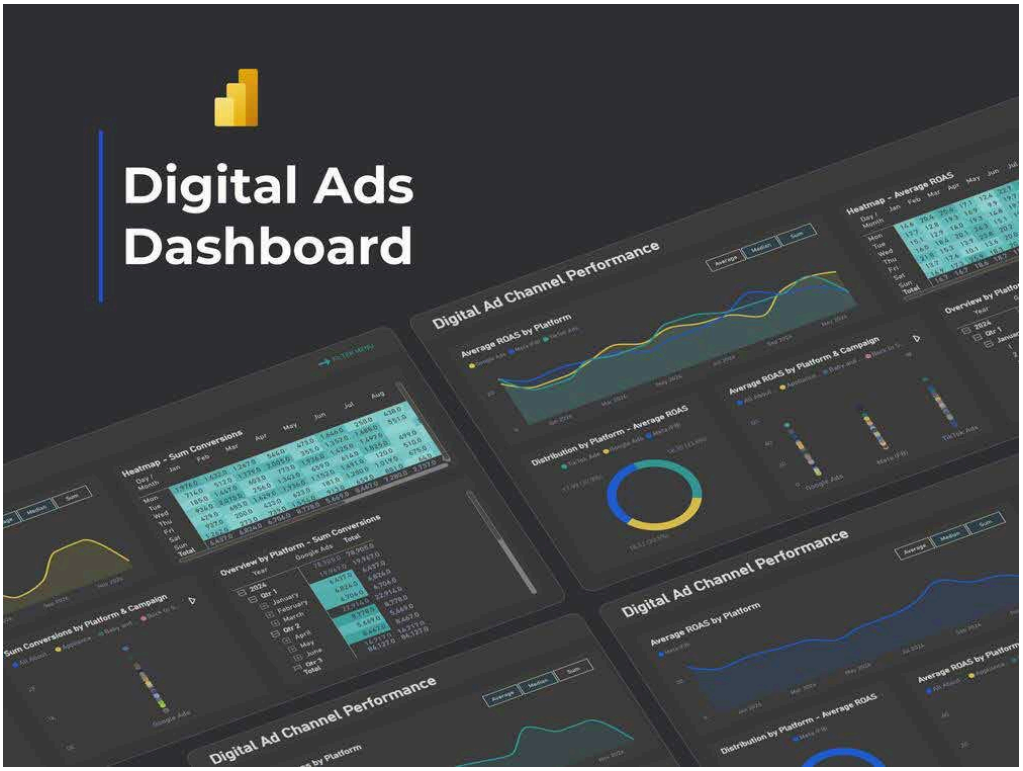

Рисунок 1.7 – Приклад Digital Ads Dashboard у Microsoft Power BI для аналізу рекламних каналів

Для узагальнення результатів порівняння основних функціональних можливостей розглянутих систем наведено таблицю 1.2.

Таблиця 1.2 – Порівняльний аналіз існуючих систем і пропонованого рішення

№	Система	Інтеграція рекламних даних	Розрахунок KPI	Дашборди і звіти	Багатоканальна аналітика	Основні особливості
1	Google Ads	Є, у межах Google	Є	Є	Обмежена	Ефективне керування кампаніями Google,

						але слабка підтримка зовнішніх джерел

Продовження таблиці 1.2

2	Meta Ads Manager	Є, у межах Meta	Є	Є	Обмежена	Детальна аналітика Facebook та Instagram, але без повної консолідації з іншими каналами
3	TikTok Ads Manager	Є, у межах TikTok	Є	Частково	Обмежена	Зручний інструмент для відеореклами, але з обмеженою наскрізною аналітикою
4	Google Analytics 4	Є, переважно веб-аналітика	Частково	Є	Середня	Сильний аналіз поведінки користувачів, але потребує інтеграції з рекламними кабінетами
5	Microsoft Power BI	Є, через конектори	Є, після налаштування	Є	Висока	Потужна BI-візуалізація, але не є готовим спеціалізованим рішенням для медіабаїнгу
6	Розроблювана система	Є, для рекламних платформ, CRM і веб-аналітики	Є	Є	Висока	Єдина система для консолідації даних, розрахунку KPI, контролю відхилень і формування звітів

Як видно з таблиці 1.2, наявні рішення забезпечують окремі функції, необхідні для медіабаїнгу: керування рекламними кампаніями, збір статистики, аналіз поведінки користувачів або побудову дашбордів. Проте більшість із них працює в межах власної екосистеми або потребує складного налаштування інтеграцій. Це створює потребу в розробленні спеціалізованої інформаційної системи бізнес-аналітики для медіабаїнгу, яка об’єднує дані рекламних платформ, CRM і веб-аналітики, автоматизує розрахунок KPI та забезпечує формування дашбордів і звітів у єдиному аналітичному середовищі.

1.3 Моделювання предметної області інформаційної системи бізнес-аналітики для медіабаїнгу

Моделювання предметної області інформаційної системи бізнес-аналітики для медіабаїнгу виконано на основі побудови UML-діаграм прецедентів, активності та послідовності, які формують узгоджену модель функціонування системи. Сукупність цих діаграм дозволяє формалізувати ролі користувачів, логіку оброблення маркетингових даних та взаємодію підсистем у часовому аспекті.

На рисунку 1.8 подано діаграму прецедентів інформаційної системи бізнес-аналітики для медіабаїнгу, яка відображає взаємодію користувачів із функціональними можливостями системи та зовнішніми джерелами маркетингових даних. У моделі окремо конкретизовано джерела даних рекламних кампаній і конверсій: Google Ads, Meta Ads, TikTok Ads, CRM-систему та Google Analytics / веб-аналітику.

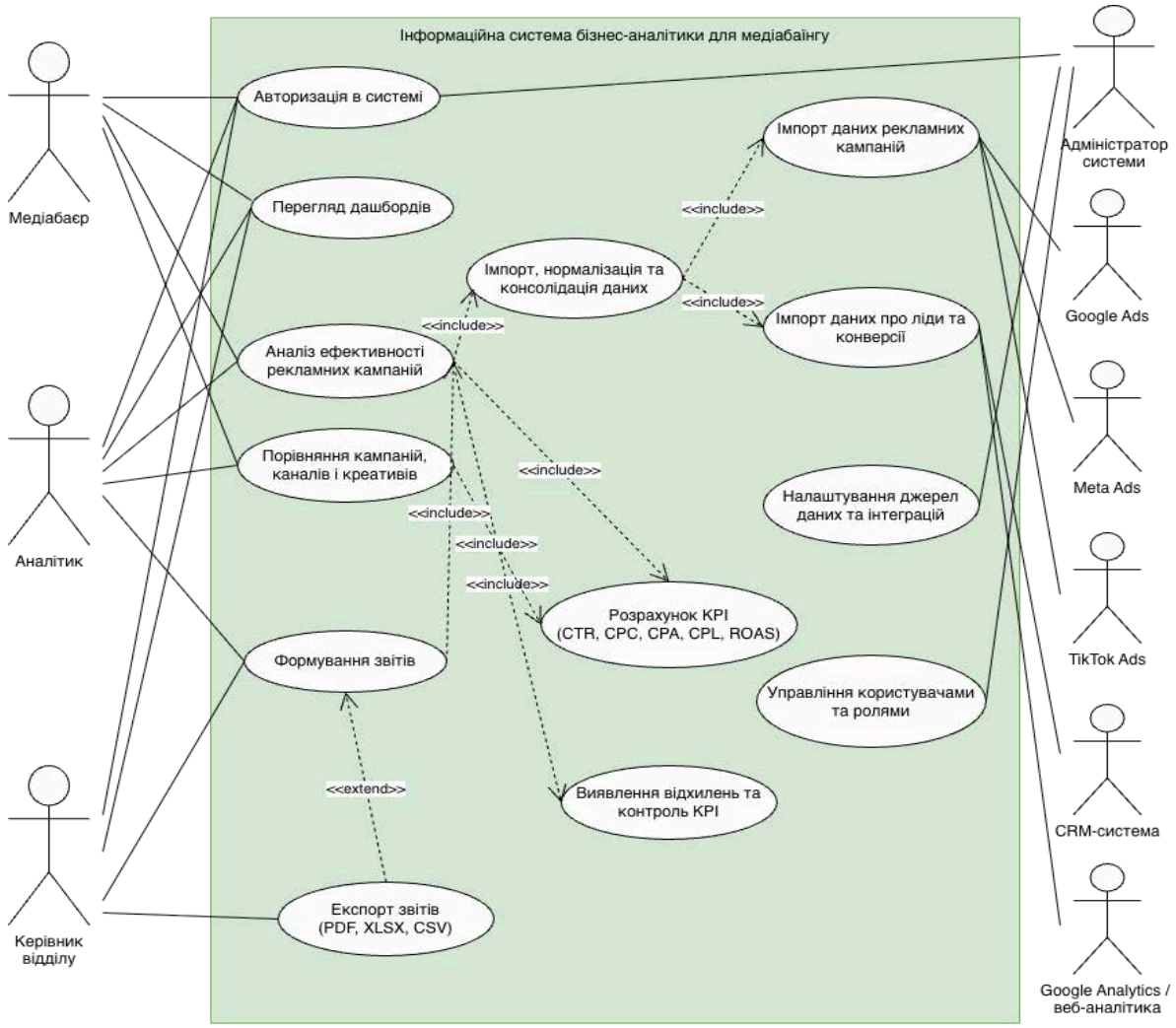

Рисунок 1.8 – Діаграма прецедентів інформаційної системи бізнес-аналітики для медіабаїнгу.

Основними користувацькими акторами системи є медіабаєр, аналітик, керівник відділу та адміністратор системи. Медіабаєр, аналітик і керівник відділу використовують систему для авторизації, перегляду дашбордів, аналізу ефективності рекламних кампаній, порівняння кампаній, каналів і креативів, а також формування та експорту звітів. Адміністратор системи відповідає за налаштування джерел даних та інтеграцій, а також за управління користувачами й ролями.

Зовнішніми акторами системи виступають Google Ads, Meta Ads, TikTok Ads, CRM-система та Google Analytics / веб-аналітика. Дані з Google Ads, Meta Ads і TikTok Ads використовуються для отримання інформації про рекламні кампанії, зокрема покази, кліки, витрати та параметри кампаній. Дані з

CRM-системи та Google Analytics / веб-аналітики застосовуються для отримання інформації про ліди, конверсії та поведінкові показники користувачів.

Центральним прецедентом є аналіз ефективності рекламних кампаній. Він включає імпорт, нормалізацію та консолідацію даних із зовнішніх джерел, розрахунок ключових показників ефективності, а також виявлення відхилень і контроль KPI. Така логіка відображає послідовність аналітичного процесу: спочатку система отримує дані з рекламних платформ, CRM та веб-аналітики, далі приводить їх до узгодженої структури, після чого виконує аналіз ефективності та розрахунок KPI.

Формування звітів базується на результатах аналізу ефективності рекламних кампаній та розрахованих KPI. Експорт звітів у форматах PDF, XLSX і CSV розглядається як додатковий сценарій, що розширює процес формування звітів. Це дозволяє відокремити основну аналітичну логіку від допоміжної функції вивантаження результатів.

На рисунку 1.9 наведено діаграму активності, яка відображає логіку виконання основного бізнес-процесу аналізу рекламних кампаній.

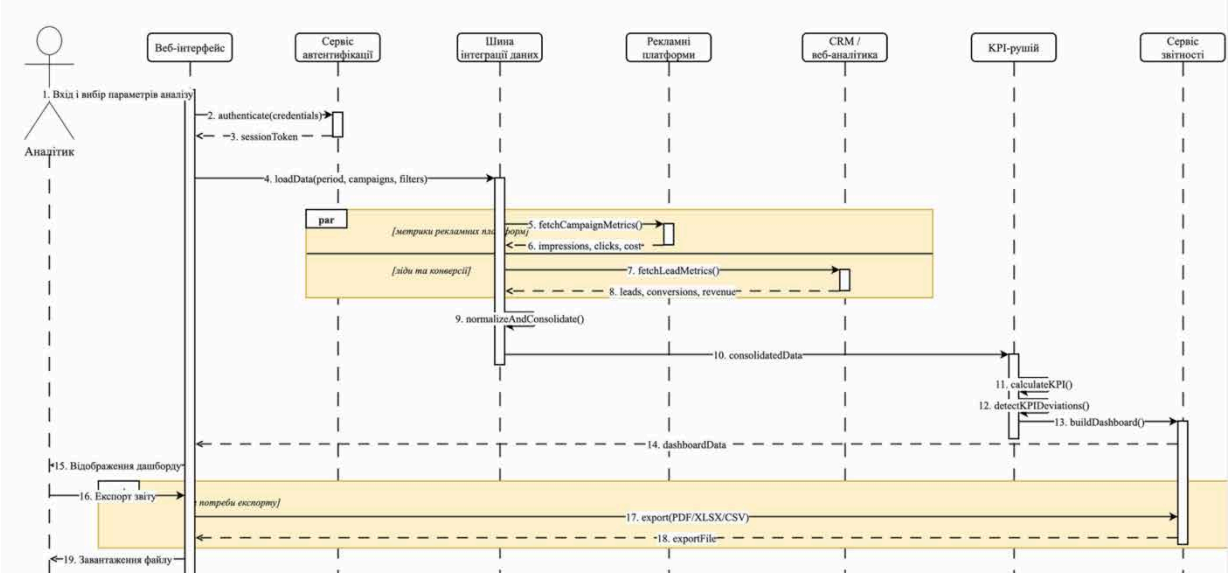

Рисунок 1.9 – Діаграма активності процесу аналізу рекламних кампаній у системі бізнес-аналітики.

Процес починається з введення облікових даних і авторизації користувача, після чого формується аналітичний запит із заданими параметрами (період,

кампанії, KPI). Наступним етапом є перевірка доступності джерел даних. У разі недоступності формується повідомлення про помилку, що завершує процес.

У випадку успішної перевірки виконується імпорт і консолідація даних із рекламних платформ і CRM/веб-аналітики. Після цього здійснюється розрахунок KPI та виявлення відхилень, що дозволяє оцінити ефективність рекламних кампаній. Результати передаються до модуля візуалізації, де формується дашборд. Завершальним етапом є перегляд результатів користувачем та, за потреби, експорт звіту.

На рисунку 1.10 подано діаграму послідовності, яка відображає динамічну взаємодію між компонентами системи.

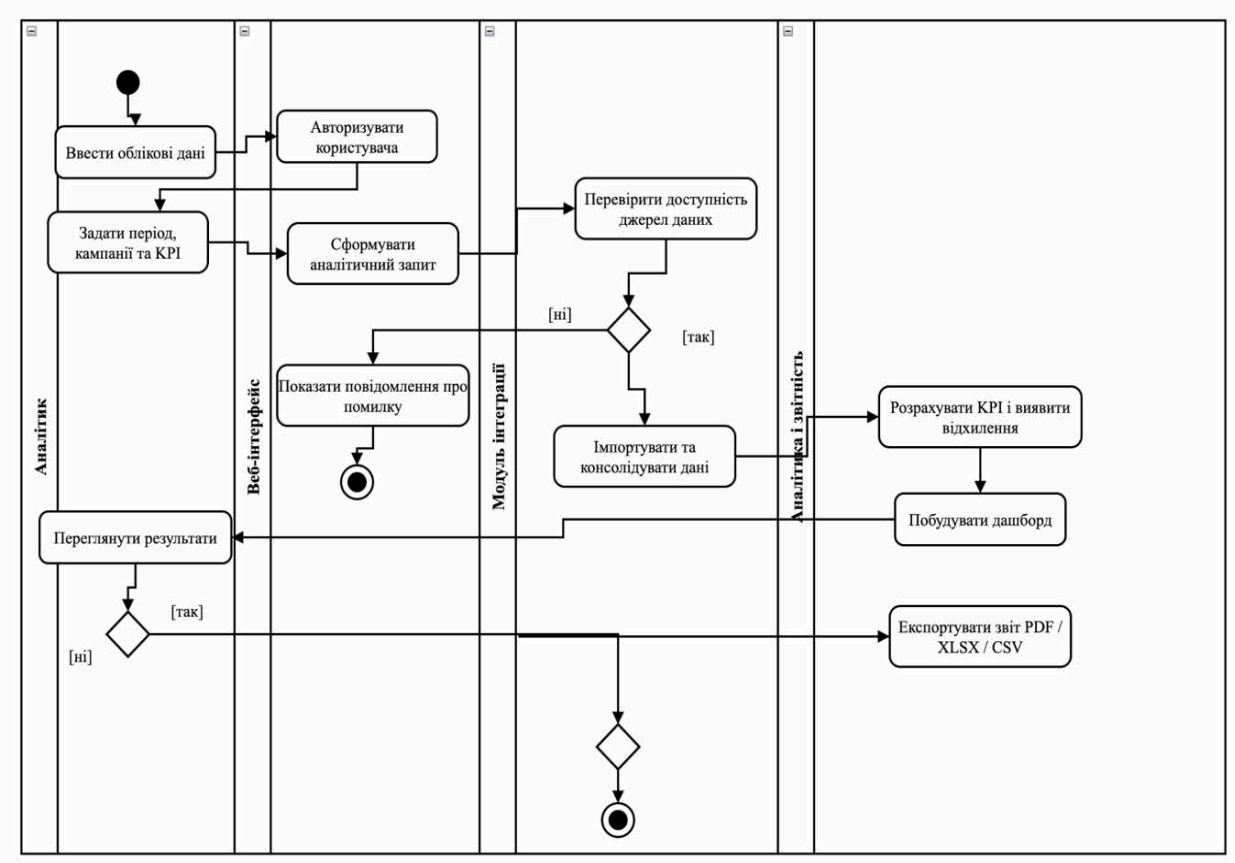

Рисунок 1.10 – Діаграма послідовності взаємодії підсистем інформаційної системи бізнес-аналітики для медіабаїнгу.

Діаграма демонструє взаємодію між веб-інтерфейсом, сервісом автентифікації, шиною інтеграції даних, рекламними платформами, CRM/веб-аналітикою, KPI-рушієм та сервісом звітності. Після авторизації

користувача інтерфейс формує запит на отримання даних, який передається до модуля інтеграції.

Інтеграційний модуль паралельно отримує дані з рекламних платформ (покази, кліки, витрати) та CRM/веб-аналітики (ліди, конверсії), після чого виконується їх нормалізація та консолідація. Отримані дані передаються до KPI-рушія, який виконує розрахунок показників ефективності та виявлення відхилень.

Результати оброблення передаються до сервісу звітності для побудови дашбордів. У разі запиту на експорт формується файл звіту у відповідному форматі (PDF, XLSX, CSV), який передається користувачу. Така послідовність взаємодій забезпечує повний цикл оброблення даних у системі.

Побудовані UML-діаграми забезпечують цілісне уявлення про структуру та логіку функціонування інформаційної системи бізнес-аналітики для медіабаїнгу. Вони дозволяють формалізувати вимоги до системи, визначити взаємозв’язки між її компонентами та створити основу для подальшого проєктування програмної архітектури.

1.4 Аналіз вимог системи

Аналіз вимог є основою проєктування інформаційної системи бізнес-аналітики для медіабаїнгу, яка поєднує механізми інтеграції маркетингових даних, аналітичної обробки та візуалізації результатів. Система повинна забезпечувати ефективне оброблення великих обсягів даних рекламних кампаній, підтримку багатоканальної аналітики, гнучку інтеграцію з зовнішніми джерелами та високий рівень надійності й безпеки.

Сукупність вимог до системи поділяється на три основні групи: функціональні, нефункціональні та вимоги до безпеки.

Функціональні вимоги визначають ключові можливості системи, необхідні для реалізації повного циклу аналітики рекламних кампаній – від отримання

даних до формування управлінських рішень. Перелік функціональних вимог наведено в таблиці 1.3.

Таблиця 1.3 – Функціональні вимоги до системи бізнес-аналітики для медіабаїнгу

№	Вимога	Опис функції	Очікуваний результат
1	Інтеграція даних із рекламних платформ	Підключення до API Google Ads, Meta Ads, TikTok Ads	Отримання даних про кампанії (покази, кліки, витрати)
2	Інтеграція з CRM і веб-аналітикою	Отримання даних про ліди, конверсії, доходи	Повна картина ефективності кампаній
3	Консолідація даних	Очищення, нормалізація та об’єднання даних	Єдиний узгоджений масив даних
4	Зберігання маркетингових даних	Організація централізованого сховища (DWH)	Структуровані історичні дані
5	Розрахунок KPI	Обчислення CTR, CPC, CPA, CPL, ROAS	Отримання аналітичних показників
6	Аналіз ефективності кампаній	Порівняння кампаній, каналів, періодів	Виявлення ефективних стратегій
7	Виявлення відхилень	Аналіз аномалій та змін KPI	Контроль якості рекламних кампаній
8	Формування дашбордів	Побудова графіків, таблиць, візуалізацій	Зручне представлення даних
9	Генерація звітів	Формування звітів у форматах PDF, XLSX, CSV	Документування результатів
10	Управління користувачами	Авторизація, ролі (аналітик, медіабаєр, адміністратор)	Контроль доступу до системи
11	Налаштування інтеграцій	Конфігурація API та джерел даних	Гнучкість підключення

Функціональні вимоги формують логічну основу системи та відображають процеси, представлені на DFD- і BPMN-діаграмах: отримання даних, їх консолідацію, розрахунок KPI та формування звітності.

Нефункціональні вимоги визначають якісні характеристики системи, що забезпечують її ефективність, масштабованість і стабільність роботи. Перелік нефункціональних вимог наведено в таблиці 1.4.

Таблиця 1.4 – Нефункціональні вимоги до системи

№	Вимога	Опис	Показник або обмеження

1	Продуктивність	Час обробки аналітичного запиту	≤ 2 с
2	Масштабованість	Підтримка великої кількості кампаній і користувачів	≥ 1000 користувачів
3	Надійність	Безперервна робота системи	Доступність ≥ 99,9 %

Продовження таблиці 1.4

4	Сумісність	Підтримка різних API і форматів	REST, JSON, CSV
5	Юзабіліті	Зручність використання інтерфейсу	Інтуїтивний UI
6	Модульність	Незалежність компонентів	Мікросервісна структура
7	Тестованість	Автоматизація тестування	Покриття ≥ 80 %
8	Логування	Запис подій і помилок	Централізоване логування

Нефункціональні вимоги забезпечують стабільність і ефективність роботи системи в умовах великого обсягу маркетингових даних і високої інтенсивності запитів.

Вимоги до безпеки спрямовані на забезпечення захисту даних користувачів, конфіденційності інформації та цілісності системи. Їх перелік наведено в таблиці 1.5.

Таблиця 1.5 – Вимоги до безпеки системи

№	Вимога	Опис	Рівень захисту
1	Автентифікація та авторизація	Використання JWT, OAuth2, RBAC	Високий
2	Шифрування даних	TLS 1.2/1.3, шифрування БД	Високий
3	Захист бази даних	Резервне копіювання, реплікація	Високий
4	Аудит дій	Логування користувачів і API	Середній
5	Захист API	Валідація запитів, захист від XSS/SQLi	Високий
6	Відновлення після збоїв	Backup і disaster recovery	Високий

Як видно з таблиць 1.3–1.5, сформовані вимоги охоплюють усі ключові аспекти функціонування системи: від інтеграції даних до їх аналітичної обробки та захисту. Такий підхід дозволяє створити масштабовану, надійну та ефективну інформаційну систему бізнес-аналітики для медіабаїнгу, що забезпечує підтримку прийняття рішень на основі даних.

1.5 Постановка завдання

Постановка завдання полягає у визначенні структури, призначення та взаємозв’язків компонентів інформаційної системи бізнес-аналітики для медіабаїнгу, яка забезпечує інтеграцію, оброблення та аналітичну інтерпретацію даних рекламних кампаній із різних джерел. Метою системи є автоматизація процесу збору, консолідації та аналізу маркетингових даних з подальшим формуванням аналітичних висновків, дашбордів і звітів для підтримки прийняття управлінських рішень.

Система повинна забезпечувати повний цикл роботи з даними рекламних кампаній, включаючи інтеграцію з рекламними платформами, CRM та системами веб-аналітики, обчислення ключових показників ефективності (KPI) та візуалізацію результатів. У загальному вигляді процес функціонування системи включає п’ять основних етапів: отримання даних, консолідація, аналітична обробка, формування результатів та їх представлення користувачу.

Вхідні дані системи охоплюють:

– дані рекламних кампаній із платформ (покази, кліки, витрати, ставки);

– інформацію про ліди, конверсії та доходи з CRM-систем;

– дані веб-аналітики (сеанси, поведінка користувачів, джерела трафіку);

– параметри аналітичного запиту (період, кампанії, KPI);

– конфігураційні дані інтеграцій і налаштувань системи.

Вихідні дані системи включають:

– розраховані KPI (CTR, CPC, CPA, CPL, ROAS);

– аналітичні показники ефективності рекламних кампаній;

– дашборди з візуалізацією даних (графіки, таблиці, діаграми);

– звіти у форматах PDF, XLSX, CSV;

– інформацію про відхилення та аномалії в показниках кампаній.

Основні підзадачі, які вирішує система:

1. збір та інтеграція даних із рекламних платформ, CRM та систем веб-аналітики за допомогою API;

2. автоматизована консолідація, очищення та нормалізація даних;
3. збереження структурованих даних у централізованому сховищі;
4. розрахунок ключових показників ефективності рекламних кампаній;
5. аналіз результатів і виявлення відхилень KPI;
6. формування дашбордів для візуалізації аналітичної інформації;
7. генерація звітів і забезпечення можливості їх експорту;
8. управління користувачами та правами доступу;
9. налаштування інтеграцій і параметрів системи.

Умови функціонування системи:

– робота в клієнт-серверній або хмарній архітектурі з можливістю масштабування;

– доступ через вебінтерфейс із підтримкою ролей користувачів (медіабаєр, аналітик, адміністратор);

– інтеграція із зовнішніми системами через REST API та обмін даними у форматі JSON;

– забезпечення обробки запитів у реальному або наближеному до реального часу;

– використання захищених каналів передачі даних (HTTPS, TLS);

– ведення журналів подій і резервне копіювання даних.

Поставлене завдання передбачає розроблення інформаційної системи, що реалізує замкнутий цикл оброблення маркетингових даних – від їх отримання до формування аналітичних результатів і підтримки прийняття рішень. Результатом має стати масштабоване, надійне та функціонально завершене програмне рішення, яке забезпечує ефективний аналіз рекламних кампаній і підвищує якість управління маркетинговими процесами у сфері медіабаїнгу.

1.6 Висновки до першого розділу

У першому розділі виконано комплексний аналіз предметної області інформаційної системи бізнес-аналітики для медіабаїнгу та визначено основні

підходи до оброблення й аналізу маркетингових даних. Обґрунтовано актуальність створення інтегрованого рішення, яке забезпечує консолідацію даних із різних джерел, їх аналітичну обробку та підтримку прийняття управлінських рішень.

У ході дослідження предметної області встановлено, що сучасні процеси медіабаїнгу характеризуються високою динамічністю, значними обсягами різнорідних даних і необхідністю їх оперативного аналізу. Визначено основні компоненти системи, інформаційні потоки та взаємозв’язки між підсистемами, що формують повний цикл оброблення даних – від їх отримання до формування аналітичних результатів.

Проведений аналіз існуючих програмних рішень (Google Ads, Meta Ads Manager, TikTok Ads Manager, Google Analytics 4, Microsoft Power BI) показав, що більшість із них реалізують лише окремі функції аналітики або працюють у межах власних екосистем. Це обмежує можливості комплексного аналізу ефективності рекламних кампаній та підтверджує доцільність розроблення спеціалізованої системи бізнес-аналітики для медіабаїнгу.

