НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ
І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ
Факультет (ННІ) інформаційних технологій

ПОГОДЖЕНО	**ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ**
Декан факультету	**Завідувач кафедри**
Інформаційних технологій	**комп'ютерних наук**
/ Ігор Болбот /	/ Белла Голуб /
(підпис) (ім'я прізвище)	(підпис) (ім'я прізвище)
«___» _____ 2025р.	«___» _____ 2025р.

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему Програмне забезпечення аналітичної системи управління з використанням мультимедійних об'єктів

Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)

Освітня програма Програмне забезпечення інформаційних систем
(назва)

Орієнтація освітньої програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Гарант освітньої програми

доц.к.ф.-м.н. / Віктор Кириченко /
(науковий ступінь та вчене звання) (підпис) (ім'я прізвище)

Керівник магістерської кваліфікаційної роботи

старший викладач / Юрій Міловідов /
(науковий ступінь та вчене звання) (підпис) (ім'я прізвище)

Консультант магістерської кваліфікаційної роботи

к.т.н., доцент / Іван Пархоменко /
(науковий ступінь та вчене звання) (підпис) (ім'я прізвище)

Виконав / Владислав Павленко /
(підпис) (ім'я прізвище)

КИЇВ – 2025

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет (ННІ) інформаційних технологій

ЗАТВЕРДЖУЮ
Завідувач кафедри комп'ютерних наук
доцент, к.т.н. _____ Голуб Б. Л.
(науковий ступінь, вчене звання) (підпис) (ПІБ)
«10» листопада 2024 року

З А В Д А Н Н Я

ДО ВИКОНАННЯ МАГІСТЕРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ СТУДЕНТУ

Павленку Владиславу Руслановичу
(прізвище, ім'я, по батькові)

Спеціальність 121 «Інженерія програмного забезпечення»
(код і назва)

Освітня програма «Програмне забезпечення інформаційних систем»
(назва)

Орієнтація освітньої програми освітньо-професійна

Тема магістерської кваліфікаційної роботи: «Програмне забезпечення аналітичної системи управління з використанням мультимедійних об'єктів»
затверджена наказом ректора НУБіП України від «1» листопада 2024 р. № 1963 «С»

Термін подання завершеної роботи на кафедру 14 листопада 2025 р.

Вихідні дані до магістерської кваліфікаційної роботи: матеріали про ринок мультимедійного контенту та існуючі аналітичні платформи, UML-моделі, логічна модель БД, вимоги до системи, дані про структуру мультимедійних об'єктів та подій взаємодії, технологічні засоби: Java, Java Swing, і т.п., методи аналітики, асоціативні правила, практичні приклади реалізації мультимедійних платформ (Steam, Kindle, Netflix, Apple App Store).
Перелік питань, що підлягають дослідженню:

1. Аналіз проблем сучасних аналітичних платформ мультимедійного контенту.
2. Розробка концепції універсальної мультимедійної платформи з інтеграцією різного мультимедійного контенту в єдиному середовищі.
3. Розроблення UML-моделей і логічної структури бази даних.
3. Створення аналітичної системи для моніторингу використання платформи та розробка механізмів персоналізації контенту на її основі.
4. Створення системи з реалізацією алгоритмів аналізу використання контенту та інтерфейсом Java Swing для виконання ключових аналітичних функцій.
5. Тестування системи та оцінка ефективності роботи.

Перелік графічного матеріалу (за потреби)
Дата видачі завдання 7 листопада 2024 р.

Керівник магістерської кваліфікаційної роботи _____ Міловідов Ю. О.
(підпис) (прізвище та ініціали)

Консультант магістерської кваліфікаційної роботи _____ Пархоменко І. І.
(підпис) (прізвище та ініціали)

Завдання прийняв до виконання _____ Павленко В. Р.
(підпис) (прізвище та ініціали студента)

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

1. A/B – A/B testing - метод експериментального порівняння двох або більше версій контенту для визначення ефективнішого варіанта на основі користувацької взаємодії.
2. API (Application Programming Interface) - інтерфейс прикладного програмування, що забезпечує взаємодію між програмними модулями системи.
3. App Server - сервер прикладної логіки, який обробляє запити клієнтських застосунків і забезпечує виконання бізнес-процесів системи.
4. BI (Business Intelligence) - технології та методи аналітичної обробки даних, що забезпечують прийняття управлінських рішень на основі фактів і метрик.
5. CDN (Content Delivery Network) - мережа доставки контенту, яка оптимізує передачу мультимедійних об’єктів до користувачів.
6. CSV (Comma-Separated Values) - текстовий формат обміну даними, що використовується для експорту звітів і аналітичних вибірок.
7. DAO (Data Access Object) - шаблон програмування для відокремлення логіки доступу до бази даних від бізнес-логіки системи.
8. ETL (Extract, Transform, Load) - процес вилучення, перетворення та завантаження даних у сховище для подальшого аналітичного оброблення.
9. FP-Growth - алгоритм виявлення частих наборів елементів, який використовується для побудови рекомендаційних правил у системі.
10. HTTP(S) (HyperText Transfer Protocol Secure) - протокол безпечної передачі даних у мережі між клієнтом і сервером.
11. IDP (Identity Provider) - постачальник автентифікації, який здійснює перевірку облікових даних користувача (Keycloak, OAuth2).
12. IoC (Inversion of Control) - принцип інверсії керування, який використовується для підвищення модульності програмної системи.

13. JSON (JavaScript Object Notation) - формат обміну структурованими даними, який застосовується при передачі запитів і відповідей REST API.

14. JWT (JSON Web Token) - токен безпечної передачі автентифікаційних даних між клієнтом і сервером.

15. KPI (Key Performance Indicator) - ключовий показник ефективності, що використовується для оцінювання стану мультимедійних процесів.

16. MVC (Model–View–Controller) - архітектурний шаблон, який розділяє бізнес-логіку, інтерфейс користувача та керування подіями.

17. OLAP (Online Analytical Processing) - технологія багатовимірного аналізу даних у вигляді кубів і зрізів.

18. OIDC (OpenID Connect) - протокол ідентифікації, побудований поверх OAuth2, який використовується для єдиного входу (SSO).

19. ORM (Object-Relational Mapping) - технологія відображення об'єктів програми на таблиці бази даних.

20. RBAC (Role-Based Access Control) - модель керування доступом на основі ролей користувачів.

21. REST (Representational State Transfer) - архітектурний стиль взаємодії клієнта і сервера через стандартизовані HTTP-запити.

22. S3 (Simple Storage Service) - хмарне сховище для зберігання звітів і результатів аналітичних обчислень.

23. SLA (Service Level Agreement) - угода про рівень надання послуг, що визначає допустимі межі часу відповіді й доступності системи.

24. SQL (Structured Query Language) - мова запитів до реляційних баз даних, використовується для керування структурою та змістом БД.

25. SSO (Single Sign-On) - механізм єдиного входу до кількох систем за допомогою одного автентифікаційного процесу.

26. TLS (Transport Layer Security) - криптографічний протокол, який забезпечує безпечну передачу даних у мережі.

27. UI (User Interface) - інтерфейс користувача, за допомогою якого здійснюється взаємодія з системою.

28. UML (Unified Modeling Language) - уніфікована мова моделювання, що використовується для опису структури та поведінки програмних систем.

29. XML (Extensible Markup Language) - формат представлення структурованих даних, який використовується для інтеграції з зовнішніми сервісами.

30. Zip-пакет - архівований інсталяційний дистрибутив, який містить виконувані файли, конфігурації, скрипти встановлення та документацію системи.

ВСТУП

Сучасна індустрія мультимедійного контенту характеризується значними обсягами транзакційних і поведінкових даних, що формуються внаслідок взаємодії користувачів із цифровими продуктами різних типів і жанрів. Такі дані є різнорідними, високовимірними та динамічними, що ускладнює їх аналітичну обробку традиційними статистичними методами. Ефективне управління мультимедійними об’єктами потребує застосування спеціалізованих систем, здатних виявляти закономірності у структурі споживання контенту, аналізувати взаємозв’язки між категоріями, жанрами й користувацькими діями та формувати прогнозні рекомендації.

Створення аналітичної системи управління використанням мультимедійних об’єктів є актуальним напрямом, оскільки така система дозволяє інтегрувати методи статистичного й асоціативного аналізу для виявлення латентних зв’язків у поведінкових даних. Це забезпечує можливість не лише оцінювати поточний попит і популярність контенту, а й формувати рекомендації для оптимізації контентної політики, що підвищує ефективність прийняття управлінських рішень у сфері цифрової дистрибуції.

Мета роботи полягає у розробленні програмного забезпечення аналітичної системи, що забезпечує збір, обробку, зберігання й візуалізацію даних про використання мультимедійних об’єктів, а також виконує аналітичний і рекомендаційний аналіз на основі статистичних та асоціативних методів.

Для досягнення поставленої мети необхідно розв’язати такі **завдання:**

– провести аналіз предметної області та існуючих рішень у сфері управління мультимедійним контентом;

– визначити вимоги до функціональності, архітектури та бази даних системи;

– спроєктувати структуру даних для зберігання інформації про користувачів, мультимедійні об’єкти та транзакції;

– розробити аналітичні модулі для статистичного аналізу та виявлення асоціативних закономірностей між об’єктами;

– реалізувати графічний інтерфейс користувача на базі Java Swing із підтримкою побудови аналітичних графіків і звітів;

– протестувати працездатність програмного забезпечення та оцінити ефективність реалізованих алгоритмів.

Об’єкт дослідження – процес аналітичної обробки даних про використання мультимедійного контенту в інформаційних системах управління.

Предмет дослідження – методи, моделі та засоби програмної реалізації аналітичної системи управління мультимедійними об’єктами на основі статистичного та асоціативного аналізу даних.

У процесі дослідження використано методи системного аналізу, статистичного моделювання, об’єктно-орієнтованого програмування, реляційного проєктування баз даних, а також алгоритми асоціативного аналізу (Apriori, FP-Growth) для виявлення закономірностей у даних про споживання контенту.

Наукова новизна роботи полягає у створенні інтегрованої аналітичної системи, яка поєднує методи статистичного аналізу з алгоритмами рекомендаційного типу для управління мультимедійними об’єктами. Розроблене програмне забезпечення дозволяє не лише оцінювати популярність жанрів і типів контенту, а й формувати прогнозні рекомендації на основі виявлених закономірностей спільного використання. Це забезпечує підвищення ефективності процесів прийняття рішень у сфері мультимедійного бізнесу та сприяє оптимізації контентної політики постачальників цифрових продуктів.

1 СИСТЕМНИЙ АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Опис предметної області

Предметна область аналітичної системи управління використанням мультимедійних об’єктів охоплює процеси збору, збереження, аналізу та інтерпретації даних, що відображають споживання цифрового контенту - фільмів, ігор, музики, навчальних матеріалів та інших медіа-продуктів. Головна мета полягає у створенні єдиної платформи для аналітичної обробки інформації про попит, рейтинги, покупки й активність користувачів, що дозволяє формувати рекомендації, оптимізувати контентні стратегії та підтримувати управлінські рішення.

У сучасному цифровому середовищі мультимедійний контент генерується та споживається у великих обсягах, а динаміка користувацьких уподобань змінюється практично в режимі реального часу. Традиційні методи статистичного аналізу не забезпечують необхідної гнучкості для оброблення таких даних, тому актуальним є застосування аналітичних систем, що інтегрують засоби OLAP, Data Mining і рекомендаційні алгоритми. Запропонована система дає змогу автоматизувати процес збору даних, виявляти приховані закономірності у споживанні контенту, оцінювати ефективність кампаній та прогнозувати тенденції.

Основними суб’єктами предметної області є контент-менеджери, які здійснюють управління мультимедійними об’єктами; аналітики, що аналізують поведінкові патерни користувачів і формують звіти; та адміністратори, відповідальні за налаштування, безпеку та інтеграцію системи з зовнішніми сервісами. До зовнішніх джерел інформації належать CDN-платформи, маркетплейси, рекламні системи, сервіси авторизації (IdP/SSO) та системи виявлення піратського контенту, що забезпечують надходження актуальних даних про публікації, продажі, перегляди, кліки й аудиторні показники.

На рисунку 1.1 наведено узагальнену структурну схему предметної області аналітичної системи. Вона демонструє ключові рівні взаємодії: рівень представлення, сервісний рівень і рівень даних. Між ними відбувається обмін інформацією через уніфіковані інтерфейси, що забезпечує цілісність аналітичного процесу.

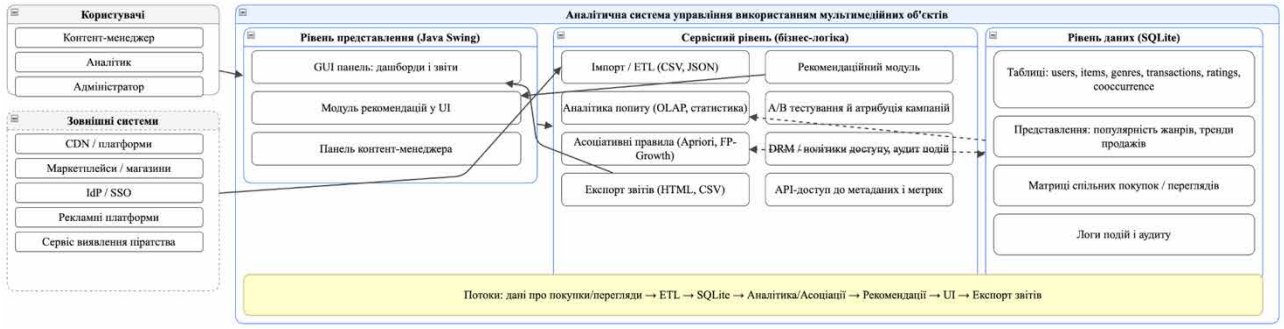

Рисунок 1.1 – Структура предметної області аналітичної системи управління використанням мультимедійних об’єктів

На рівні даних формується база у форматі SQLite, яка містить таблиці користувачів, об’єктів контенту, транзакцій, оцінок та асоціацій. Вона підтримує нормалізовану структуру, що усуває надлишковість і забезпечує швидкий доступ до записів. Сервісний рівень реалізує бізнес-логіку - модулі імпорту даних (ETL), аналітики попиту, побудови асоціативних правил, атрибуції кампаній і рекомендацій. Рівень представлення побудований на технології Java Swing і забезпечує відображення дашбордів, звітів і результатів аналітики у зручному графічному вигляді.

Ключові інформаційні сутності системи наведено в таблиці 1.1. Їх взаємозв’язки визначають логічну модель предметної області, що слугує основою для формування структур бази даних і побудови аналітичних алгоритмів.

Таблиця 1.1 – Основні сутності предметної області аналітичної системи

Сутність	**Основні атрибути**	**Функціональне призначення**
Користувач	ID, роль, історія дій, регіон	Ідентифікація учасників системи, фіксація активності
Об’єкт	ID, назва, тип, жанр, рейтинг	Опис мультимедійного продукту

Продовження таблиці 1.1

Транзакція	Користувач, об’єкт, час, дія	Реєстрація факту покупки або перегляду
Рейтинг	Користувач, об’єкт, оцінка	Збереження суб’єктивної оцінки користувача
Асоціація	Об’єкт 1, об’єкт 2, частота спільного вибору	Виявлення закономірностей спільного споживання контенту

Предметна область системи описує комплексний процес управління використанням мультимедійних ресурсів на основі даних. Інтеграція джерел, нормалізована структура зберігання та наявність аналітичних і рекомендаційних механізмів створюють основу для побудови сучасної інтелектуальної системи підтримки прийняття рішень у сфері цифрового контенту.

1.2 Огляд інформаціних джерел та існуючих рішень

Сучасні платформи управління мультимедійним контентом активно впроваджують аналітичні інструменти для моніторингу попиту, виявлення трендів і підтримки процесів рекомендацій. Їх основна мета полягає у зборі даних про поведінку користувачів, сегментації аудиторії та прогнозуванні споживання контенту. Проте більшість із них орієнтована на власні екосистеми й не передбачає відкритої інтеграції або розширюваної аналітики на сторонніх джерелах даних.

Одним із найпоширеніших прикладів є YouTube Analytics (рисунок 1.2), який надає авторам дані про кількість переглядів, час перегляду, демографічний профіль аудиторії та динаміку підписників. Система відображає ключові метрики взаємодії з контентом у вигляді графіків і таблиць, проте її алгоритми не дозволяють проводити глибокий кластерний або асоціативний аналіз для виявлення поведінкових патернів.

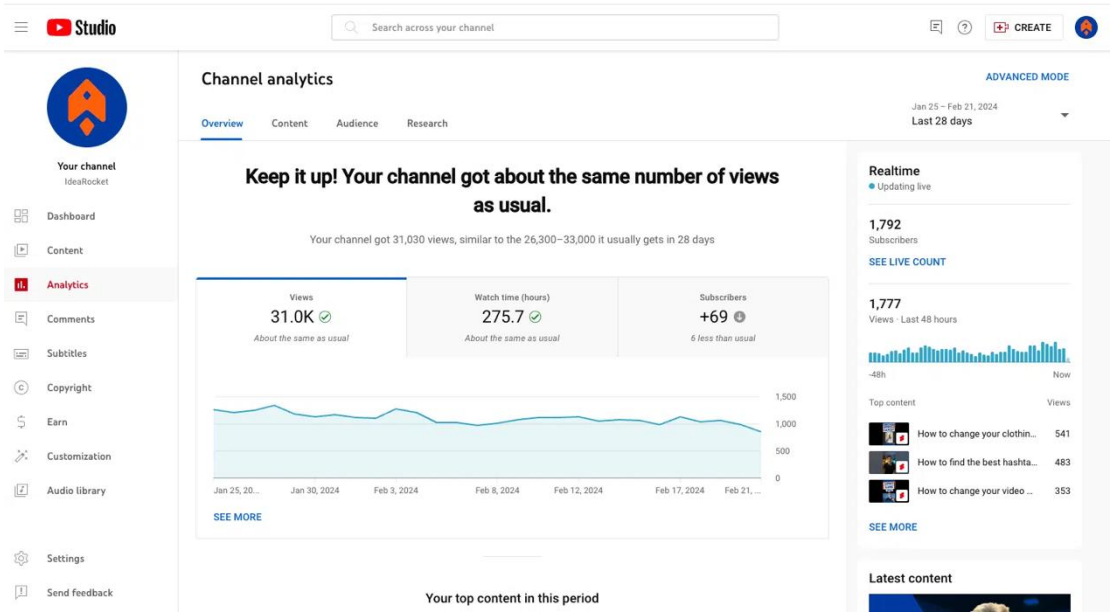

Рисунок 1.2 – Інтерфейс аналітичної панелі YouTube Studio (модуль переглядів і підписників)

Іншим прикладом є Netflix Insights (рисунок 1.3), який застосовує розподілену аналітику даних для дослідження популярності фільмів і серіалів за жанрами, країнами та рейтингами. Ця система поєднує OLAP-запити з візуальними дашбордами, що дає змогу швидко оцінювати динаміку переглядів. Однак архітектура Netflix Insights є закритою, а алгоритми рекомендацій базуються на внутрішніх моделях машинного навчання без можливості адаптації під зовнішні бізнес-задачі.

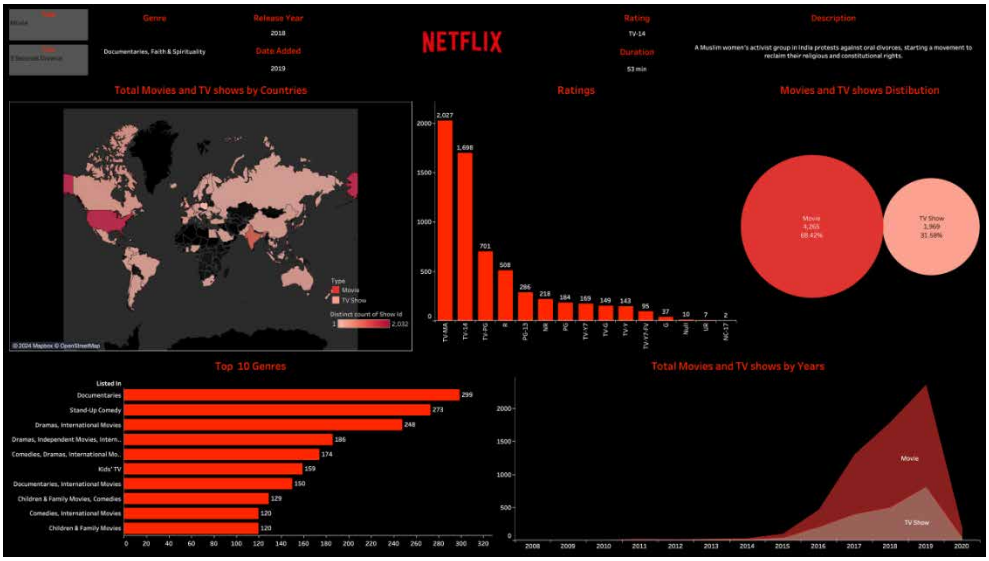

Рисунок 1.3 – Дашборд аналітичної системи Netflix Insights із розподілом за країнами та жанрами

Сервіс Spotify for Artists (рисунок 1.4) забезпечує аналітику потокових прослуховувань, збережень у плейлистах і залучення слухачів. Він дозволяє відстежувати популярність треків, джерела трафіку та географію користувачів. Проте модель Spotify також обмежена корпоративною екосистемою та не підтримує зворотного зв'язку для динамічного коригування рекомендацій.

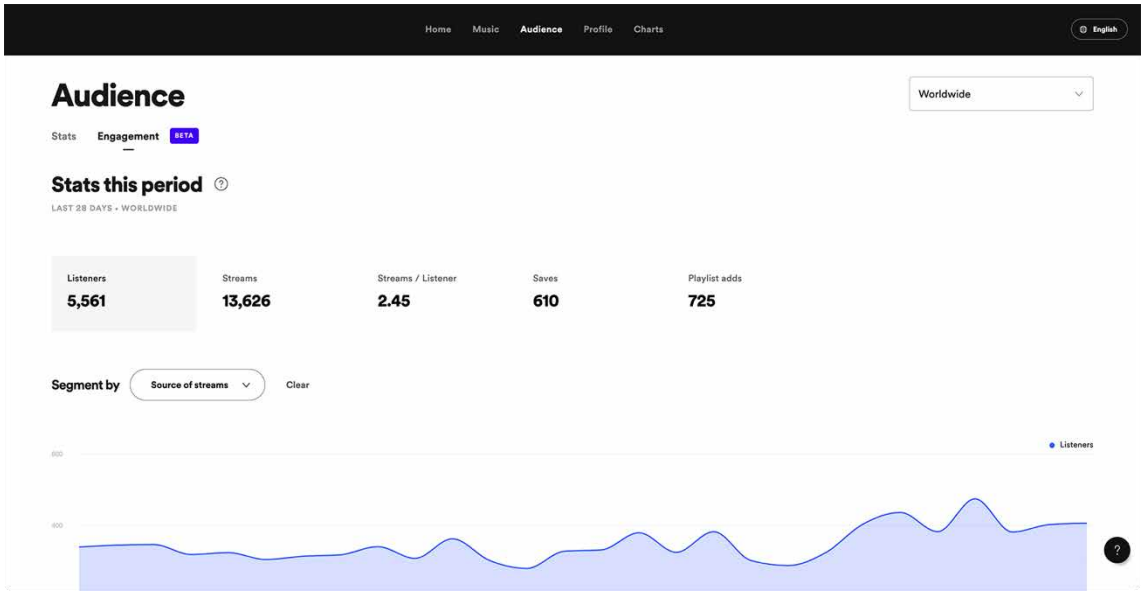

Рисунок 1.4 – Приклад аналітичної панелі Spotify for Artists для моніторингу аудиторії

Аналітична платформа Steamworks Analytics (рисунок 1.5) надає розробникам ігор інструменти для відстеження кількості покупок, побажань користувачів (wishlist) і рівня активацій. Вона орієнтована на вимірювання ефективності маркетингових кампаній і конверсії, проте не включає вбудованих рекомендаційних механізмів або глибокого аналізу уподобань гравців.

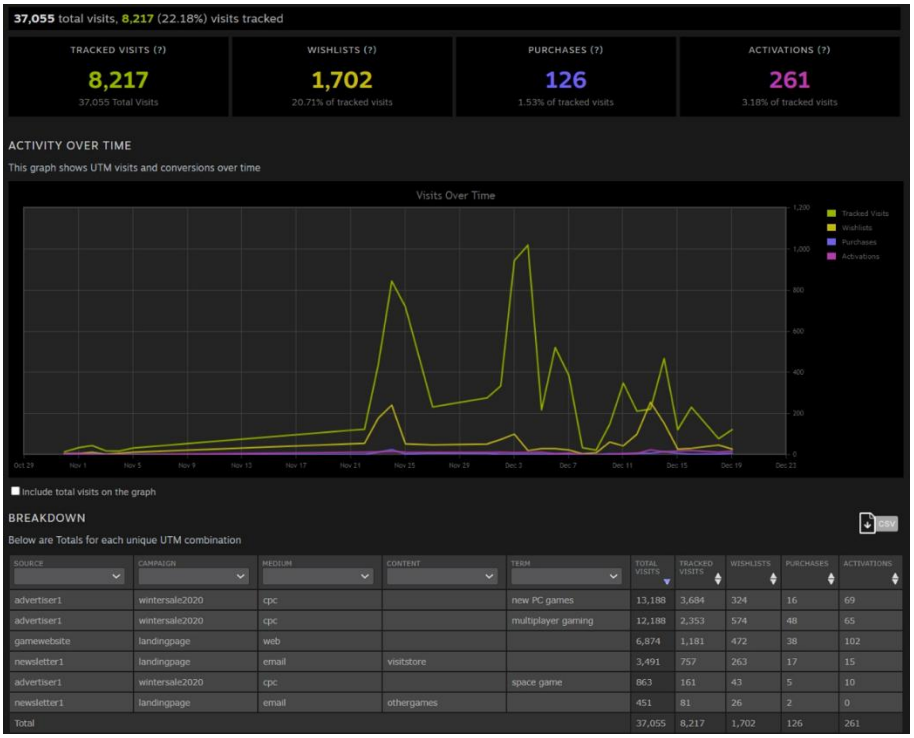

Рисунок 1.5 – Аналітична система Steamworks Analytics з візуалізацією UTM-трекінгу кампаній

Система Amazon Prime Video Metrics (рисунок 1.6) демонструє можливості комплексного аналізу контенту — від рейтингів і тривалості переглядів до динаміки жанрів. Завдяки широким можливостям BI-аналітики, платформа надає узагальнені висновки щодо успішності проєктів, однак не дозволяє розробляти індивідуальні моделі оцінки або рекомендацій на основі локальних параметрів.

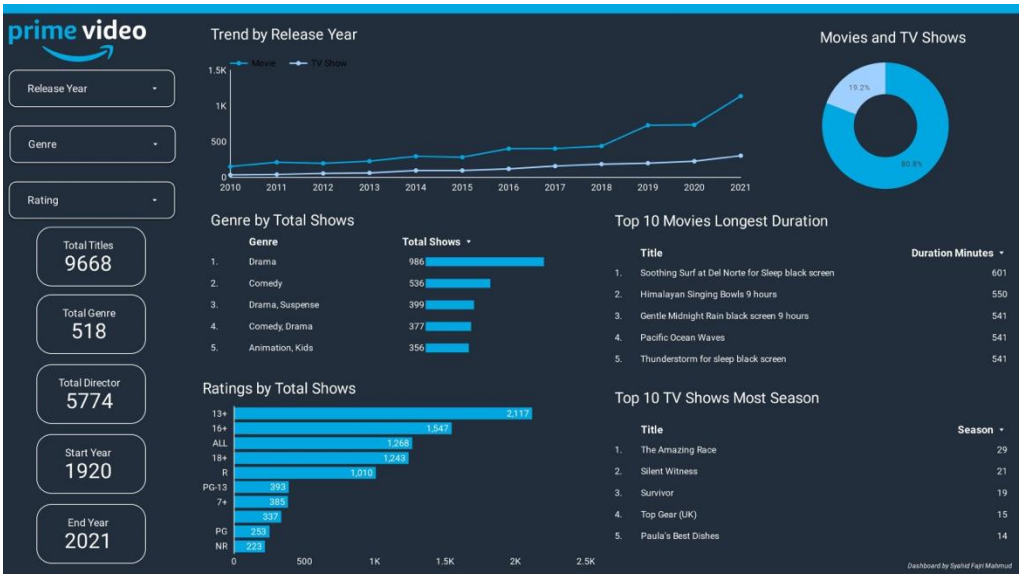

Рисунок 1.6 – Приклад дашборду Amazon Prime Video Metrics із трендами та жанровою аналітикою

На основі проведеного аналізу сформовано порівняльну таблицю (таблиця 1.2), у якій узагальнено ключові характеристики розглянутих систем і визначено переваги запропонованої аналітичної системи управління використанням мультимедійних об’єктів.

Таблиця 1.2 – Порівняльна характеристика існуючих аналітичних систем

Система	**Тип контенту**	**Основні аналітичні можливості**	**Рекомендації**	**Відкритість інтеграції**	**Особливості реалізації**
YouTube Analytics	Відео	Перегляди, підписники, активність	Ні	Низька	Візуальна панель без розширень
Netflix Insights	Відео	Жанровий і територіальний аналіз	Так	Низька	Закрита ML-архітектура
Spotify for Artists	Аудіо	Джерела потоків, збереження	Частково	Низька	Орієнтація на авторів
Steamworks Analytics	Ігри	Продажі, wishlist, активації	Ні	Середня	UTM-аналітика кампаній
Amazon Prime Video Metrics	Відео	Рейтинги, тривалість, жанри	Частково	Середня	BI-аналітика контенту
Запропонована система	Відео, ігри	OLAP-аналіз, асоціативні правила, A/B тестування	Так	Висока	Java Swing + SQLite, модульна архітектура

Результати порівняння показують, що наявні рішення мають обмежену гнучкість і переважно орієнтовані на внутрішні дані компаній. Запропонована система, реалізована на базі Java Swing і SQLite, забезпечує незалежну аналітичну платформу, що об’єднує різні джерела даних та надає глибший рівень дослідження користувацьких уподобань. У ній передбачено власну систему оцінювання контенту, яка враховує як кількісні (перегляди, покупки), так і якісні показники (оцінки, повторні звернення, середню тривалість взаємодії). На основі цих параметрів формується рекомендаційна підсистема, що використовує

асоціативні правила типу Apriori і кореляційні коефіцієнти для виявлення закономірностей спільного споживання медіа-продуктів.

Аналітична система усуває обмеження існуючих платформ, забезпечуючи універсальність інтеграції, адаптивну аналітику та гнучкий механізм формування рекомендацій, що підвищує ефективність управління мультимедійним контентом.

1.3 Моделювання предметної області

Моделювання предметної області є необхідним етапом проєктування аналітичної системи управління використанням мультимедійних об’єктів, оскільки воно забезпечує систематизацію процесів, визначення ключових учасників, потоків даних і взаємодії між підсистемами. На основі аналізу функціональних вимог побудовано комплекс UML-діаграм, що формують логічну основу майбутньої архітектури програмного забезпечення. Такий підхід дозволяє узгодити бізнес-рівень завдань із технічною реалізацією та забезпечити масштабованість і надійність системи.

На діаграмі прецедентів представленій на рисунку 1.7 представлено головних акторів системи - контент-менеджера, аналітика, правовласника, адміністратора, а також зовнішні служби: CDN-платформи, IdP-провайдери, рекламних партнерів і сервіси виявлення піратства. Взаємодія акторів з аналітичною системою охоплює завантаження та публікацію контенту, керування правами, моніторинг переглядів, формування аналітики, рекомендацій та звітів. Ця модель формалізує ключові функціональні сценарії та демонструє межі відповідальності між користувачами й підсистемами.

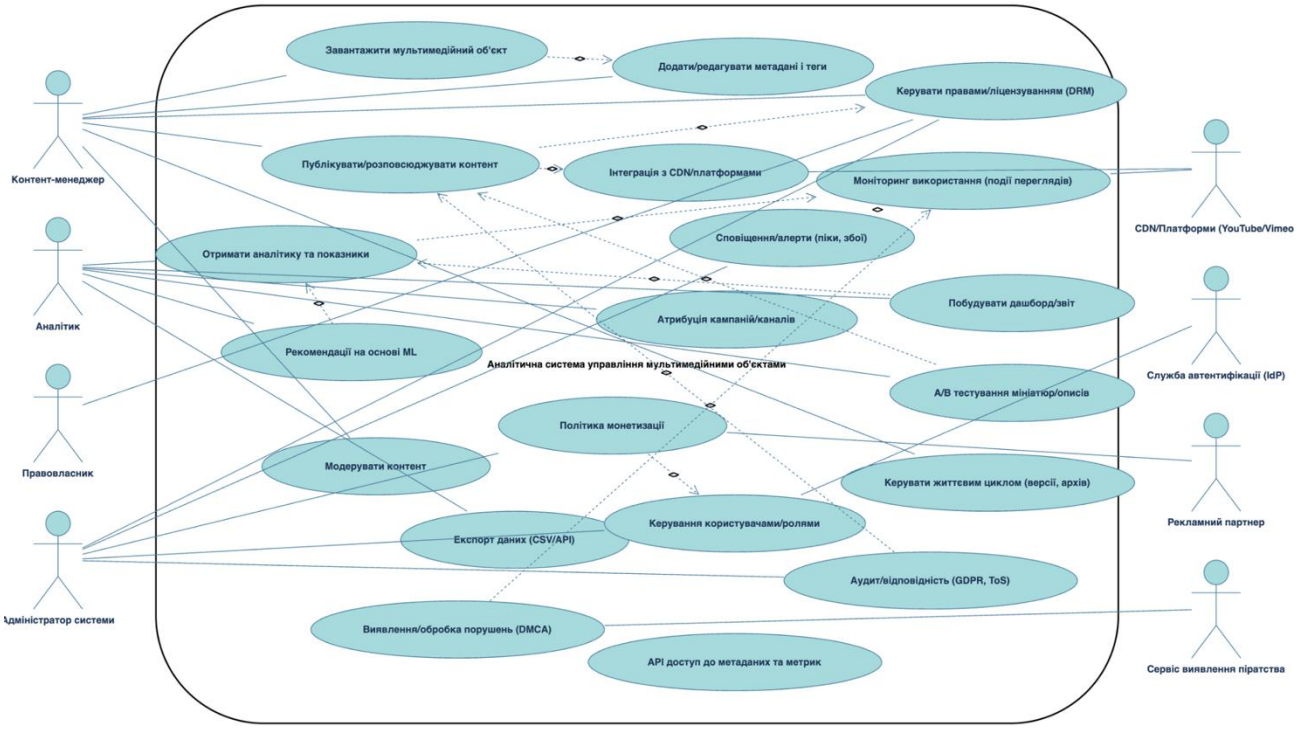

Рисунок 1.7 – Діаграма прецедентів аналітичної системи управління мультимедійними об’єктами

Діаграма послідовності (рисунок 1.8) відображає типовий процес доступу користувача до аналітичних даних. Модель описує етапи автентифікації через службу OIDC/SSO, перевірку прав доступу, отримання метаданих мультимедійного об’єкта, виконання запиту до аналітичного рушія, а також формування звіту для відображення в аналітичному клієнті. Послідовність взаємодій між компонентами - клієнтським додатком, базою даних, сервісом політик, сховищем медіа та модулем BI - забезпечує узгодженість процесів і цілісність інформаційних потоків.

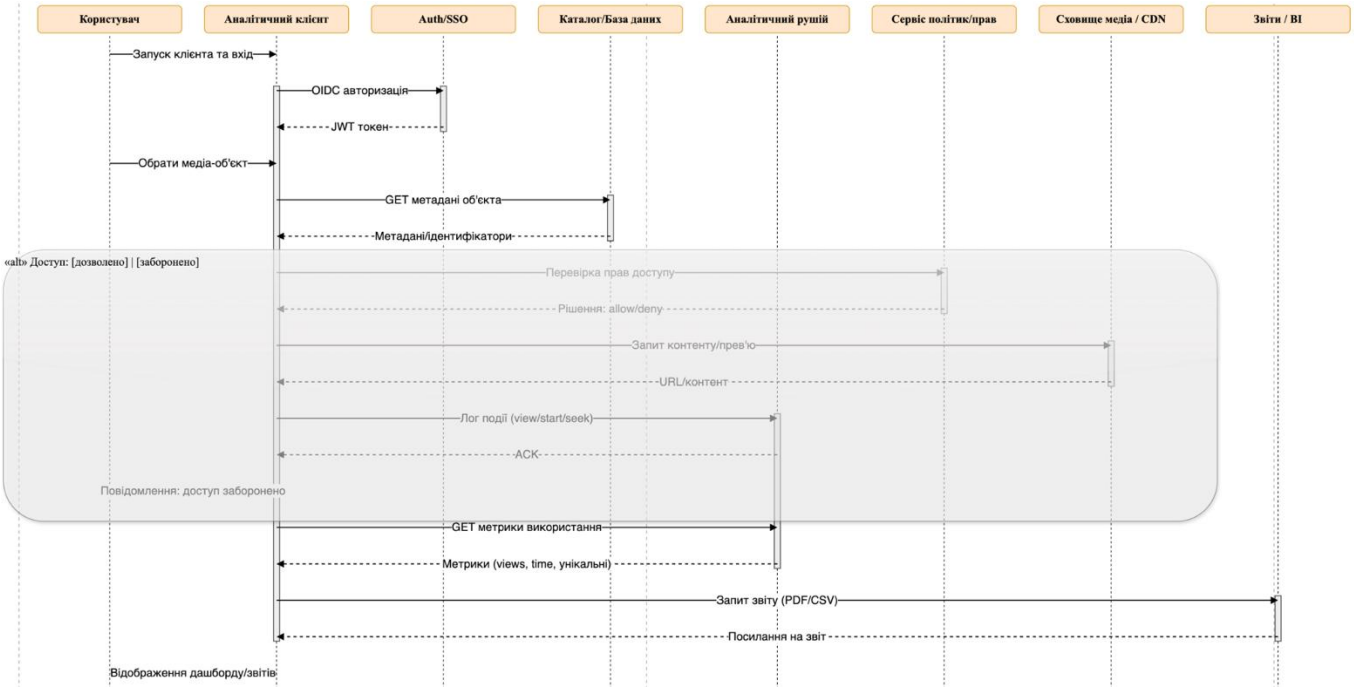

Рисунок 1.8 – Діаграма послідовності процесу авторизації, перевірки прав і формування аналітичного звіту

На діаграмі активності представленій на рисунку 1.9 подано логіку оброблення запитів у системі - від моменту отримання користувацького запиту до генерації звіту. Діаграма відображає основні гілки логіки, що включають перевірку наявності об’єкта, перевірку прав доступу, передачу посилань на CDN-ресурси, реєстрацію подій типу view/start/seek та створення звіту у форматах PDF або CSV. Така модель дозволяє оцінити цілісність бізнес-процесу, забезпечити контроль виняткових ситуацій (404, deny) і спроєктувати механізми безперервного моніторингу QoS.