На основі побудованих UML-діаграм (прецедентів, активності та послідовності) формалізовано структуру системи, визначено ролі користувачів, основні сценарії взаємодії та логіку виконання бізнес-процесів. Це дозволило узгодити функціональні можливості системи з її архітектурними компонентами та інформаційними потоками.

У результаті аналізу вимог визначено функціональні, нефункціональні та безпекові характеристики системи, які забезпечують її ефективність, масштабованість, надійність і захищеність. Сформовано перелік ключових функцій, серед яких інтеграція даних, їх консолідація, розрахунок KPI, візуалізація результатів і формування звітності.

У першому розділі сформовано теоретичну та методологічну основу для подальшого проєктування інформаційної системи бізнес-аналітики для медіабаїнгу. Отримані результати є базою для розроблення архітектури системи,

вибору технологічних рішень та реалізації програмного забезпечення у наступних розділах роботи.

2 ПРОЕКТУВАННЯ ІНФОРМАЦІЙНОГО ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Логічна модель даних у вигляді ER-діаграми

Логічна модель даних інформаційної системи бізнес-аналітики для медіабаїнгу забезпечує формалізоване подання структури маркетингових даних, необхідних для збору, зберігання, оброблення та аналізу ефективності рекламних кампаній. ER-модель, наведена на рисунку 2.1, побудована з урахуванням принципів нормалізації до третьої нормальної форми, що дозволяє мінімізувати надлишковість даних, уникнути аномалій оновлення та забезпечити цілісність інформації.

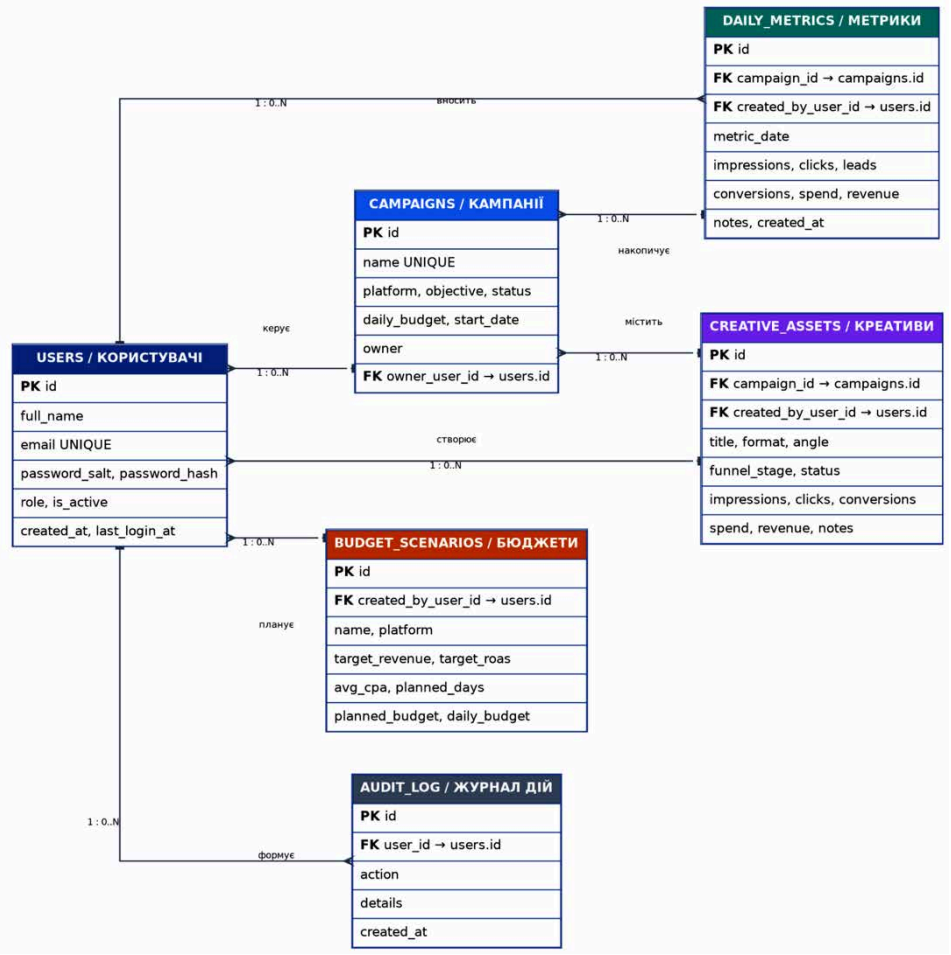

Рисунок 2.1 – ER-діаграма логічної моделі даних інформаційної системи бізнес-аналітики для медіабаїнгу.

У моделі виділено ключові сутності, що відображають основні компоненти предметної області: користувачів системи, рекламні кампанії, метрики ефективності, креативи, бюджетні сценарії та журнал дій. Такий поділ дозволяє забезпечити чітке розмежування функціональних зон системи та підтримує масштабованість аналітичних процесів.

Центральне місце в моделі займають дані про користувачів, які визначають права доступу до системи, виконують керування кампаніями, формують аналітичні запити та здійснюють контроль результатів. Через механізм зовнішніх ключів забезпечується зв’язок користувачів із іншими компонентами системи, що дозволяє відстежувати їхню участь у створенні та зміні даних.

Основою аналітичної структури є інформація про рекламні кампанії, яка містить параметри їх функціонування, зокрема платформу, цілі, статус, бюджетні обмеження та часові характеристики. Кампанії виступають ядром моделі, оскільки саме до них прив’язуються інші дані, що формують аналітичну картину.

Динамічна складова моделі представлена показниками ефективності, що накопичуються у часовому розрізі та включають кількість показів, кліків, лідів, конверсій, витрат і доходу. Такий підхід забезпечує можливість проведення аналізу ефективності рекламних кампаній у динаміці, а також виявлення трендів і відхилень ключових показників.

Окремий блок даних становлять рекламні креативи, які характеризують контентну складову кампаній. Їх збереження та аналіз дозволяють оцінювати вплив окремих рекламних матеріалів на результативність кампаній і оптимізувати маркетингові стратегії.

Для підтримки процесів планування у моделі передбачено зберігання бюджетних сценаріїв, що включають цільові показники ефективності, заплановані витрати та часові параметри. Це дає змогу здійснювати прогнозування результатів і порівняння фактичних показників із плановими.

Додатково модель містить механізм журналювання дій користувачів, який забезпечує фіксацію змін у системі, контроль виконаних операцій і підтримку

вимог безпеки. Такий підхід підвищує прозорість функціонування системи та дозволяє виконувати аудит дій користувачів.

Модель передбачає використання зв’язків типу «один-до-багатьох» між основними сутностями, що відповідає реальній структурі предметної області. Наприклад, один користувач може керувати кількома кампаніями, одна кампанія може містити багато креативів і записів метрик, а також один користувач може створювати декілька бюджетних сценаріїв.

Особливістю логічної моделі є чітке розділення статичних і динамічних даних. До статичних належать користувачі, кампанії та бюджетні сценарії, тоді як динамічні дані представлені метриками та журналом дій. Такий підхід дозволяє ефективно обробляти великі обсяги даних у часовому розрізі та виконувати складні аналітичні запити.

Для узагальнення структури логічної моделі основні сутності та їх функціональне призначення наведено в таблиці 2.1.

Таблиця 2.1 – Основні сутності логічної моделі даних системи

Сутність	Ключові атрибути	Функціональне призначення
Users	id, email, role	Зберігання даних користувачів і прав доступу
Campaigns	id, name, platform, objective, status, daily_budget, start_date, owner, owner_user_id	Зберігання активних, тестових, призупинених та архівних рекламних кампаній. Логічне архівування реалізується через значення Archived у полі status
Daily_Metrics	id, campaign_id, created_by_user_id, metric_date, impressions, clicks, leads, conversions, spend, revenue, notes, created_at	Зберігання фактичних динамічних метрик рекламних кампаній у часовому розрізі. KPI розраховуються на основі цих метрик

Продовження таблиці 2.1

Creative_Assets	id, campaign_id, format	Аналіз ефективності креативів

Budget_Scenarios	id, created_by_user_id, name, platform, target_revenue, target_roas, avg_cpa, planned_days, planned_budget, daily_budget	Зберігання планових параметрів і цільових показників, які мають умовно статичний характер у межах сценарію планування
Audit_Log	id, user_id, action	Аудит дій користувачів

У логічній моделі даних KPI не виділяються як окрема сутність, оскільки вони є розрахунковими аналітичними показниками. Їх значення формуються на основі фактичних метрик рекламних кампаній, що зберігаються в сутності Daily_Metrics. До таких метрик належать покази, кліки, ліди, конверсії, витрати та дохід. На їх основі система розраховує CTR, CPC, CPA, CPL, ROAS та інші показники ефективності.

Фактичні показники мають динамічний характер, оскільки змінюються залежно від дати, рекламної кампанії, каналу трафіку та періоду аналізу. Тому вони зберігаються у часовому розрізі з прив’язкою до конкретної кампанії та дати. Натомість планові показники мають умовно статичний характер у межах заданого сценарію планування. Вони зберігаються в сутності Budget_Scenarios і використовуються для прогнозування бюджету, очікуваної кількості конверсій та цільового ROAS.

Архів попередніх рекламних кампаній у системі реалізовано не через окрему таблицю, а через поле status у сутності Campaigns. Значення Archived використовується для позначення завершених або неактивних кампаній, які залишаються в загальному реєстрі кампаній і можуть бути використані для подальшого аналізу. Історичні метрики таких кампаній зберігаються в Daily_Metrics, що дозволяє аналізувати результати як активних, так і архівних кампаній.

API рекламних платформ, CRM та веб-аналітики в межах моделі розглядаються як джерела первинного отримання або синхронізації даних. Водночас аналітична обробка виконується на основі локально збережених даних, що дає змогу не залежати від постійного звернення до зовнішніх сервісів. У

реалізованій версії системи дані кампаній і метрик зберігаються в локальній базі даних, а інтеграція із зовнішніми API може бути розглянута як напрям подальшого розвитку.

Як видно з таблиці 2.1, логічна модель даних забезпечує структуроване та нормалізоване представлення інформації, що дозволяє ефективно реалізувати аналітичні запити, зокрема аналіз ефективності кампаній у часовому розрізі, порівняння каналів і креативів, а також оцінювання відповідності фактичних результатів запланованим показникам.

Запропонована ER-модель створює основу для побудови масштабованої інформаційної системи бізнес-аналітики для медіабаїнгу, забезпечуючи цілісність даних, ефективність їх оброблення та підтримку прийняття рішень на основі аналітики.

2.2 Діаграма класів і кооперації розроблювальної системи

Діаграма класів, подана на рисунку 2.2, відображає логічну структуру об’єктної моделі інформаційної системи бізнес-аналітики для медіабаїнгу. Вона визначає основні класи предметної області, їхні атрибути, операції та асоціативні зв’язки між ними. Модель побудована з урахуванням принципів об’єктного проєктування, що дозволяє розмежувати відповідальність між класами та забезпечити подальше масштабування системи без порушення її внутрішньої узгодженості.

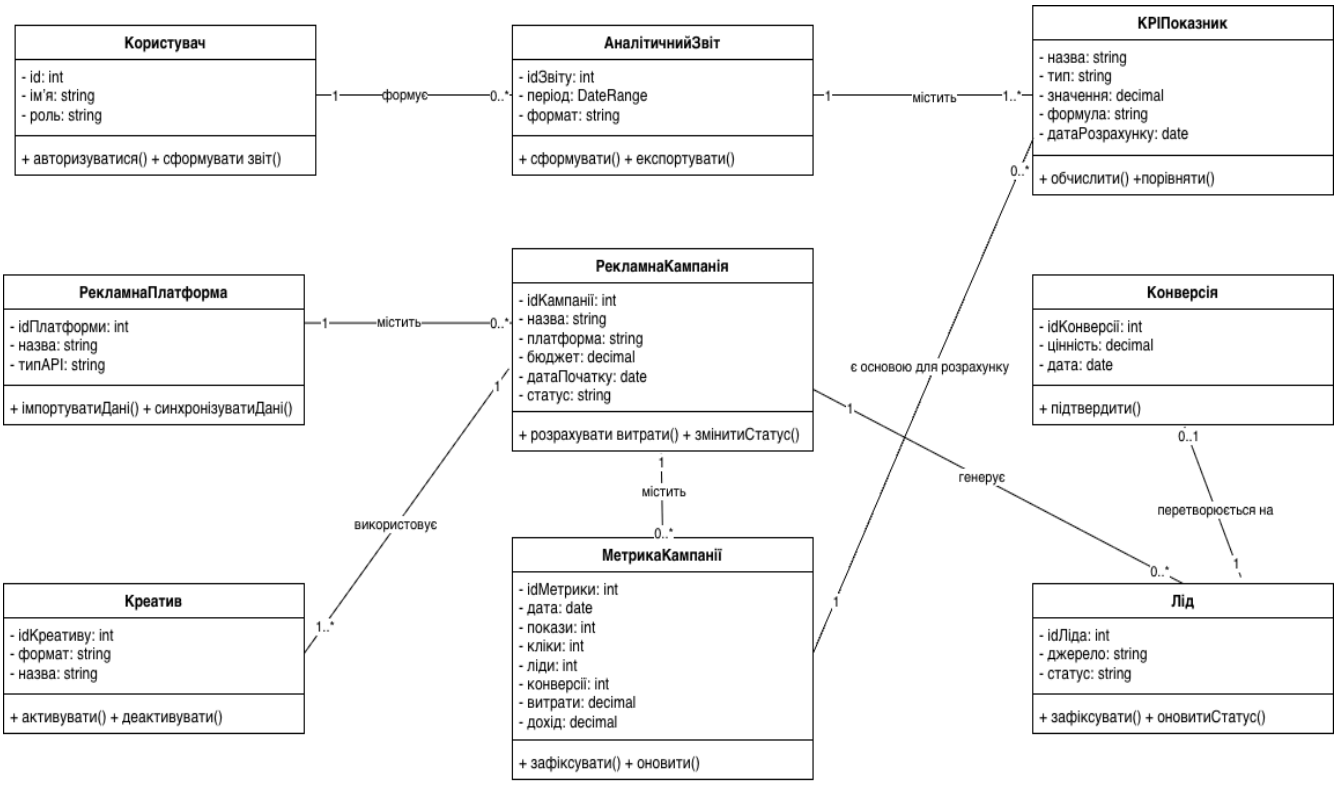

Рисунок 2.2 – Діаграма класів інформаційної системи бізнес-аналітики для медіабаїнгу

На рисунку 2.2 наведено діаграму класів інформаційної системи бізнес-аналітики для медіабаїнгу. Вона відображає основні об’єктні сутності системи, їхні атрибути, операції та взаємозв’язки між користувачами, рекламними платформами, кампаніями, креативами, лідами, конверсіями, метриками, KPI та аналітичними звітами.

Основними класами моделі є Користувач, РекламнаПлатформа, РекламнаКампанія, Креатив, МетрикиКампанії, Лід, Конверсія, КРІПоказник та АналітичнийЗвіт. Клас Користувач відповідає за авторизацію та формування звітів. Клас РекламнаПлатформа описує джерело рекламних даних і забезпечує імпорт та синхронізацію інформації. Клас РекламнаКампанія є центральним об’єктом моделі, оскільки об’єднує дані про бюджет, платформу, статус кампанії, пов’язані креативи, ліди, конверсії, фактичні метрики та розраховані KPI.

Клас МетрикиКампанії використовується для зберігання фактичних показників рекламної кампанії у часовому розрізі. До таких показників належать покази, кліки, ліди, конверсії, витрати та дохід. Саме ці дані є основою для

розрахунку KPI. Клас KPIПоказник у діаграмі класів відображає не окрему таблицю бази даних, а об’єктну модель розрахованого аналітичного показника. Його значення формується на основі фактичних метрик кампанії за відповідний період. Такий підхід узгоджується з логічною моделлю даних, у якій метрики зберігаються в Daily_Metrics, а KPI розраховуються під час аналітичної обробки.

Клас РекламнаКампанія містить атрибут статус, який визначає поточний стан кампанії. Значення Archived використовується для логічного архівування завершених або неактивних кампаній. Тому архів у системі не виділяється в окремий клас, а реалізується через стан об’єкта рекламної кампанії та збереження пов’язаних із нею історичних метрик. Це дозволяє аналізувати результати як активних, так і завершених рекламних кампаній без дублювання структури даних.

Клас АналітичнийЗвіт агрегує результати аналізу рекламних кампаній і містить набір розрахованих KPI-показників. Один звіт може містити декілька KPI, що характеризують ефективність кампаній за обраний період. Звіт може бути сформований користувачем та експортований у визначеному форматі.

Зв’язки між класами відображають реальну логіку предметної області. Один користувач може формувати декілька аналітичних звітів, одна рекламна платформа може містити багато рекламних кампаній, а кожна кампанія може використовувати кілька креативів, генерувати ліди та накопичувати фактичні метрики, на основі яких розраховуються KPI-показники. Лід у межах аналітичного процесу може перетворюватися на конверсію, яка надалі враховується під час обчислення результативності кампанії. Така структура забезпечує повний цикл аналізу рекламної активності - від джерела трафіку до фінального показника ефективності.

На рисунку 2.3 наведено кооперацію процесу формування аналітичного звіту.

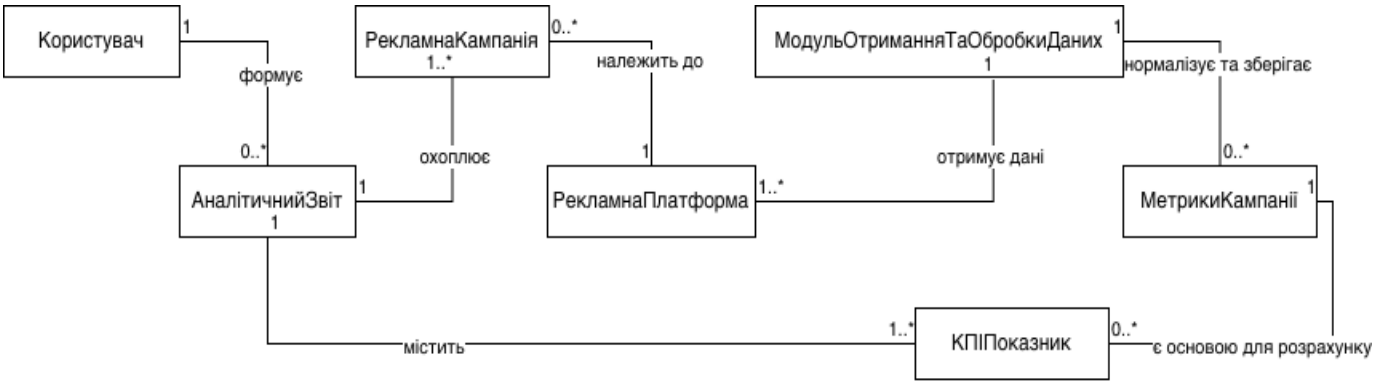

Рисунок 2.3 – Кооперація формування аналітичного звіту рекламної кампанії

На рисунку 2.3 наведено кооперацію класів, що відображає взаємодію між користувачем, аналітичним звітом, рекламною кампанією, рекламною платформою, модулем отримання та обробки даних, метриками кампанії та KPI-показниками. Користувач ініціює формування аналітичного звіту, який охоплює одну або кілька рекламних кампаній. Кожна рекламна кампанія належить до певної рекламної платформи, з якої модуль отримання та обробки даних отримує вхідні показники.

Модуль отримання та обробки даних виконує нормалізацію, підготовку та збереження фактичних метрик кампанії. До таких метрик належать покази, кліки, ліди, конверсії, витрати та дохід. На основі збережених метрик формуються KPI-показники, які надалі включаються до аналітичного звіту. Така кооперація демонструє повний шлях даних: від рекламної платформи до метрик, розрахунку KPI та формування звітності.

На рисунку 2.4 подано кооперацію аналізу рекламної кампанії та креативів.

Рисунок 2.4 – Кооперація аналізу рекламної кампанії, креативів і KPI-показників

На рисунку 2.4 наведено кооперацію класів, що деталізує взаємозв’язок між рекламною платформою, рекламною кампанією, креативами, лідами, конверсіями, метриками кампанії та KPI-показниками. Рекламна платформа виступає джерелом даних про рекламні кампанії, а модуль отримання та обробки даних забезпечує отримання, нормалізацію та збереження фактичних показників.

Рекламна кампанія є центральним об’єктом аналізу, у межах якого генеруються ліди, використовуються креативи та накопичуються метрики кампанії. Ліди можуть перетворюватися на конверсії, що враховується під час оцінювання результативності кампанії. Креативи впливають не безпосередньо на KPI, а на фактичні метрики кампанії, зокрема покази, кліки, ліди, конверсії, витрати та дохід.

Метрики кампанії є основою для розрахунку KPI-показників, таких як CTR, CPC, CPA, CPL і ROAS. Таким чином, у кооперації явно відображено повний шлях даних: від рекламної платформи через модуль отримання та обробки даних до фактичних метрик, розрахунку KPI та подальшого використання результатів в аналітичній системі.

На рисунку 2.5 представлено кооперацію процесу оброблення лідів і конверсій.

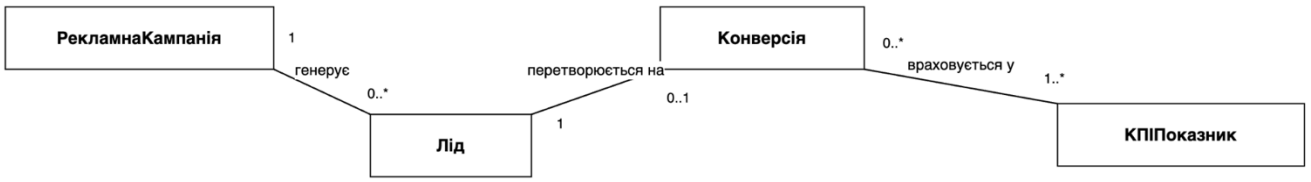

Рисунок 2.5 – Кооперація оброблення лідів, конверсій і KPI-показників

Кооперація відображає зв’язок між рекламною кампанією, лідами, конверсіями та KPI. Кампанія генерує ліди, частина яких перетворюється на конверсії. Конверсії враховуються під час розрахунку показників ефективності, зокрема CPA, CPL і ROAS. Такий підхід дозволяє простежити шлях користувача від рекламної взаємодії до результативної дії та забезпечує коректну оцінку ефективності рекламного бюджету.

Для узагальнення ролей основних класів у моделі розроблювальної системи наведено таблицю 2.2.

Таблиця 2.2 – Функціональні ролі класів у моделі системи

Клас	Призначення	Ключові обмеження	Роль в аналітичному циклі
Користувач	Представлення користувача системи та його ролі	Один користувач може формувати багато звітів	Ініціює аналітичні запити
РекламнаПлатформа	Джерело рекламних даних	Одна платформа містить багато кампаній	Забезпечує імпорт даних
РекламнаКампанія	Основна одиниця аналізу реклами	Кампанія належить одній платформі та може мати багато креативів, лідів і записів метрик	Об’єднує дані про кампанію; через атрибут статус підтримує логічне архівування
Креатив	Рекламний матеріал кампанії	Креатив пов’язаний із кампанією	Впливає на ефективність реклами

Продовження таблиці 2.2

Лід	Потенційний клієнт, отриманий із кампанії	Лід може мати не більше однієї конверсії	Відображає проміжний результат реклами

Конверсія	Завершена цільова дія	Конверсія формується з ліда	Враховується у фінальній ефективності
КПІПоказник	Об’єктна модель розрахованого показника ефективності	Формується на основі метрик кампанії та використовується у звітах	Забезпечує кількісну оцінку результатів без виділення KPI в окрему таблицю бази даних
АналітичнийЗвіт	Узагальнений результат аналізу	Містить одну або кілька кампаній і KPI	Формує звітність для прийняття рішень
МетрикиКампанії	Зберігання фактичних показників рекламної кампанії у часовому розрізі	Одна кампанія може мати багато записів метрик за різні дати	Є основою для розрахунку KPI

Як видно з таблиці 2.2, класи моделі охоплюють основні елементи аналітичного циклу медіабаїнгу: джерела даних, рекламні кампанії, креативи, ліди, конверсії, KPI та звітність. Це забезпечує логічну узгодженість між об’єктною моделлю, ER-діаграмою та бізнес-процесами системи.

Діаграма класів і кооперації, наведені на рисунках 2.2–2.5, формують об’єктну основу розроблювальної системи. Вони визначають структуру основних класів, їхні зв’язки та типові сценарії взаємодії, що є необхідним для подальшого проєктування компонентів, програмної архітектури та реалізації інформаційної системи бізнес-аналітики для медіабаїнгу.

2.3 Діаграма компонентів

Структурна організація програмних модулів інформаційної системи бізнес-аналітики для медіабаїнгу наведена на рисунку 2.6, який відображає логічне групування компонентів та їх взаємодію в межах сервіс-орієнтованої архітектури. Діаграма компонентів фіксує розподіл функціональності між підсистемами, що забезпечують інтеграцію даних, їх консолідацію, аналітичну обробку та формування звітності, без деталізації внутрішньої реалізації окремих модулів.

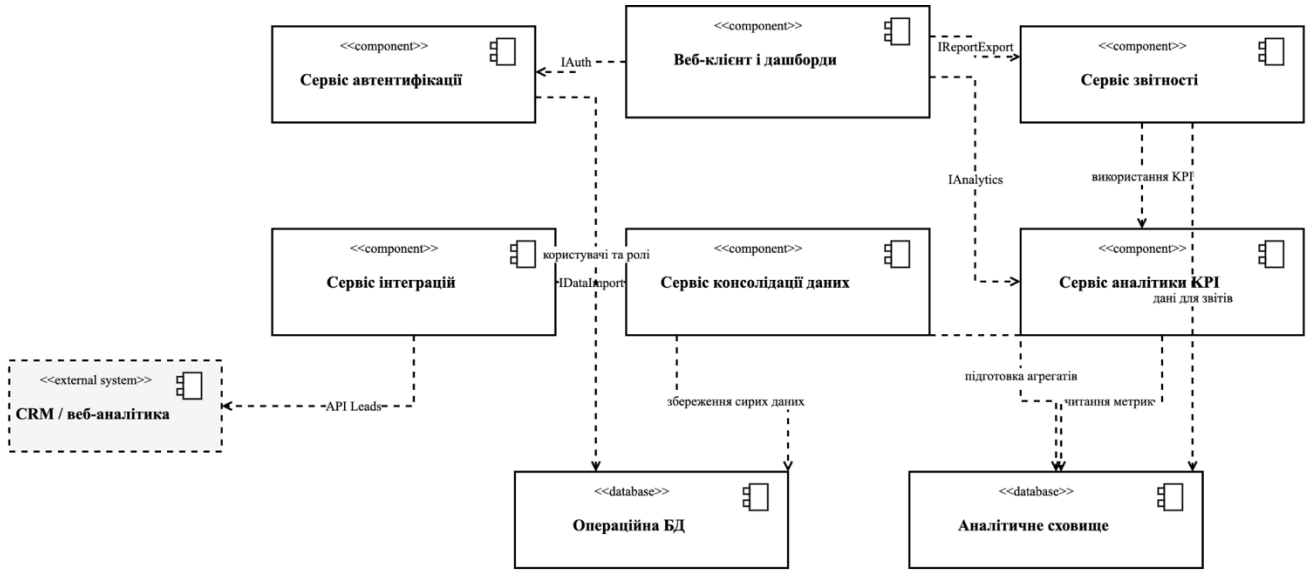

Рисунок 2.6 – Діаграма компонентів інформаційної системи бізнес-аналітики для медіабаїнгу

Представлена модель включає такі ключові компоненти: веб-клієнт і дашборди, сервіс автентифікації, сервіс інтеграції, сервіс консолідації даних, сервіс аналітики KPI, сервіс звітності, операційну базу даних та аналітичне сховище. Взаємодія між компонентами здійснюється через інтерфейси API, що забезпечує слабку зв’язність та незалежність окремих підсистем.

Веб-клієнт і дашборди реалізують прикладний рівень системи та забезпечують взаємодію користувача з аналітичними даними. Через сервіс автентифікації здійснюється перевірка облікових даних і контроль доступу до функціональності системи. Сервіс інтеграції відповідає за отримання даних із зовнішніх джерел, зокрема рекламних платформ, CRM та систем веб-аналітики, забезпечуючи їх уніфікацію та передачу до внутрішніх компонентів.

Сервіс консолідації даних виконує об’єднання, очищення та нормалізацію отриманої інформації з подальшим збереженням у операційній базі даних. Такий підхід дозволяє забезпечити цілісність і узгодженість даних перед їх аналітичною обробкою. Аналітичне сховище використовується для зберігання агрегованих показників і історичних даних, що застосовуються під час побудови звітів і виконання аналітичних запитів.

Сервіс аналітики KPI є ключовим елементом системи, оскільки виконує обчислення показників ефективності рекламних кампаній, виявлення відхилень і підготовку агрегованих даних для подальшого використання. Сервіс звітності забезпечує формування аналітичних звітів і їх експорт у різних форматах (PDF, XLSX, CSV), використовуючи результати роботи аналітичного модуля.

Така структура компонентів дозволяє реалізувати наскрізний процес оброблення даних – від їх надходження із зовнішніх систем до формування аналітичних результатів і візуалізації для користувача. Розділення компонентів за функціональними ролями забезпечує масштабованість, можливість незалежного розвитку модулів та підвищення надійності системи.

Узагальнені характеристики ролей і функцій компонентів наведено в таблиці 2.3.

Таблиця 2.3 – Узагальнена структурна характеристика компонентів системи

Рівень	Характеристика	Призначення
Прикладний рівень	Веб-клієнт, дашборди	Взаємодія користувача з системою та візуалізація даних
Інтеграційний рівень	API, сервіс інтеграції	Отримання даних із зовнішніх джерел
Оброблювальний рівень	Сервіс консолідації даних	Очищення, нормалізація та уніфікація даних
Аналітичний рівень	Сервіс аналітики KPI	Розрахунок показників ефективності
Рівень звітності	Сервіс звітності	Формування та експорт аналітичних звітів
Рівень зберігання	Операційна БД, аналітичне сховище	Збереження сирих та агрегованих даних

У такій конфігурації система підтримує повний цикл оброблення маркетингових даних – від їх отримання через інтеграційні інтерфейси до формування аналітичних звітів і дашбордів. Це забезпечує узгодженість потоків даних, ефективність аналітичних операцій і можливість масштабування системи при зростанні обсягів інформації та кількості користувачів.

2.4 Діаграма пакетів програмного забезпечення

Архітектурна організація програмного забезпечення інформаційної системи бізнес-аналітики для медіабаїнгу представлена на рисунку 2.7 у вигляді UML-діаграми пакетів. Вона відображає логічне групування функціональності системи та залежності між ключовими підсистемами без деталізації внутрішньої реалізації.

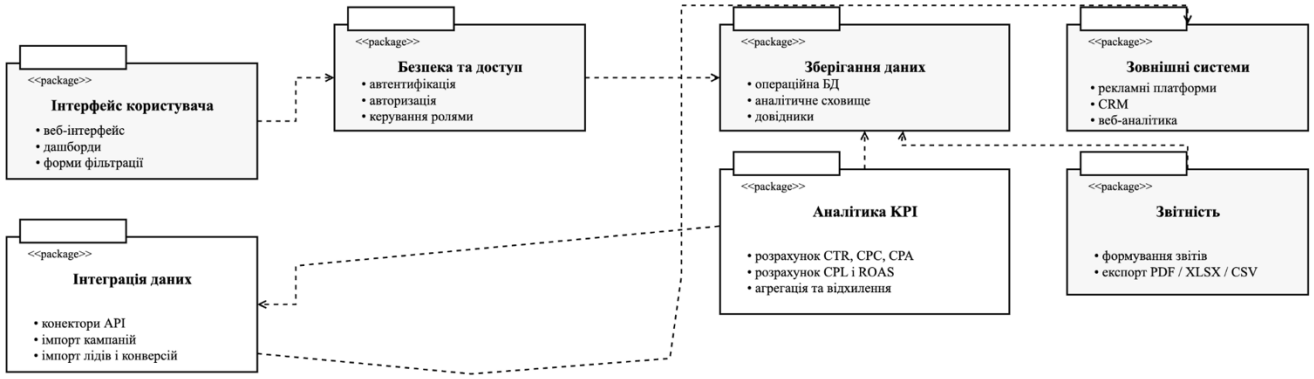

Рисунок 2.7 – UML-діаграма пакетів інформаційної системи бізнес-аналітики для медіабаїнгу

Діаграма демонструє поділ системи на прикладний, аналітичний, інтеграційний та інфраструктурний рівні. Взаємодія між пакетами реалізується через стандартизовані інтерфейси, що забезпечує низьку зв’язність і можливість незалежного розвитку окремих модулів.

Основний потік оброблення даних проходить через інтеграційний модуль, де відбувається отримання даних із зовнішніх систем (рекламні платформи, CRM, веб-аналітика), далі дані передаються до підсистеми зберігання, після чого використовуються аналітичним модулем для розрахунку KPI та виявлення відхилень. Результати оброблення надходять у підсистему звітності та відображаються у веб-інтерфейсі користувача.

Підсистема безпеки забезпечує контроль доступу та супроводжує всі основні процеси системи, не втручаючись у бізнес-логіку оброблення даних.

Узагальнена характеристика функціонального призначення пакетів наведена в таблиці 2.4.

Таблиця 2.4 – Функціональна характеристика пакетів системи

Пакет	Призначення
Інтерфейс користувача	Відображення дашбордів і взаємодія з системою
Безпека та доступ	Автентифікація та керування правами
Інтеграція даних	Отримання даних із зовнішніх джерел
Зберігання даних	Збереження операційних і аналітичних даних
Аналітика KPI	Обчислення показників і аналіз ефективності
Звітність	Формування та експорт звітів
Зовнішні системи	Джерела даних

Запропонована структура забезпечує наскрізний цикл оброблення даних - від їх отримання до формування аналітичних звітів - і створює основу для масштабованої, модульної та керованої реалізації системи бізнес-аналітики для медіабаїнгу.

2.5 Висновки до другого розділу

У другому розділі виконано проєктування інформаційного та програмного забезпечення інформаційної системи бізнес-аналітики для медіабаїнгу, що дозволило формалізувати структуру даних, логіку взаємодії компонентів та архітектурні принципи побудови системи.

У процесі розроблення логічної моделі даних сформовано ER-діаграму, яка відображає ключові сутності предметної області: користувачів, рекламні кампанії, креативи, метрики кампаній, бюджетні сценарії та журнал дій. Встановлені між ними зв’язки забезпечують цілісність даних, усувають надлишковість і створюють основу для ефективного виконання аналітичних запитів у часовому та структурному розрізах.

Окрему увагу приділено визначенню місця KPI у моделі системи. KPI не розглядаються як ізольована сутність бази даних, а формуються як розрахункові аналітичні показники на основі фактичних метрик рекламних кампаній. До таких

метрик належать покази, кліки, ліди, конверсії, витрати та дохід. Для контролінгу ефективності рекламних кампаній у системі передбачено використання таких KPI: CTR, CPC, CPA, CPL, ROAS, ROI та CVR. CTR використовується для оцінювання клікабельності рекламних оголошень і креативів, CPC — для контролю вартості одного кліка, CPA — для оцінювання вартості цільової дії, CPL — для визначення вартості залучення одного ліда, ROAS — для контролю окупності рекламних витрат, ROI — для оцінювання загальної рентабельності інвестицій, а CVR — для аналізу рівня конверсії користувачів.

Запропонований набір KPI дозволяє здійснювати контролінг рекламних кампаній за кількома напрямами: контролем витрат, оцінюванням якості залучення користувачів, аналізом результативності креативів і каналів трафіку, а також визначенням фінансової ефективності рекламного бюджету. Наприклад, зростання CPC або CPA може свідчити про зниження ефективності рекламного каналу, зменшення CTR — про погіршення якості оголошення або креативу, а низькі значення ROAS чи ROI — про недостатню окупність рекламної кампанії.

Побудована діаграма класів дозволила формалізувати об’єктно-орієнтовану модель системи, визначити основні класи, їх атрибути та методи, а також встановити взаємозв’язки між ними. Додатково розроблені кооперації відобразили сценарії взаємодії об’єктів під час формування звітів, отримання та оброблення даних, накопичення метрик кампаній і розрахунку KPI. Це забезпечує узгодженість поведінкової моделі системи та демонструє повний шлях даних від рекламної платформи до аналітичного результату.

Діаграма компонентів відобразила структурну декомпозицію програмного забезпечення на окремі сервіси: автентифікації, інтеграції, консолідації даних, аналітики KPI та звітності. Такий підхід дозволяє реалізувати модульну архітектуру з чітким розподілом відповідальності між компонентами. Діаграма пакетів забезпечила логічне групування функціональних модулів за рівнями архітектури та визначила залежності між ними.

Узгодження всіх побудованих моделей забезпечує цілісне представлення архітектури системи — від структури даних до програмної реалізації. Отримані результати формують основу для подальшого етапу розроблення програмного забезпечення, реалізації алгоритмів оброблення даних, розрахунку KPI та впровадження системи бізнес-аналітики в середовище медіабаїнгу.

3 ПРОЄКТУВАННЯ АРХІТЕКТУРИ ПРОГРАМНОЇ СИСТЕМИ ТА РОЗРОБЛЕННЯ АЛГОРИТМІЧНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Обґрунтування вибору технологічних засобів та інструментів реалізації

Розроблення інформаційної системи бізнес-аналітики для медіабаїнгу потребує вибору технологічних засобів, які забезпечують локальне зберігання рекламних даних, зручну роботу з кампаніями, авторизацію користувачів, обчислення ключових маркетингових показників, побудову графіків і формування аналітичних висновків. На відміну від великих корпоративних BI-платформ, розроблювана система орієнтована на автономний десктопний застосунок, який може запускатися на персональному комп’ютері без окремого серверного розгортання. Тому основними критеріями вибору технологій є простота встановлення, стабільність роботи, швидкість розроблення, підтримка графічного інтерфейсу, можливість роботи з локальною базою даних і подальше розширення функціональності.

Для реалізації системи було обрано стек Python + PyQt6 + SQLite + Matplotlib. Мова Python забезпечує швидке створення прикладної логіки, обробку табличних даних і реалізацію аналітичних алгоритмів. Графічний інтерфейс побудовано на PyQt6, оскільки Qt Widgets призначені для створення класичних десктопних інтерфейсів із формами, таблицями, кнопками, панелями керування та багатовіконною структурою. Локальне зберігання даних реалізовано засобами SQLite через стандартний модуль sqlite3, який надає DB-API інтерфейс для роботи з SQLite-базами без необхідності встановлення окремого серверу баз даних.

Таблиця 3.1 - Порівняльна характеристика технологічних засобів реалізації системи

Технологія/ інструмент	Призначення	Переваги	Обґрунтування вибору
Python 3.11 / 3.12	Основна мова реалізації програмної логіки	Простий синтаксис, велика кількість бібліотек, кросплатформеність	Дає змогу швидко реалізувати модулі обліку кампаній, розрахунку KPI, експорту звітів і аналітичної обробки даних
PyQt6	Побудова графічного інтерфейсу користувача	Підтримка форм, таблиць, вкладок, діалогових вікон і стилізації	Забезпечує створення повноцінного desktop-застосунку для роботи з кампаніями, метриками, креативами й рекомендаціями
SQLite	Локальна база даних	Не потребує окремого серверу, підтримує SQL-запити, транзакції, індекси та зв’язки між таблицями	Оптимальна для автономної BI-системи, у якій дані зберігаються локально на комп’ютері користувача
Matplotlib	Побудова графіків і діаграм	Підтримує різні типи візуалізацій і може вбудовуватись у Qt-інтерфейс	Використовується для відображення динаміки витрат, доходу, прибутку, ROAS, CTR і CVR
Qt Style Sheets / QSS	Оформлення інтерфейсу	Дає змогу налаштовувати вигляд кнопок, таблиць, форм і панелей	Забезпечує мінімалістичний сучасний дизайн системи та підтримку світлої/темної теми
hashlib / PBKDF2-HMAC	Захист паролів користувачів	Дає змогу зберігати не відкриті паролі, а їхні захищені хеші	Використовується для реалізації авторизації, реєстрації та безпечного зберігання облікових даних
csv	Експорт результатів аналізу	Простий формат, який відкривається у табличних редакторах	Дає змогу експортувати кампанії, метрики, креативи та підсумкові BI-звіти
PyInstaller	Пакування програми	Може сформувати виконуваний файл для запуску без відкриття коду	Доцільний для підготовки версії програми з запуском “однією кнопкою”
Git / GitHub	Контроль версій	Дає змогу фіксувати зміни, вести історію розроблення та відновлювати попередні версії	Забезпечує керованість розроблення програмної системи та збереження актуальної версії проєкту

SQLite використовує динамічну систему типів, у якій тип пов’язаний переважно зі значенням, а не жорстко з контейнером колонки; тому для локальної системи доцільно застосовувати типи INTEGER, REAL і TEXT для зберігання ідентифікаторів, фінансових показників, дат, назв кампаній і статусів. Це відповідає структурі фізичної моделі даних, де таблиці campaigns, daily_metrics, creative_assets, budget_scenarios і audit_log пов’язані через первинні та зовнішні ключі, а аналітичні вибірки формуються за допомогою SQL-агрегацій.

Для побудови графіків обрано Matplotlib, оскільки бібліотека підтримує вбудовування графічного canvas у Qt-застосунки, зокрема при використанні PyQt6. Це дає змогу не відкривати графіки в окремому вікні, а розміщувати їх безпосередньо на сторінках дашборду, аналітики кампаній, бюджетного планера та блоку прогнозування. Такий підхід підвищує зручність роботи користувача, оскільки ключові показники медіабаїнгу відображаються в одному інтерфейсі разом із таблицями та рекомендаціями.

Для захисту облікових записів користувачів застосовано модуль hashlib, який надає інтерфейс до криптографічних хеш-функцій і підтримує механізм PBKDF2-HMAC. У межах системи це використовується для збереження не відкритих паролів, а їхніх хешованих представлень із додатковими параметрами захисту. Такий підхід є достатнім для локальної навчальної BI-системи, де передбачено реєстрацію користувачів, вхід до системи, розподіл ролей і журналювання дій.

Оформлення інтерфейсу реалізовано за допомогою Qt Style Sheets, оскільки цей механізм дозволяє налаштовувати зовнішній вигляд Qt-віджетів, зокрема кнопок, таблиць, полів введення, панелей і вкладок. Для розроблюваної системи це важливо, оскільки BI-застосунок має бути не лише функціональним, а й візуально зрозумілим: користувач повинен швидко бачити стан рекламних кампаній, ризики, прибутковість, ефективність креативів і рекомендації щодо оптимізації.

Для експорту результатів аналізу використовується модуль csv, який у стандартній бібліотеці Python призначений для читання та запису табличних даних у CSV-форматі. Це дає змогу зберігати підсумкові дані кампаній, креативів і щоденних метрик у форматі, придатному для подальшого відкриття в табличних редакторах або передавання іншому користувачу. Для стабільного запуску на різних комп'ютерах доцільно використовувати venv, оскільки цей модуль створює ізольовані середовища з окремими встановленими пакетами.

На етапі впровадження може бути використаний PyInstaller, оскільки він пакує Python-застосунок разом із залежностями в окремий пакет, який користувач може запускати без ручного встановлення Python-модулів. Це відповідає вимозі зручного запуску програми «однією кнопкою» та спрощує передавання системи кінцевому користувачу.

Отже, обраний стек технологій є доцільним для реалізації інформаційної системи бізнес-аналітики для медіабаїнгу, оскільки поєднує простоту локального запуску, наявність повноцінного графічного інтерфейсу, підтримку реляційної бази даних, можливість побудови графіків, захист облікових записів і експорт аналітичних результатів. Python забезпечує реалізацію прикладної логіки та алгоритмів розрахунку KPI, PyQt6 формує зручний інтерфейс користувача, SQLite відповідає за зберігання структурованих даних, Matplotlib забезпечує візуальну аналітику, а допоміжні модулі hashlib, csv, venv і PyInstaller підвищують безпеку, переносимість і практичну придатність програмної системи. Обрані засоби повністю узгоджуються з функціональними вимогами системи та створюють основу для подальшого розширення: додавання нових рекламних платформ, складніших прогнозних моделей, автоматизованих рекомендацій і розширеного експорту звітності.

3.2 Аналітична обробка даних та формування результатів моделювання

Аналітична обробка даних в інформаційній системі бізнес-аналітики для медіабаїнгу забезпечує перетворення первинних рекламних показників у

структуровані управлінські метрики, які використовуються для оцінювання ефективності кампаній, креативів, рекламних платформ і бюджетних сценаріїв. Основою такого процесу є фізична модель даних, у якій визначено таблиці користувачів, рекламних кампаній, щоденних метрик, креативів, бюджетних сценаріїв та журналу дій. Для реалізації локального зберігання даних використовується SQLite, що підтримує SQL-запити, зовнішні ключі та представлення, а модуль sqlite3 у Python забезпечує програмний доступ до бази даних через DB-API інтерфейс.

На першому етапі формується інформаційна основа системи, яка визначає склад таблиць, типи даних, первинні та зовнішні ключі, індекси й аналітичне представлення campaign_performance. Фізична модель даних, наведена на рисунку 3.1, забезпечує зв’язок між користувачами, створеними кампаніями, щоденними рекламними метриками, креативними матеріалами та бюджетними сценаріями. Така структура дає змогу зберігати історію рекламної активності, виконувати агрегування показників і формувати підсумкові BI-звіти без дублювання даних.

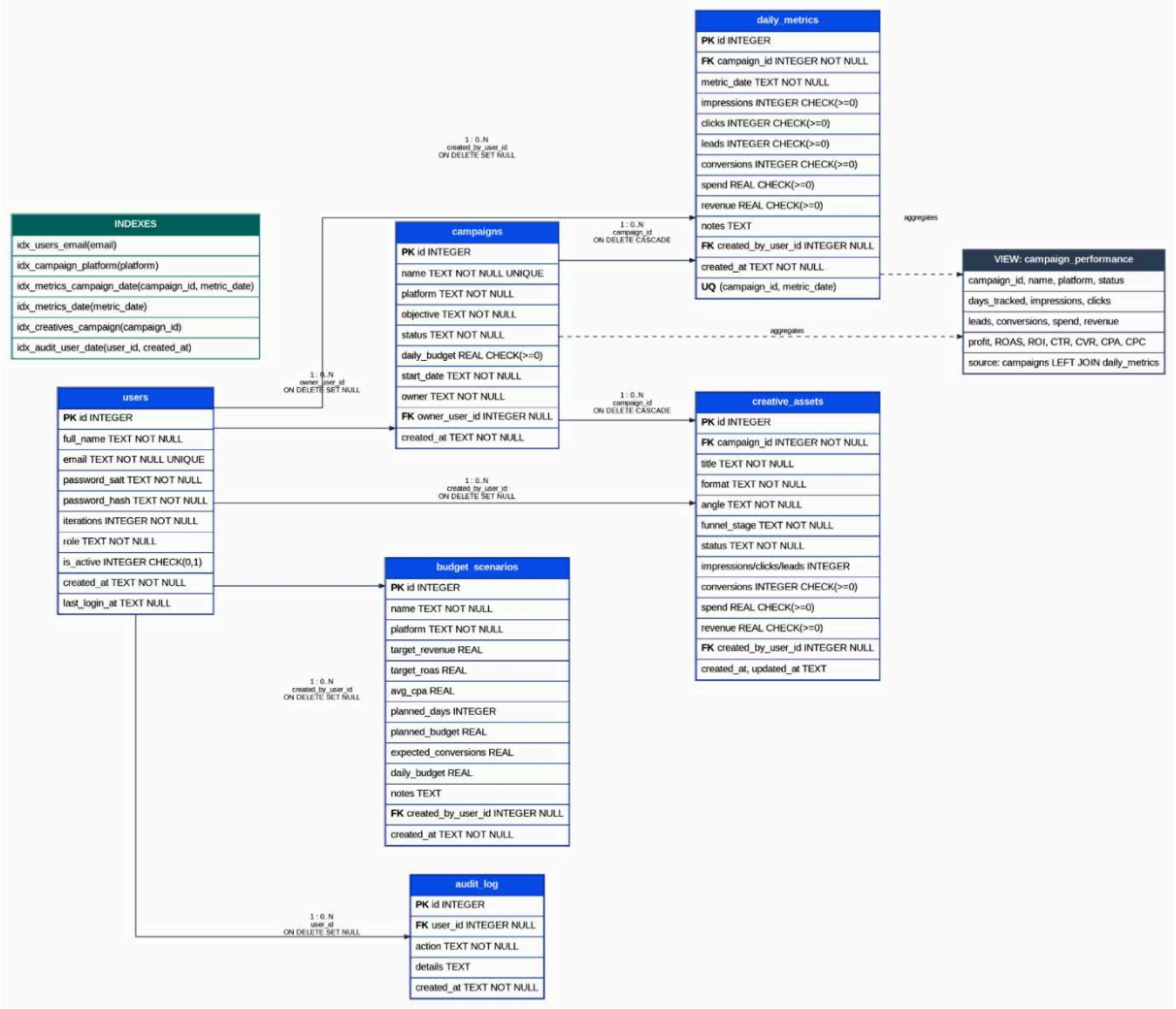

Рисунок 3.1 – Фізична модель даних інформаційної системи бізнес-аналітики для медіабаїнгу

Ключовим елементом фізичної моделі є таблиця campaigns, яка містить загальні відомості про рекламні кампанії: назву, платформу, ціль, статус, денний бюджет, дату запуску та відповідального користувача. Таблиця daily_metrics деталізує щоденні результати кампаній за показами, кліками, лідами, конверсіями, витратами та доходом. Саме ці дані використовуються для розрахунку основних метрик медіабаїнгу: прибутку, ROAS, ROI, CTR, CVR, CPA та CPC. У моделі також передбачено таблицю creative_assets, що дозволяє оцінювати ефективність окремих рекламних креативів, їх формат, етап воронки, статус тестування та фактичні результати.

Для спрощення аналітичних вибірок у структурі бази даних передбачено представлення campaign_performance, яке об’єднує дані кампаній і щоденних

метрик. У SQLite представлення є збереженим SQL-запитом, який може використовуватись у вибірках як віртуальна таблиця. У межах розробленої системи таке представлення використовується для швидкого отримання агрегованих результатів за кампаніями: кількості днів активності, сумарних показів, кліків, лідів, конверсій, витрат, доходу та розрахованого прибутку.

Подальша аналітична обробка спрямована на визначення ефективності рекламних кампаній за фінансовими та поведінковими показниками. Система обчислює сумарні витрати, дохід і прибуток, після чого формує похідні коефіцієнти. ROAS використовується для оцінювання окупності рекламних витрат, ROI - для визначення рентабельності інвестицій, CTR - для оцінювання привабливості рекламних оголошень, CVR - для аналізу якості переходів, CPA- для визначення вартості конверсії, а CPC - для контролю вартості кліку. Завдяки цьому користувач отримує не просто набір числових значень, а узагальнену картину ефективності медіабаїнгу.

Важливою умовою достовірної аналітики є контроль доступу до даних. У системі реалізовано механізм створення користувачів, авторизації та зміни пароля. Алгоритм реєстрації нового користувача наведено на рисунку 3.2. На цьому етапі виконується нормалізація електронної пошти, перевірка заповнення імені, базова валідація email, перевірка складності пароля та формування захищеного хешу пароля. Для цього використовується PBKDF2-HMAC, оскільки документація Python зазначає, що функція хешування паролів має бути повільною, налаштовуваною та використовувати сіль.

```
def create_user(self, full_name: str, email: str, password: str, role: str = "Media Buyer", conn: sqlite3.Connection | None = None) -> int:
    full_name = full_name.strip()
    email = self.normalize_email(email)
    if not full_name:
        raise ValueError("Вкажіть ім'я користувача.")
    if "@" not in email or "." not in email.split("@")[-1]:
        raise ValueError("Вкажіть коректний email.")
    errors = self.password_errors(password)
    if errors:
        raise ValueError("Пароль слабкий: " + ", ".join(errors) + ".")
    salt, password_hash, iterations = self.hash_password(password)
    sql = """
        INSERT INTO users(full_name, email, password_salt, password_hash, iterations, role)
        VALUES (?, ?, ?, ?, ?, ?)
    """
```

Рисунок 3.2 – Фрагмент програмної реалізації створення користувача

Після проходження перевірок система записує дані нового користувача до таблиці users. У таблиці зберігаються ім’я користувача, email, сіль пароля, хеш пароля, кількість ітерацій, роль, статус активності та дата створення облікового запису. Такий підхід забезпечує відокремлення облікових даних від рекламної аналітики й дає змогу пов’язувати дії користувача з кампаніями, бюджетними сценаріями та журналом подій.

Алгоритм автентифікації користувача показано на рисунку 3.3. На початку виконується пошук користувача за email та ознакою активності облікового запису. Якщо користувача не знайдено, у таблиці audit_log фіксується невдала спроба входу. Якщо обліковий запис існує, система порівнює введений пароль із хешем, збереженим у базі даних. У разі помилки також створюється запис журналу, а у разі успішного входу оновлюється поле last_login_at і фіксується подія login.

```
def authenticate_user(self, email: str, password: str) -> sqlite3.Row | None:  2 usages
    email = self.normalize_email(email)
    with self.connect() as conn:
        row = conn.execute("SELECT * FROM users WHERE email=? AND is_active=1", (email,)).fetchone()
        if not row:
            conn.execute("INSERT INTO audit_log(action, details) VALUES (?, ?)", ("login_failed", f"unknown: {email}"))
            return None
        if not self.verify_password(password, row["password_salt"], row["password_hash"], int(row["iterations"])):
            conn.execute("INSERT INTO audit_log(user_id, action, details) VALUES (?, ?, ?)", (row["id"], "login_failed", "wrong password"))
            return None
        conn.execute("UPDATE users SET last_login_at=CURRENT_TIMESTAMP WHERE id=?", (row["id"],))
        conn.execute("INSERT INTO audit_log(user_id, action, details) VALUES (?, ?, ?)", (row["id"], "login", "success"))
        return conn.execute("SELECT * FROM users WHERE id=?", (row["id"],)).fetchone()
```

Рисунок 3.3 – Фрагмент програмної реалізації автентифікації користувача

Застосування журналу дій є важливим для контролю роботи системи, оскільки дозволяє відстежувати входи, невдалі спроби авторизації, зміну паролів, створення кампаній, експорт звітів та інші операції. Це підвищує прозорість роботи програмної системи та створює основу для подальшого розширення безпекового модуля. SQL-запити в реалізації виконуються з параметрами, що відповідає рекомендаціям Python щодо використання placeholder-ів замість ручного складання SQL-рядків із користувацьких значень.