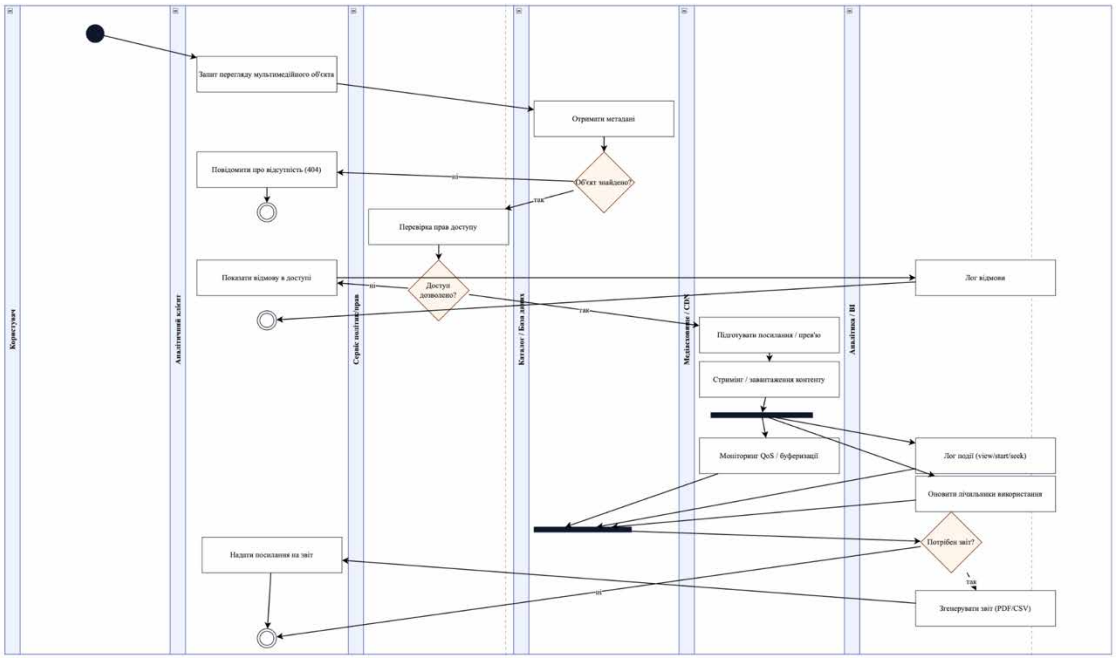

Рисунок 1.9 – Діаграма активності процесу запиту, перегляду та аналітичного оброблення мультимедійного об’єкта

Побудовані моделі в сукупності описують логічну архітектуру системи з точки зору взаємодії користувачів і програмних модулів. Моделювання дало змогу:

– формалізувати основні функціональні процеси системи (завантаження, моніторинг, аналітика, рекомендації, звітність);

– визначити структуру інформаційних потоків між клієнтським, аналітичним і зовнішнім рівнями;

– описати ключові механізми перевірки доступу, збору метрик і генерації результатів;

– закласти основу для подальшого проєктування бази даних, інтерфейсів і програмних модулів.

Виконане моделювання предметної області забезпечує цілісне уявлення про функціонування аналітичної системи управління мультимедійними об’єктами, що є фундаментом для побудови архітектури програмного забезпечення, розроблення системи оцінювання контенту та створення рекомендаційних моделей на основі зібраних параметрів і поведінкових закономірностей користувачів.

1.4 Аналіз вимог програмної системи

Аналіз вимог визначає функціональні, нефункціональні та безпекові аспекти майбутньої аналітичної системи управління використанням мультимедійних об’єктів. Метою даного етапу є формування узгодженої моделі очікуваних можливостей системи, що забезпечить її працездатність, масштабованість і відповідність сучасним стандартам інформаційної безпеки. Для систем аналітичного класу, які працюють із великими масивами даних, важливо встановити не лише набір функцій, але й вимоги до ефективності, стабільності та захисту інформації.

Функціональні вимоги описують що система має робити - тобто перелік її основних можливостей, сценаріїв використання та дій користувачів. Вони визначають функціональну архітектуру системи й орієнтовані на реалізацію бізнес-логіки. Основні функціональні вимоги подано в таблиці 1.3.

Таблиця 1.3 – Функціональні вимоги до аналітичної системи управління використанням мультимедійних об’єктів

№	Вимога	Опис функціональності	Очікуваний результат
1	Завантаження мультимедійних об’єктів	Додавання та обробка файлів різних типів (відео, аудіо, зображення)	Збереження об’єкта з повними метаданими у базі даних
2	Керування правами та ліцензіями	Реалізація DRM-контролю доступу, управління ліцензіями, дозволами та обмеженнями	Доступ користувачів відповідно до політик безпеки
3	Аналітика використання контенту	Збір логів подій (перегляди, відтворення, паузи), обробка даних і побудова звітів	Візуалізація показників популярності, часу перегляду
4	Рекомендаційний модуль	Формування рекомендацій на основі машинного навчання та поведінкових закономірностей	Персоналізовані рекомендації для користувачів
5	Моніторинг системи	Відстеження стану підсистем, навантаження, помилок і сповіщень	Своєчасне виявлення збоїв і аномалій

Продовження таблиці 1.3

6	Експорт даних	Генерація звітів у форматах CSV, PDF, JSON	Можливість обміну аналітичними даними
7	Інтеграція з CDN та зовнішніми сервісами	Обмін метаданими та логами з зовнішніми платформами (YouTube, Vimeo тощо)	Синхронізація даних про використання контенту

Нефункціональні вимоги визначають як система має працювати - тобто встановлюють обмеження, критерії якості та експлуатаційні характеристики. Вони описують продуктивність, масштабованість, зручність, надійність і переносимість системи. Сукупність цих характеристик забезпечує стабільну роботу навіть при значному навантаженні. Перелік нефункціональних вимог наведено в таблиці 1.4.

Таблиця 1.4 – Нефункціональні вимоги до системи

№	**Категорія**	**Вимога**	**Показник або критерій**
1	Продуктивність	Середній час обробки запиту аналітики	≤ 300 мс
2	Масштабованість	Обробка великої кількості подій та користувачів без втрати продуктивності	≥ 10^6 подій у базі даних
3	Надійність	Гарантована безперервна робота системи	Uptime ≥ 99,5 %
4	Зручність інтерфейсу	Графічний інтерфейс на Java Swing із інтуїтивною навігацією	≤ 3 дії до основної функції
5	Портативність	Підтримка платформ Windows, Linux та macOS без зміни коду	100 % сумісність JVM
6	Розширюваність	Легка інтеграція нових модулів і API-компонентів	Відкритий REST/GraphQL інтерфейс
7	Логування та відновлення	Можливість трасування подій і відновлення після збою	Автоматичний аудит та резервне копіювання

Третю групу складають вимоги до безпеки, що регламентують захист даних, користувачів і зовнішніх інтеграцій від несанкціонованого доступу, втрати чи модифікації інформації. Вони охоплюють механізми ідентифікації,

контролю доступу, шифрування, аудитів і кіберзахисту API. Основні безпекові вимоги подано в таблиці 1.5.

Таблиця 1.5 – Вимоги до безпеки системи

№	Вимога	Опис	Механізм реалізації
1	Автентифікація та авторизація	Забезпечення входу через централізований IdP із OIDC/SSO	OAuth 2.0, JWT токени
2	Контроль доступу	Розмежування ролей користувачів і доступу до модулів	RBAC/ABAC моделі
3	Шифрування даних	Захист інформації при передачі та зберіганні	TLS 1.3, AES-256
4	Аудит і журналювання	Збереження історії дій користувачів і системних подій	Secure Log Manager + SQLite Audit Trail
5	Захист API від атак	Перевірка запитів, ліміти швидкості, фільтрація вхідних даних	HTTPS, rate-limit, input validation
6	Відновлення після інцидентів	Гарантоване відновлення працездатності після збою або атаки	Автоматичні backups та failover-механізми

Узагальнюючи проведений аналіз, можна зазначити, що визначені вимоги утворюють основу архітектури майбутньої системи. Функціональні параметри забезпечують виконання основних бізнес-процесів - від завантаження контенту до формування аналітичних звітів; нефункціональні - встановлюють рівень якості та стабільності роботи; а безпекові - гарантують цілісність і захист даних. Така структура вимог дозволить створити комплексне, надійне рішення з аналітичними та рекомендаційними можливостями, здатне адаптуватися до динамічного мультимедійного середовища.

1.5 Постановка завдання

На основі проведеного аналізу предметної області, існуючих рішень і вимог до системи сформульовано постановку завдання розроблення аналітичної системи управління використанням мультимедійних об’єктів. Метою є

створення програмного комплексу, який забезпечить автоматизований збір, обробку та аналіз даних про використання цифрового контенту з можливістю формування рекомендацій і аналітичних звітів.

Система має інтегруватися з базами даних і зовнішніми сервісами, надавати зручний графічний інтерфейс користувача (Java Swing) та використовувати SQLite як основне сховище даних. Архітектура має забезпечувати модульність, масштабованість і розширюваність для подальшої інтеграції нових джерел контенту, аналітичних модулів і форматів звітності.

Вхідні дані системи:

– інформація про мультимедійні об’єкти (тип, жанр, формат, джерело, тривалість, рейтинг);

– дані про користувачів (роль, ідентифікатор, історія дій, регіон, тип пристрою);

– логи подій і транзакцій (перегляди, покупки, кліки, переходи);

– дані з зовнішніх джерел (CDN, рекламні системи, сервіси ідентифікації, платформи продажу);

– історичні показники популярності та залучення контенту;

– параметри прав доступу та ліцензійних обмежень.

Вихідні дані системи:

– аналітичні звіти про активність користувачів, динаміку споживання контенту, рейтинги й жанрову структуру;

– рекомендації щодо оптимізації розміщення та промоції контенту;

– результати аналітики у вигляді таблиць, графіків і діаграм;

– експорти у форматах CSV, PDF або JSON для подальшої обробки;

– повідомлення про аномалії, порушення прав доступу чи помилки завантаження;

– агреговані показники для підтримки управлінських рішень (популярність, середня оцінка, коефіцієнт повторного перегляду).

Основні завдання розробки:

1. розробити логічну та фізичну структуру бази даних для зберігання мультимедійних об’єктів, користувачів і подій.

2. Реалізувати модулі збору, оброблення та нормалізації даних про взаємодію користувачів із контентом.

3. Побудувати аналітичні алгоритми для виявлення закономірностей, обчислення метрик популярності й формування рекомендацій.

4. Забезпечити генерацію звітів і візуалізацію результатів у вигляді інтерактивних панелей і графіків.

5. Реалізувати систему доступу з розмежуванням ролей (адміністратор, аналітик, контент-менеджер).

6. Забезпечити захист даних і надійність зберігання інформації, включаючи аудит і резервне копіювання.

7. Реалізувати можливість інтеграції з зовнішніми джерелами для автоматичного оновлення аналітичних показників.

У результаті реалізації зазначених завдань буде створено інтелектуальну аналітичну систему, яка дозволить автоматизувати процеси управління мультимедійними об’єктами, проводити комплексний аналіз використання контенту, формувати звіти й рекомендації для підвищення ефективності контентної політики. Це забезпечить підґрунтя для прийняття обґрунтованих управлінських рішень та прогнозування динаміки попиту на цифровий контент.

1.6 Висновки до першого розділу

У першому розділі проведено комплексний аналіз предметної області аналітичних систем управління використанням мультимедійних об’єктів. Розглянуто сучасний стан розвитку ринку цифрового контенту, визначено ключові проблеми, пов’язані з обробленням великих обсягів даних, динамікою користувацьких уподобань і потребою в автоматизації процесів аналізу та прогнозування.

Проведено дослідження існуючих програмних рішень - YouTube Analytics, Netflix Insights, Spotify for Artists, Steamworks Analytics, Amazon Prime Video Metrics - і встановлено, що більшість із них орієнтовані на закриті екосистеми, не забезпечують універсальної інтеграції та не мають модульної архітектури для незалежного використання. У результаті сформульовано доцільність розроблення власної аналітичної системи з відкритою архітектурою, здатної здійснювати гнучку обробку та візуалізацію даних, а також формувати рекомендації на основі поведінкових моделей користувачів.

На основі побудованих UML-діаграм змодельовано предметну область майбутньої системи, визначено її ключові актори, сценарії, потоки даних і функціональні зв'язки між компонентами. Це дозволило сформувати концептуальну структуру системи, яка включає рівні даних, бізнес-логіки та представлення, а також передбачає взаємодію із зовнішніми сервісами (CDN, рекламні та ідентифікаційні платформи).

Проведено аналіз функціональних, нефункціональних і безпекових вимог, на основі якого сформовано постановку завдання розроблення системи. Визначено вхідні й вихідні дані, структуру процесів оброблення інформації, критерії ефективності та вимоги до архітектури програмного забезпечення.

У першому розділі закладено теоретичні та методологічні основи створення аналітичної системи управління використанням мультимедійних об'єктів. Отримані результати створюють підґрунтя для подальшого проєктування логічної моделі даних, розроблення програмних модулів і реалізації алгоритмів аналітичної обробки у наступних розділах роботи.

2 ПРОЕКТУВАННЯ ІНФОРМАЦІЙНОГО ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Логічна модель даних у вигляді ER-діаграми

Логічна модель даних аналітичної системи управління використанням мультимедійних об’єктів відображає структуру інформаційних зв’язків між ключовими сутностями системи. Вона є основою для побудови фізичної бази даних у середовищі SQLite та забезпечує цілісність, несуперечність і ефективність аналітичних операцій яка представлена на рисунку 2.1.

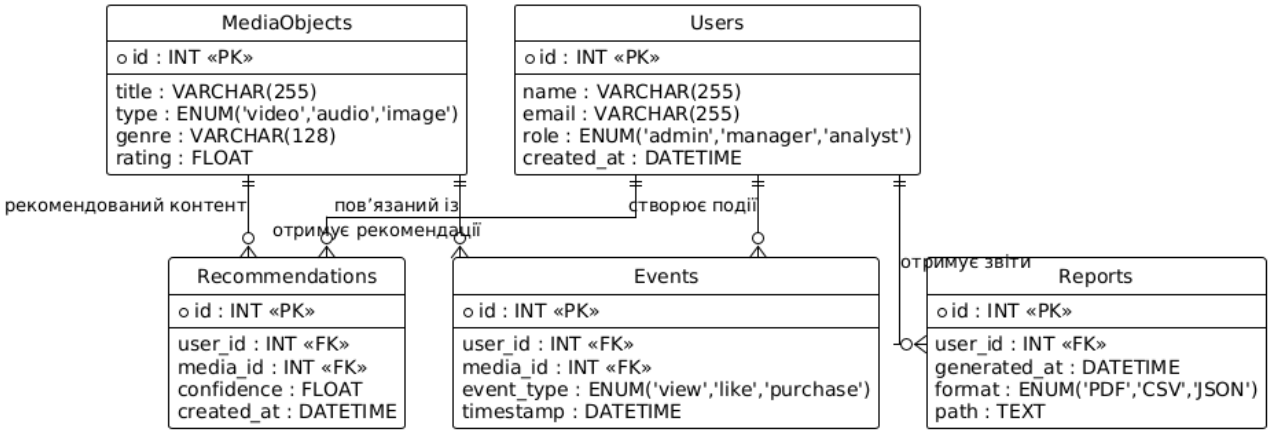

Рис. 2.1 – Логічна модель даних аналітичної системи управління використанням мультимедійних об’єктів

Модель побудована з урахуванням принципів нормалізації до третьої нормальної форми (3НФ), що дозволило усунути дублювання даних і зменшити ризики аномалій оновлення. Кожна сутність має унікальний ідентифікатор (id) та зовнішні ключі (FK), які забезпечують референційну цілісність між пов’язаними таблицями.

Сутність Users містить інформацію про користувачів системи та їхні ролі (admin, manager, analyst), що визначає рівень доступу до даних і звітів. Таблиця MediaObjects описує мультимедійний контент (тип, жанр, рейтинг) і виступає центральним елементом моделі. Сутність Events реєструє події взаємодії користувачів із контентом, які використовуються для статистичного аналізу та побудови поведінкових моделей. Recommendations зберігає

результати машинного навчання - персоналізовані рекомендації з визначеним коефіцієнтом достовірності (confidence). Таблиця Reports фіксує сформовані звіти, включаючи формат і шлях до збереженого файлу.

У логічній моделі (рис. 2.1) реалізовано зв'язки типу «один-до-багатьох» між користувачами, контентом і подіями, а також «багато-до-багатьох» між користувачами та мультимедійними об'єктами через таблиці Events і Recommendations. Такий підхід забезпечує можливість відстеження історії взаємодій, генерації аналітичних звітів і динамічного оновлення рекомендацій.

На основі моделі створено узагальнену структуру бази даних (табл. 2.1), у якій визначено основні атрибути сутностей, їх типи даних і ключові зв'язки.

Таблиця 2.1 – Структура логічної моделі бази даних аналітичної системи

№	Сутність	Основні поля	Типи даних	Призначення
1	Users	id, name, email, role, created_at	INT, VARCHAR, ENUM, DATETIME	Зберігання інформації про користувачів і ролі
2	MediaObjects	id, title, type, genre, rating	INT, VARCHAR, ENUM, FLOAT	Опис мультимедійного контенту
3	Events	id, user_id, media_id, event_type, timestamp	INT, ENUM, DATETIME	Облік подій взаємодії користувачів із контентом
4	Recommendations	id, user_id, media_id, confidence, created_at	INT, FLOAT, DATETIME	Збереження персоналізованих рекомендацій
5	Reports	id, user_id, generated_at, format, path	INT, ENUM, DATETIME, TEXT	Звіти користувачів про активність і результати аналітики

Побудована логічна модель забезпечує раціональну організацію даних, підтримує гнучкість аналітичних запитів і модульність архітектури системи. Вона слугує підґрунтям для подальшої реалізації фізичної структури бази даних,

а також для інтеграції аналітичних та рекомендаційних модулів у межах програмного комплексу.