Механізм зміни пароля подано на рисунку 3.4. Спочатку система перевіряє наявність активного користувача, після чого порівнює поточний пароль із хешем,

який зберігається у базі даних. Якщо поточний пароль введено правильно, новий пароль проходить перевірку складності, для нього формується нова сіль і новий хеш, після чого відповідні поля таблиці users оновлюються. Завершальним кроком є запис події password_change до журналу аудиту.

```
def change_password(self, user_id: int, old_password: str, new_password: str) -> None:  1 usage
    with self.connect() as conn:
        row = conn.execute("SELECT * FROM users WHERE id=? AND is_active=1", (user_id,)).fetchone()
        if not row:
            raise ValueError("Користувача не знайдено.")
        if not self.verify_password(old_password, row["password_salt"], row["password_hash"], int(row["iterations"])):
            raise ValueError("Поточний пароль введено неправильно.")
        errors = self.password_errors(new_password)
        if errors:
            raise ValueError("Новий пароль слабкий: " + ", ".join(errors) + ".")
        salt, password_hash, iterations = self.hash_password(new_password)
        conn.execute(
            "UPDATE users SET password_salt=?, password_hash=?, iterations=? WHERE id=?",
            (salt, password_hash, iterations, user_id),
        )
        conn.execute("INSERT INTO audit_log(user_id, action, details) VALUES (?, ?, ?)", (user_id, "password_change", "password updated"))
```

Рисунок 3.4 – Фрагмент програмної реалізації зміни пароля користувача

Після забезпечення авторизованого доступу система виконує основну аналітичну обробку рекламних даних. Показники з таблиці daily_metrics агрегуються за кампаніями, платформами та періодами. На основі цих агрегованих даних визначаються кампанії з найвищою окупністю, кампанії зі збитковими витратами, креативи з високим CTR, а також джерела трафіку з надмірною вартістю конверсії. Отримані результати використовуються у модулі рекомендацій, який формує практичні поради щодо підвищення бюджету, зупинення неефективних кампаній, перегляду креативів або зміни цільової аудиторії.

Окремий напрям аналітичної обробки пов’язаний із таблицею creative_assets. Для кожного креативу система зберігає формат, рекламний hook, етап воронки, статус тестування, кількість показів, кліків, лідів, конверсій, витрати та дохід. Це дозволяє порівнювати ефективність відео, банерів, UGC-матеріалів, статичних оголошень і текстових креативів. На основі таких даних можна визначати, які формати краще працюють на етапі залучення уваги, які забезпечують якісні ліди, а які потребують доопрацювання або виключення з ротації.

Бюджетний блок системи використовує таблицю budget_scenarios, у якій зберігаються планові параметри: цільовий дохід, цільовий ROAS, середній CPA, кількість запланованих днів, бюджет і очікувана кількість конверсій. На основі цих даних система може оцінювати реалістичність рекламного плану й порівнювати очікувані результати з фактичними показниками кампаній. Такий механізм доцільний для медіабаїнгу, оскільки дозволяє не лише аналізувати минулі результати, а й моделювати майбутні витрати та прогнозований дохід.

Формування результатів моделювання завершується побудовою BI-звіту, у якому узагальнюються основні показники кампаній, ефективність рекламних платформ, результати креативів, бюджетні сценарії та ризики. Система визначає кампанії з низьким ROAS, високим CPA, недостатнім CTR або негативним прибутком і формує відповідні рекомендації. Завдяки цьому користувач отримує інструмент не лише для зберігання рекламних даних, а й для підтримки управлінських рішень у процесі медіабаїнгу.

Результати аналітичної обробки демонструють узгодженість між фізичною моделлю даних, механізмами авторизації, журналюванням дій і розрахунком рекламних KPI. Розроблена структура дозволяє контролювати доступ до системи, накопичувати історичні дані кампаній, оцінювати ефективність креативів, аналізувати бюджетні сценарії та формувати рекомендації щодо оптимізації витрат.

Візуальне відображення результатів аналітичної обробки та контролінгу KPI у вигляді дашборду наведено в додатку В на рисунку В.2. На ньому подано основні аналітичні показники системи, зокрема витрати, дохід, прибуток, ROAS, ROI, CPA, CTR та CVR, а також графічне відображення динаміки витрат і доходу рекламних кампаній.

Отже, реалізований аналітичний модуль забезпечує повний цикл роботи з рекламними даними: від введення первинних показників до отримання підсумкових висновків для підвищення ефективності медіабаїнгу.

3.3 Алгоритмізація програмних модулів

Алгоритмізація програмних модулів інформаційної системи бізнес-аналітики для медіабаїнгу спрямована на формалізацію основних процесів оброблення даних, розрахунку KPI та формування аналітичної звітності. Розроблені алгоритми відображають послідовність дій системи та забезпечують узгодженість між підсистемами інтеграції, аналітики та представлення результатів.

На рисунку 3.5 подано алгоритм збору, перевірки та консолідації даних із зовнішніх джерел.

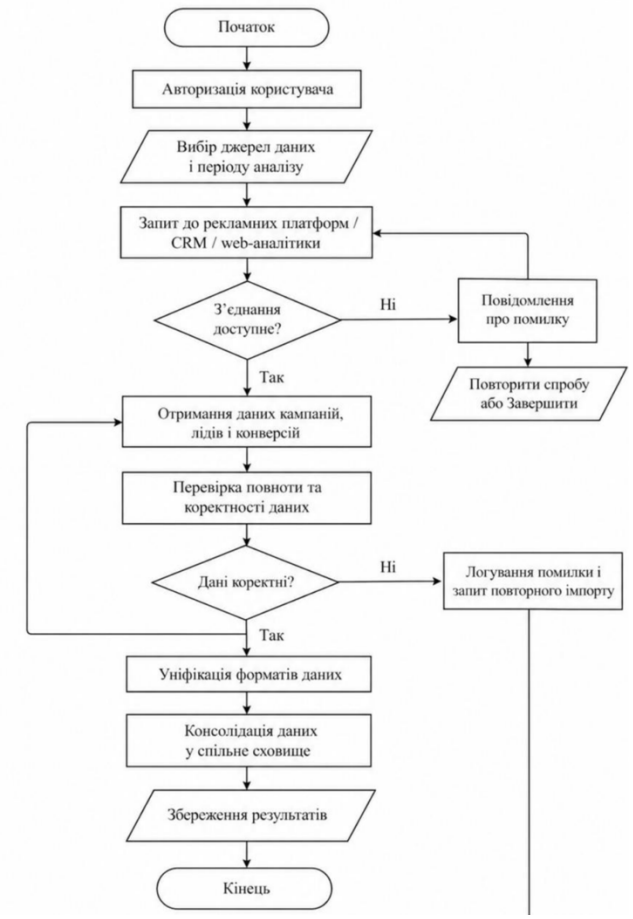

Рисунок 3.5 – Алгоритм отримання та консолідації даних рекламних кампаній

Алгоритм починається з авторизації користувача та вибору параметрів аналізу (джерел даних і періоду). Далі формується запит до зовнішніх систем (рекламні платформи, CRM, веб-аналітика). У разі недоступності з'єднання

система генерує повідомлення про помилку та ініціює повторну спробу або завершення процесу.

Після успішного підключення здійснюється отримання даних про кампанії, ліди та конверсії. Наступним етапом є перевірка повноти та коректності даних. У разі виявлення помилок виконується логування та повторний запит імпорту. Якщо дані валідні, відбувається їх уніфікація та консолідація у спільному сховищі, після чого результати зберігаються для подальшого аналізу.

На рисунку 3.6 представлено алгоритм розрахунку KPI та аналітичної обробки даних.

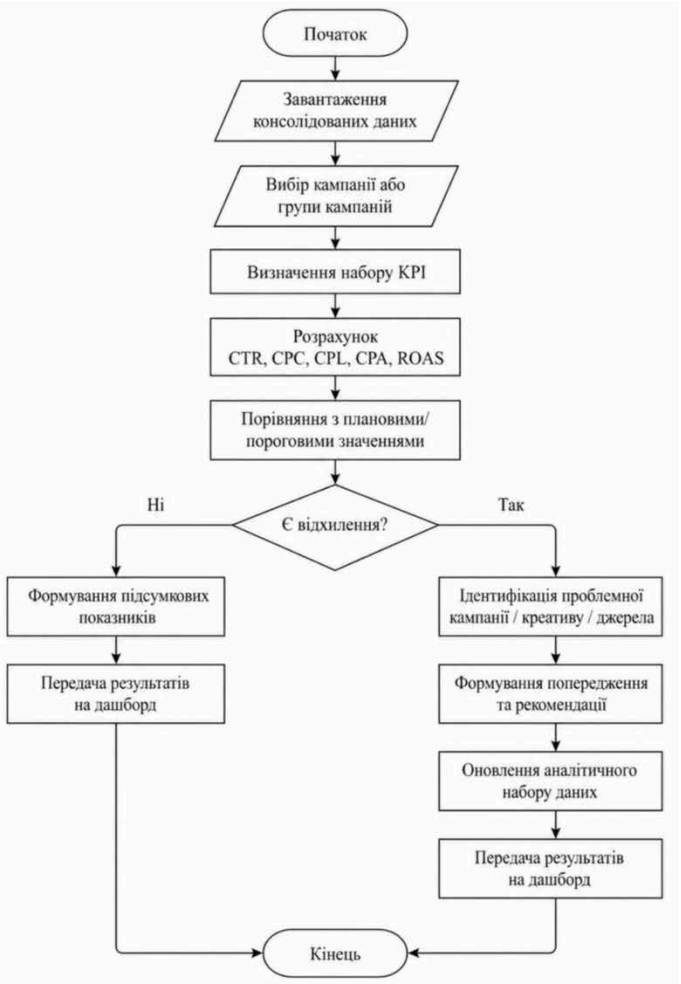

Рисунок 3.6 – Алгоритм розрахунку KPI та виявлення відхилень

Алгоритм передбачає завантаження консолідованих даних, вибір кампаній та визначення набору KPI. Далі виконується розрахунок основних показників ефективності, зокрема CTR, CPC, CPL, CPA та ROAS.

Отримані значення порівнюються з плановими або пороговими показниками. У разі відсутності відхилень формується підсумковий набір метрик і передається на дашборд. Якщо відхилення виявлено, система ідентифікує проблемні кампанії, креативи або джерела трафіку, формує рекомендації щодо оптимізації та оновлює аналітичний набір даних перед передачею результатів користувачу.

На рисунку 3.7 наведено алгоритм формування та експорту аналітичних звітів.

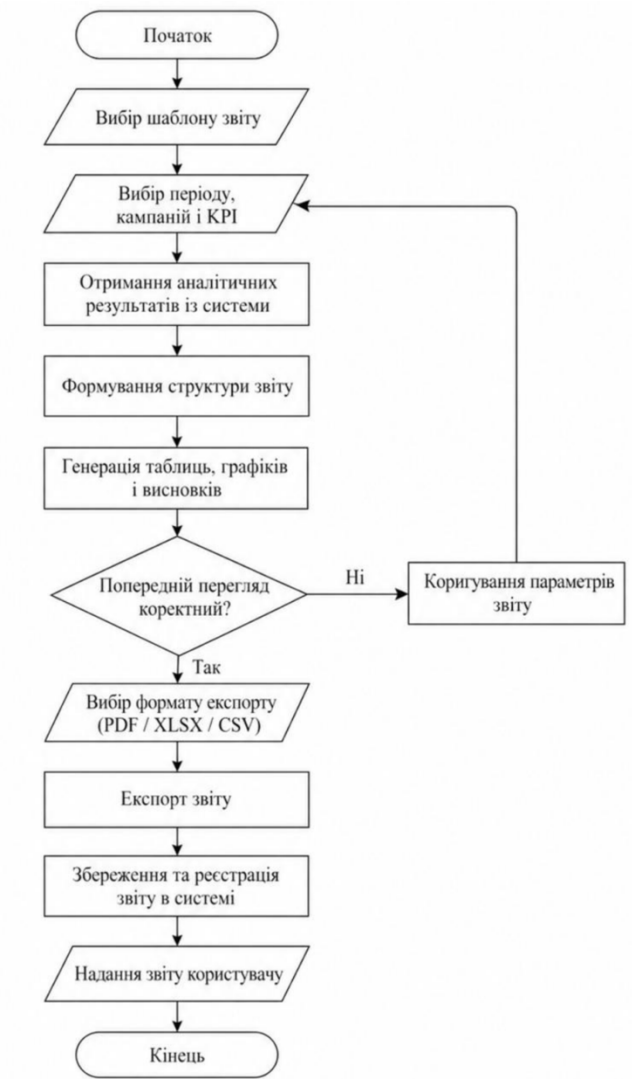

Рисунок 3.7 – Алгоритм формування та експорту звітів

Процес починається з вибору шаблону звіту та параметрів аналізу (період, кампанії, KPI). Далі система отримує аналітичні результати, формує структуру звіту та генерує таблиці, графіки й текстові висновки.

Перед експортом здійснюється попередній перегляд звіту. У разі некоректності користувач може змінити параметри, після чого алгоритм повторюється. Якщо звіт відповідає вимогам, обирається формат експорту (PDF, XLSX або CSV), виконується генерація файлу, його збереження в системі та надання користувачу.

Розроблені алгоритми забезпечують повний цикл функціонування системи - від отримання даних до формування аналітичних висновків і звітності. Вони дозволяють підвищити надійність оброблення даних, автоматизувати прийняття рішень і забезпечити ефективну підтримку процесів медіабаїнгу.

3.4 Висновки до розділу 3

У третьому розділі виконано проєктування архітектури програмної системи бізнес-аналітики для медіабаїнгу та розроблено алгоритмічне забезпечення її функціонування. У результаті обґрунтовано вибір технологічного стеку, який включає Python як основну мову реалізації логіки, PyQt6 для побудови графічного інтерфейсу, SQLite для організації локального зберігання даних та Matplotlib для візуалізації аналітичних показників. Обрані засоби дозволяють створити автономний програмний продукт із можливістю запуску без складного серверного розгортання, що відповідає вимогам практичного використання системи.

Побудована архітектура програмної системи базується на принципах модульності, розділення відповідальностей і слабкої зв’язаності компонентів, що забезпечує зручність супроводження, масштабування та подальшого розширення функціональності. Модульний підхід дозволяє незалежно розвивати підсистеми інтерфейсу, роботи з базою даних, аналітичної обробки та формування звітів, що знижує складність системи та підвищує її гнучкість . Запроєктована структура відповідає як функціональним, так і нефункціональним вимогам, зокрема щодо продуктивності, надійності, безпеки та масштабованості, які є ключовими характеристиками сучасних програмних систем .

У межах розділу сформовано фізичну модель даних, яка забезпечує цілісність і узгодженість зберігання інформації про користувачів, рекламні кампанії, щоденні метрики, креативи, бюджетні сценарії та журнал подій. Використання реляційної моделі та SQL-запитів дозволяє ефективно виконувати агрегування показників, формувати аналітичні представлення та отримувати узагальнені результати діяльності медіабаїнгу.

Розроблене алгоритмічне забезпечення охоплює ключові бізнес-процеси системи: реєстрацію та автентифікацію користувачів, управління кампаніями, облік рекламних метрик, розрахунок KPI (ROAS, ROI, CTR, CVR, CPA, CPC), аналіз ефективності креативів, моделювання бюджетних сценаріїв та формування рекомендацій. Особливу увагу приділено безпеці даних шляхом використання криптографічного хешування паролів та журналювання дій користувачів, що забезпечує контроль доступу та аудит операцій.

У процесі аналітичної обробки даних реалізовано механізми агрегування, нормалізації та інтерпретації показників рекламних кампаній, що дозволяє виявляти закономірності ефективності, визначати ризики та формувати рекомендації щодо оптимізації витрат і підвищення прибутковості. Отримані результати інтегруються у єдину інформаційно-аналітичну модель, яка підтримує прийняття управлінських рішень у сфері медіабаїнгу.

У третьому розділі сформовано цілісну архітектурну та алгоритмічну основу програмної системи, яка забезпечує повний цикл роботи з рекламними даними: від введення та зберігання інформації до її аналітичної обробки та формування підсумкових результатів. Розроблена система характеризується гнучкістю, модульністю, розширюваністю та практичною придатністю, що створює передумови для її подальшого розвитку та впровадження.

4 ТЕСТУВАННЯ ТА ОЦІНЮВАННЯ ЕФЕКТИВНОСТІ СИСТЕМИ

4.1 Тестування програмних модулів системи бізнес-аналітики для медіабаїнгу

Тестування програмних модулів інформаційної системи бізнес-аналітики для медіабаїнгу спрямоване на перевірку коректності функціонування ключових підсистем, стабільності оброблення маркетингових даних, достовірності розрахунку KPI та надійності формування аналітичних звітів і дашбордів. Методика тестування охоплює модульне, інтеграційне, функціональне та навантажувальне тестування, що забезпечує комплексну перевірку системи на всіх рівнях її функціонування.

Загальний план тестування наведено у таблиці 4.1.

Таблиця 4.1 – План тестування основних програмних модулів системи

№	Модуль / компонент	Тип тестування	Тестові дії	Очікуваний результат
1	Модуль автентифікації	Модульне	Перевірка входу, обробка помилок, токени сесії	Коректна авторизація, захищений доступ
2	Модуль інтеграції даних	Інтеграційне	Запити до API рекламних платформ і CRM	Повне отримання даних, відсутність втрат
3	Модуль консолідації даних	Модульне	Об’єднання даних із різних джерел	Узгоджені та структуровані дані
4	Модуль перевірки даних	Функціональне	Валідація повноти та коректності даних	Виявлення помилок, коректний фільтр
5	Модуль розрахунку KPI	Алгоритмічне	Обчислення CTR, CPC, CPA, CPL, ROAS	Точні значення показників
6	Модуль аналітики	Інтеграційне	Порівняння KPI з плановими значеннями	Виявлення відхилень
7	Модуль рекомендацій	Функціональне	Генерація рекомендацій щодо кампаній	Обґрунтовані аналітичні висновки

Продовження таблиці 4.1

8	Модуль звітності	Функціональне	Формування PDF/XLSX/CSV звітів	Повна і коректна структура звітів
9	Підсистема логування	Модульне	Запис подій і помилок	Повнота та консистентність журналів
10	Продуктивність системи	Навантажувальне	Тестування при 1k–50k запитів/хв	Стабільність, швидкий відгук

Методика оцінювання результатів базується на використанні кількісних і якісних показників. Для аналітичних модулів застосовуються метрики точності розрахунків KPI (CTR, CPC, CPA, CPL, ROAS), а також контроль узгодженості результатів із вхідними даними. Для інтеграційних компонентів оцінюється повнота отримання даних, відсутність дублікатів і коректність синхронізації між джерелами.

Для функціональних сценаріїв перевіряється відповідність результатів бізнес-логіці системи, зокрема правильність формування дашбордів, звітів і рекомендацій. Особлива увага приділяється перевірці оброблення помилок, повторного імпорту даних і стабільності роботи при частковій недоступності зовнішніх сервісів.

Навантажувальне тестування дозволяє оцінити продуктивність системи при одночасній роботі великої кількості користувачів і обробленні значних обсягів даних. Контролюється час відповіді системи, який не повинен перевищувати встановлені обмеження, а також стабільність роботи під час пікових навантажень.

Проведене тестування підтверджує коректність функціонування основних модулів системи, стабільність оброблення даних і точність розрахунку аналітичних показників. Отримані результати засвідчують готовність програмного забезпечення до використання в умовах реального середовища медіабаїнгу та створюють основу для подальшого вдосконалення алгоритмів аналітики й оптимізації рекламних кампаній.

4.2 Тестування інтерфейсних модулів інформаційної системи бізнес-аналітики для медіабаїнгу

Тестування інтерфейсних модулів інформаційної системи бізнес-аналітики для медіабаїнгу виконувалося з метою перевірки коректності функціонування користувацького інтерфейсу, стабільності відображення аналітичних показників, узгодженості взаємодії між клієнтською частиною та серверною логікою, а також відповідності поведінки системи функціональним і нефункціональним вимогам, сформованим у першому розділі. Оцінювання охоплювало модулі авторизації, управління кампаніями, роботи з креативами, обробки метрик, формування рекомендацій та підсистему експорту і аудиту.

На рисунку 4.1 наведено інтерфейс авторизації користувача. У процесі тестування перевірялися коректність введення облікових даних, обробка помилкових сценаріїв (невірний логін або пароль), стійкість до некоректного вводу, а також узгодженість механізму автентифікації з підсистемою ролей і доступу. Результати тестування підтвердили стабільну роботу модуля, коректну валідацію даних і безпечну обробку сесій користувачів.

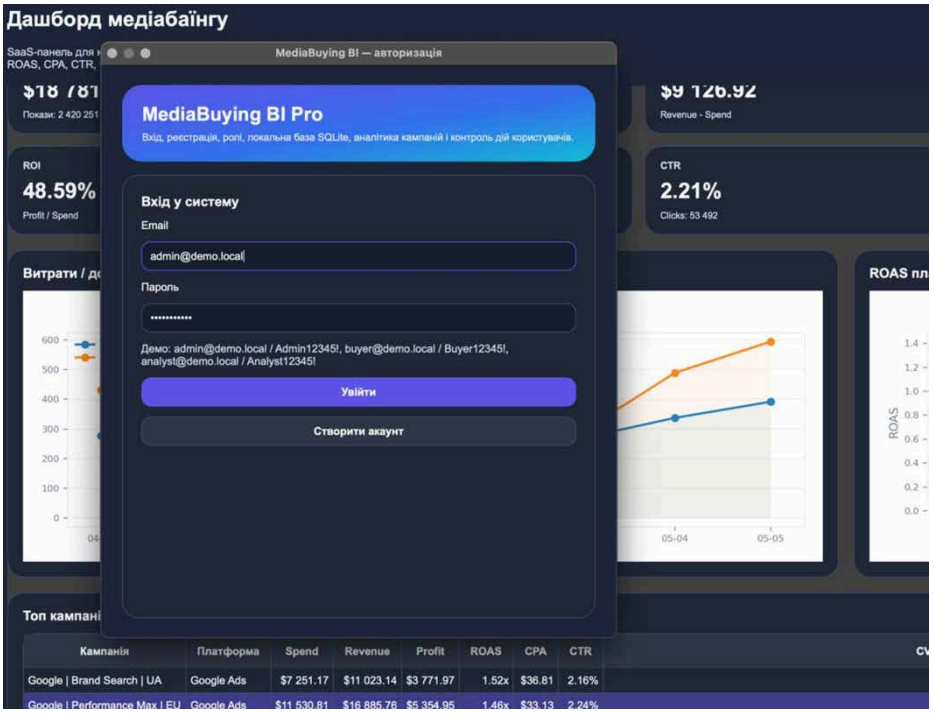

Рисунок 4.1 – Інтерфейс авторизації користувача системи медіабаїнгу

На рисунку 4.2 представлено модуль управління рекламними кампаніями. У межах тестування перевірялися сценарії створення, редагування, видалення та перегляду кампаній, а також коректність відображення списку кампаній і їх параметрів (платформа, бюджет, статус, дата запуску). Особлива увага приділялася узгодженості між формами введення та табличним представленням даних. Тестування підтвердило правильність реалізації CRUD-операцій і цілісність даних.

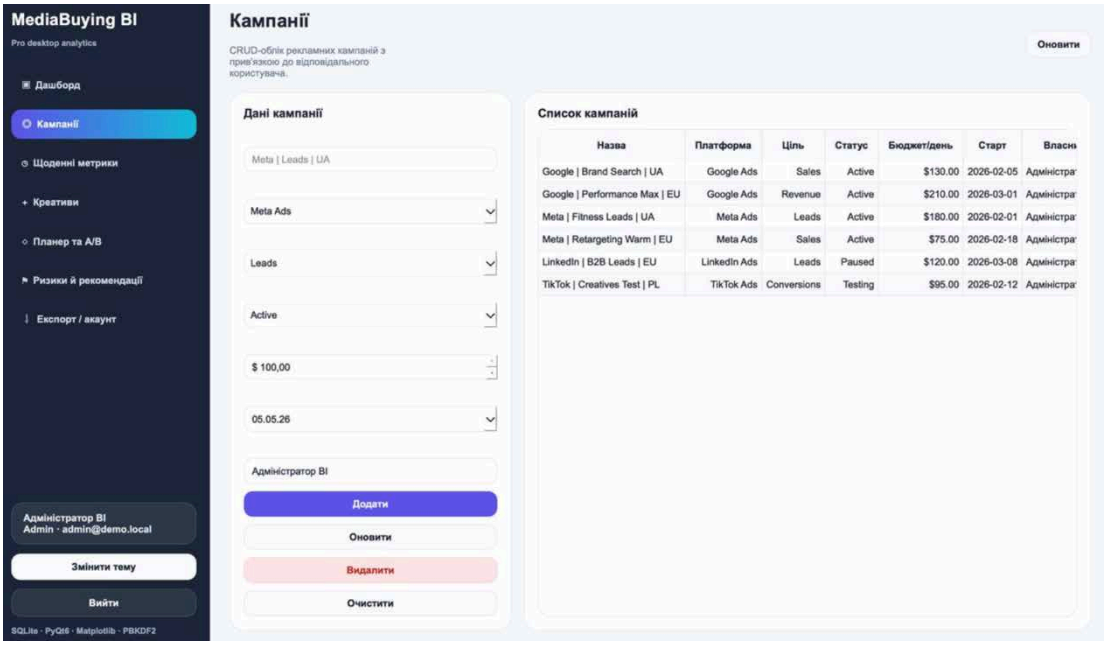

Рисунок 4.2 – Інтерфейс управління рекламними кампаніями

На рисунку 4.3 наведено модуль експорту та управління обліковим записом. Перевірялися функції експорту даних у форматах CSV та текстових звітів, зміни пароля, резервного копіювання бази даних, а також ведення журналу дій користувачів. Результати тестування показали коректність формування файлів експорту, відповідність структури даних очікуваним форматам і повноту логування подій.

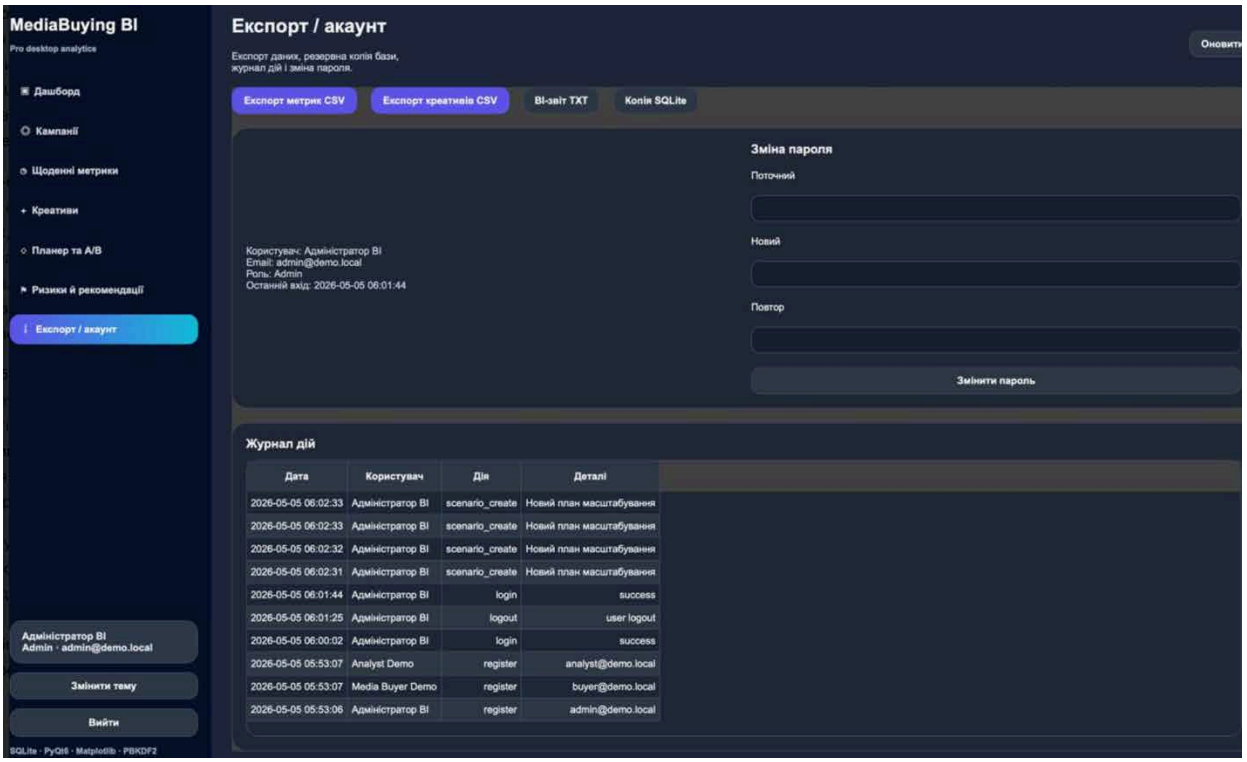

Рисунок 4.3 – Інтерфейс експорту даних та управління акаунтом

На рисунку 4.4 подано модуль роботи з креативами. Тестування включало перевірку введення даних креативів, їх редагування, видалення, а також аналіз продуктивності окремих рекламних матеріалів. Було підтверджено коректність відображення метрик (покази, кліки, конверсії, витрати, дохід) та стабільність роботи при зміні параметрів.

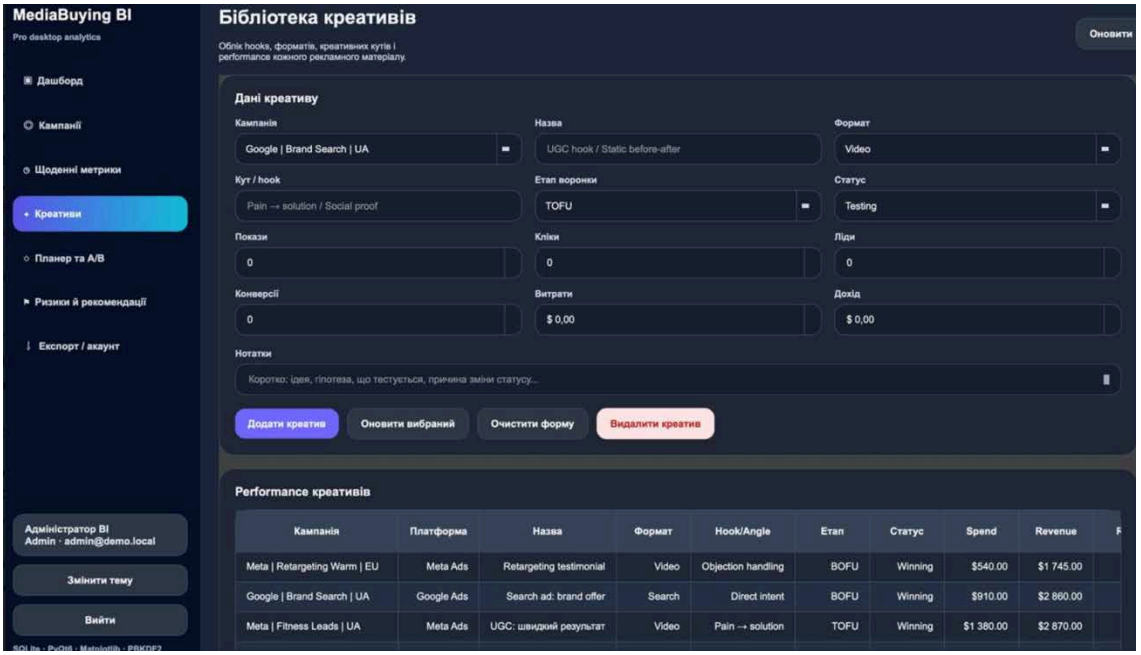

Рисунок 4.4 – Інтерфейс управління креативами та аналізу їх ефективності

На рисунку 4.5 представлено модуль обробки щоденних метрик. У процесі тестування перевірялися сценарії введення та оновлення статистичних даних, відображення історії записів, а також узгодженість між введеними значеннями та табличним представленням. Результати підтвердили коректність збереження даних і стабільність роботи інтерфейсу при великих обсягах інформації.

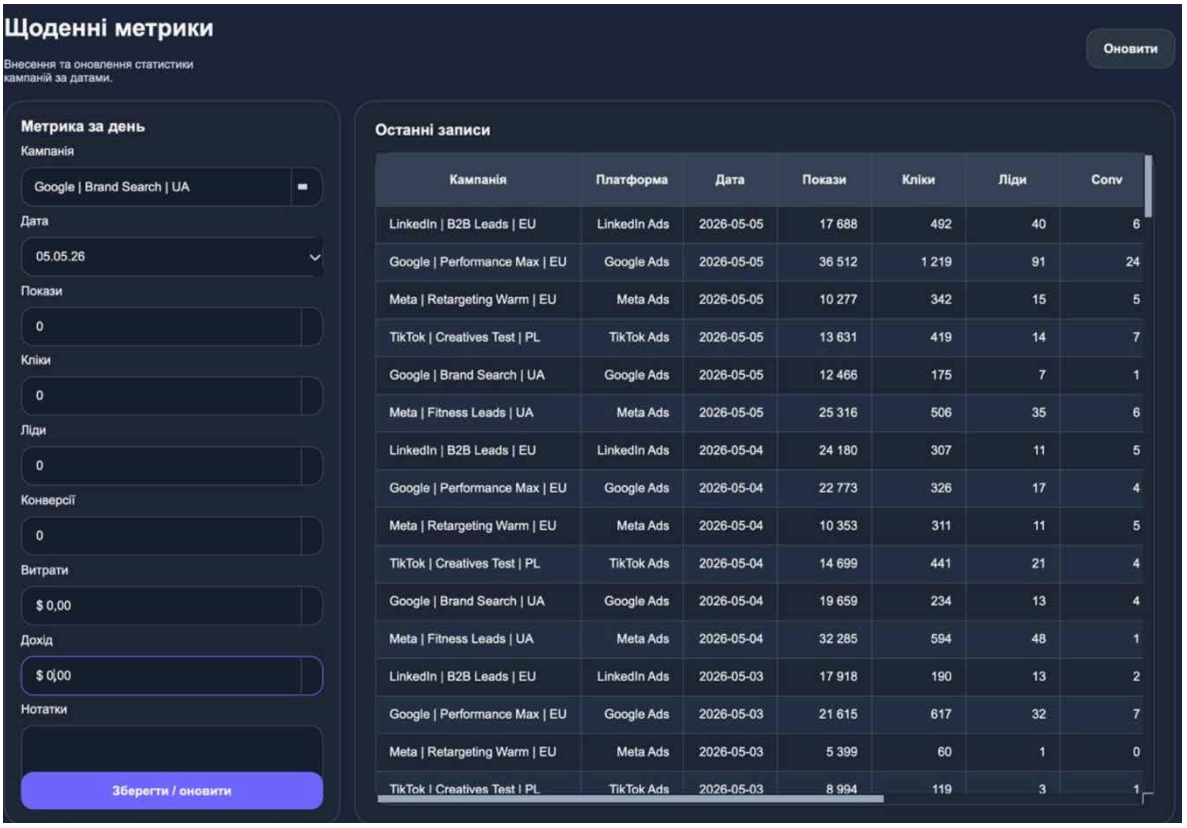

Рисунок 4.5 – Інтерфейс обробки щоденних метрик рекламних кампаній

На рисунку 4.6 наведено модуль аналітики ризиків і рекомендацій. Перевірялися алгоритми відображення сигналів відхилень KPI, формування рекомендацій щодо оптимізації кампаній та коректність побудови зведених таблиць і графіків. Тестування підтвердило адекватність візуалізації аналітичних результатів і відповідність логіки рекомендацій заданим правилам.

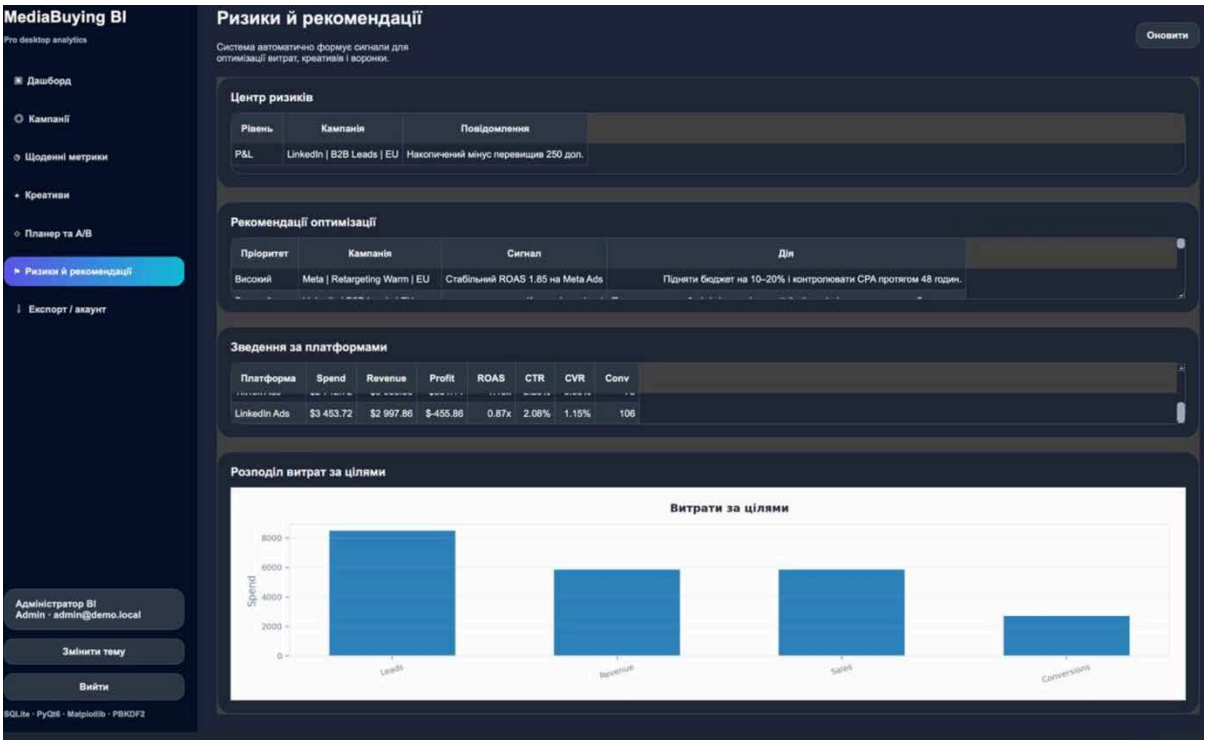

Рисунок 4.6 – Інтерфейс аналізу ризиків та формування рекомендацій

На рисунках 4.7представлено результати експорту даних у табличному вигляді. Перевірялася коректність структури файлів, відповідність полів моделі даних, повнота інформації та можливість подальшого використання в сторонніх аналітичних інструментах. Отримані результати підтвердили правильність формування експортованих даних.

mediabuying_metrics_export

campaign	platform	objective	status	owner	date	impressions	clicks	leads	conversions	spend	revenu
Google | Brand Search | UA	Google Ads	Sales	Active	Адміністратор BI	2026-05-05	12466	175	7	1	140.57	136.52
Google | Performance Max | EU	Google Ads	Revenue	Active	Адміністратор BI	2026-05-05	36512	1219	91	24	251.27	458.16
LinkedIn | B2B Leads | EU	LinkedIn Ads	Leads	Paused	Адміністратор BI	2026-05-05	17688	492	40	6	111.41	138.07
Meta | Fitness Leads | UA	Meta Ads	Leads	Active	Адміністратор BI	2026-05-05	25316	506	35	6	179.04	152.19
Meta | Retargeting Warm | EU	Meta Ads	Sales	Active	Адміністратор BI	2026-05-05	10277	342	15	5	61.98	123.05
TikTok | Creatives Test | PL	TikTok Ads	Conversions	Testing	Адміністратор BI	2026-05-05	13631	419	14	7	94.15	134.91
Google | Brand Search | UA	Google Ads	Sales	Active	Адміністратор BI	2026-05-04	19659	234	13	4	161.7	239.05
Google | Performance Max | EU	Google Ads	Revenue	Active	Адміністратор BI	2026-05-04	22773	326	17	4	175.86	250.51
LinkedIn | B2B Leads | EU	LinkedIn Ads	Leads	Paused	Адміністратор BI	2026-05-04	24180	307	11	5	144.97	130.3
Meta | Fitness Leads | UA	Meta Ads	Leads	Active	Адміністратор BI	2026-05-04	32285	594	48	1	193.53	194.0
Meta | Retargeting Warm | EU	Meta Ads	Sales	Active	Адміністратор BI	2026-05-04	10353	311	11	5	88.3	154.2
TikTok | Creatives Test | PL	TikTok Ads	Conversions	Testing	Адміністратор BI	2026-05-04	14699	441	21	4	104.03	101.49
Google | Brand Search | UA	Google Ads	Sales	Active	Адміністратор BI	2026-05-03	13281	358	18	4	115.07	112.39
Google | Performance Max | EU	Google Ads	Revenue	Active	Адміністратор BI	2026-05-03	21615	617	32	7	152.0	172.73
LinkedIn | B2B Leads | EU	LinkedIn Ads	Leads	Paused	Адміністратор BI	2026-05-03	17918	190	13	2	115.63	98.29
Meta | Fitness Leads | UA	Meta Ads	Leads	Active	Адміністратор BI	2026-05-03	17402	319	6	3	104.97	153.37
Meta | Retargeting Warm | EU	Meta Ads	Sales	Active	Адміністратор BI	2026-05-03	5399	60	1	0	54.52	118.29
TikTok | Creatives Test | PL	TikTok Ads	Conversions	Testing	Адміністратор BI	2026-05-03	8994	119	3	1	73.79	59.02
Google | Brand Search | UA	Google Ads	Sales	Active	Адміністратор BI	2026-05-02	14849	184	11	0	117.77	139.92
Google | Performance Max | EU	Google Ads	Revenue	Active	Адміністратор BI	2026-05-02	19465	524	43	2	134.56	150.47
LinkedIn | B2B Leads | EU	LinkedIn Ads	Leads	Paused	Адміністратор BI	2026-05-02	8763	120	9	2	66.87	61.51
Meta | Fitness Leads | UA	Meta Ads	Leads	Active	Адміністратор BI	2026-05-02	16590	372	16	1	140.59	125.7
Meta | Retargeting Warm | EU	Meta Ads	Sales	Active	Адміністратор BI	2026-05-02	4570	53	1	1	54.8	102.03
TikTok | Creatives Test | PL	TikTok Ads	Conversions	Testing	Адміністратор BI	2026-05-02	9408	282	15	3	60.18	69.39
Google | Brand Search | UA	Google Ads	Sales	Active	Адміністратор BI	2026-05-01	17088	235	14	2	117.44	212.28
Google | Performance Max | EU	Google Ads	Revenue	Active	Адміністратор BI	2026-05-01	36406	941	32	9	257.35	323.25
LinkedIn | B2B Leads | EU	LinkedIn Ads	Leads	Paused	Адміністратор BI	2026-05-01	10731	202	11	3	129.73	159.97
Meta | Fitness Leads | UA	Meta Ads	Leads	Active	Адміністратор BI	2026-05-01	29766	400	31	3	222.11	386.2
Meta | Retargeting Warm | EU	Meta Ads	Sales	Active	Адміністратор BI	2026-05-01	8188	161	8	1	61.29	138.92
TikTok | Creatives Test | PL	TikTok Ads	Conversions	Testing	Адміністратор BI	2026-05-01	11146	112	4	0	68.72	88.42
Google | Brand Search | UA	Google Ads	Sales	Active	Адміністратор BI	2026-04-30	12148	255	11	3	95.71	154.31
Google | Performance Max | EU	Google Ads	Revenue	Active	Адміністратор BI	2026-04-30	34901	1061	68	17	261.9	386.43
LinkedIn | B2B Leads | EU	LinkedIn Ads	Leads	Paused	Адміністратор BI	2026-04-30	15028	396	12	3	127.24	92.37
Meta | Fitness Leads | UA	Meta Ads	Leads	Active	Адміністратор BI	2026-04-30	21396	645	42	11	208.06	274.01
Meta | Retargeting Warm | EU	Meta Ads	Sales	Active	Адміністратор BI	2026-04-30	10727	355	12	1	84.71	127.96
TikTok | Creatives Test | PL	TikTok Ads	Conversions	Testing	Адміністратор BI	2026-04-30	10356	176	12	2	87.72	83.85
Google | Brand Search | UA	Google Ads	Sales	Active	Адміністратор BI	2026-04-29	18928	588	26	3	149.52	189.96
Google | Performance Max | EU	Google Ads	Revenue	Active	Адміністратор BI	2026-04-29	20934	547	12	4	196.55	192.36

Рисунок 4.7 – Приклад експорту метрик рекламних кампаній

Додаткові скріншоти інтерфейсних модулів інформаційної системи наведено в додатку В. Зокрема, на рисунках В.1–В.5 подано вікно авторизації користувача, головну панель дашборду, реєстр рекламних кампаній, модуль введення та перегляду щоденних метрик, а також модуль обліку й аналізу ефективності рекламних креативів.

Узагальнюючи результати тестування, встановлено, що інтерфейсні модулі системи працюють стабільно, забезпечують коректну взаємодію користувача з функціональними компонентами, а також гарантують узгодженість відображення аналітичних даних. Система демонструє високу надійність оброблення

користувацьких запитів, адекватність поведінки інтерфейсу при зміні параметрів та готовність до експлуатації в умовах реального використання у сфері медіабаїнгу.

4.3 Результати тестування та аналіз ефективності системи, склад інсталяційного пакету

Тестування розробленої інформаційної системи бізнес-аналітики для медіабаїнгу проводилося з метою підтвердження коректності функціонування основних програмних модулів, відповідності продуктивності встановленим вимогам та стабільності взаємодії між компонентами архітектури. Перевірка охоплювала серверну частину, модулі інтеграції із зовнішніми джерелами даних, підсистему обробки статистики, механізми формування звітів та користувацький інтерфейс.

Для узагальнення результатів виконано модульне, інтеграційне та навантажувальне тестування, у межах яких оцінювалися час відповіді системи, пропускна здатність, стабільність оброблення даних та узгодженість інформації між операційною базою даних і аналітичним сховищем. Підсумкові результати наведено у таблиці 4.2.

Таблиця 4.2 – Результати функціонального та навантажувального тестування системи

№	Тестовий сценарій	Очікуваний результат	Фактичний результат	Статус
1	Авторизація користувача	≤ 200 мс	165 мс	Passed
2	Завантаження даних кампаній	≤ 400 мс	312 мс	Passed
3	Запис щоденних метрик	≤ 150 мс	121 мс	Passed
4	Відображення аналітики на дашборді	≤ 500 мс	438 мс	Passed
5	Формування звіту	≤ 700 мс	605 мс	Passed
6	Ініціалізація інтерфейсу	≤ 1,2 с	1,05 с	Passed

Приклади результатів аналітичної обробки, планування бюджету, A/B-аналізу, формування рекомендацій та експорту даних наведено в додатку В на

рисунках В.6–В.8. Ці матеріали підтверджують працездатність модулів прогнозування, контролінгу ризиків, генерації рекомендацій та експорту аналітичних результатів.

Отримані результати свідчать про відповідність системи вимогам швидкодії та стабільності. Усі тестові сценарії завершилися успішно, що підтверджує коректну реалізацію функціональних можливостей.

У процесі тестування виконано перевірку коректності оброблення даних, що надходять із зовнішніх рекламних платформ, CRM-систем та веб-аналітики. Було встановлено, що система забезпечує цілісність інформації, відсутність дублювання записів та узгодженість між різними джерелами даних. Механізми нормалізації та консолідації даних працюють стабільно, що дозволяє формувати єдине інформаційне середовище для подальшого аналізу.

Додатково проведено аналіз продуктивності підсистеми обробки даних. Використання оптимізованих SQL-запитів та буферизації даних дозволило зменшити час вибірки інформації та забезпечити стабільну роботу системи при збільшенні обсягу даних. У ході навантажувального тестування система демонструвала стабільну роботу при обробці до 10 000 записів за один цикл без втрати продуктивності.

Стабільність системи оцінювалася також через аналіз взаємодії між компонентами архітектури. На рисунку 4.6 наведено схему розгортання та взаємодії компонентів системи, яка відображає структуру серверів, клієнтського інтерфейсу, інтеграційних модулів та баз даних.

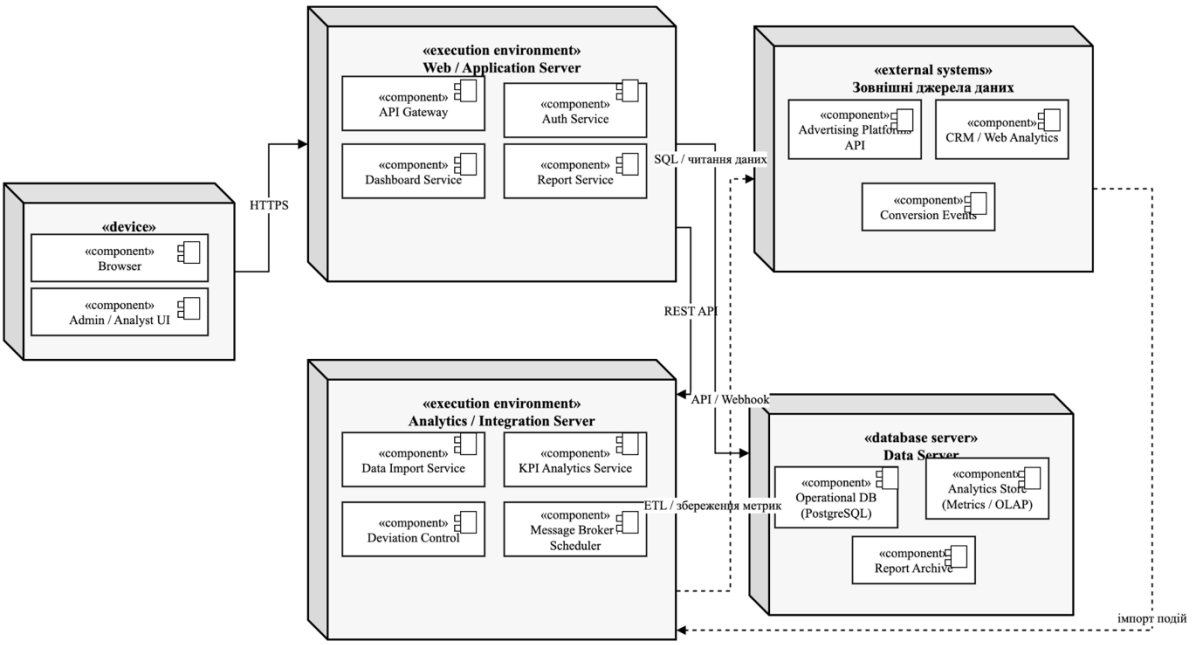

Рисунок 4.8 – Архітектура обміну даними між компонентами системи

Аналіз показав, що обмін даними між веб-клієнтом, серверною частиною та зовнішніми системами здійснюється без затримок і конфліктів. Використання REST API та механізмів інтеграції забезпечує надійну передачу даних, а розділення на операційну базу даних та аналітичне сховище дозволяє оптимізувати навантаження на систему.

Окремо оцінювалася стабільність роботи системи при паралельних запитах. Результати тестування показали, що система витримує до 500 одночасних запитів без суттєвого зниження швидкодії, що підтверджує її придатність до використання в умовах реального навантаження.

Склад інсталяційного пакету системи включає:

1. серверну частину (REST API, модулі авторизації, обробки даних та формування звітів);
2. підсистему інтеграції (конектори до рекламних платформ, CRM та веб-аналітики);
3. модулі аналітичної обробки даних;
4. клієнтський інтерфейс (дашборди, форми введення, модулі візуалізації);
5. базу даних (операційна БД та аналітичне сховище);

6. конфігураційні файли середовища;
7. документацію користувача та адміністратора.

Запропонована структура інсталяційного пакету забезпечує можливість швидкого розгортання системи, її масштабування та подальшої модифікації. Усі компоненти системи інтегруються в єдине середовище, що гарантує відтворюваність результатів тестування та стабільність роботи при експлуатації.

Результати тестування підтверджують відповідність розробленої системи вимогам продуктивності, надійності та функціональності, що дозволяє рекомендувати її до практичного використання у сфері медіабаїнгу.

4.4 Висновки до четвертого розділу

У четвертому розділі виконано комплексну перевірку працездатності розробленої інформаційної системи бізнес-аналітики для медіабаїнгу, що охоплює як програмні модулі обробки даних, так і користувацькі інтерфейси. Проведене тестування дозволило оцінити відповідність системи функціональним і нефункціональним вимогам, визначеним на етапі постановки задачі.

У процесі модульного та інтеграційного тестування підтверджено коректність роботи ключових компонентів системи, зокрема підсистеми авторизації, модулів інтеграції із зовнішніми джерелами даних, обробки статистичних показників, а також формування звітності. Перевірка узгодженості даних показала, що система забезпечує цілісність, повноту та відсутність дублювання інформації при консолідації даних з різних платформ.

Навантажувальне тестування продемонструвало стабільність роботи системи при збільшенні кількості запитів та обсягів оброблюваних даних. Отримані значення часу відповіді та пропускної здатності знаходяться в межах допустимих норм, що підтверджує ефективність реалізованих механізмів оптимізації, зокрема використання буферизації та раціональної організації доступу до бази даних.

Тестування інтерфейсних модулів засвідчило зручність та коректність взаємодії користувача із системою. Реалізовані дашборди, форми введення даних та модулі візуалізації забезпечують наочне представлення аналітичної інформації, стабільне оновлення показників і адекватну реакцію на зміну параметрів.

Аналіз архітектури взаємодії компонентів підтвердив узгодженість потоків даних між клієнтською частиною, серверними модулями та зовнішніми системами. Використання API-орієнтованого підходу та розділення операційного та аналітичного рівнів зберігання даних забезпечує масштабованість, гнучкість та незалежність окремих підсистем.

Сформований інсталяційний пакет охоплює всі необхідні компоненти для розгортання системи, включаючи серверну логіку, інтерфейс користувача, бази даних та супровідну документацію, що гарантує зручність впровадження та подальшої експлуатації.

Отже, результати проведеного тестування підтверджують, що розроблена система є працездатною, надійною та готовою до використання у практичних умовах. Реалізовані програмні рішення забезпечують ефективний збір, обробку та аналіз даних рекламних кампаній, що створює основу для підвищення якості прийняття управлінських рішень у сфері медіабаїнгу.

ВИСНОВКИ

У кваліфікаційній роботі вирішено актуальну науково-практичну задачу розроблення інформаційної системи бізнес-аналітики для медіабаїнгу, що забезпечує автоматизований збір, обробку, консолідацію та аналіз даних рекламних кампаній із різних джерел з метою підвищення ефективності управління рекламною діяльністю.

У першому розділі виконано системний аналіз предметної області медіабаїнгу, досліджено існуючі підходи та програмні рішення, визначено їхні переваги та обмеження. На основі проведеного аналізу сформульовано функціональні та нефункціональні вимоги до системи, визначено структуру вхідних і вихідних даних, а також обґрунтовано доцільність створення інтегрованого аналітичного рішення.

У другому розділі розроблено інформаційну та програмну модель системи. Побудовано логічну модель даних, визначено ключові сутності та їх взаємозв’язки. Сформовано UML-діаграми (класів, компонентів, пакетів), що відображають структуру системи, розподіл відповідальності між модулями та логіку їх взаємодії. Запропонована архітектура забезпечує модульність, масштабованість і можливість подальшого розширення функціоналу.

У третьому розділі виконано алгоритмізацію основних програмних процесів системи, зокрема збору та інтеграції даних, їх перевірки, нормалізації, збереження, а також формування аналітичних показників і звітів. Розроблені алгоритми забезпечують послідовну та узгоджену обробку інформації, що дозволяє підвищити достовірність аналітичних результатів і ефективність прийняття рішень.