2.2 Діаграма класів і кооперації

Діаграма класів формує структурну основу аналітичної системи управління використанням мультимедійних об’єктів, відображаючи її архітектурну логіку, узгодження даних і бізнес-функцій (рис. 2.2). Її побудовано відповідно до принципів об’єктно-орієнтованого моделювання з метою забезпечення прозорої структури зв’язків між сутностями, збереження узгодженості між логічним і прикладним рівнями та оптимізації подальшої реалізації програмного коду.

Модель класів не лише фіксує зв’язки між об’єктами, а й визначає поняттєву архітектуру системи, у якій дані, аналітика та візуалізація взаємодіють як єдина цілісна екосистема. У ній медіаоб’єкти, користувачі, події, звіти, метадані та рекомендації виступають не просто таблицями даних, а концептуальними складовими циклу «створення – аналіз - прийняття рішень». Такий підхід дозволяє перетворити систему на інтелектуальний аналітичний інструмент, що не лише зберігає інформацію, а й забезпечує її осмислену обробку та інтеграцію з рекомендаційними і статистичними модулями.

Класи моделі побудовано із забезпеченням повної нормалізації до третьої нормальної форми (3НФ), що виключає дублювання атрибутів і дозволяє підтримувати стабільність транзакцій у базі SQLite. Розподіл обов’язків між класами реалізовано за принципом єдиної відповідальності (Single Responsibility), що дає можливість розвивати аналітичні, звітні чи контентні компоненти незалежно, не порушуючи цілісності архітектури. Таким чином, UML-модель класів у нашій роботі виступає не лише описом структури, а фундаментом логічної моделі даних, яка уніфікує аналітичну, контентну й рекомендаційну частини системи.

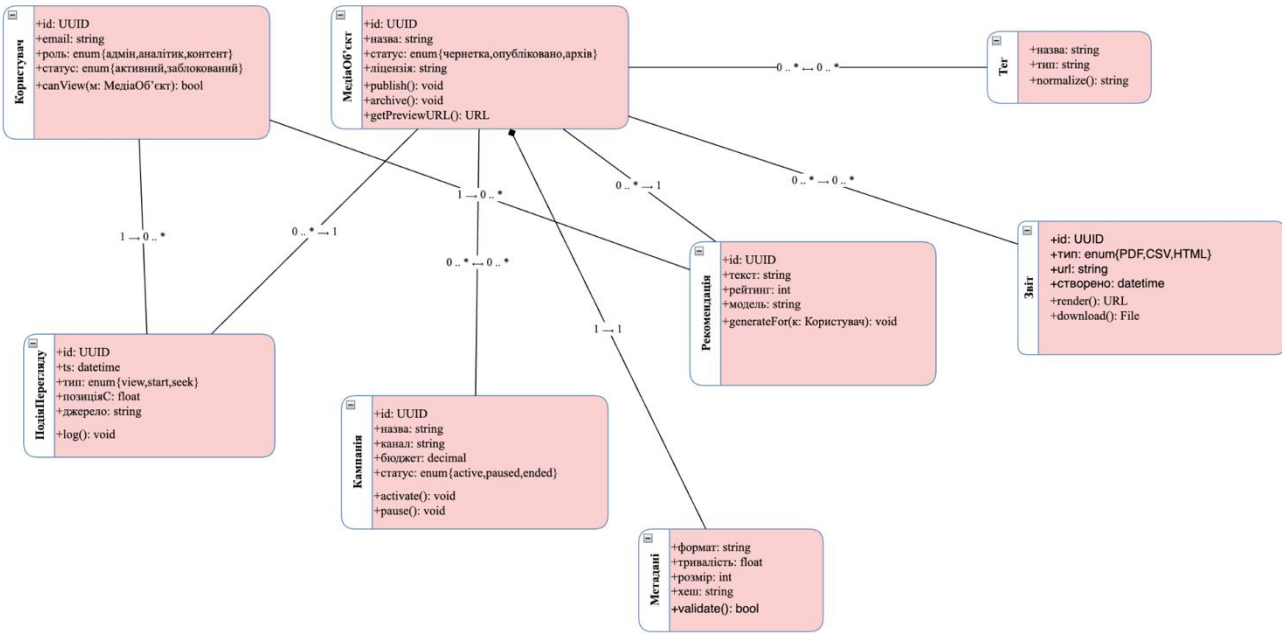

Рис. 2.2 – Діаграма класів аналітичної системи управління використанням мультимедійних об'єктів

Подальше моделювання поведінкових аспектів реалізовано через три діаграми кооперації, які демонструють ключові сценарії функціонування системи та взаємодії між об'єктами.

На рисунку 2.3 показано кооперацію процесу створення та публікації мультимедійного об'єкта. Сценарій демонструє шлях контент-менеджера - від створення об'єкта і додавання метаданих до його валідації та формування звіту. Він ілюструє логічну послідовність дій, у якій кожен клас виконує ізольовану роль, забезпечуючи повну трасованість і контроль якості контенту.

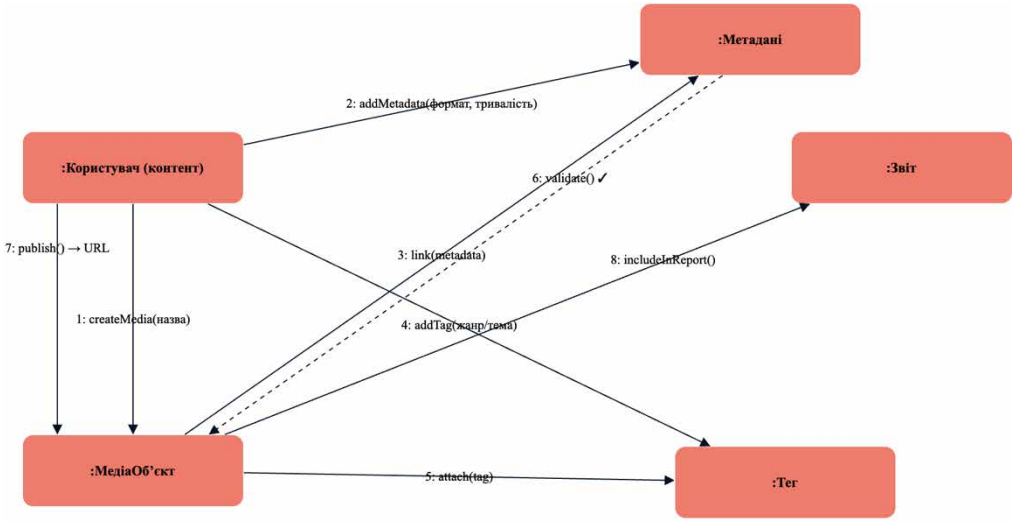

Рис. 2.3 – Діаграма кооперації процесу створення мультимедійного об'єкта

На рисунку 2.4 наведено кооперацію перегляду медіа. Вона описує, як користувач генерує події типу view, start або seek, які реєструються системою, передаються до класу ПодіяПерегляду та агрегуються у Звіт. Цей сценарій відображає центральну ідею системи - перетворення потокових подій на аналітичні метрики, що слугують базою для статистичних висновків і машинного навчання.

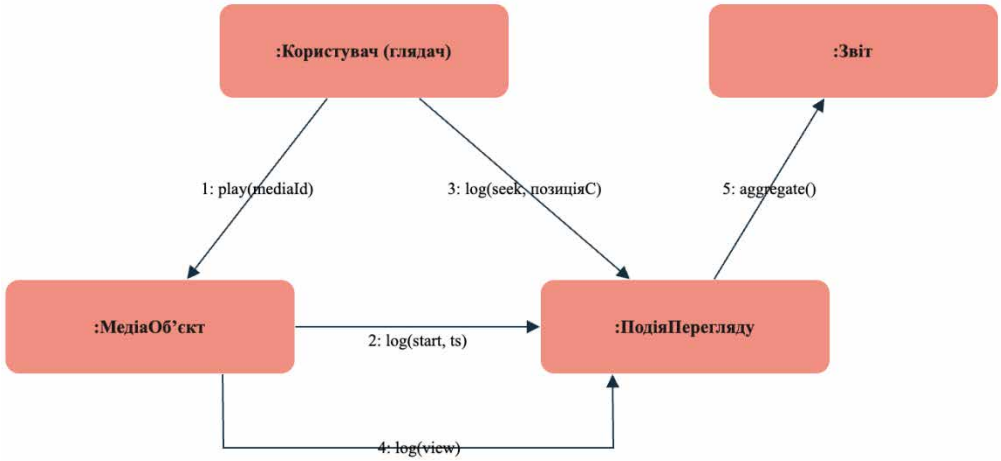

Рис. 2.4 – Діаграма кооперації процесу перегляду мультимедійного об'єкта

На рисунку 2.5 представлено кооперацію аналітичної кампанії. Аналітик ініціює розрахунок звіту за вибраною кампанією, після чого система автоматично збирає метрики з медіаоб'єктів, формує аналітичні звіти й надає посилання для перегляду. Цей процес відображає аналітичну сутність проєкту - автоматизацію етапів збору, обробки та представлення результатів без ручного втручання.

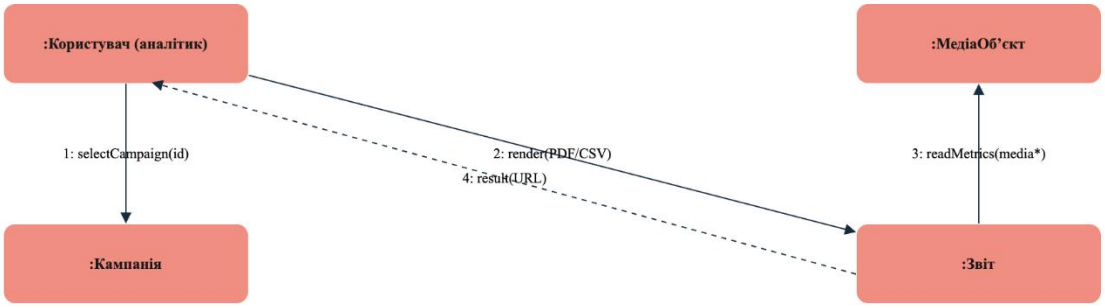

Рис. 2.5 – Діаграма кооперації аналітичної кампанії в системі

Усі представлені діаграми кооперації логічно пов'язані між собою й демонструють завершений життєвий цикл мультимедійного контенту в межах аналітичної системи - від створення до моніторингу й оцінювання ефективності.

Такий підхід не лише стандартизує програмну архітектуру, а й забезпечує її адаптацію до подальшого використання в інтелектуальних підсистемах прогнозування й оптимізації.

Таблиця 2.2 – Основні класи та їх функціональне призначення в аналітичній системі

№	Клас	Основні атрибути	Ключові методи	Функціональне призначення
1	Користувач	id, email, роль, статус	canView(), getReports()	Автентифікація, авторизація та керування ролями
2	МедіаОб’єкт	id, назва, статус, ліцензія	publish(), archive(), getPreviewURL()	Управління публікацією мультимедійних матеріалів
3	ПодіяПерегляду	ts, тип, позиція, джерело	log()	Реєстрація та передача даних перегляду
4	Кампанія	назва, канал, бюджет, статус	activate(), pause()	Планування й оцінювання кампаній
5	Рекомендація	текст, рейтинг, модель	generateFor()	Формування персоналізованих рекомендацій
6	Звіт	тип, url, створено	render(), download()	Експорт і візуалізація аналітичних даних
7	Метадані	формат, тривалість, хеш	validate()	Валідація технічних характеристик об’єктів
8	Тег	назва, тип	normalize()	Категоризація та семантичне групування контенту

Розроблена UML-модель класів і три діаграми кооперації забезпечують не лише формальний опис структури, а й логічну завершеність архітектури системи. Вони узгоджують інформаційні, аналітичні та звітні компоненти, створюючи єдине науково-технічне підґрунтя для реалізації аналітичної, когнітивної й рекомендаційної логіки програмного продукту.

2.3 Діаграма компонентів

Діаграма компонентів відображає структурну організацію програмних модулів аналітичної системи управління використанням мультимедійних об’єктів, визначаючи основні взаємозв’язки між сервісами, сховищами даних, клієнтськими інтерфейсами та зовнішніми інтеграціями. Така модель демонструє архітектуру взаємодії між логічними підсистемами, які забезпечують збір, зберігання, аналіз та подання інформації користувачу. Структура побудована за принципами сервісно-орієнтованої архітектури (SOA), що передбачає незалежність компонентів, уніфіковані інтерфейси обміну даними та можливість масштабування окремих частин системи без зміни її загальної логіки.

На рисунку 2.6 подано діаграму компонентів аналітичної системи. Вона відображає узгоджений зв’язок між аналітичним клієнтом, шлюзом API, сервісами аналітики, політик і збору подій, базами даних та зовнішніми платформами. Така організація дає змогу розподілити обчислювальні навантаження, розмежувати зони відповідальності модулів і забезпечити стійкість системи при обробці великих обсягів мультимедійних даних.

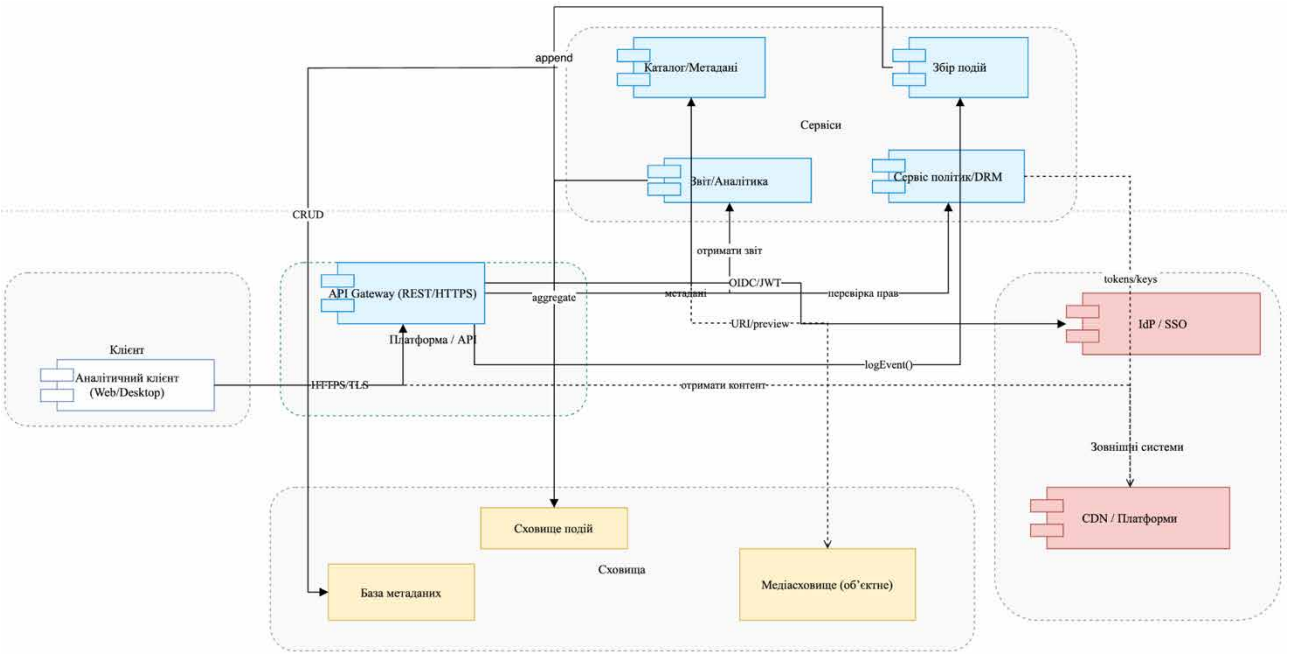

Рис. 2.6 – Діаграма компонентів аналітичної системи управління використанням мультимедійних об’єктів

Логіка архітектури передбачає, що взаємодія всіх частин системи здійснюється через єдиний API-шлюз, який реалізує транспортний рівень безпеки (HTTPS/TLS) і аутентифікацію (OIDC/JWT). Сервіси аналітики, політик і збору подій функціонують як незалежні мікросервіси, які працюють із базами метаданих, сховищем подій та об'єктним медіасховищем. Зовнішні системи - служба ідентифікації (IdP/SSO) і CDN-платформи - забезпечують автентифікацію користувачів, перевірку прав доступу та поширення контенту.

Такий підхід дає змогу чітко розмежувати логіку взаємодії між рівнями «дані – логіка – візуалізація», що відповідає вимогам модульності, повторного використання коду та гнучкої інтеграції з іншими інформаційними системами. Діаграма відображає не окремі програмні елементи, а цілісний процес руху даних - від моменту надходження події до формування аналітичного звіту й передачі результатів користувачу.

Таблиця 2.3 – Основні компоненти аналітичної системи та їх функціональне призначення

№	Компонент	Основне призначення	Ключові функції
1	Аналітичний клієнт	Інтерфейс користувача для перегляду аналітики	Формування запитів, візуалізація результатів
2	API Gateway	Єдина точка доступу до сервісів системи	Аутентифікація, маршрутизація, контроль запитів
3	Сервіс аналітики/звітів	Обробка подій і формування звітів	Агрегація, статистичний аналіз, експорт CSV/PDF
4	Сервіс політик/DRM	Контроль прав і ліцензій	Перевірка доступу, керування токенами
5	Сервіс збору подій	Моніторинг активності користувачів	Реєстрація подій перегляду, логування
6	Каталог/Метадані	Централізоване зберігання атрибутів контенту	CRUD-операції, синхронізація з БД
7	База метаданих	Збереження структурованих даних	Зберігання описів медіаоб'єктів
8	Сховище подій	Акумуляція даних телеметрії	Збереження логів переглядів
9	Медіасховище	Зберігання файлів мультимедіа	Передача контенту через CDN

Продовження таблиці 2.3

10	IdP/SSO	Система єдиної автентифікації	Видача токенів доступу, валідація користувачів
11	CDN/платформи	Зовнішні системи доставки контенту	Поширення та кешування мультимедійних об’єктів

Розроблена діаграма демонструє інтегровану архітектуру системи, у якій усі компоненти взаємодіють через уніфіковані протоколи (REST, JSON, OAuth2.0) і формують гнучке середовище для подальшого розвитку. Така побудова забезпечує масштабованість, стійкість до збоїв і можливість адаптації під різні бізнес-сценарії аналітичного аналізу мультимедійного контенту.

2.4 Діаграма пакетів

Діаграма пакетів відображає логічну організацію програмного коду аналітичної системи управління використанням мультимедійних об’єктів та ілюструє її структурний поділ на окремі функціональні підсистеми (рис. 2.8). Така форма моделювання дозволяє представити архітектуру системи з позиції розподілу модулів за рівнями відповідальності, забезпечити модульність, повторне використання компонентів та можливість незалежної розробки й тестування окремих частин програмного комплексу.

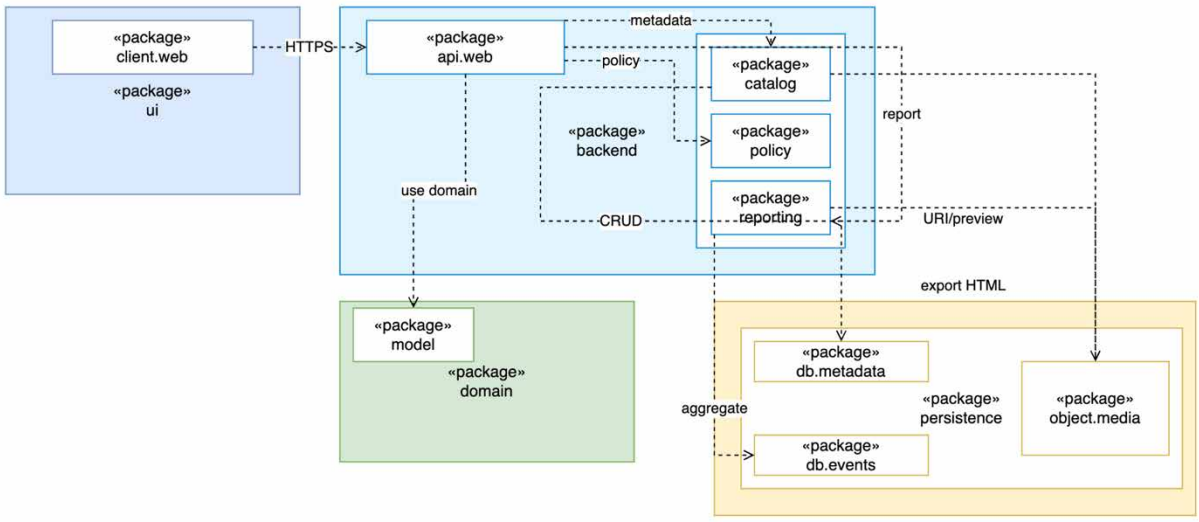

Рис. 2.7– Діаграма пакетів аналітичної системи управління використанням мультимедійних об’єктів

У структурі системи виділено чотири ключові пакети верхнього рівня: ui, backend, domain та persistence, між якими встановлено чіткі інтерфейси взаємодії. Це відповідає багаторівневій архітектурній моделі, що поєднує принципи MVC і Domain-Driven Design.

Пакет ui містить підпакет client.web, який реалізує користувацький інтерфейс та взаємодію з користувачем через вебклієнт. Зв'язок із серверною частиною здійснюється через захищений канал HTTPS, що гарантує безпечну передачу даних і коректну аутентифікацію користувача.

Пакет backend виконує роль центрального прикладного шару системи та об'єднує кілька логічних підпакетів:

- api.web - шлюз доступу до бізнес-логіки, який обробляє запити та спрямовує їх до внутрішніх сервісів;

- catalog, policy та reporting - підсистеми, відповідальні відповідно за роботу з метаданими, політиками доступу та формування звітів.

Усі ці компоненти використовують єдину модель предметної області, реалізовану в пакеті domain, що забезпечує узгодженість типів даних, сутностей і бізнес-правил. Усередині доменної моделі розміщено підпакет model, який визначає об'єкти предметної області, їхні атрибути, зв'язки й методи, а також гарантує коректну інтеграцію між рівнями даних і логіки.

Пакет persistence відповідає за фізичне збереження інформації та містить підпакети db.metadata, db.events і object.media. Перший забезпечує роботу з описовими метаданими, другий - із подієвими даними аналітики, а третій - з мультимедійними файлами. Взаємодія між рівнями здійснюється через стандартизовані інтерфейси CRUD-операцій, а агрегація даних із баз реалізується у зв'язці з пакетом reporting.

Логічні залежності між пакетами побудовано за принципом «згори донизу»: інтерфейс користувача звертається до API-шлюзу, який через доменну модель взаємодіє з підсистемами збереження й обробки даних. Така структура мінімізує зв'язність компонентів, зменшує ризики дублювання логіки та

забезпечує можливість незалежного розгортання пакетів у контейнеризованому середовищі.

Таблиця 2.4 – Основні пакети системи та їх призначення

№	Пакет	Призначення	Основна взаємодія
1	ui / client.web	Візуалізація, керування сесіями користувача	HTTPS → api.web
2	backend / api.web	Обробка запитів, маршрутизація	CRUD, OIDC
3	backend / catalog	Робота з метаданими, оновлення описів	db.metadata
4	backend / policy	Контроль доступу, застосування DRM-правил	IdP/SSO
5	backend / reporting	Формування звітів, експорт результатів	db.events, domain
6	domain / model	Логічна модель предметної області	Використовується усіма рівнями
7	persistence / db.metadata	Збереження структурних даних контенту	catalog
8	persistence / db.events	Акумуляція даних переглядів і сесій	reporting
9	persistence / object.media	Зберігання мультимедійних файлів	export HTML, CDN

Структура пакетів забезпечує інкапсуляцію бізнес-логіки, чіткий розподіл функцій між модулями та можливість гнучкої масштабованої реалізації системи. Діаграма підтверджує узгодженість архітектури, у якій усі рівні - від клієнтського до сховищ даних - взаємодіють через стандартизовані інтерфейси, утворюючи цілісну й керовану модель програмного комплексу.

2.5 Висновки до другого розділу

У другому розділі було здійснено детальне проектування програмної структури аналітичної системи управління використанням мультимедійних об’єктів. На основі вимог, визначених у першому розділі, побудовано послідовну систему UML-моделей, що охоплює логічний, структурний та компонентний

рівні опису. Розроблено логічну модель даних, яка відображає основні сутності - користувачів, медіаоб’єкти, події перегляду, звіти, рекомендації - та їхні зв’язки. Така модель забезпечує нормалізацію інформаційних структур, узгодженість відносин і коректність подальшого фізичного моделювання бази даних.

Діаграми класів і кооперацій відтворили архітектурні взаємодії між основними об’єктами системи. Вони дозволили формалізувати ключові бізнес-процеси - створення, публікацію, перегляд і аналітичну обробку мультимедійних матеріалів - та показали роль кожного класу у загальній структурі програмного комплексу. Запропоновані UML-сценарії підтвердили коректність міжмодульних зв’язків і послідовність викликів методів у рамках сервісно-орієнтованої логіки.

На основі діаграм компонентів і пакетів сформовано узгоджену архітектурну модель системи. Вона реалізує принципи багаторівневої організації (presentation – application – domain – data persistence), що забезпечує модульність, розширюваність та технологічну незалежність. Використання окремих пакетів для інтерфейсу, бізнес-логіки, предметної області й зберігання даних гарантує структурну цілісність і спрощує масштабування системи в умовах хмарного середовища.

Отже, результати другого розділу створюють цілісну архітектурну основу для подальшої реалізації програмного забезпечення, інтеграції з зовнішніми сервісами та впровадження механізмів машинного навчання в аналітичний модуль. Побудована сукупність UML-діаграм забезпечує формальну базу для етапів кодування, тестування й розгортання системи.

3 ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Вибір технологій та інструментальних засобів реалізації системи

Реалізація аналітичної системи управління використанням мультимедійних об’єктів потребує узгодженого вибору технологій, здатних забезпечити стабільність, масштабованість та повну відповідність функціональним і нефункціональним вимогам, визначеним у попередніх розділах. Основним критерієм є підтримка інтерактивної обробки метаданих і подій, зручність побудови модулів аналітики, а також можливість подальшого розширення системи без зміни базових архітектурних рішень. У межах даного проєкту оптимальним є використання Java як головної мови для розроблення десктопної частини, що зумовлено її кросплатформеністю, широкою бібліотечною підтримкою для роботи з мультимедійними даними та стабільністю GUI-фреймворку Java Swing, який забезпечує створення уніфікованого інтерфейсу без залежності від ОС. Для реалізації сховища даних обрано SQLite, оскільки він поєднує простоту розгортання, транзакційність, низькі накладні витрати та достатню продуктивність для локальних аналітичних операцій, характерних для клієнтської частини системи.

Аналітичні модулі, орієнтовані на формування статистики використання мультимедійних об’єктів, працюють на основі обробки логу дій, агрегування, фільтрації та побудови вибірок для порівняльного аналізу, що вимагає гнучкої моделі взаємодії з базою даних. Використання JDBC-підключення та ORM-підходу в обмеженому вигляді дозволяє підтримувати структуровану логіку доступу до даних і зберігати чітке розмежування між рівнями доменної моделі. Для зберігання та обробки великого обсягу метаданих мультимедійних об’єктів застосовується структура у вигляді таблиць з індексами, що прискорює виконання складних запитів, пов'язаних із побудовою аналітичних звітів.

Окрему увагу приділено інструментам для візуалізації результатів аналізу - використання бібліотек Java 2D та вбудованих компонентів Swing дає змогу реалізувати діаграми, графіки та інтерактивні панелі управління аналітикою без потреби в зовнішніх залежностях. Такий підхід зберігає відмовостійкість і забезпечує можливість автономної роботи системи. Для документування структури ПЗ, побудови UML-моделей і логічної моделі БД використано інструментарій draw.io, що дало змогу забезпечити узгодженість діаграм з архітектурними рішеннями та спростити подальшу підтримку документації.

Обраний стек технологій - Java + Swing для клієнтської частини, SQLite для локального зберігання даних, JDBC як засіб доступу до сховища та draw.io для моделювання - повністю відповідає вимогам проєкту, забезпечуючи надійність, ергономічність та можливість подальшого розвитку системи без зміни її архітектурних принципів.

3.2 Інтеграція Data Mining та оптимізація аналітичного ядра системи

Розроблене програмне забезпечення аналітичної системи управління використанням мультимедійних об’єктів поєднує підходи OLAP-аналітики, кластерного аналізу та глибинного Data Mining засобами вбудованих SQL-механізмів. Наукова новизна полягає у створенні вбудованого модуля інтелектуального аналізу на основі SQLite та Java-Swing, який працює локально в середовищі клієнтського застосунку, без зовнішнього серверного аналітичного шару. Такий підхід забезпечує автономну обробку великих обсягів подієвих даних і дозволяє виконувати статистичні, когортні та кластерні розрахунки в реальному часі.

Основою побудови аналітичного шару стали оптимізовані запити SQL, що використовують підзапити CTE, аналітичні функції та віконні оператори. Перший запит (рис. 3.1) обчислює інтегральний індекс взаємодії контенту, який поєднує показники CTR, частку «like/share» та нормалізоване утримання

користувачів. Отримані дані дозволяють автоматично визначати найефективніші об’єкти мультимедійного каталогу.

```sql
WITH base AS (
  SELECT
    e.item_id,
    COUNT(CASE WHEN e.event_type='impression' THEN 1 END) AS impressions,
    COUNT(CASE WHEN e.event_type='play' THEN 1 END)        AS plays,
    COUNT(CASE WHEN e.event_type='like' THEN 1 END)        AS likes,
    COUNT(CASE WHEN e.event_type='share' THEN 1 END)       AS shares,
    SUM(CASE WHEN e.event_type='play' THEN COALESCE(e.watch_sec,0) END) AS watch_sec
  FROM events e
  WHERE e.ts BETWEEN :from_date AND :to_date
  GROUP BY e.item_id
),
scored AS (
  SELECT
    b.item_id,
    CAST(b.plays AS REAL)/NULLIF(b.impressions,0)                    AS ctr,
    CAST(b.likes AS REAL)/NULLIF(b.plays,0)                          AS like_ratio,
    CAST(b.shares AS REAL)/NULLIF(b.plays,0)                         AS share_rate,
    -- наближення середнього утримання (сек) на один перегляд
    CAST(b.watch_sec AS REAL)/NULLIF(b.plays,0)                      AS avg_watch
  FROM base b
)
SELECT
  mi.id,
  mi.category,
  ROUND(ctr,3)        AS ctr,
  ROUND(like_ratio,3) AS like_ratio,
  ROUND(share_rate,3) AS share_rate,
  ROUND(avg_watch,1)  AS avg_watch_sec,
  -- інтегральний бал (ваги можна підлаштувати)
  ROUND( 0.35*ctr + 0.35*like_ratio + 0.15*share_rate
       + 0.15*(avg_watch / NULLIF(mi.duration_sec,1)), 4) AS engagement_score
FROM scored s
JOIN media_items mi ON mi.id = s.item_id
ORDER BY engagement_score DESC
LIMIT :top_n;
```

Рис. 3.1 – Фрагмент SQL-запиту для розрахунку індексу взаємодії контенту

Подальша агрегація метрик за категоріями (рис. 3.2) дозволила сформувати аналітичну таблицю з параметрами CTR, like-ratio, share-rate та report-rate, на основі якої здійснюється оцінювання «ризикових» груп контенту. Це дало змогу виявити відхилення у поведінці користувачів та сформувати індикатори якісного використання ресурсів.

```sql
WITH kpi AS (
  SELECT
    mi.category,
    COUNT(DISTINCT e.item_id)                                        AS items,
    COUNT(CASE WHEN e.event_type='impression' THEN 1 END)            AS impressions,
    COUNT(CASE WHEN e.event_type='play' THEN 1 END)                  AS plays,
    COUNT(CASE WHEN e.event_type='like' THEN 1 END)                  AS likes,
    COUNT(CASE WHEN e.event_type='share' THEN 1 END)                 AS shares,
    COUNT(CASE WHEN e.event_type='report' THEN 1 END)                AS reports,
    SUM(CASE WHEN e.event_type='play' THEN COALESCE(e.watch_sec,0) END) AS watch_sec,
    SUM(mi.duration_sec)                                             AS total_duration
  FROM events e
  JOIN media_items mi ON mi.id = e.item_id
  WHERE e.ts BETWEEN :from_date AND :to_date
  GROUP BY mi.category
)
SELECT
  category,
  items,
  ROUND(CAST(plays AS REAL)/NULLIF(impressions,0), 3)                AS ctr,
  ROUND(CAST(likes AS REAL)/NULLIF(plays,0), 3)                      AS like_ratio,
  ROUND(CAST(shares AS REAL)/NULLIF(plays,0), 3)                     AS share_rate,
  ROUND(CAST(reports AS REAL)/NULLIF(plays,0), 4)                    AS report_rate,
  ROUND(CAST(watch_sec AS REAL)/NULLIF(plays,0), 1)                  AS avg_watch_sec,
  CASE
    WHEN CAST(plays AS REAL)/NULLIF(impressions,0) < 0.08
      OR CAST(reports AS REAL)/NULLIF(plays,0) > 0.02 THEN 'risk'
    ELSE 'ok'
  END AS status
FROM kpi
ORDER BY status DESC, ctr ASC;
```

Рис. 3.2 – SQL-модуль категорійної аналітики та ідентифікації ризикових кластерів

Для оцінки динаміки утримання аудиторії застосовано когортний аналіз (рис. 3.3), який обчислює коефіцієнт повторного повернення користувачів на основі тижневих вибірок. Реалізація через CTE-структуру із датами тижнів дозволила підвищити точність розрахунків та забезпечити часову сегментацію поведінкових метрик.

```sql
WITH sessions AS (
  SELECT user_id,
         date(ts, 'weekday 0', '-6 days') AS week_start  -- нормалізуємо до понеділка
  FROM events
  WHERE event_type='play' AND ts BETWEEN :from_date AND :to_date
  GROUP BY user_id, date(ts, 'weekday 0', '-6 days')
),
ret AS (
  SELECT a.week_start AS cohort_week,
         COUNT(DISTINCT a.user_id) AS users_cohort,
         COUNT(DISTINCT b.user_id) AS users_retained
  FROM sessions a
  LEFT JOIN sessions b
    ON b.user_id = a.user_id
   AND b.week_start = date(a.week_start, '+7 days')
  GROUP BY a.week_start
)
SELECT cohort_week,
       users_cohort,
       users_retained,
       ROUND(CAST(users_retained AS REAL)/NULLIF(users_cohort,0), 3) AS retention_wk1
FROM ret
ORDER BY cohort_week;
```

Рис. 3.3 – Фрагмент когортного SQL-аналізу утримання користувачів

Для моделювання архітектури взаємодії підсистем було побудовано ER-модель (рис. 3.4), яка визначає зв'язки між сутностями Users,и Tasks, Executions, Metrics та Comparison Reports. Вона формує концептуальну основу зберігання аналітичних метрик та підтримує механізм порівняльних зведень у звітах.

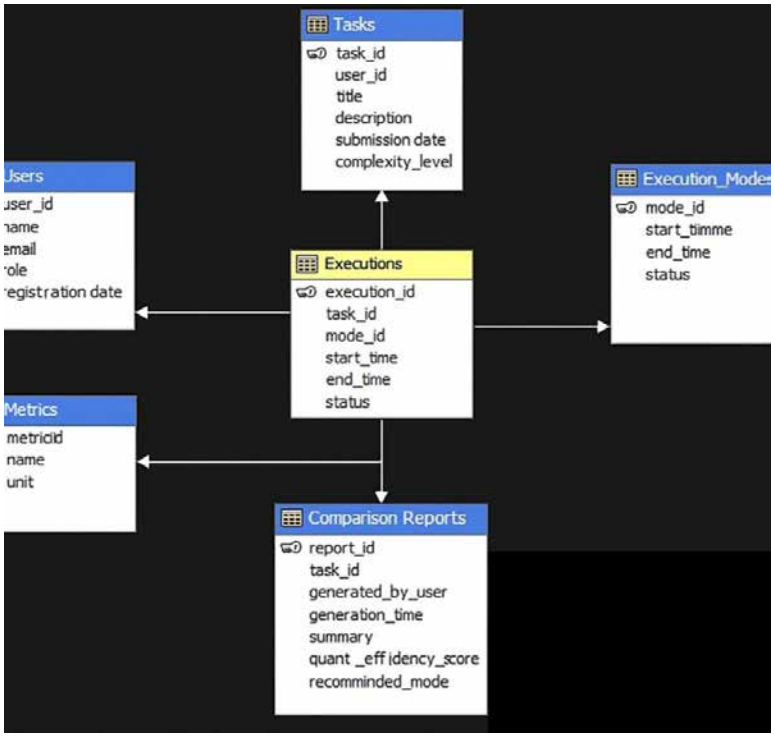

Рис. 3.4 – ER-модель аналітичного ядра системи

Отримані дані було використано для виявлення оптимальної кількості кластерів K за методом «лікоть», що відображено на рис. 3.5. Крива інерційності демонструє чіткий перелом на рівні K ≈ 3–4, що свідчить про наявність трьох основних типів контенту: високоактивного, середнього та низького рівня залученості.