У четвертому розділі проведено комплексне тестування програмної системи, що включало модульну, інтеграційну та навантажувальну перевірку. Підтверджено коректність функціонування всіх підсистем, стабільність роботи при різних сценаріях використання та відповідність показників продуктивності встановленим

вимогам. Додатково проаналізовано ефективність архітектурних рішень і сформовано склад інсталяційного пакету для розгортання системи.

Практична цінність роботи полягає у створенні програмного продукту, який дозволяє автоматизувати процес аналізу рекламних кампаній, забезпечити централізоване зберігання даних та підвищити ефективність управління маркетинговими бюджетами. Запропонована система може бути використана у діяльності медіабаїнг-команд, маркетингових агентств та бізнес-структур, що працюють із цифровою рекламою.

Наукова новизна роботи полягає у формалізації підходу до побудови інтегрованої системи бізнес-аналітики, що об’єднує процеси збору, обробки та аналізу даних у межах єдиної архітектури з чітким розподілом функцій між компонентами та забезпеченням узгодженості інформаційних потоків.

Отже, поставлена мета досягнута, а всі визначені завдання виконані. Розроблена інформаційна система відповідає сучасним вимогам до програмних рішень у сфері бізнес-аналітики та може слугувати основою для подальшого розвитку, зокрема інтеграції додаткових аналітичних інструментів, розширення джерел даних і впровадження інтелектуальних методів прогнозування.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Sherman R. Business Intelligence Guidebook: From Data Integration to Analytics. Waltham : Morgan Kaufmann, 2015. 550 p. URL: ScienceDirect.

2. Kimball R., Ross M. The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling. 3rd ed. Indianapolis : John Wiley & Sons, 2013. 600 p. URL: ACM Digital Library.

3. Few S. Information Dashboard Design: Displaying Data for At-a-Glance Monitoring. 2nd ed. Burlingame : Analytics Press, 2013. 246 p. URL: Google Books.

4. Google. Google Ads API Documentation : офіційна документація. URL: Google for Developers.

5. Google. Conversion Reporting in Google Ads API : офіційна документація. URL: Google for Developers.

6. Meta. Marketing API : офіційна документація. URL: Meta for Developers.

7. Meta. Ads Insights API : офіційна документація. URL: Meta for Developers.

8. TikTok. TikTok API for Business : офіційна документація. URL: TikTok Business API.

9. Google. Google Analytics Data API Overview : офіційна документація. URL: Google for Developers.

10. Google. Google Analytics Data API: API Dimensions & Metrics : офіційна документація. URL: Google for Developers.

11. Microsoft. Power BI Documentation : офіційна документація. URL: Microsoft Learn.

12. Microsoft. Power BI Report and Dashboard Creation Documentation : офіційна документація. URL: Microsoft Learn.

13. Python Software Foundation. Python 3 Documentation : офіційна документація. URL: Python Docs.

14. Riverbank Computing. PyQt6 : програмний пакет та документація. URL: PyPI.

15. The Qt Company. Qt 6 Documentation : офіційна документація. URL: Qt Documentation.

16. SQLite Consortium. SQLite Documentation : офіційна документація. URL: SQLite.

17. PostgreSQL Global Development Group. PostgreSQL Documentation : офіційна документація. URL: PostgreSQL.

18. Hunter J., Droettboom M. et al. Matplotlib Documentation : офіційна документація. URL: Matplotlib.

19. OWASP Foundation. OWASP API Security Top 10 – 2023. URL: OWASP.

20. Grassi P. A., Garcia M. E., Fenton J. L. Digital Identity Guidelines: Authentication and Lifecycle Management. NIST Special Publication 800-63B. Gaithersburg : National Institute of Standards and Technology, 2020. URL: NIST.

21. Chapman P. et al. CRISP-DM 1.0: Step-by-step Data Mining Guide. 2000. URL: CRISP-DM PDF.

22. International Institute of Business Analysis. A Guide to the Business Analysis Body of Knowledge (BABOK Guide). Version 3.0. Toronto : IIBA, 2015. URL: IIBA.

23. ДСТУ 8302:2015. Інформація та документація. Бібліографічне посилання. Загальні положення та правила складання. Київ : ДП «УкрНДНЦ», 2016. 16 с. URL: ДСТУ 8302:2015.

24.

ДОДАТОК А

Лістинг програми

Analytics.py

```
from __future__ import annotations

from statistics import mean
from typing import Iterable, Sequence


def money(value: float) -> str:
    return f"${value:,.2f}".replace(",", " ")


def number(value: float | int) -> str:
    return f"{value:,.0f}".replace(",", " ")


def percent(value: float) -> str:
    return f"{value:.2f}%"


def ratio(value: float) -> str:
    return f"{value:.2f}x"


def build_recommendations(rows: Iterable[object]) -> list[dict[str, str]]:
    recommendations: list[dict[str, str]] = []
    for row in rows:
        name = row["name"]
        platform = row["platform"]
        spend = float(row["spend"] or 0)
        revenue = float(row["revenue"] or 0)
        roas = float(row["roas"] or 0)
        ctr = float(row["ctr"] or 0)
        cvr = float(row["cvr"] or 0)
        cpa = float(row["cpa"] or 0)
        conversions = int(row["conversions"] or 0)

        if spend < 1:
            recommendations.append({"priority": "Низький", "campaign": name,
"signal": "Недостатньо даних", "action": "Додайте щоденні метрики кампанії."})
            continue
        if roas >= 1.65 and conversions >= 8:
            recommendations.append({"priority": "Високий", "campaign": name,
"signal": f"Стабільний ROAS {roas:.2f} на {platform}", "action": "Підняти бюджет на
10-20% і контролювати CPA протягом 48 годин."})
        if roas < 0.9 and spend > 250:
            recommendations.append({"priority": "Високий", "campaign": name,
"signal": f"Низький ROAS {roas:.2f} при витратах {spend:.0f}", "action": "Зупинити
слабкі групи, перевірити офер, аудиторії та посадкову сторінку."})
        if ctr < 1.15 and spend > 150:
            recommendations.append({"priority": "Середній", "campaign": name,
"signal": f"CTR лише {ctr:.2f}%", "action": "Оновити hook, перший екран і візуальну
подачу креативу."})
        if cvr < 1.7 and conversions > 0:
```

```
            recommendations.append({"priority": "Середній", "campaign": name,
"signal": f"CVR {cvr:.2f}%", "action": "Перевірити відповідність реклами посадковій
сторінці та форму заявки."})
        if cpa > 45 and conversions >= 3:
            recommendations.append({"priority": "Середній", "campaign": name,
"signal": f"CPA {cpa:.2f}", "action": "Відключити дорогі плейсменти та змістити
бюджет у сегменти з ROAS > 1.3."})
        if revenue > 0 and spend > 0 and revenue - spend < 0:
            recommendations.append({"priority": "Високий", "campaign": name,
"signal": "Кампанія у мінусі", "action": "Поставити денний ліміт і перевірити
attribution window перед масштабуванням."})

    if not recommendations:
        recommendations.append({"priority": "Низький", "campaign": "Усі кампанії",
"signal": "Критичних відхилень немає", "action": "Продовжити A/B тестування
креативів і щоденний контроль витрат."})
    priority_order = {"Високий": 0, "Середній": 1, "Низький": 2}
    return sorted(recommendations, key=lambda x: priority_order.get(x["priority"],
9))[:14]


def build_alerts(rows: Iterable[object]) -> list[dict[str, str]]:
    alerts: list[dict[str, str]] = []
    for row in rows:
        spend = float(row["spend"] or 0)
        revenue = float(row["revenue"] or 0)
        budget = float(row["daily_budget"] or 0)
        roas = float(row["roas"] or 0)
        ctr = float(row["ctr"] or 0)
        conversions = int(row["conversions"] or 0)
        days = int(row["days_tracked"] or 0)
        name = row["name"]
        if budget > 0 and spend > budget * max(days, 1) * 1.2:
            alerts.append({"level": "Бюджет", "campaign": name, "message": "Фактичні
витрати перевищують плановий бюджет більше ніж на 20%."})
        if roas < 0.85 and spend > 300:
            alerts.append({"level": "Ризик", "campaign": name, "message": "ROAS
нижче беззбиткового рівня при суттєвих витратах."})
        if ctr < 1.0 and spend > 150:
            alerts.append({"level": "Креатив", "campaign": name, "message": "Низький
CTR: потрібне оновлення hook/visual angle."})
        if spend > 200 and conversions == 0:
            alerts.append({"level": "Конверсії", "campaign": name, "message": "Є
витрати, але немає конверсій."})
        if revenue - spend < -250:
            alerts.append({"level": "P&L", "campaign": name, "message": "Накопичений
мінус перевищив 250 дол."})
    return alerts[:20]


def ab_compare(a: object | None, b: object | None) -> dict[str, str]:
    if not a or not b:
        return {"winner": "—", "summary": "Оберіть два елементи для порівняння.",
"roas_delta": "—", "cpa_delta": "—", "ctr_delta": "—", "cvr_delta": "—"}
    roas_a = float(a["roas"] or 0)
    roas_b = float(b["roas"] or 0)
    cpa_a = float(a["cpa"] or 0)
    cpa_b = float(b["cpa"] or 0)
    ctr_a = float(a["ctr"] or 0)
    ctr_b = float(b["ctr"] or 0)
    cvr_a = float(a["cvr"] or 0)
    cvr_b = float(b["cvr"] or 0)
    score_a = roas_a * 2 + cvr_a / 10 + ctr_a / 20 - (cpa_a / 200 if cpa_a else 0)
```

```
    score_b = roas_b * 2 + cvr_b / 10 + ctr_b / 20 - (cpa_b / 200 if cpa_b else 0)
    winner = str(a["name"] if "name" in a.keys() else a["title"]) if score_a >=
score_b else str(b["name"] if "name" in b.keys() else b["title"])
    return {
        "winner": winner,
        "summary": f"Переможець за комбінованим score: {winner}. Оцінка враховує
ROAS, CPA, CTR та CVR.",
        "roas_delta": f"{roas_a - roas_b:+.2f}x",
        "cpa_delta": f"{cpa_a - cpa_b:+.2f}",
        "ctr_delta": f"{ctr_a - ctr_b:+.2f}%",
        "cvr_delta": f"{cvr_a - cvr_b:+.2f}%",
    }


def forecast_from_trend(rows: Sequence[object], days_forward: int = 7) ->
list[dict[str, float | str]]:
    recent = list(rows)[-14:]
    if not recent:
        return []
    avg_spend = mean(float(r["spend"] or 0) for r in recent)
    avg_revenue = mean(float(r["revenue"] or 0) for r in recent)
    result: list[dict[str, float | str]] = []
    # simple dampened trend based on last 7 versus previous 7
    first = recent[:7]
    last = recent[-7:]
    first_rev = mean(float(r["revenue"] or 0) for r in first) if first else
avg_revenue
    last_rev = mean(float(r["revenue"] or 0) for r in last) if last else avg_revenue
    trend_factor = 1.0
    if first_rev > 0:
        trend_factor = max(0.85, min(1.18, last_rev / first_rev))
    for i in range(1, days_forward + 1):
        factor = trend_factor ** min(i, 5)
        result.append({"day": f"+{i} день", "spend": avg_spend, "revenue":
avg_revenue * factor, "profit": avg_revenue * factor - avg_spend})
    return result
```

ДОДАТОК Б

Робота з базою даних

```
from __future__ import annotations

import csv
import hashlib
import hmac
import secrets
import sqlite3
from contextlib import contextmanager
from dataclasses import dataclass
from datetime import date, timedelta
from pathlib import Path
from random import Random
from typing import Any, Iterable

ROOT_DIR = Path(__file__).resolve().parents[1]
DATA_DIR = ROOT_DIR / "app_data"
EXPORT_DIR = ROOT_DIR / "exports"
DB_PATH = DATA_DIR / "mediabuying_bi.sqlite"
PASSWORD_ITERATIONS = 260_000


@dataclass(frozen=True)
class CampaignInput:
    name: str
    platform: str
    objective: str
    status: str
    daily_budget: float
    start_date: str
    owner: str
    owner_user_id: int | None = None


@dataclass(frozen=True)
class MetricInput:
    campaign_id: int
    metric_date: str
    impressions: int
    clicks: int
    leads: int
    conversions: int
    spend: float
    revenue: float
    notes: str = ""


@dataclass(frozen=True)
class CreativeInput:
    campaign_id: int
    title: str
    format: str
    angle: str
    funnel_stage: str
    status: str
    impressions: int
    clicks: int
```

```
    leads: int
    conversions: int
    spend: float
    revenue: float
    notes: str = ""


@dataclass(frozen=True)
class ScenarioInput:
    name: str
    platform: str
    target_revenue: float
    target_roas: float
    avg_cpa: float
    planned_days: int
    notes: str = ""


class DatabaseManager:
    """SQLite data layer for MediaBuying BI."""

    def __init__(self, db_path: Path | str = DB_PATH) -> None:
        self.db_path = Path(db_path)
        self.db_path.parent.mkdir(parents=True, exist_ok=True)
        EXPORT_DIR.mkdir(parents=True, exist_ok=True)
        self.init_db()

    @contextmanager
    def connect(self) -> Iterable[sqlite3.Connection]:
        conn = sqlite3.connect(self.db_path)
        conn.row_factory = sqlite3.Row
        conn.execute("PRAGMA foreign_keys = ON;")
        try:
            yield conn
            conn.commit()
        finally:
            conn.close()

    def init_db(self) -> None:
        with self.connect() as conn:
            conn.executescript(
                """
                CREATE TABLE IF NOT EXISTS users (
                    id INTEGER PRIMARY KEY AUTOINCREMENT,
                    full_name TEXT NOT NULL,
                    email TEXT NOT NULL UNIQUE,
                    password_salt TEXT NOT NULL,
                    password_hash TEXT NOT NULL,
                    iterations INTEGER NOT NULL,
                    role TEXT NOT NULL DEFAULT 'Media Buyer',
                    is_active INTEGER NOT NULL DEFAULT 1 CHECK(is_active IN (0, 1)),
                    created_at TEXT NOT NULL DEFAULT CURRENT_TIMESTAMP,
                    last_login_at TEXT
                );

                CREATE TABLE IF NOT EXISTS campaigns (
                    id INTEGER PRIMARY KEY AUTOINCREMENT,
                    name TEXT NOT NULL UNIQUE,
                    platform TEXT NOT NULL,
                    objective TEXT NOT NULL,
                    status TEXT NOT NULL DEFAULT 'Active',
                    daily_budget REAL NOT NULL DEFAULT 0 CHECK(daily_budget >= 0),
                    start_date TEXT NOT NULL,
```

```
                    owner TEXT NOT NULL DEFAULT 'Media Buyer',
                    owner_user_id INTEGER,
                    created_at TEXT NOT NULL DEFAULT CURRENT_TIMESTAMP,
                    FOREIGN KEY(owner_user_id) REFERENCES users(id) ON DELETE SET
NULL
                );

                CREATE TABLE IF NOT EXISTS daily_metrics (
                    id INTEGER PRIMARY KEY AUTOINCREMENT,
                    campaign_id INTEGER NOT NULL,
                    metric_date TEXT NOT NULL,
                    impressions INTEGER NOT NULL DEFAULT 0 CHECK(impressions >= 0),
                    clicks INTEGER NOT NULL DEFAULT 0 CHECK(clicks >= 0),
                    leads INTEGER NOT NULL DEFAULT 0 CHECK(leads >= 0),
                    conversions INTEGER NOT NULL DEFAULT 0 CHECK(conversions >= 0),
                    spend REAL NOT NULL DEFAULT 0 CHECK(spend >= 0),
                    revenue REAL NOT NULL DEFAULT 0 CHECK(revenue >= 0),
                    notes TEXT DEFAULT '',
                    created_by_user_id INTEGER,
                    created_at TEXT NOT NULL DEFAULT CURRENT_TIMESTAMP,
                    FOREIGN KEY(campaign_id) REFERENCES campaigns(id) ON DELETE
CASCADE,
                    FOREIGN KEY(created_by_user_id) REFERENCES users(id) ON DELETE
SET NULL,
                    UNIQUE(campaign_id, metric_date)
                );

                CREATE TABLE IF NOT EXISTS creative_assets (
                    id INTEGER PRIMARY KEY AUTOINCREMENT,
                    campaign_id INTEGER NOT NULL,
                    title TEXT NOT NULL,
                    format TEXT NOT NULL,
                    angle TEXT NOT NULL,
                    funnel_stage TEXT NOT NULL,
                    status TEXT NOT NULL DEFAULT 'Testing',
                    impressions INTEGER NOT NULL DEFAULT 0 CHECK(impressions >= 0),
                    clicks INTEGER NOT NULL DEFAULT 0 CHECK(clicks >= 0),
                    leads INTEGER NOT NULL DEFAULT 0 CHECK(leads >= 0),
                    conversions INTEGER NOT NULL DEFAULT 0 CHECK(conversions >= 0),
                    spend REAL NOT NULL DEFAULT 0 CHECK(spend >= 0),
                    revenue REAL NOT NULL DEFAULT 0 CHECK(revenue >= 0),
                    notes TEXT DEFAULT '',
                    created_by_user_id INTEGER,
                    created_at TEXT NOT NULL DEFAULT CURRENT_TIMESTAMP,
                    updated_at TEXT NOT NULL DEFAULT CURRENT_TIMESTAMP,
                    FOREIGN KEY(campaign_id) REFERENCES campaigns(id) ON DELETE
CASCADE,
                    FOREIGN KEY(created_by_user_id) REFERENCES users(id) ON DELETE
SET NULL
                );

                CREATE TABLE IF NOT EXISTS budget_scenarios (
                    id INTEGER PRIMARY KEY AUTOINCREMENT,
                    name TEXT NOT NULL,
                    platform TEXT NOT NULL,
                    target_revenue REAL NOT NULL DEFAULT 0,
                    target_roas REAL NOT NULL DEFAULT 1,
                    avg_cpa REAL NOT NULL DEFAULT 0,
                    planned_days INTEGER NOT NULL DEFAULT 30,
                    planned_budget REAL NOT NULL DEFAULT 0,
                    expected_conversions REAL NOT NULL DEFAULT 0,
                    daily_budget REAL NOT NULL DEFAULT 0,
                    notes TEXT DEFAULT '',
```

```
                    created_by_user_id INTEGER,
                    created_at TEXT NOT NULL DEFAULT CURRENT_TIMESTAMP,
                    FOREIGN KEY(created_by_user_id) REFERENCES users(id) ON DELETE
SET NULL
                );

                CREATE TABLE IF NOT EXISTS audit_log (
                    id INTEGER PRIMARY KEY AUTOINCREMENT,
                    user_id INTEGER,
                    action TEXT NOT NULL,
                    details TEXT DEFAULT '',
                    created_at TEXT NOT NULL DEFAULT CURRENT_TIMESTAMP,
                    FOREIGN KEY(user_id) REFERENCES users(id) ON DELETE SET NULL
                );

                CREATE INDEX IF NOT EXISTS idx_metrics_campaign_date ON
daily_metrics(campaign_id, metric_date);
                CREATE INDEX IF NOT EXISTS idx_metrics_date ON
daily_metrics(metric_date);
                CREATE INDEX IF NOT EXISTS idx_campaign_platform ON
campaigns(platform);
                CREATE INDEX IF NOT EXISTS idx_creatives_campaign ON
creative_assets(campaign_id);
                CREATE INDEX IF NOT EXISTS idx_users_email ON users(email);
                CREATE INDEX IF NOT EXISTS idx_audit_user_date ON audit_log(user_id,
created_at);
                """
            )
            self._ensure_column(conn, "campaigns", "owner_user_id", "INTEGER")
            self._ensure_column(conn, "daily_metrics", "created_by_user_id",
"INTEGER")
            conn.executescript(
                """
                DROP VIEW IF EXISTS campaign_performance;
                CREATE VIEW campaign_performance AS
                SELECT
                    c.id AS campaign_id,
                    c.name,
                    c.platform,
                    c.objective,
                    c.status,
                    c.daily_budget,
                    c.owner,
                    c.owner_user_id,
                    COUNT(m.id) AS days_tracked,
                    COALESCE(SUM(m.impressions), 0) AS impressions,
                    COALESCE(SUM(m.clicks), 0) AS clicks,
                    COALESCE(SUM(m.leads), 0) AS leads,
                    COALESCE(SUM(m.conversions), 0) AS conversions,
                    COALESCE(SUM(m.spend), 0) AS spend,
                    COALESCE(SUM(m.revenue), 0) AS revenue,
                    COALESCE(SUM(m.revenue) - SUM(m.spend), 0) AS profit,
                    CASE WHEN SUM(m.spend) > 0 THEN SUM(m.revenue) / SUM(m.spend)
ELSE 0 END AS roas,
                    CASE WHEN SUM(m.spend) > 0 THEN (SUM(m.revenue) - SUM(m.spend))
/ SUM(m.spend) * 100 ELSE 0 END AS roi,
                    CASE WHEN SUM(m.impressions) > 0 THEN SUM(m.clicks) * 100.0 /
SUM(m.impressions) ELSE 0 END AS ctr,
                    CASE WHEN SUM(m.clicks) > 0 THEN SUM(m.conversions) * 100.0 /
SUM(m.clicks) ELSE 0 END AS cvr,
                    CASE WHEN SUM(m.conversions) > 0 THEN SUM(m.spend) /
SUM(m.conversions) ELSE 0 END AS cpa,
                    CASE WHEN SUM(m.clicks) > 0 THEN SUM(m.spend) / SUM(m.clicks)
```

```
ELSE 0 END AS cpc
                FROM campaigns c
                LEFT JOIN daily_metrics m ON c.id = m.campaign_id
                GROUP BY c.id;
                """
            )
        self.seed_default_users()
        self.seed_if_empty()

    def _ensure_column(self, conn: sqlite3.Connection, table: str, column: str, ddl:
str) -> None:
        columns = {row[1] for row in conn.execute(f"PRAGMA
table_info({table})").fetchall()}
        if column not in columns:
            conn.execute(f"ALTER TABLE {table} ADD COLUMN {column} {ddl}")

    # ---------- generic ----------

    def fetch_all(self, sql: str, params: tuple[Any, ...] = ()) ->
list[sqlite3.Row]:
        with self.connect() as conn:
            return conn.execute(sql, params).fetchall()

    def fetch_one(self, sql: str, params: tuple[Any, ...] = ()) -> sqlite3.Row |
None:
        with self.connect() as conn:
            return conn.execute(sql, params).fetchone()

    # ---------- auth ----------

    @staticmethod
    def normalize_email(email: str) -> str:
        return email.strip().lower()

    @staticmethod
    def hash_password(password: str, salt: bytes | None = None, iterations: int =
PASSWORD_ITERATIONS) -> tuple[str, str, int]:
        salt = salt or secrets.token_bytes(16)
        digest = hashlib.pbkdf2_hmac("sha256", password.encode("utf-8"), salt,
iterations)
        return salt.hex(), digest.hex(), iterations

    @staticmethod
    def verify_password(password: str, salt_hex: str, hash_hex: str, iterations:
int) -> bool:
        digest = hashlib.pbkdf2_hmac("sha256", password.encode("utf-8"),
bytes.fromhex(salt_hex), int(iterations))
        return hmac.compare_digest(digest.hex(), hash_hex)

    @staticmethod
    def password_errors(password: str) -> list[str]:
        errors: list[str] = []
        if len(password) < 8:
            errors.append("мінімум 8 символів")
        if not any(ch.isupper() for ch in password):
            errors.append("хоча б одна велика літера")
        if not any(ch.islower() for ch in password):
            errors.append("хоча б одна мала літера")
        if not any(ch.isdigit() for ch in password):
            errors.append("хоча б одна цифра")
        return errors

    def seed_default_users(self) -> None:
```

```
        with self.connect() as conn:
            if conn.execute("SELECT COUNT(*) FROM users").fetchone()[0]:
                return
            self.create_user("Адміністратор BI", "admin@demo.local", "Admin12345!",
"Admin", conn=conn)
            self.create_user("Media Buyer Demo", "buyer@demo.local", "Buyer12345!",
"Media Buyer", conn=conn)
            self.create_user("Analyst Demo", "analyst@demo.local", "Analyst12345!",
"Analyst", conn=conn)

    def create_user(self, full_name: str, email: str, password: str, role: str =
"Media Buyer", conn: sqlite3.Connection | None = None) -> int:
        full_name = full_name.strip()
        email = self.normalize_email(email)
        if not full_name:
            raise ValueError("Вкажіть ім'я користувача.")
        if "@" not in email or "." not in email.split("@")[-1]:
            raise ValueError("Вкажіть коректний email.")
        errors = self.password_errors(password)
        if errors:
            raise ValueError("Пароль слабкий: " + ", ".join(errors) + ".")
        salt, password_hash, iterations = self.hash_password(password)
        sql = """
            INSERT INTO users(full_name, email, password_salt, password_hash,
iterations, role)
            VALUES (?, ?, ?, ?, ?, ?)
        """
        params = (full_name, email, salt, password_hash, iterations, role)
        target = conn
        if target is not None:
            cur = target.execute(sql, params)
            user_id = int(cur.lastrowid)
            target.execute("INSERT INTO audit_log(user_id, action, details) VALUES
(?, ?, ?)", (user_id, "register", email))
            return user_id
        with self.connect() as local:
            cur = local.execute(sql, params)
            user_id = int(cur.lastrowid)
            local.execute("INSERT INTO audit_log(user_id, action, details) VALUES
(?, ?, ?)", (user_id, "register", email))
            return user_id

    def authenticate_user(self, email: str, password: str) -> sqlite3.Row | None:
        email = self.normalize_email(email)
        with self.connect() as conn:
            row = conn.execute("SELECT * FROM users WHERE email=? AND is_active=1",
(email,)).fetchone()
            if not row:
                conn.execute("INSERT INTO audit_log(action, details) VALUES (?, ?)",
("login_failed", f"unknown: {email}"))
                return None
            if not self.verify_password(password, row["password_salt"],
row["password_hash"], int(row["iterations"])):
                conn.execute("INSERT INTO audit_log(user_id, action, details) VALUES
(?, ?, ?)", (row["id"], "login_failed", "wrong password"))
                return None
            conn.execute("UPDATE users SET last_login_at=CURRENT_TIMESTAMP WHERE
id=?", (row["id"],))
            conn.execute("INSERT INTO audit_log(user_id, action, details) VALUES (?,
?, ?)", (row["id"], "login", "success"))
            return conn.execute("SELECT * FROM users WHERE id=?",
(row["id"],)).fetchone()
```

```
    def change_password(self, user_id: int, old_password: str, new_password: str) ->
None:
        with self.connect() as conn:
            row = conn.execute("SELECT * FROM users WHERE id=? AND is_active=1",
(user_id,)).fetchone()
            if not row:
                raise ValueError("Користувача не знайдено.")
            if not self.verify_password(old_password, row["password_salt"],
row["password_hash"], int(row["iterations"])):
                raise ValueError("Поточний пароль введено неправильно.")
            errors = self.password_errors(new_password)
            if errors:
                raise ValueError("Новий пароль слабкий: " + ", ".join(errors) + ".")
            salt, password_hash, iterations = self.hash_password(new_password)
            conn.execute(
                "UPDATE users SET password_salt=?, password_hash=?, iterations=?
WHERE id=?",
                (salt, password_hash, iterations, user_id),
            )
            conn.execute("INSERT INTO audit_log(user_id, action, details) VALUES (?,
?, ?)", (user_id, "password_change", "password updated"))

    def log_event(self, user_id: int | None, action: str, details: str = "") ->
None:
        with self.connect() as conn:
            conn.execute("INSERT INTO audit_log(user_id, action, details) VALUES (?,
?, ?)", (user_id, action, details))

    def recent_audit_logs(self, limit: int = 100) -> list[sqlite3.Row]:
        return self.fetch_all(
            """
            SELECT a.created_at, COALESCE(u.full_name, 'Система') AS user_name,
a.action, a.details
            FROM audit_log a
            LEFT JOIN users u ON u.id = a.user_id
            ORDER BY a.id DESC
            LIMIT ?
            """,
            (limit,),
        )

    # ---------- seed ----------

    def seed_if_empty(self) -> None:
        with self.connect() as conn:
            if conn.execute("SELECT COUNT(*) FROM campaigns").fetchone()[0]:
                return
            admin = conn.execute("SELECT id, full_name FROM users WHERE role='Admin'
ORDER BY id LIMIT 1").fetchone()
            owner_id = int(admin["id"]) if admin else None
            owner = admin["full_name"] if admin else "Адміністратор BI"
            campaigns = [
                CampaignInput("Meta | Fitness Leads | UA", "Meta Ads", "Leads",
"Active", 180, "2026-02-01", owner, owner_id),
                CampaignInput("Google | Brand Search | UA", "Google Ads", "Sales",
"Active", 130, "2026-02-05", owner, owner_id),
                CampaignInput("TikTok | Creatives Test | PL", "TikTok Ads",
"Conversions", "Testing", 95, "2026-02-12", owner, owner_id),
                CampaignInput("Meta | Retargeting Warm | EU", "Meta Ads", "Sales",
"Active", 75, "2026-02-18", owner, owner_id),
                CampaignInput("Google | Performance Max | EU", "Google Ads",
"Revenue", "Active", 210, "2026-03-01", owner, owner_id),
                CampaignInput("LinkedIn | B2B Leads | EU", "LinkedIn Ads", "Leads",
```

```
"Paused", 120, "2026-03-08", owner, owner_id),
            ]
            ids = [self.add_campaign(item, conn=conn, actor_user_id=None) for item
in campaigns]
            rng = Random(44)
            today = date.today()
            multipliers = [1.25, 1.42, 1.05, 1.78, 1.36, 0.88]
            for idx, campaign_id in enumerate(ids):
                base = campaigns[idx].daily_budget
                for offset in range(59, -1, -1):
                    day = today - timedelta(days=offset)
                    weekday = 0.76 if day.weekday() in (5, 6) else 1.0
                    spend = round(base * weekday * rng.uniform(0.72, 1.28), 2)
                    impressions = int(spend * rng.uniform(80, 175))
                    clicks = max(1, int(impressions * rng.uniform(0.010, 0.034)))
                    leads = max(0, int(clicks * rng.uniform(0.020, 0.085)))
                    conversions = max(0, int(clicks * rng.uniform(0.003, 0.020)))
                    if conversions == 0 and idx in (1, 3, 4) and rng.random() >
0.48:
                        conversions = 1
                    revenue = round(spend * multipliers[idx] * rng.uniform(0.68,
1.45), 2)
                    metric = MetricInput(campaign_id, day.isoformat(), impressions,
clicks, leads, conversions, spend, revenue, "demo seed")
                    self.upsert_metric(metric, conn=conn, actor_user_id=None)
            creative_titles = [
                (ids[0], "UGC: швидкий результат", "Video", "Pain → solution",
"TOFU", "Winning", 62000, 1860, 122, 29, 1380, 2870),
                (ids[0], "Static: before/after", "Image", "Social proof", "MOFU",
"Testing", 44000, 970, 74, 11, 760, 1120),
                (ids[1], "Search ad: brand offer", "Search", "Direct intent",
"BOFU", "Winning", 31000, 1580, 80, 43, 910, 2860),
                (ids[2], "TikTok hook 3 sec", "Video", "Curiosity hook", "TOFU",
"Testing", 88000, 2140, 95, 14, 1210, 1360),
                (ids[3], "Retargeting testimonial", "Video", "Objection handling",
"BOFU", "Winning", 27000, 1040, 61, 28, 540, 1745),
                (ids[4], "PMax asset group A", "Mixed", "Value stack", "MOFU",
"Scaling", 76000, 2110, 130, 35, 2310, 3750),
            ]
            for data in creative_titles:
                self.add_creative(CreativeInput(*data, notes="demo creative"),
conn=conn, actor_user_id=None)
            self.save_scenario(ScenarioInput("Scale winners Meta + Google", "Mixed",
15000, 1.45, 38, 30, "demo plan"), conn=conn, actor_user_id=None)

    # ---------- campaigns ----------

    def add_campaign(self, item: CampaignInput, conn: sqlite3.Connection | None =
None, actor_user_id: int | None = None) -> int:
        sql = """
            INSERT INTO campaigns(name, platform, objective, status, daily_budget,
start_date, owner, owner_user_id)
            VALUES (?, ?, ?, ?, ?, ?, ?, ?)
        """
        params = (item.name.strip(), item.platform, item.objective, item.status,
float(item.daily_budget), item.start_date, item.owner, item.owner_user_id)
        target = conn
        if target is not None:
            cur = target.execute(sql, params)
            campaign_id = int(cur.lastrowid)
            if actor_user_id:
                target.execute("INSERT INTO audit_log(user_id, action, details)
VALUES (?, ?, ?)", (actor_user_id, "campaign_create", item.name))
```

```
            return campaign_id
        with self.connect() as local:
            cur = local.execute(sql, params)
            campaign_id = int(cur.lastrowid)
            if actor_user_id:
                local.execute("INSERT INTO audit_log(user_id, action, details)
VALUES (?, ?, ?)", (actor_user_id, "campaign_create", item.name))
            return campaign_id

    def update_campaign(self, campaign_id: int, item: CampaignInput, actor_user_id:
int | None = None) -> None:
        with self.connect() as conn:
            conn.execute(
                """
                UPDATE campaigns
                SET name=?, platform=?, objective=?, status=?, daily_budget=?,
start_date=?, owner=?, owner_user_id=?
                WHERE id=?
                """,
                (item.name.strip(), item.platform, item.objective, item.status,
float(item.daily_budget), item.start_date, item.owner, item.owner_user_id,
campaign_id),
            )
            if actor_user_id:
                conn.execute("INSERT INTO audit_log(user_id, action, details) VALUES
(?, ?, ?)", (actor_user_id, "campaign_update", item.name))

    def delete_campaign(self, campaign_id: int, actor_user_id: int | None = None) ->
None:
        with self.connect() as conn:
            row = conn.execute("SELECT name FROM campaigns WHERE id=?",
(campaign_id,)).fetchone()
            conn.execute("DELETE FROM campaigns WHERE id=?", (campaign_id,))
            if actor_user_id:
                conn.execute("INSERT INTO audit_log(user_id, action, details) VALUES
(?, ?, ?)", (actor_user_id, "campaign_delete", row["name"] if row else
str(campaign_id)))

    def campaigns(self) -> list[sqlite3.Row]:
        return self.fetch_all("SELECT * FROM campaigns ORDER BY status, platform,
name")

    def campaign_choices(self) -> list[sqlite3.Row]:
        return self.fetch_all("SELECT id, name FROM campaigns ORDER BY name")

    def campaign_by_id(self, campaign_id: int) -> sqlite3.Row | None:
        return self.fetch_one("SELECT * FROM campaigns WHERE id=?", (campaign_id,))

    # ---------- metrics ----------

    def upsert_metric(self, item: MetricInput, conn: sqlite3.Connection | None =
None, actor_user_id: int | None = None) -> None:
        sql = """
            INSERT INTO daily_metrics(campaign_id, metric_date, impressions, clicks,
leads, conversions, spend, revenue, notes, created_by_user_id)
            VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?)
            ON CONFLICT(campaign_id, metric_date)
            DO UPDATE SET impressions=excluded.impressions,
                          clicks=excluded.clicks,
                          leads=excluded.leads,
                          conversions=excluded.conversions,
                          spend=excluded.spend,
                          revenue=excluded.revenue,
```

```
                        notes=excluded.notes,
                        created_by_user_id=excluded.created_by_user_id
        """
        params = (item.campaign_id, item.metric_date, item.impressions, item.clicks,
item.leads, item.conversions, item.spend, item.revenue, item.notes, actor_user_id)
        target = conn
        if target is not None:
            target.execute(sql, params)
            if actor_user_id:
                target.execute("INSERT INTO audit_log(user_id, action, details)
VALUES (?, ?, ?)", (actor_user_id, "metric_upsert", f"campaign={item.campaign_id},
date={item.metric_date}"))
            return
        with self.connect() as local:
            local.execute(sql, params)
            if actor_user_id:
                local.execute("INSERT INTO audit_log(user_id, action, details)
VALUES (?, ?, ?)", (actor_user_id, "metric_upsert", f"campaign={item.campaign_id},
date={item.metric_date}"))

    def recent_metrics(self, limit: int = 160) -> list[sqlite3.Row]:
        return self.fetch_all(
            """
            SELECT m.id, c.name AS campaign, c.platform, m.metric_date,
m.impressions, m.clicks, m.leads,
                   m.conversions, m.spend, m.revenue, m.notes
            FROM daily_metrics m
            JOIN campaigns c ON c.id=m.campaign_id
            ORDER BY m.metric_date DESC, m.id DESC
            LIMIT ?
            """,
            (limit,),
        )

    # ---------- creatives ----------

    def add_creative(self, item: CreativeInput, conn: sqlite3.Connection | None =
None, actor_user_id: int | None = None) -> int:
        sql = """
            INSERT INTO creative_assets(campaign_id, title, format, angle,
funnel_stage, status, impressions, clicks, leads, conversions, spend, revenue,
notes, created_by_user_id)
            VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)
        """
        params = (item.campaign_id, item.title.strip(), item.format,
item.angle.strip(), item.funnel_stage, item.status, item.impressions, item.clicks,
item.leads, item.conversions, item.spend, item.revenue, item.notes, actor_user_id)
        target = conn
        if target is not None:
            cur = target.execute(sql, params)
            creative_id = int(cur.lastrowid)
            if actor_user_id:
                target.execute("INSERT INTO audit_log(user_id, action, details)
VALUES (?, ?, ?)", (actor_user_id, "creative_create", item.title))
            return creative_id
        with self.connect() as local:
            cur = local.execute(sql, params)
            creative_id = int(cur.lastrowid)
            if actor_user_id:
                local.execute("INSERT INTO audit_log(user_id, action, details)
VALUES (?, ?, ?)", (actor_user_id, "creative_create", item.title))
            return creative_id
```

```
    def update_creative(self, creative_id: int, item: CreativeInput, actor_user_id:
int | None = None) -> None:
        with self.connect() as conn:
            conn.execute(
                """
                UPDATE creative_assets
                SET campaign_id=?, title=?, format=?, angle=?, funnel_stage=?,
status=?, impressions=?, clicks=?, leads=?, conversions=?, spend=?, revenue=?,
notes=?, updated_at=CURRENT_TIMESTAMP
                WHERE id=?
                """,
                (item.campaign_id, item.title.strip(), item.format,
item.angle.strip(), item.funnel_stage, item.status, item.impressions, item.clicks,
item.leads, item.conversions, item.spend, item.revenue, item.notes, creative_id),
            )
            if actor_user_id:
                conn.execute("INSERT INTO audit_log(user_id, action, details) VALUES
(?, ?, ?)", (actor_user_id, "creative_update", item.title))

    def delete_creative(self, creative_id: int, actor_user_id: int | None = None) ->
None:
        with self.connect() as conn:
            row = conn.execute("SELECT title FROM creative_assets WHERE id=?",
(creative_id,)).fetchone()
            conn.execute("DELETE FROM creative_assets WHERE id=?", (creative_id,))
            if actor_user_id:
                conn.execute("INSERT INTO audit_log(user_id, action, details) VALUES
(?, ?, ?)", (actor_user_id, "creative_delete", row["title"] if row else
str(creative_id)))

    def creatives(self) -> list[sqlite3.Row]:
        return self.fetch_all(
            """
            SELECT cr.*, c.name AS campaign, c.platform,
                   CASE WHEN cr.spend > 0 THEN cr.revenue / cr.spend ELSE 0 END AS
roas,
                   CASE WHEN cr.impressions > 0 THEN cr.clicks * 100.0 /
cr.impressions ELSE 0 END AS ctr,
                   CASE WHEN cr.clicks > 0 THEN cr.conversions * 100.0 / cr.clicks
ELSE 0 END AS cvr,
                   CASE WHEN cr.conversions > 0 THEN cr.spend / cr.conversions ELSE
0 END AS cpa
            FROM creative_assets cr
            JOIN campaigns c ON c.id=cr.campaign_id
            ORDER BY roas DESC, cr.updated_at DESC
            """
        )

    def creative_by_id(self, creative_id: int) -> sqlite3.Row | None:
        return self.fetch_one("SELECT * FROM creative_assets WHERE id=?",
(creative_id,))

    # ---------- planner ----------

    def save_scenario(self, item: ScenarioInput, conn: sqlite3.Connection | None =
None, actor_user_id: int | None = None) -> int:
        target_roas = max(float(item.target_roas), 0.1)
        planned_days = max(int(item.planned_days), 1)
        planned_budget = float(item.target_revenue) / target_roas if
item.target_revenue else 0
        daily_budget = planned_budget / planned_days if planned_days else 0
        expected_conversions = planned_budget / float(item.avg_cpa) if item.avg_cpa
else 0
```

```
        sql = """
            INSERT INTO budget_scenarios(name, platform, target_revenue,
target_roas, avg_cpa, planned_days, planned_budget, expected_conversions,
daily_budget, notes, created_by_user_id)
            VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)
        """
        params = (item.name.strip(), item.platform, item.target_revenue,
target_roas, item.avg_cpa, planned_days, planned_budget, expected_conversions,
daily_budget, item.notes, actor_user_id)
        target = conn
        if target is not None:
            cur = target.execute(sql, params)
            scenario_id = int(cur.lastrowid)
            if actor_user_id:
                target.execute("INSERT INTO audit_log(user_id, action, details)
VALUES (?, ?, ?)", (actor_user_id, "scenario_create", item.name))
            return scenario_id
        with self.connect() as local:
            cur = local.execute(sql, params)
            scenario_id = int(cur.lastrowid)
            if actor_user_id:
                local.execute("INSERT INTO audit_log(user_id, action, details)
VALUES (?, ?, ?)", (actor_user_id, "scenario_create", item.name))
            return scenario_id

    def scenarios(self) -> list[sqlite3.Row]:
        return self.fetch_all("SELECT * FROM budget_scenarios ORDER BY id DESC LIMIT
80")

    # ---------- analytics ----------

    def platforms(self) -> list[str]:
        return [r["platform"] for r in self.fetch_all("SELECT DISTINCT platform FROM
campaigns ORDER BY platform")]

    def performance(self, days: int | None = None, platform: str | None = None) ->
list[sqlite3.Row]:
        params: list[Any] = []
        where: list[str] = []
        if days is not None:
            where.append("campaign_id IN (SELECT campaign_id FROM daily_metrics
WHERE metric_date >= date('now', ?))")
            params.append(f"-{int(days)} days")
        if platform and platform != "Усі платформи":
            where.append("platform = ?")
            params.append(platform)
        clause = "WHERE " + " AND ".join(where) if where else ""
        return self.fetch_all(f"SELECT * FROM campaign_performance {clause} ORDER BY
roas DESC, spend DESC", tuple(params))

    def totals(self, days: int | None = None, platform: str | None = None) ->
dict[str, float]:
        rows = self.performance(days=days, platform=platform)
        spend = sum(float(r["spend"] or 0) for r in rows)
        revenue = sum(float(r["revenue"] or 0) for r in rows)
        impressions = sum(int(r["impressions"] or 0) for r in rows)
        clicks = sum(int(r["clicks"] or 0) for r in rows)
        conversions = sum(int(r["conversions"] or 0) for r in rows)
        leads = sum(int(r["leads"] or 0) for r in rows)
        profit = revenue - spend
        return {
            "spend": spend,
            "revenue": revenue,
```

```
            "profit": profit,
            "roas": revenue / spend if spend else 0,
            "roi": profit / spend * 100 if spend else 0,
            "ctr": clicks * 100 / impressions if impressions else 0,
            "cvr": conversions * 100 / clicks if clicks else 0,
            "cpa": spend / conversions if conversions else 0,
            "cpc": spend / clicks if clicks else 0,
            "clicks": clicks,
            "leads": leads,
            "conversions": conversions,
            "impressions": impressions,
        }

    def trend(self, days: int = 30, platform: str | None = None) ->
list[sqlite3.Row]:
        params: list[Any] = [f"-{int(days)} days"]
        platform_clause = ""
        if platform and platform != "Усі платформи":
            platform_clause = "AND c.platform = ?"
            params.append(platform)
        return self.fetch_all(
            f"""
            SELECT m.metric_date,
                   COALESCE(SUM(m.spend),0) AS spend,
                   COALESCE(SUM(m.revenue),0) AS revenue,
                   COALESCE(SUM(m.conversions),0) AS conversions,
                   COALESCE(SUM(m.clicks),0) AS clicks,
                   COALESCE(SUM(m.impressions),0) AS impressions
            FROM daily_metrics m
            JOIN campaigns c ON c.id=m.campaign_id
            WHERE m.metric_date >= date('now', ?) {platform_clause}
            GROUP BY m.metric_date
            ORDER BY m.metric_date
            """,
            tuple(params),
        )

    def platform_summary(self, days: int | None = None) -> list[sqlite3.Row]:
        params: list[Any] = []
        metric_clause = ""
        if days is not None:
            metric_clause = "WHERE m.metric_date >= date('now', ?)"
            params.append(f"-{int(days)} days")
        return self.fetch_all(
            f"""
            SELECT c.platform,
                   COALESCE(SUM(m.spend),0) AS spend,
                   COALESCE(SUM(m.revenue),0) AS revenue,
                   COALESCE(SUM(m.revenue)-SUM(m.spend),0) AS profit,
                   CASE WHEN SUM(m.spend)>0 THEN SUM(m.revenue)/SUM(m.spend) ELSE 0
END AS roas,
                   CASE WHEN SUM(m.impressions)>0 THEN
SUM(m.clicks)*100.0/SUM(m.impressions) ELSE 0 END AS ctr,
                   CASE WHEN SUM(m.clicks)>0 THEN
SUM(m.conversions)*100.0/SUM(m.clicks) ELSE 0 END AS cvr,
                   COALESCE(SUM(m.conversions),0) AS conversions
            FROM campaigns c
            LEFT JOIN daily_metrics m ON m.campaign_id=c.id
            {metric_clause}
            GROUP BY c.platform
            ORDER BY roas DESC
            """,
            tuple(params),
```

```
        )

    def objective_summary(self, days: int | None = None) -> list[sqlite3.Row]:
        params: list[Any] = []
        metric_clause = ""
        if days is not None:
            metric_clause = "WHERE m.metric_date >= date('now', ?)"
            params.append(f"-{int(days)} days")
        return self.fetch_all(
            f"""
            SELECT c.objective,
                   COALESCE(SUM(m.spend),0) AS spend,
                   COALESCE(SUM(m.revenue),0) AS revenue,
                   CASE WHEN SUM(m.spend)>0 THEN SUM(m.revenue)/SUM(m.spend) ELSE 0
END AS roas
            FROM campaigns c
            LEFT JOIN daily_metrics m ON m.campaign_id=c.id
            {metric_clause}
            GROUP BY c.objective
            ORDER BY spend DESC
            """,
            tuple(params),
        )

    def export_metrics_csv(self, actor_user_id: int | None = None) -> Path:
        rows = self.fetch_all(
            """
            SELECT c.name AS campaign, c.platform, c.objective, c.status, c.owner,
m.metric_date,
                   m.impressions, m.clicks, m.leads, m.conversions, m.spend,
m.revenue,
                   ROUND(CASE WHEN m.spend>0 THEN m.revenue/m.spend ELSE 0 END, 3)
AS roas,
                   ROUND(CASE WHEN m.clicks>0 THEN m.spend/m.clicks ELSE 0 END, 3)
AS cpc,
                   ROUND(CASE WHEN m.conversions>0 THEN m.spend/m.conversions ELSE 0
END, 3) AS cpa,
                   m.notes
            FROM daily_metrics m JOIN campaigns c ON c.id=m.campaign_id
            ORDER BY m.metric_date DESC, c.name
            """
        )
        path = EXPORT_DIR / "mediabuying_metrics_export.csv"
        with path.open("w", newline="", encoding="utf-8-sig") as fh:
            writer = csv.writer(fh)
            writer.writerow(["campaign", "platform", "objective", "status", "owner",
"date", "impressions", "clicks", "leads", "conversions", "spend", "revenue", "roas",
"cpc", "cpa", "notes"])
            for row in rows:
                writer.writerow([row[key] for key in row.keys()])
        self.log_event(actor_user_id, "export_csv", path.name)
        return path

    def export_creatives_csv(self, actor_user_id: int | None = None) -> Path:
        rows = self.creatives()
        path = EXPORT_DIR / "mediabuying_creatives_export.csv"
        with path.open("w", newline="", encoding="utf-8-sig") as fh:
            writer = csv.writer(fh)
            writer.writerow(["campaign", "platform", "title", "format", "angle",
"funnel_stage", "status", "impressions", "clicks", "leads", "conversions", "spend",
"revenue", "roas", "ctr", "cvr", "cpa", "notes"])
            for row in rows:
                writer.writerow([row["campaign"], row["platform"], row["title"],
```

```
row["format"], row["angle"], row["funnel_stage"], row["status"], row["impressions"],
row["clicks"], row["leads"], row["conversions"], row["spend"], row["revenue"],
row["roas"], row["ctr"], row["cvr"], row["cpa"], row["notes"]])
        self.log_event(actor_user_id, "export_creatives_csv", path.name)
        return path

    def export_summary_txt(self, days: int | None = 30, actor_user_id: int | None =
None) -> Path:
        totals = self.totals(days=days)
        rows = self.performance(days=days)[:10]
        path = EXPORT_DIR / "mediabuying_bi_summary.txt"
        with path.open("w", encoding="utf-8") as fh:
            fh.write("MEDIA BUYING BI SUMMARY\n")
            fh.write(f"Period: {'all time' if days is None else str(days) + '
days'}\n")
            fh.write(f"Spend: {totals['spend']:.2f}\nRevenue:
{totals['revenue']:.2f}\nProfit: {totals['profit']:.2f}\nROAS:
{totals['roas']:.2f}\nCPA: {totals['cpa']:.2f}\n\n")
            for row in rows:
                fh.write(f"- {row['name']} | {row['platform']} |
spend={row['spend']:.2f} | revenue={row['revenue']:.2f} | roas={row['roas']:.2f}\n")
        self.log_event(actor_user_id, "export_summary", path.name)
        return path
```