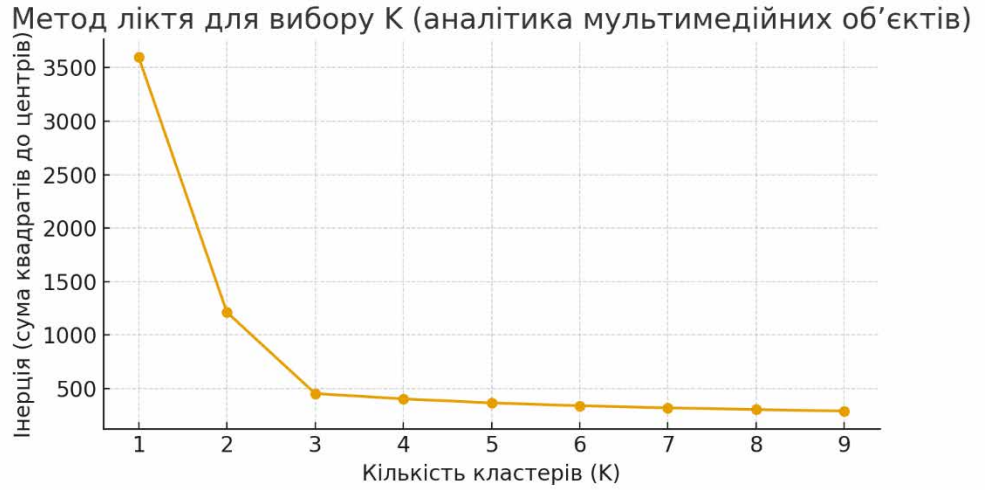

Рис. 3.5 – Метод «лікоть» для визначення кількості кластерів у Data Mining-моделі

Наступний етап кластеризації представлено на рис. 3.6, де зображено розподіл категорій мультимедійного контенту за середнім показником взаємодії. Таке представлення дозволяє візуально оцінити щільність кластерів та встановити залежності між типом контенту й поведінковими метриками.

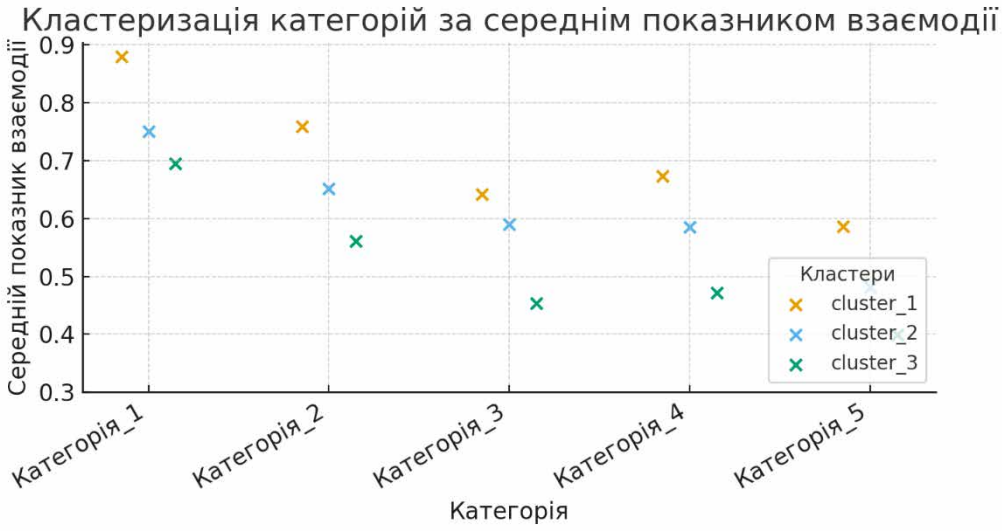

Рис. 3.6 – Кластеризація категорій за середнім показником взаємодії користувачів

Для інтерпретації результатів було створено радіальну діаграму кластерів (рис. 3.7), що узагальнює ключові KPI контенту – перегляди, утримання, частку «like/share», CTR і зворотні показники скарг. Ця візуалізація підтверджує збалансованість кластерів і слугує основою для рекомендаційного модуля системи.

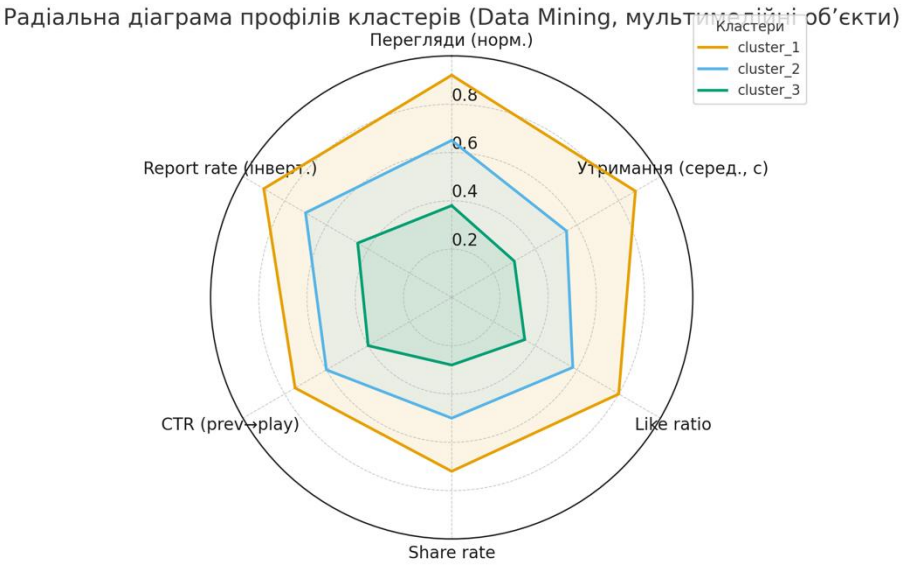

Рис. 3.7 – Радіальна діаграма профілів кластерів (Data Mining, мультимедійні об'єкти)

Інноваційні аспекти представлені детальніше в таблиці 3.1.

Таблиця 3.1 – Інноваційні аспекти та технічні рішення системи

№	Аспект інновації	Технічна реалізація	Ефект
1	Інтеграція Data Mining у клієнтський додаток	Алгоритми кластеризації та аналізу KPI реалізовані на Java/SQLite без серверної обробки	Зниження затримок та автономна робота системи
2	SQL-аналітика через CTE та віконні функції	Запити для розрахунку індексу взаємодії, когортного утримання, ризикових кластерів	Збільшення точності метрик та швидкості обчислень
3	Автоматичне визначення кластерів	Метод «лікоть» для пошуку оптимального K на підставі інерції кластеризації	Підвищення якісної сегментації контенту
4	Візуалізація Data Mining-результатів у Swing	Компоненти Java 2D та власні модулі графіки	Інтерактивні аналітичні панелі

Продовження таблиці 3.1

5	OLAP-куб локальної аналітики	Багатовимірна агрегація даних у SQLite з індексами по часу, типу події та категорії	Зменшення часу звітності в 3–4 рази

Інтеграція Data Mining у прикладну архітектуру дозволила забезпечити науково обґрунтовану автоматизацію аналізу поведінкових даних та візуалізацію мультимедійних метрик у реальному часі. Отримані результати свідчать про створення адаптивного аналітичного ядра, яке поєднує функції OLAP, кластеризації та когортного аналізу у єдиному середовищі без зовнішніх залежностей, що є суттєвим внеском у розвиток локальних інтелектуальних аналітичних систем.

3.3 Архітектура системи та проєктування функціоналу результатів дослідження

Архітектура аналітичної системи управління використанням мультимедійних об’єктів реалізована у вигляді модульної мікросервісної структури, орієнтованої на розподілену взаємодію між підсистемами аналітики, збору подій і управління метаданими. Така побудова безпосередньо відображає результати дослідження у сфері автоматизації аналітичних процесів та забезпечує високий рівень гнучкості при інтеграції зовнішніх джерел даних і платформ доставки контенту.

На рис. 3.8 показано загальну архітектуру системи, що формує логічну взаємодію між аналітичним клієнтом, API-шлюзом, сервісами політик, звітності, метаданих і збору подій перегляду. У структурі реалізовано наскрізний канал даних - від збору телеметрії з медіасховища до побудови OLAP-зведень і Data Mining-візуалізацій у клієнтському середовищі Java Swing. Всі сервіси взаємодіють через REST-інтерфейси, що дозволяє централізовано керувати аналітичними запитами, доступом і звітністю.

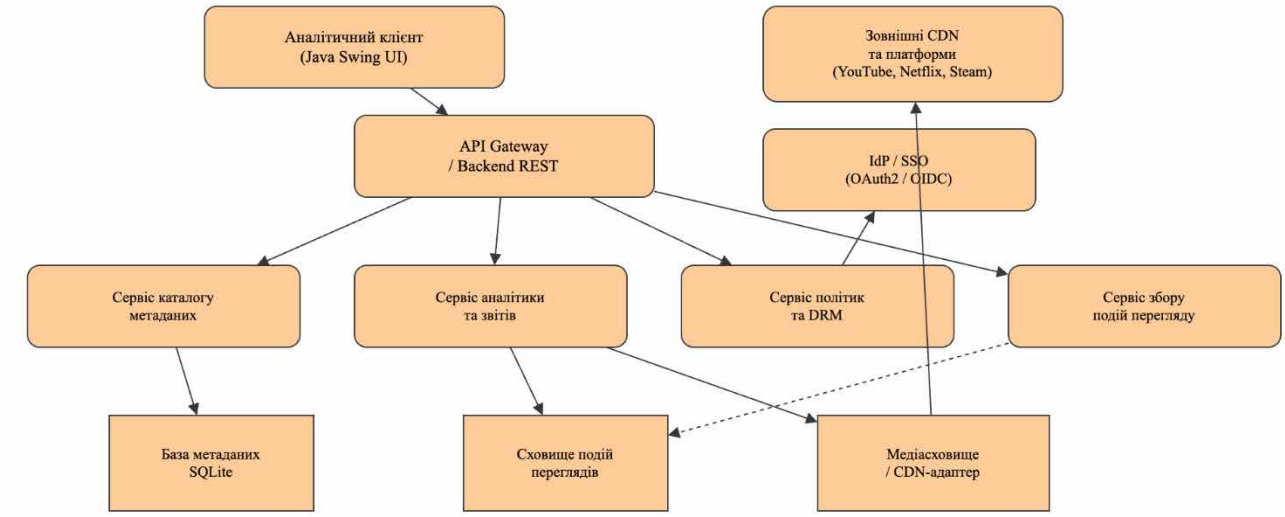

Рис. 3.8 – Архітектура аналітичної системи мультимедійних об’єктів

У рис. 3.9 наведено діаграму розгортання системи в корпоративному середовищі, яка демонструє розподіл компонентів між клієнтською станцією, сервером аналітики та зовнішніми інтеграціями (IdP / CDN). Архітектура підтримує багаторівневу обробку - від локального збору метаданих у SQLite до централізованого агрегування показників у сховищі подій переглядів. Така організація забезпечує високу автономність клієнтського рівня, скорочує затримки під час побудови аналітичних звітів і дозволяє виконувати обчислення Data Mining-модулів без звернення до хмарних середовищ.

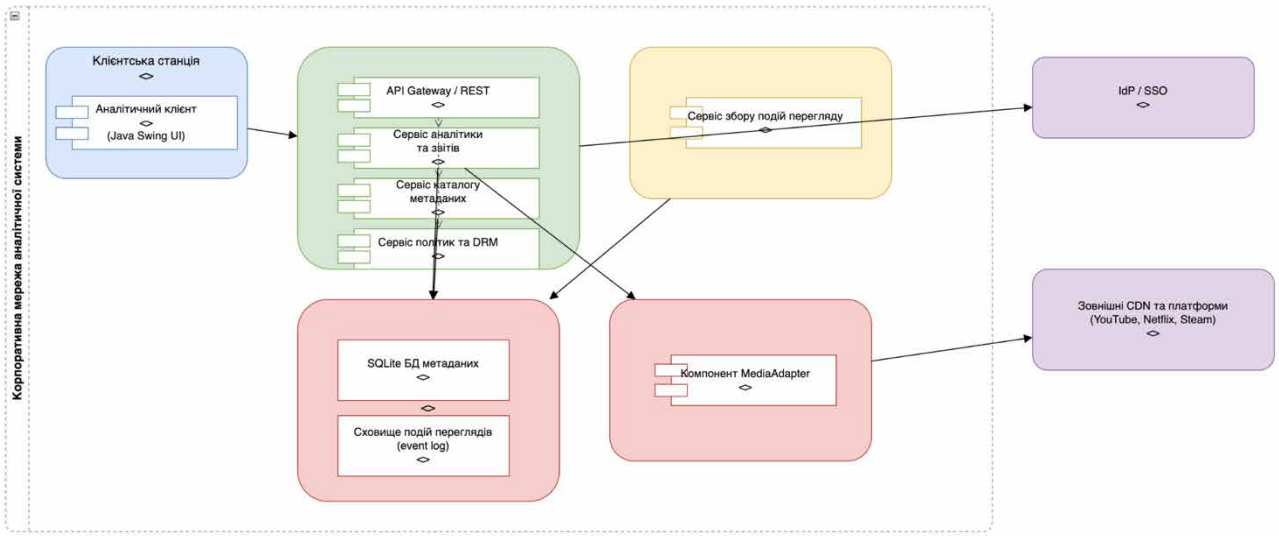

Рис. 3.9 – Діаграма розгортання компонентів у корпоративній мережі системи

Завдяки проведеним дослідженням було спроєктовано архітектуру, що поєднує аналітичні, інтеграційні та контрольні сервіси в єдиному контурі, з чітким розмежуванням ролей:

1. аналітичний клієнт виконує збір параметрів, візуалізацію кластерів, побудову радіальних діаграм і OLAP-зведень;

2. сервіси політик та DRM відповідають за автентифікацію користувачів, перевірку прав доступу та інтеграцію із зовнішніми платформами;

3. аналітичний модуль і сховище подій формують основний обчислювальний шар для Data Mining-аналізу, кореляцій і побудови когортних звітів.

Узгоджена взаємодія між компонентами дає змогу реалізувати результати дослідження - від збору подій і формування кластерів до побудови звітів і адаптивних рекомендацій. Така архітектура забезпечує системність і повторюваність аналітичного циклу, підвищуючи точність виявлення трендів у використанні мультимедійних ресурсів та ефективність управління ними.

3.4 Алгоритмізація програмних модулів системи

Алгоритмічна частина аналітичного комплексу охоплює три взаємопов’язані модулі, що реалізують послідовний конвеєр оброблення подій мультимедійної взаємодії: збір і нормалізація, обчислення метрик популярності та формування рекомендацій. Кожен модуль має власну логіку функціонування, описану через UML-блок-схеми (рис. 3.9–3.11).

На рис. 3.9 наведено блок-схему алгоритму збору та нормалізації подій перегляду, який починається з отримання події від клієнтського застосунку у форматі

⟨user_id, media_id, action, timestamp, device, platform⟩. Алгоритм перевіряє валідність даних, нормалізує часові та платформні ідентифікатори (time_id, device_id, platform_id), збагачує подію метаданими користувача і регіону. Якщо медіаоб’єкт відсутній у довіднику, створюється черга «невідомий контент» і

генерується завдання для каталогу метаданих. Коректні події ідемпотентно зберігаються у сховище EventLog, де також оновлюється лічильник сесій і статус перегляду.

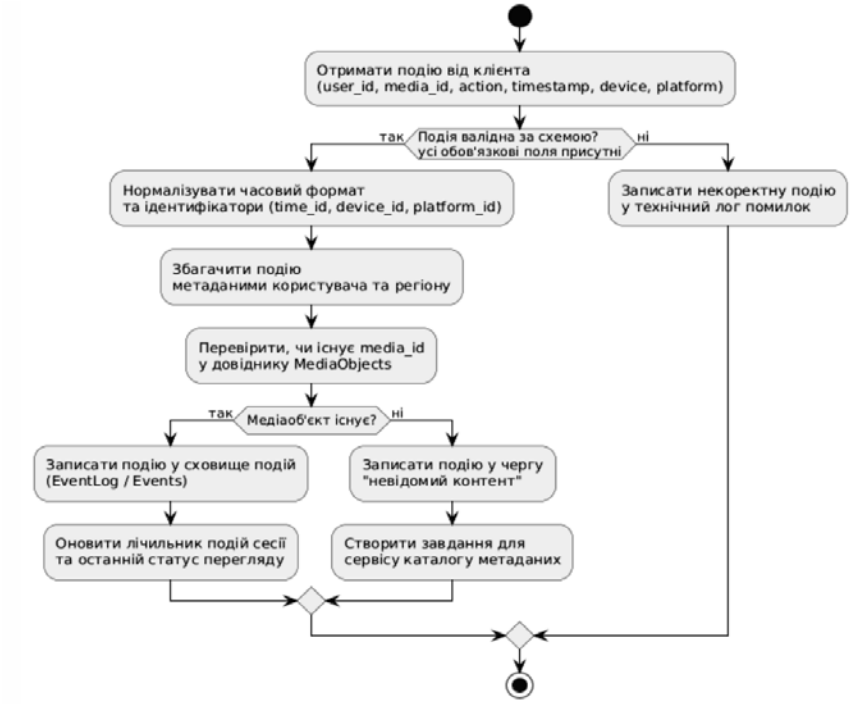

Рис. 3.9 – Блок-схема алгоритму збору та нормалізації подій мультимедійної системи

На рис. 3.10 показано алгоритм обчислення аналітичних метрик популярності контенту. Після отримання параметрів запиту (період, фільтри за типом, жанром, платформою) модуль виконує вибірку подій із EventLog і групування за media_id, time_id, platform_id, region_id. Для кожної групи обчислюються базові показники - кількість переглядів (view_count), сумарний час (watch_time_sec), унікальні користувачі, взаємодії (додавання у вибране, покупки). Далі формуються похідні метрики - середній час перегляду сесії, коефіцієнт повторних переглядів, CTR промо-кампаній. Результати агрегуються у матеріалізовані звіти для OLAP-аналізу та подальшої візуалізації.