ДОДАТОК В

Скріншоти інтерфейсу та результатів роботи інформаційної системи

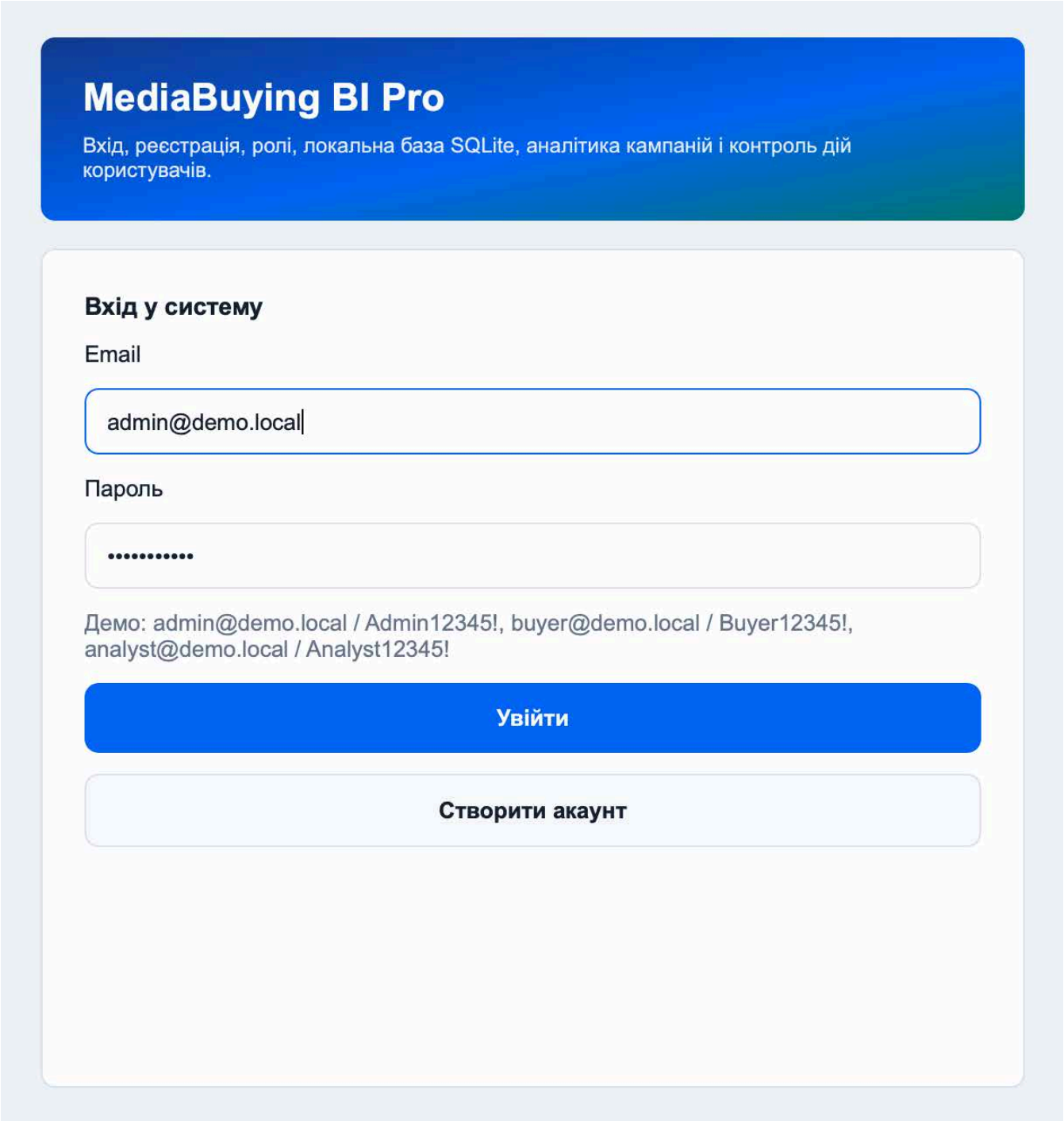

Рисунок В.1 – Вікно авторизації користувача інформаційної системи

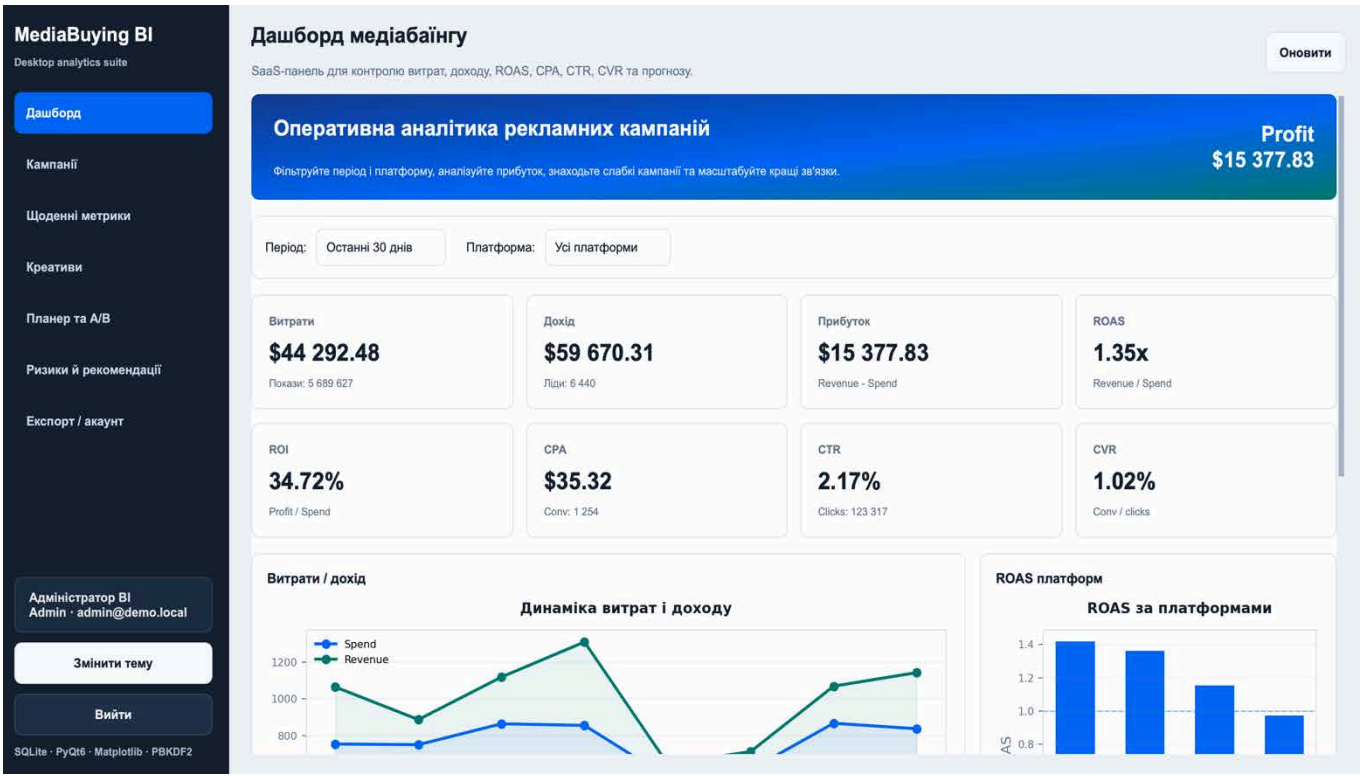

Рисунок В.2 – Дашборд з KPI-показниками рекламних кампаній

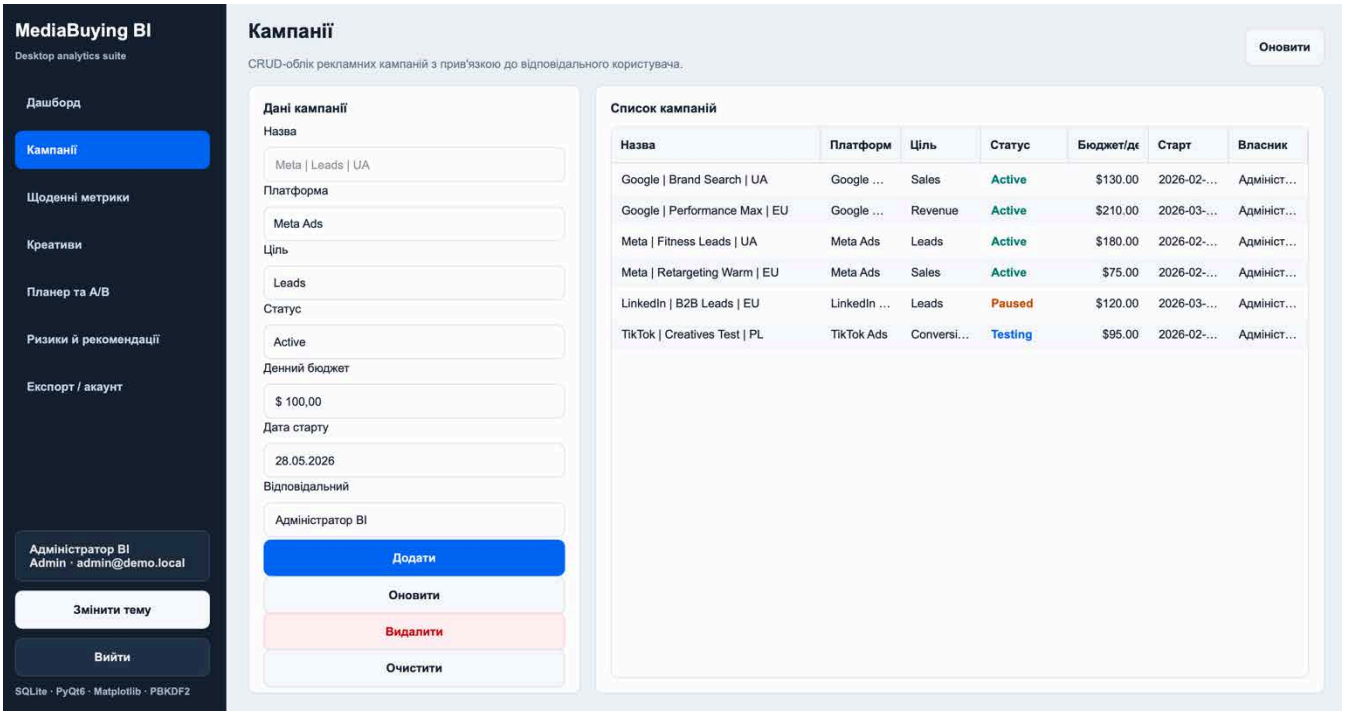

Рисунок В.3 – Реєстр рекламних кампаній у системі

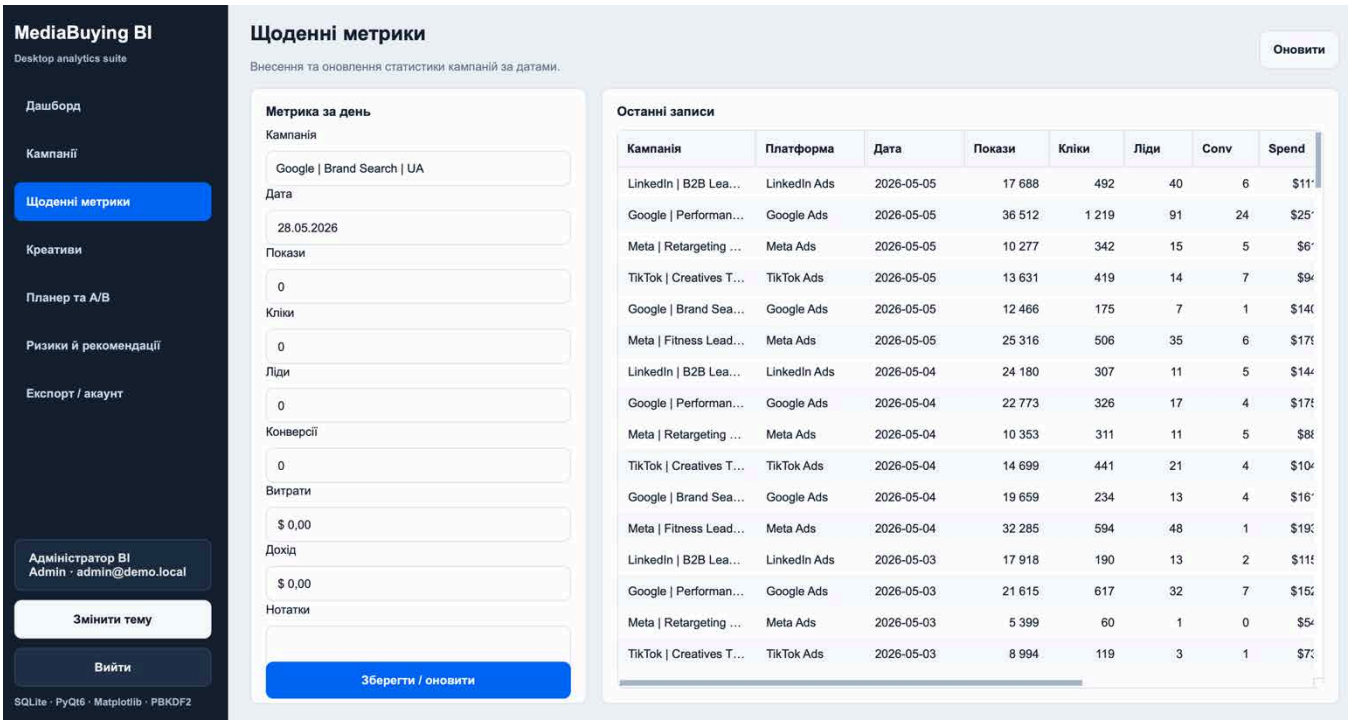

Рисунок В.4 – Введення та перегляд щоденних метрик рекламної кампанії

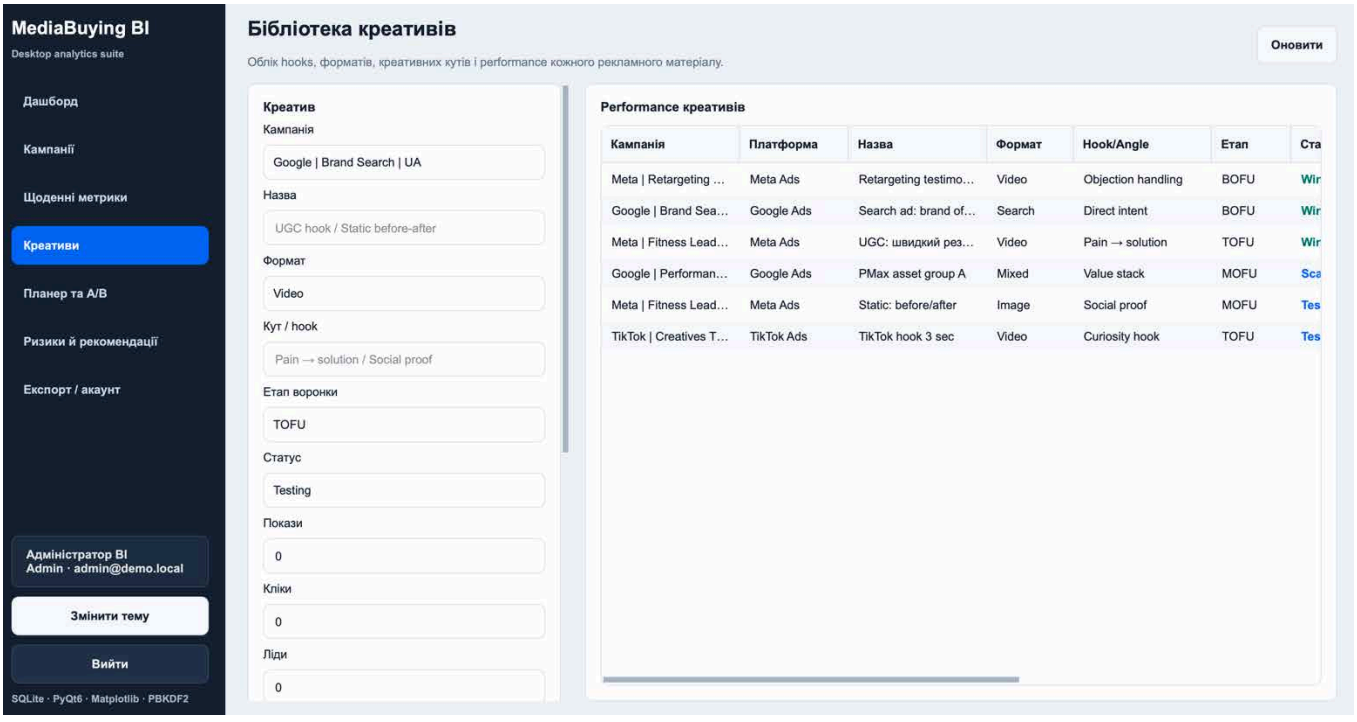

Рисунок В.5 – Облік і аналіз ефективності рекламних креативів

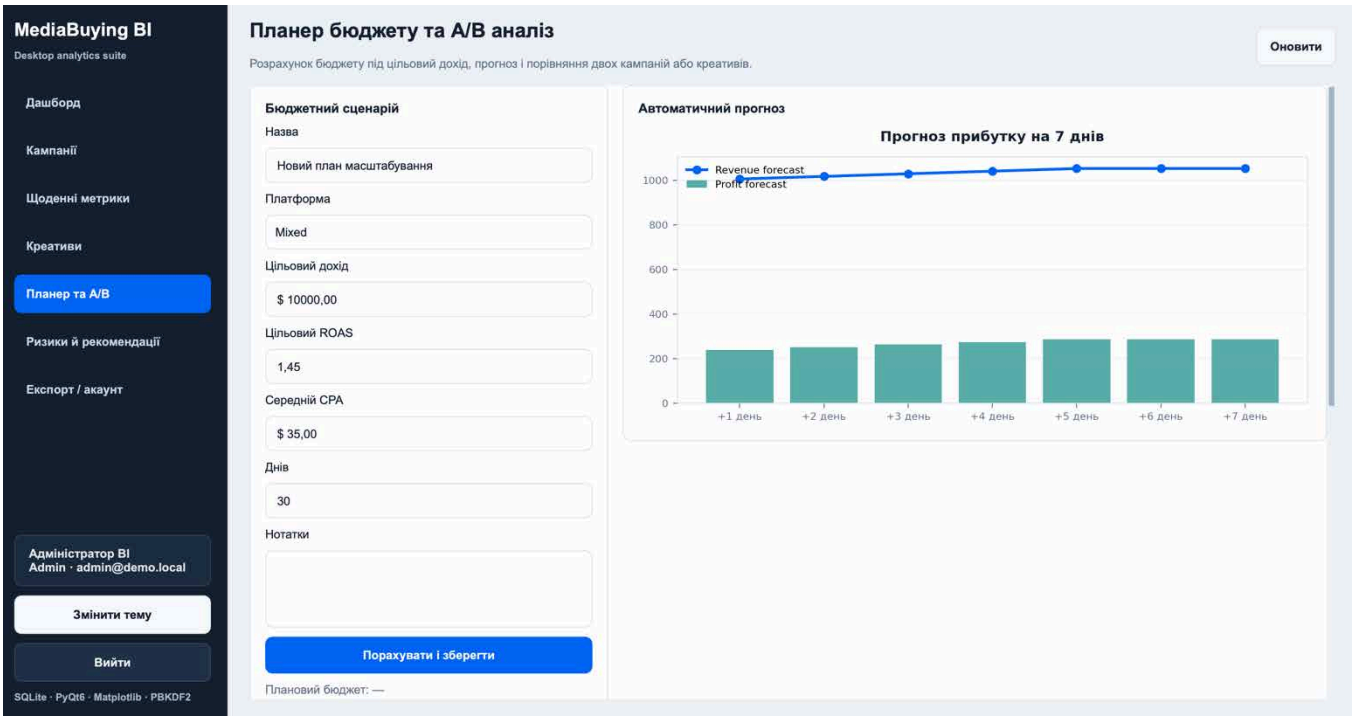

Рисунок В.6 – Планування бюджету та A/B-аналіз рекламних кампаній

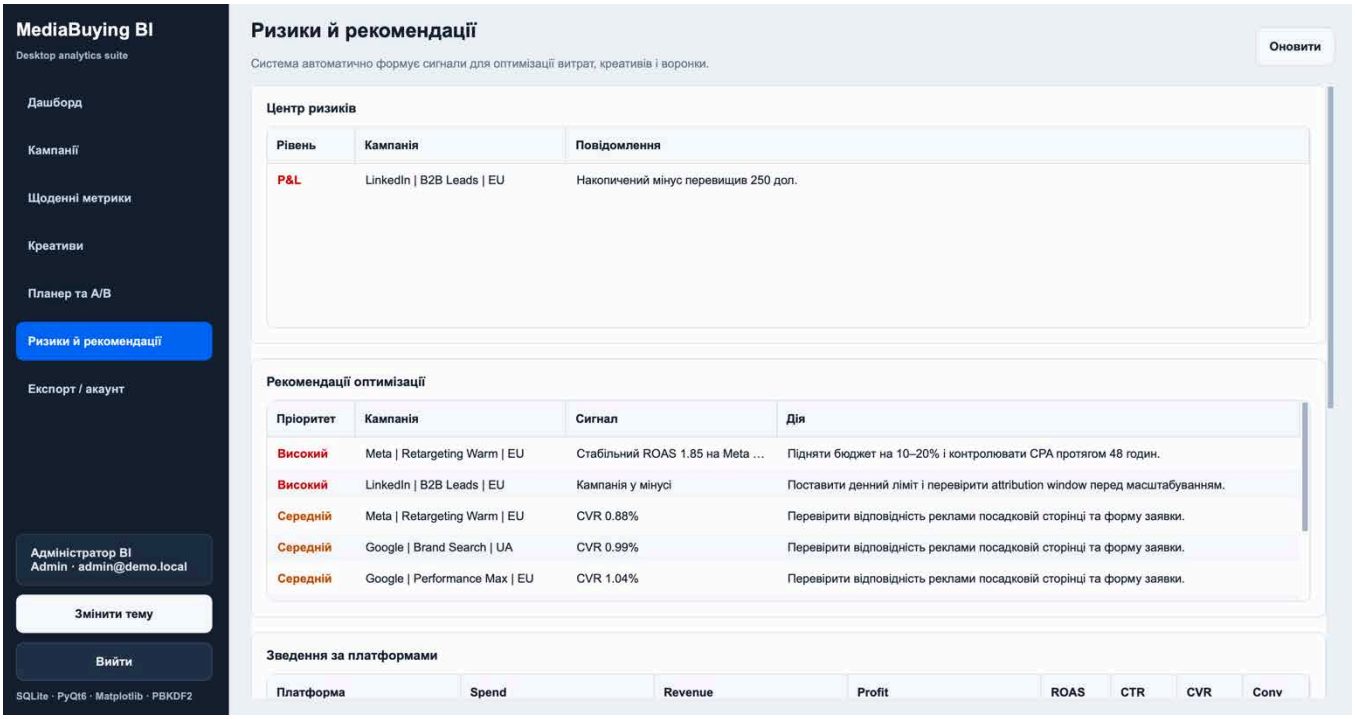

Рисунок В.7 – Виявлення ризиків і формування рекомендацій щодо оптимізації кампаній

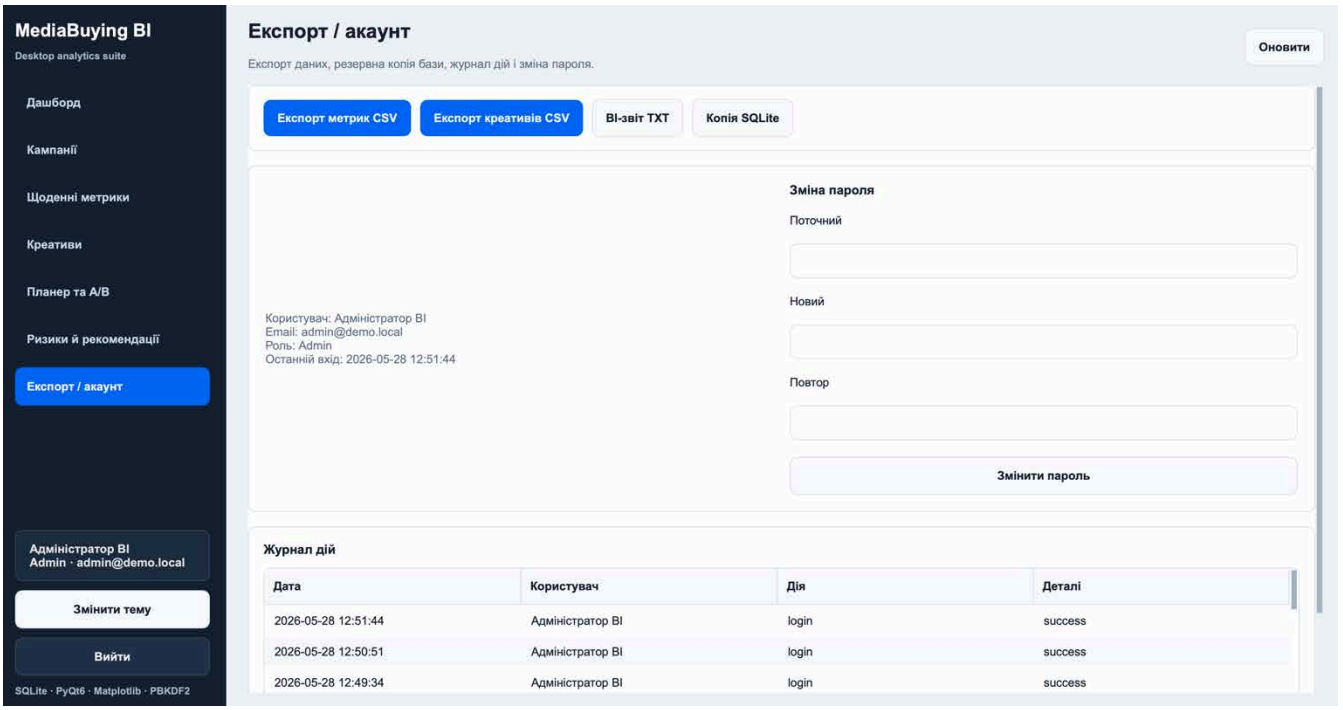

Рисунок В.8 – Експорт аналітичних даних та журнал дій користувача

ДОДАТОК Г

Календарний план

№ з/п	Назва етапів виконання бакалаврської кваліфікаційної роботи	Строк виконання етапів бакалаврської кваліфікаційної роботи	Примітки
1	Одержання завдання на бакалаврську кваліфікаційну роботу, уточнення теми, визначення структури та розроблення календарного плану.	01.12.2025 – 05.12.2025	Затверджений календарний план
2	Дослідження предметної області бізнес-аналітики у сфері медіабаїнгу та аналіз процесів обліку рекламних кампаній.	06.12.2025 – 15.12.2025	Огляд предметної області та джерел
3	Аналіз існуючих програмних рішень для рекламної аналітики та порівняння їх функціональних можливостей.	16.12.2025 – 24.12.2025	Порівняльний аналіз аналогів
4	Формулювання мети, завдань, об’єкта, предмета дослідження та функціональних вимог до інформаційної системи.	25.12.2025 – 03.01.2026	Сформовані вимоги до системи
5	Моделювання предметної області, побудова DFD- та BPMN-діаграм бізнес-процесу аналізу рекламних кампаній.	04.01.2026 – 12.01.2026	DFD- та BPMN-діаграми
6	Проєктування логічної моделі даних інформаційної системи та побудова ER-діаграми бази даних.	13.01.2026 – 20.01.2026	ER-діаграма та схема БД
7	Проєктування UML-діаграм програмної системи: діаграми класів, компонентів, пакетів і кооперації.	21.01.2026 – 28.01.2026	Комплект UML-діаграм
8	Оформлення та редагування першого і другого розділів пояснювальної записки.	29.01.2026 – 05.02.2026	Текст розділів 1 та 2
9	Ініціалізація програмного проєкту Python, налаштування структури каталогів, середовища розробки та репозиторію.	06.02.2026 – 12.02.2026	Початкова кодова база
10	Проєктування та реалізація бази даних SQLite для зберігання даних про кампанії, креативи, щоденні метрики та KPI.	13.02.2026 – 19.02.2026	Структура таблиць SQLite
11	Розроблення модуля роботи з даними: додавання, редагування, видалення та збереження інформації про рекламні кампанії.	20.02.2026 – 27.02.2026	CRUD-операції для кампаній

12	Реалізація обліку щоденних маркетингових метрик: показів, кліків, витрат, лідів, конверсій і доходу.	28.02.2026 – 06.03.2026	Модуль введення метрик
13	Розроблення аналітичного модуля розрахунку KPI-показників CTR, CPC, CPA, CPL, ROAS, ROI та CVR.	07.03.2026 – 14.03.2026	Алгоритми обчислення KPI
14	Реалізація механізмів аналізу ефективності креативів, виявлення ризиків та формування рекомендацій щодо оптимізації кампаній.	15.03.2026 – 22.03.2026	Модуль аналітичних рекомендацій
15	Проєктування графічного інтерфейсу користувача інформаційної системи з використанням бібліотеки PyQt6.	23.03.2026 – 29.03.2026	Макети та базові вікна PyQt6
16	Реалізація головного дашборду системи для перегляду зведених показників рекламних кампаній.	30.03.2026 – 05.04.2026	Інтерфейс дашборду
17	Розроблення таблиць, фільтрів і форм перегляду статистики за кампаніями, креативами та періодами.	06.04.2026 – 12.04.2026	Інтерактивні таблиці та фільтри
18	Реалізація модулів візуалізації даних: побудова графіків, діаграм та аналітичних віджетів.	13.04.2026 – 19.04.2026	Графіки та діаграми системи
19	Розроблення функціоналу формування аналітичних звітів та експорту результатів аналізу.	20.04.2026 – 25.04.2026	Модуль звітності
20	Наповнення бази тестовими наборами даних, наближеними до реальних рекламних кампаній, для перевірки роботи системи.	26.04.2026 – 29.04.2026	Тестові дані для перевірки
21	Виконання функціонального тестування модулів введення, редагування, збереження та оброблення даних.	30.04.2026 – 03.05.2026	Таблиці функціонального тестування
22	Перевірка правильності розрахунку KPI-показників та відповідності аналітичних результатів очікуваним значенням.	04.05.2026 – 06.05.2026	Верифікація формул і метрик
23	Тестування інтерфейсних модулів, перевірка зручності взаємодії користувача з дашбордом, фільтрами та звітами.	07.05.2026 – 09.05.2026	Результати UI-тестування
24	Підготовка інсталяційного пакету, перевірка запуску програми та опис складу програмного забезпечення.	10.05.2026 – 12.05.2026	Готовий інсталяційний пакет
25	Оформлення третього та четвертого розділів пояснювальної записки з описом реалізації, тестування та результатів роботи системи.	13.05.2026 – 14.05.2026	Опис третього та четвертого розділів

26	Формування вступу, висновків, переліку умовних позначень, списку використаних джерел та додатків.	15.05.2026 – 16.05.2026	Фінальна структура записки
27	Перевірка тексту на академічну доброчесність, остаточне нормоконтрольне редагування та підготовка матеріалів до захисту.	17.05.2026 – 22.05.2026	Готова бакалаврська робота

Студент _____ **Богдан ЧЕХІРКІН**
(підпис)

Керівник бакалаврської кваліфікаційної роботи _____ **Таїсія САЯПІНА**
(підпис)

НАЦІОНАЛЬНИЙ УНІВЕРСТИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Декларація про академічну доброчесність та використання ШІ

ДЕКЛАРАЦІЯ ЗДОБУВАЧ

Я, Чехіркін Богдан Олексійович, здобувач НУБіП України, усвідомлюючи відповідальність за порушення академічної доброчесності, цією заявою підтверджую, що:

1. Моя бакалаврська кваліфікаційна робота на тему «Інформаційна система бізнес-аналітики для медіабаїнгу» є результатом моєї особистої праці.

2. Усі запозичення з текстів інших авторів мають відповідні посилання згідно з чинними стандартами.

3. Щодо використання інструментів ШІ зазначаю, що (обрати необхідне):

- [ ] інструменти генеративного ШІ не використовувалися.
- [ ] інструменти ШІ (вказати назву: ChatGPT, Gemini, Claude, DeepL, _____інші (вказати назву)) були використані для:
  - [ ] перекладу іншомовних джерел;
  - [ ] стилістичного редагування та перевірки граматики;
  - [ ] допомоги у написанні/відлагодженні програмного коду;
  - [ ] інше: _____.

Я розумію, що надання недостовірної інформації може призвести до скасування результатів захисту та відрахування з числа здобувачів НУБіП України.

«__» _____2026р. _____ Богдан ЧЕХІРКІН

(підпис) **(Ім'я та ПРІЗВИЩЕ)**