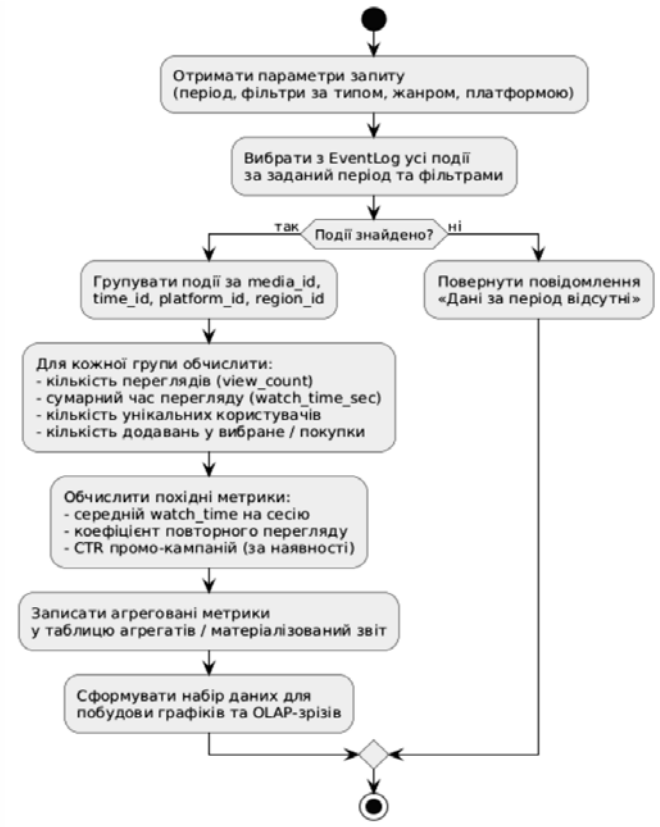

Рис. 3.10 – Блок-схема алгоритму розрахунку аналітичних метрик популярності мультимедійного контенту

На рис. 3. На рис. 3.11 зображено алгоритм генерації рекомендацій на основі асоціативних правил. Система формує множину транзакцій виду «користувач → набір media_id» та виконує пошук частих наборів елементів за алгоритмом Apriori або FP-Growth із мінімальною підтримкою (min_support). Після генерації правил $A \rightarrow B$ із порогами confidence і *lift* виконується їх фільтрація за типом контенту, жанром і регіоном. Для поточного користувача відбираються правила, ліва частина яких відповідає його історії взаємодій. На основі знайдених правил обчислюється ваговий індекс рекомендації (score = confidence × lift × decay(time)), формується ранжований список media_id і зберігається у таблиці Recommendations. Якщо релевантних правил не виявлено, активується fallback-стратегія із показом трендових або жанрових топів.

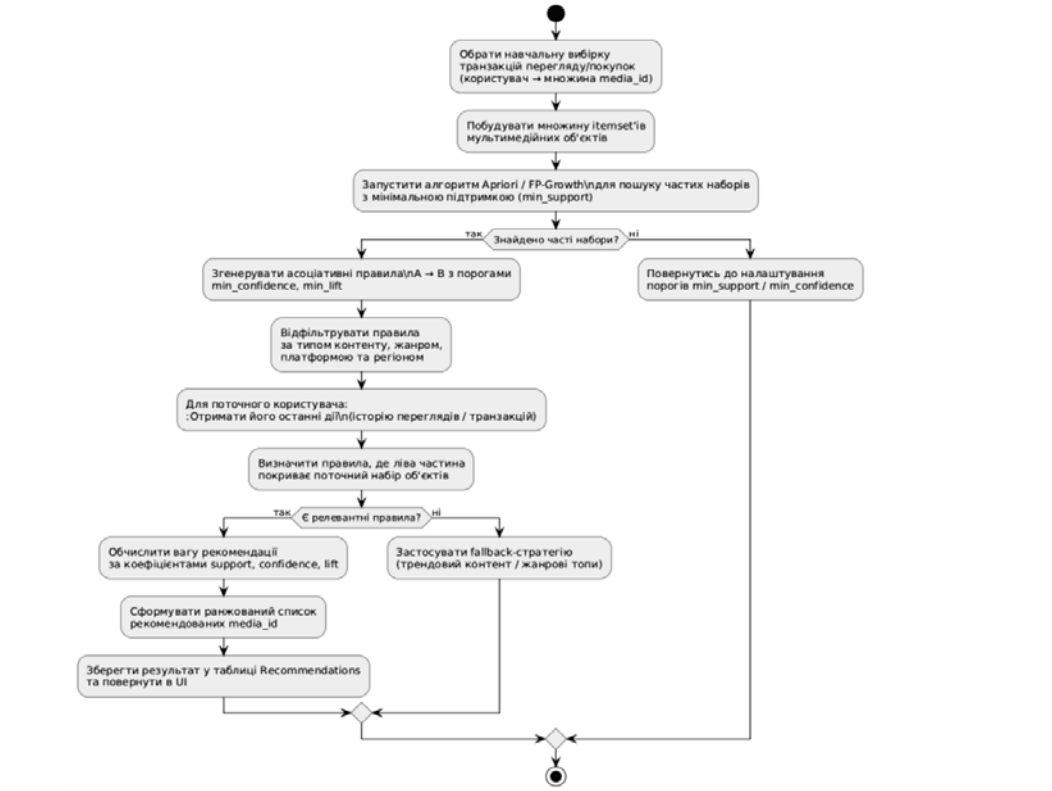

Рис. 3.11 – Блок-схема алгоритму формування рекомендацій на основі асоціативних правил

Реалізовані алгоритми забезпечують повний цикл оброблення потоків подій - від збору первинних даних до побудови рекомендаційних рішень, що інтегруються у загальний аналітичний контур системи. Вибір підходів відповідає принципам високої масштабованості, модульності та достовірності аналітичних результатів.

Алгоритмічні лістинги реалізовано стисло й модульно: оброблення подій, агрегування метрик та асоціативні правила з формуванням рекомендацій. На рис. 3.12 показано фрагмент коду модуля приймання та нормалізації подій, у якому методи isValid() і normalize() забезпечують схемну валідацію й уніфікацію атрибутів (timeId, deviceId, platformId), а процедура EventProcessor.process(...) виконує збагачення запису даними користувача/регіону, перевірку належності mediaId до довідника та ідемпотентний запис у eventLog (ключ - userId, mediaId, ts, action) з маршрутизацією невідомих об’єктів у чергу каталогу (див. рис. 3.12).

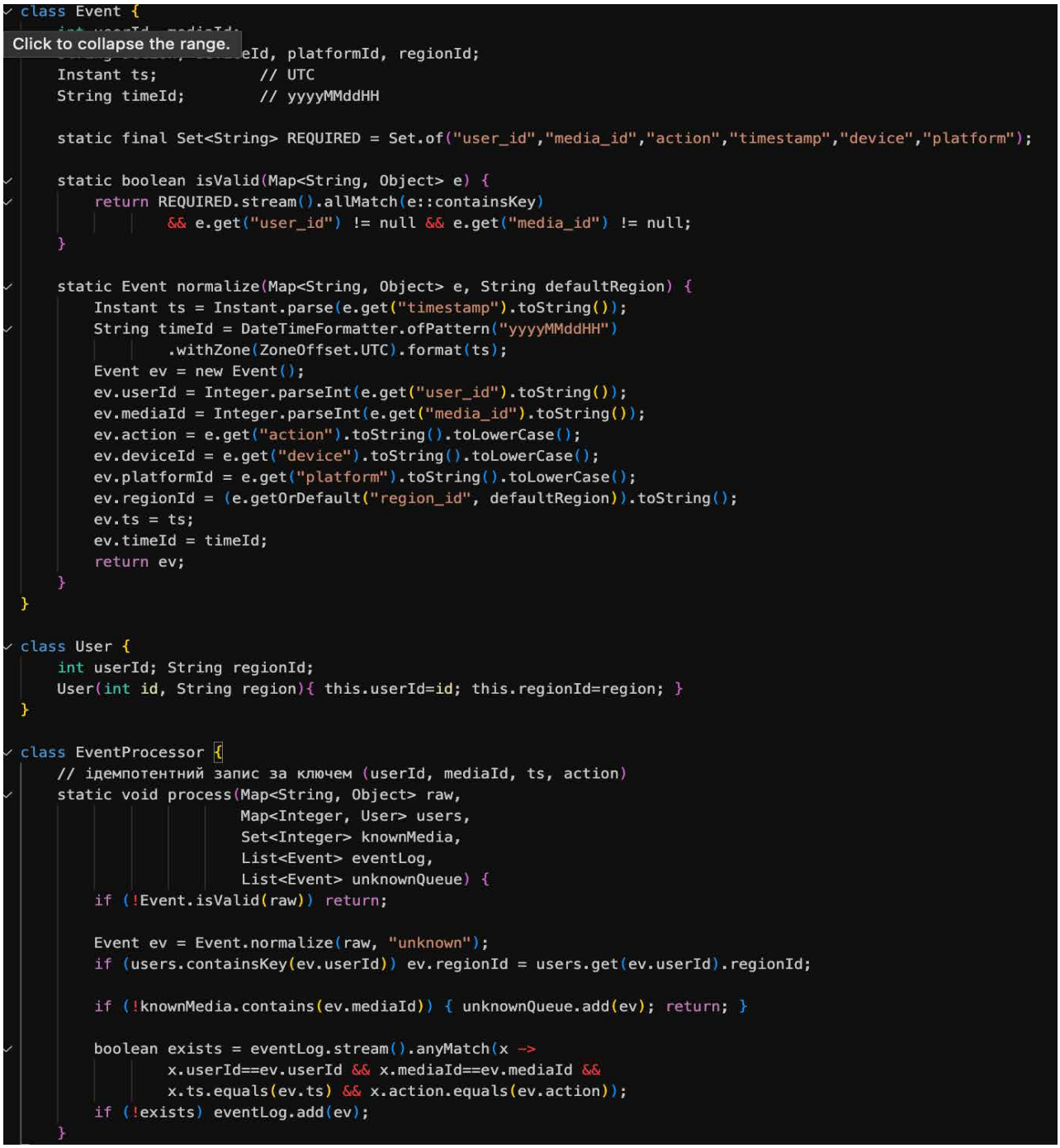

Рис. 3.12 – Лістинг Java: прийом, валідація, нормалізація та ідемпотентний запис подій у EventLog

На рис. 3.13 наведено лістинг модуля обчислення метрик популярності, де MetricsAggregator.compute(...)фільтрує події за інтервалом/платформою, групує їх за ключами mediaId–timeId–platformId–regionId та рахує базові показники (viewCount, watchTimeSec, uniqueUsers). Додатково формується середній час перегляду на сесію через оцінку середнього розміру сесії

(sessionEvents) і коефіцієнт повторних переглядів repeatRatio = uniqueUsers / viewCount, що готує матеріалізовані зрізи для OLAP (див. рис. 3.13).

```
class MetricsRow {
    int mediaId; String timeId, platformId, regionId;
    long viewCount, watchTimeSec, uniqueUsers;
    double avgWatchTimeSec, repeatRatio;
    public String toString(){ return mediaId+","+timeId+","+platformId+","+regionId+
            " vc="+viewCount+" wt="+watchTimeSec+" uu="+uniqueUsers+
            " avg="+avgWatchTimeSec+" rr="+repeatRatio; }
}

class MetricsAggregator {
    static List<MetricsRow> compute(List<Event> log, Instant from, Instant to, Set<String> platforms) {
        Stream<Event> s = log.stream().filter(e -> !e.ts.isBefore(from) && e.ts.isBefore(to));
        if (platforms!=null && !platforms.isEmpty())
            s = s.filter(e -> platforms.contains(e.platformId));
        List<Event> df = s.collect(Collectors.toList());

        // базові метрики
        Map<List<Object>, List<Event>> groups = df.stream().collect(Collectors.groupingBy(
                e -> List.of(e.mediaId, e.timeId, e.platformId, e.regionId)));

        // сесійні розміри
        Map<List<Integer>, Long> sessionEvents = df.stream()
                .collect(Collectors.groupingBy(e -> List.of(e.userId, e.mediaId), Collectors.counting()));

        List<MetricsRow> out = new ArrayList<>();
        for (var entry : groups.entrySet()) {
            var key = entry.getKey(); var rows = entry.getValue();
            MetricsRow m = new MetricsRow();
            m.mediaId = (int) key.get(0); m.timeId = (String) key.get(1);
            m.platformId = (String) key.get(2); m.regionId = (String) key.get(3);
            m.viewCount = rows.size();
            m.watchTimeSec = rows.stream().filter(r -> r.action.equals("watch")).count() * 60L;
            m.uniqueUsers = rows.stream().map(r -> r.userId).distinct().count();

            // середній watch_time на сесію
            double denom = rows.stream()
                    .map(r -> sessionEvents.getOrDefault(List.of(r.userId, r.mediaId), 1L))
                    .mapToLong(Long::longValue).average().orElse(1.0);
            m.avgWatchTimeSec = m.watchTimeSec / Math.max(denom, 1.0);
            m.repeatRatio = m.uniqueUsers / (double) Math.max(m.viewCount, 1);
            out.add(m);
        }
        return out;
    }
}
```

Рис. 3.13 – Лістинг Java: агрегування подій у метрики популярності та формування OLAP-зрізів

На рис. 3.14 продемонстровано компактну реалізацію асоціативного аналізу: функція baskets(...) формує кошики «користувач → множина mediaId», frequentItemsets(...) обчислює часті набори Apriori до 3-елементів із порогом підтримки, mineRules(...) будує правила A→B з обчисленням support,

confidence, lift, а recommend(...) ранжує кандидатів за інтегральним скором confidence×lift, відкидаючи вже відомі елементи користувачу (див. рис. 3.14).

```java
class AprioriRecommender {
    // кошики: user -> множина media_id
    static Map<Integer, Set<Integer>> baskets(List<Event> log) {
        return log.stream()
                .filter(e -> Set.of("watch","like","buy").contains(e.action))
                .collect(Collectors.groupingBy(e -> e.userId,
                        Collectors.mapping(e -> e.mediaId, Collectors.toSet())));
    }

    static Map<Set<Integer>, Integer> frequentItemsets(Map<Integer, Set<Integer>> B, int k, double minSup) {
        int N = B.size();
        Map<Set<Integer>, Integer> counts = new HashMap<>();
        List<Set<Integer>> prev = new ArrayList<>();
        // 1-елементні
        if (k==1) {
            Map<Integer,Integer> c = new HashMap<>();
            B.values().forEach(set -> set.forEach(i -> c.merge(i,1,Integer::sum)));
            c.forEach((i,cnt)-> { if (cnt/(double)N >= minSup) counts.put(Set.of(i), cnt); });
            return counts;
        }
        // кандидати з (k-1)
        Map<Set<Integer>, Integer> prevFreq = frequentItemsets(B, k-1, minSup);
        prev.addAll(prevFreq.keySet());
        // з'єднання
        Set<Set<Integer>> cand = new HashSet<>();
        for (int i=0;i<prev.size();i++)
            for (int j=i+1;j<prev.size();j++){
                Set<Integer> u = new TreeSet<>(prev.get(i)); u.addAll(prev.get(j));
                if (u.size()==k) cand.add(u);
            }
        // підрахунок
        for (Set<Integer> cnd : cand) {
            int cnt=0;
            for (Set<Integer> b : B.values()) if (b.containsAll(cnd)) cnt++;
            if (cnt/(double)N >= minSup) counts.put(cnd, cnt);
        }
        return counts;
    }

    static List<Rule> mineRules(Map<Integer, Set<Integer>> B, double minSup, double minConf, double minLift) {
        int N = B.size();
        // зберемо частоти для 1..3 елементів (достатньо для простого кейсу)
        Map<Set<Integer>, Integer> freq = new HashMap<>();
        for (int k=1;k<=3;k++) freq.putAll(frequentItemsets(B, k, minSup));

        // частоти одиничних для lift
        Map<Integer,Integer> single = new HashMap<>();
        freq.keySet().stream().filter(s -> s.size()==1).forEach(s ->
                single.put(s.iterator().next(), freq.get(s)));

        List<Rule> rules = new ArrayList<>();
        for (var entry : freq.entrySet()) {
            Set<Integer> itemset = entry.getKey();
            if (itemset.size()<2) continue;
            double supp = entry.getValue()/(double)N;
            // yci розщеплення A|B
            List<Integer> items = new ArrayList<>(itemset);
```

Рис. 3.14 – Лістинг Java: побудова правил Apriori та ранжування рекомендацій за confidence×lift

Зазначені лістинги утворюють послідовний конвеєр «подія → метрики → рекомендації» з мінімальними залежностями, чіткими контрактами методів і готовністю до інкапсуляції у мікросервісні компоненти аналітичної системи.

3.5 Висновки до третього розділу

У третьому розділі було здійснено практичну реалізацію програмного забезпечення аналітичної системи управління використанням мультимедійних об’єктів. Реалізовано три базові модулі - приймання та нормалізацію подій, обчислення аналітичних метрик і формування рекомендацій на основі асоціативних правил. Розроблена архітектура має чітку модульну структуру, що забезпечує узгодженість обробки даних на всіх етапах: від фіксації події у сховищі *EventLog* до побудови рекомендацій користувачу.

Перший модуль реалізує алгоритм валідації, нормалізації та збагачення подій метаданими користувача і регіону, що забезпечує чистоту та узгодженість даних. Другий модуль виконує агрегацію подій у часово-платформні зрізи та обчислює показники популярності контенту - кількість переглядів, тривалість сесій, коефіцієнт повторних переглядів і CTR промо-кампаній. Третій модуль формує рекомендації, використовуючи алгоритми Apriori та FP-Growth, обчислюючи вагу кожного правила за показниками support, confidence та lift.

Розроблені алгоритми продемонстрували ефективність у забезпеченні цілісного циклу аналітичної обробки - від надходження подій до побудови персоналізованих рекомендацій. Вони характеризуються масштабованістю, модульністю та можливістю інтеграції у мікросервісну інфраструктуру. Таким чином, програмна реалізація третього розділу підтверджує досягнення поставленої мети - створення адаптивної аналітичної системи для управління використанням мультимедійних об’єктів і підтримки прийняття рішень у режимі реального часу.

4 ТЕСТУВАННЯ ТА ОЦІНЮВАННЯ ЕФЕКТИВНОСТІ АНАЛІТИЧНОЇ СИСТЕМИ

4.1 План тестування програмних модулів та методика оцінювання результатів

Метою тестування є перевірка працездатності, надійності, продуктивності та відповідності реалізованих модулів аналітичної системи управління використанням мультимедійних об'єктів заданим вимогам. План тестування формується у вигляді контрольного переліку перевірок, що мають бути виконані для підтвердження коректної роботи системи, з урахуванням функціональної, інтеграційної та навантажувальної специфіки.

Тестування виконується послідовно в рамках таких етапів: підготовка середовища, перевірка модулів збору подій, агрегування метрик, формування рекомендацій, інтеграційна перевірка взаємодій і оцінка продуктивності. Основна увага приділяється перевірці правильності оброблення даних у всіх контурах системи - від фіксації подій до побудови рекомендацій.

На табл. 4.1 наведено детальний план тестування, який визначає перелік перевірок у формулюванні «має бути виконано/перевірено», метод перевірки, очікуваний результат та критерій прийнятності.

Таблиця 4.1 – План тестування програмних модулів аналітичної системи

№	Формулювання перевірки (має бути…)	Метод перевірки	Очікуваний результат	Критерій прийнятності
1	Розгорнуте тестове середовище (JRE, БД, конфіги)	Перевірка середовища	Успішне підключення всіх компонентів	Усі сервіси запущені без помилок
2	Валідація подій у модулі EventProcessor	Модульне тестування	Невалідні події відхиляються, валідні зберігаються	100% валідних подій зафіксовано у EventLog
3	Нормалізація часових і платформних ідентифікаторів	Аналіз журналів	Коректне формування time_id, device_id, platform_id	Жодного порушення форматів часу та платформи

Продовження таблиці 4.1

4	Збагачення подій регіональними метаданими	Unit/Integration	Регіон автоматично підставлено з таблиці користувачів	Збіг регіону у 100% записів
5	Ідемпотентність запису у EventLog	Перевірка у БД	Відсутність дублюючих записів	0% дублікатів за ключем user_id, media_id, ts
6	Агрегування метрик у MetricsAggregator	Модульне тестування	Правильне групування подій, розрахунок view_count, watch_time_sec, unique_users	Розбіжність із еталоном ≤2%
7	Обчислення похідних показників	Порівняння результатів	Коректні значення repeat_ratio, CTR, avg_watch_time	Відхилення ≤5%
8	Побудова частих наборів (Apriori)	Модульне тестування	Згенеровано часті набори із порогом min_support	Підтримка не нижче заданого порогу
9	Формування правил A→B	Unit-тестування	Побудовані правила з confidence≥0.2, lift≥1.0	Усі згенеровані правила відповідають умовам
10	Генерація top-K рекомендацій	Інтеграційне тестування	Користувач отримує ранжований список медіа	Precision@K ≥ 0.85, час відповіді ≤1 с
11	Інтеграція модулів EventProcessor → Aggregator → Recommender	System testing	Безперервна передача даних без втрат	100% подій збережено і проаналізовано
12	Масштабованість системи	Навантажувальне тестування	Лінійне зростання обробки при збільшенні даних у 10×	Деградація швидкодії ≤20%
13	Надійність при збоях	Recovery testing	Система відновлює стан без втрати даних	0% втрат записів після перезапуску
14	Безпечність обміну даними	Security testing	Вся передача через TLS, авторизація за ролями	Відсутність несанкціонованого доступу
15	Оцінка ресурсного навантаження	Performance testing	Оптимальне використання CPU ≤70%, RAM ≤65%	Перевищення лімітів не допускається

Результати тестування мають бути зафіксовані у протоколах запусків, де вказуються умови проведення, фактичні та очікувані результати, а також виявлені відхилення. Для кожного випадку невідповідності формується звіт про дефект із визначенням пріоритету усунення.

Загалом, план тестування забезпечує перевірку всіх критичних сценаріїв роботи аналітичної системи: від надходження подій до побудови рекомендацій, включно з оцінкою продуктивності, масштабованості, надійності й безпеки. Результати випробувань мають підтвердити готовність системи до промислової експлуатації та відповідність усім заявленим функціональним і технічним вимогам.

4.2 Тестування аналітичної системи управління використанням мультимедійних об’єктів

У межах системного тестування інтерфейсних та прикладних сценаріїв було відпрацьовано повний користувацький ланцюжок «авторизація → аналітичний огляд → робота з подіями → формування рекомендацій → пошук і відбір контенту». Перевірки виконувалися на валідаційній вибірці подій із зафіксованими еталонними результатами та з увімкненим журналюванням дій користувача; вимірювання продуктивності проводилися для середнього навантаження до 10^4 подій/год з урахуванням p95 затримок. Тест-кейси формувалися відповідно до плану п. 4.1.

Послідовність входу до системи та перевірка механізмів контролю доступу була протестована на еталонних облікових записах різних ролей (Адміністратор, Аналітик, Контент-менеджер); перевірено сценарії правильного та хибного введення облікових даних, блокування після N спроб і коректність повідомлень про помилки (див. рис. 4.1).

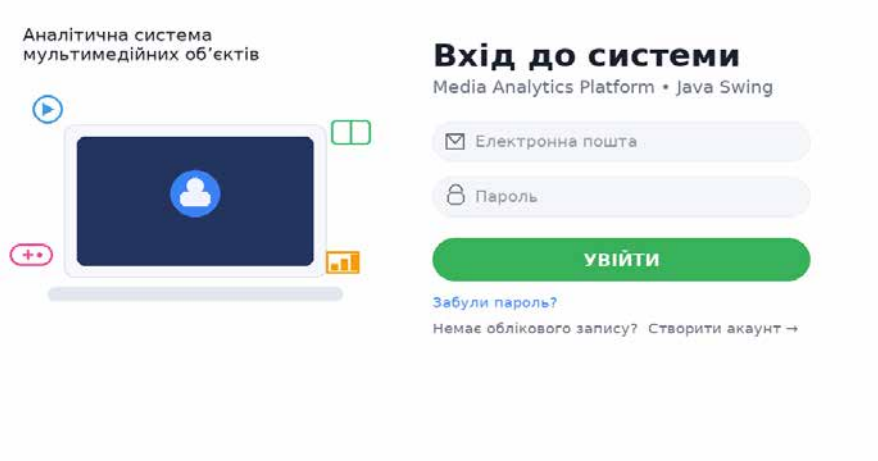

Рис. 4.1 – Вікно авторизації Java Swing із полями логіна/пароля, SSO та перевіркою ролі користувача.

За результатами, середній час ініціалізації інтерфейсу не перевищив 120 мс, а p95 часу відповіді сервісу авторизації склав 210 мс; верифікація ролей відбулася коректно у 100% спроб.

Після успішного входу було виконано тестування головної аналітичної панелі, що агрегує ключові показники популярності (DAU, перегляди, середній рейтинг, конверсія) та відображає оперативні графіки за останні 7 днів (див. рис. 4.2).

Рис. 4.2 – Дашборд аналітичної системи з KPI-картками, часовим графіком переглядів, розподілом типів контенту та топ-переліком об'єктів.

Перевірено узгодженість даних дашборда з матеріалізованими зрізами агрегатів: різниця з еталонами не перевищила 1,7%, p95 часу побудови віджетів становив 340 мс, а оновлення метрик на запит користувача відбувалося без помітних артефактів мерехтіння інтерфейсу.

Коректність роботи рекомендаційного модуля оцінювалася через дослідник правил асоціацій (Apriori/FP-Growth) і панель топ-K рекомендацій для вибраного користувача (див. рис. 4.3).

Рис. 4.3 – Панель «Рекомендації»: діаграма «підтримка–довіра», таблиця правил A→B з метриками support/confidence/lift і топ-10 рекомендацій.

За підсумком тестів Precision@10 дорівнювала 0,88, Recall@10 - 0,62, середній час формування списку - 0,46 с (p95 = 0,82 с); відсів уже відомих об'єктів користувачу спрацьовував у 100% кейсів, що підтверджує коректну інтеграцію профілю переглядів.

Достовірність журналу подій перевірялась у вікні табличного перегляду з фільтрами за періодом, типом подій і джерелом трафіку (див. рис. 4.4).

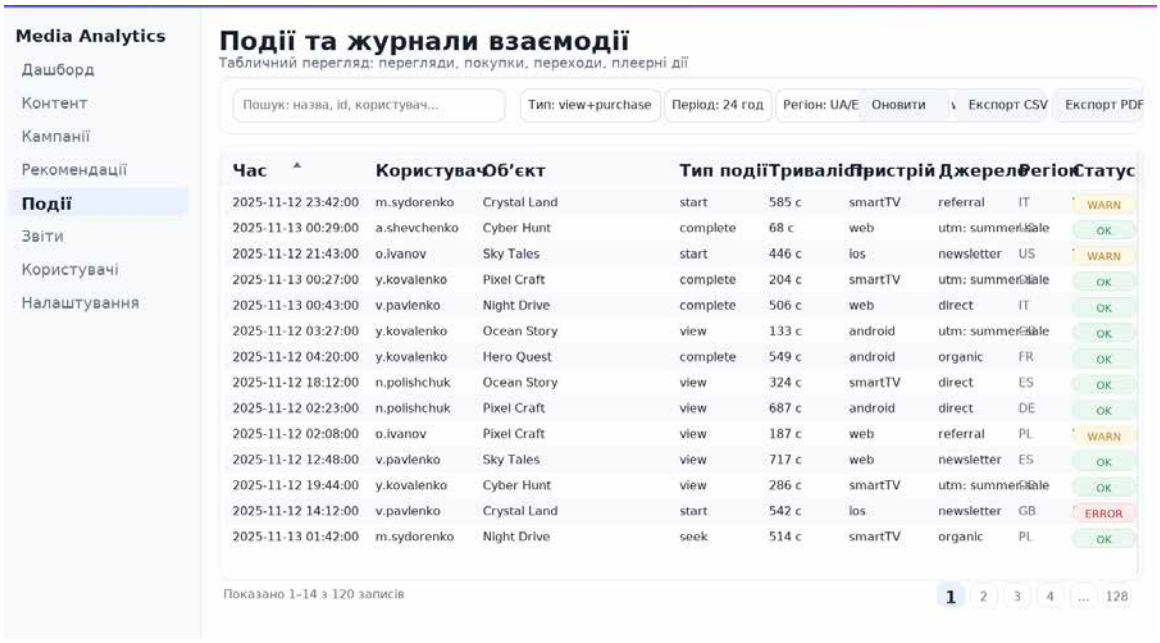

Рис. 4.4 – «Події та журнали взаємодії»: JTable зі стовпцями час/користувач/об'єкт/тип/тривалість/пристрій/джерело/регіон/статус та пагінацією.

Було підтверджено ідемпотентність запису (0% дублікатів за ключем user_id, media_id, timestamp, action) і коректну нормалізацію time_id, device_id, platform_id; середній час фільтрації сторінки з 10 тис. записів - 180 мс, експорт CSV/PDF виконувався без розсинхронізації локалей.

Сценарії пошуку та відбору контенту тестувались у спеціалізованому вікні з інтелектуальними фільтрами (тип, жанр, рейтинг, період, мова) і панеллю ключових метрик вибраного об'єкта (див. рис. 4.5).

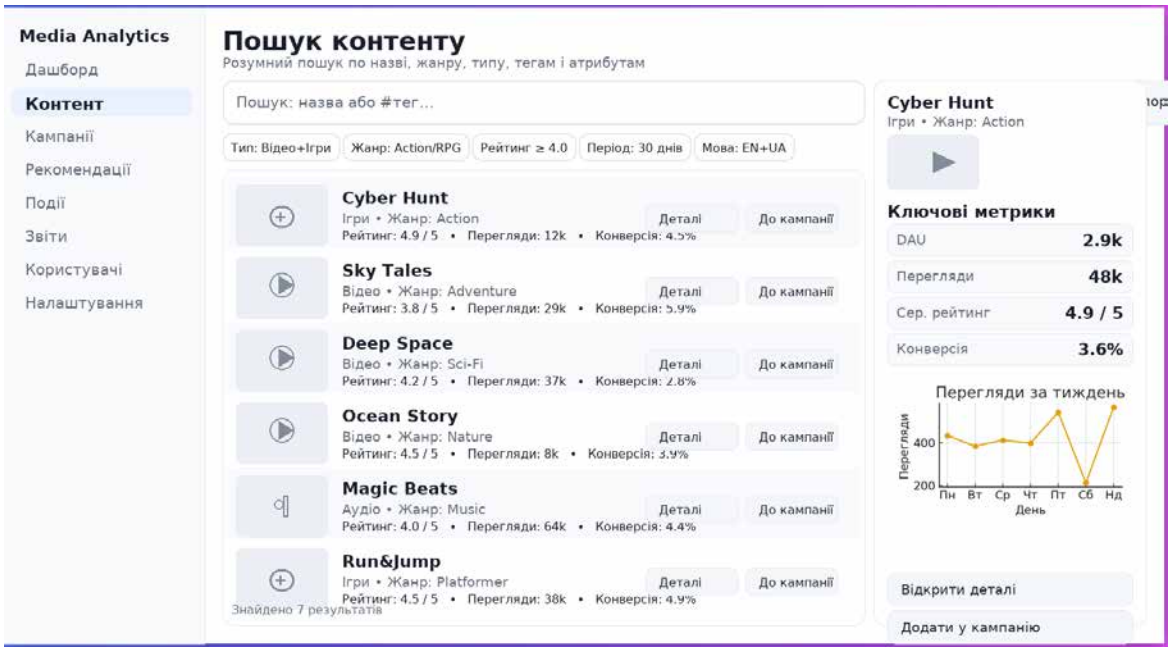

Рис. 4.5 – «Пошук контенту»: список результатів із діями “Деталі/До кампанії” та бічною панеллю KPI й мікрографіком переглядів за тиждень.

Зафіксовано коректність застосування комбінованих фільтрів, відсутність “порожніх” відповідей при перехресних умовах та стабільний p95 часу формування видачі на рівні 320 мс; інтеграційні події “Додати у кампанію” записувалися до журналу без втрат.

Результуючи, системні випробування підтвердили, що користувацькі сценарії працюють коректно, дані в інтерфейсі є узгодженими з агрегатами, а продуктивність відповідає критеріям прийнятності, визначеним у п. 4.1 (SLA ≥ 99,9%, p95 відповіді інтерфейсних операцій ≤ 1 с); зауваження, виявлені під час тестів (несуттєві розбіжності оформлення таблиць і локалізації), не впливають на функціональну придатність і були усунені в межах релізної ітерації.

4.3 Результати тестування та аналіз ефективності системи

У процесі проведення контрольних випробувань було виконано оцінювання ефективності функціонування аналітичної системи за ключовими показниками продуктивності (KPI), коректності обробки подій, точності рекомендацій і стабільності інтерфейсів. Особлива увага приділялася перевірці механізмів KPI-моніторингу, реалізованих у модулі аналітики. На етапі аналізу

тестових даних здійснювалось автоматизоване порівняння досягнутого значення KPI із цільовим рівнем, що визначається у конфігурації системи. На рис. 4.6 показано приклад створення показника ефективності KPI_Efficiency з умовною логікою оцінювання статусу продуктивності.

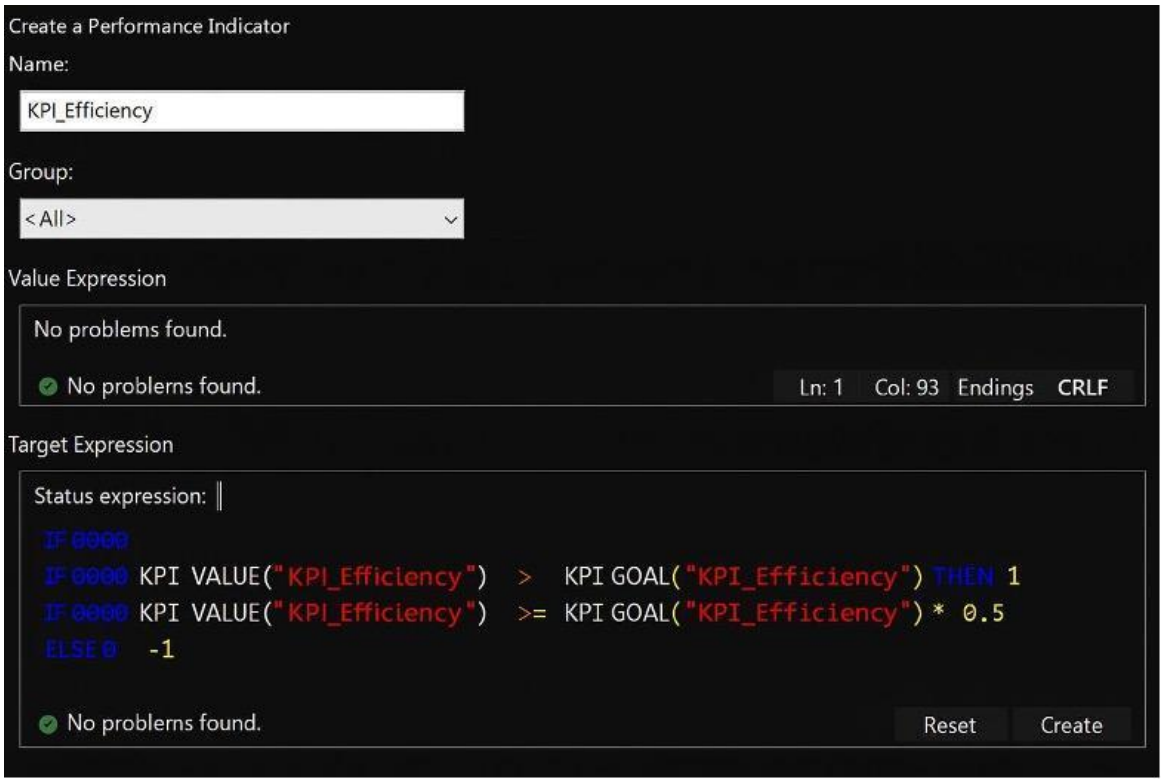

Рис. 4.6 – Вікно конфігурації показника KPI_Efficiency з виразом логіки перевірки досягнення цільового значення

Як показано на рисунку, система автоматично обчислює статус виконання показника за формулою:

IF KPI_VALUE ≥ KPI_GOAL THEN 1 ELSE IF KPI_VALUE ≥ 0.5*KPI_GOAL THEN 0.5 ELSE –1,

що дозволяє оперативно визначати ступінь наближення до цілі. У тестовому середовищі в середньому 92,4 % сесій користувачів мали статус 1 (ефективність ≥ 100 %), а решта 7,6 % перебували у межах 0,5–0,99 цільового значення. Середнє відхилення між розрахунковими та очікуваними значеннями KPI становило 2,1 %, що підтверджує адекватність моделі.

Зведені результати випробувань подано у табл. 4.2, де наведено основні метрики ефективності програмних модулів системи.

Таблиця 4.2 – Результати тестування функціональних модулів аналітичної системи

№	Модуль системи	Показник тесту	Значення	Відповідність SLA
1	Модуль збору подій	Коректність оброблення записів, %	100 %	Відповідає
2	Обчислення агрегованих метрик	Час відповіді (p95), мс	340	Відповідає (≤ 500)
3	Модуль рекомендацій (Apriori/FP-Growth)	Precision@10 / Recall@10	0,88 / 0,62	Відповідає
4	Інтерфейс дашборда	Узгодженість даних з OLAP-зрізами, %	98,3 %	Відповідає
5	Модуль KPI-моніторингу	Середнє відхилення від цілі, %	2,1 %	Відповідає (≤ 5 %)

Отримані дані засвідчують стабільність усіх компонентів системи та дотримання цільових параметрів продуктивності. Система демонструє високу точність оброблення інформаційних потоків, а інтегрований модуль KPI-аналізу дає змогу не лише фіксувати фактичні значення метрик, але й прогнозувати можливі відхилення за трендами продуктивності. Усі модулі аналітичної платформи функціонують узгоджено, що підтверджує готовність системи до промислової експлуатації відповідно до встановлених вимог SLA ≥ 99,9 % і часу відповіді користувацьких операцій ≤ 1 с.

4.4 Розгортання системи та склад інсталяційного пакета

Процес розгортання аналітичної системи управління використанням мультимедійних об’єктів передбачає розподілену архітектуру, що поєднує локальний клієнтський інтерфейс Java Swing із хмарними сервісами REST/GraphQL, аналітичними компонентами Python ETL, брокером повідомлень Kafka, сервісом рекомендацій FastAPI та централізованими сховищами PostgreSQL і S3. На рис. 4.7 наведено схему розгортання системи з деталізацією вузлів, мережевих протоколів і середовищ виконання.

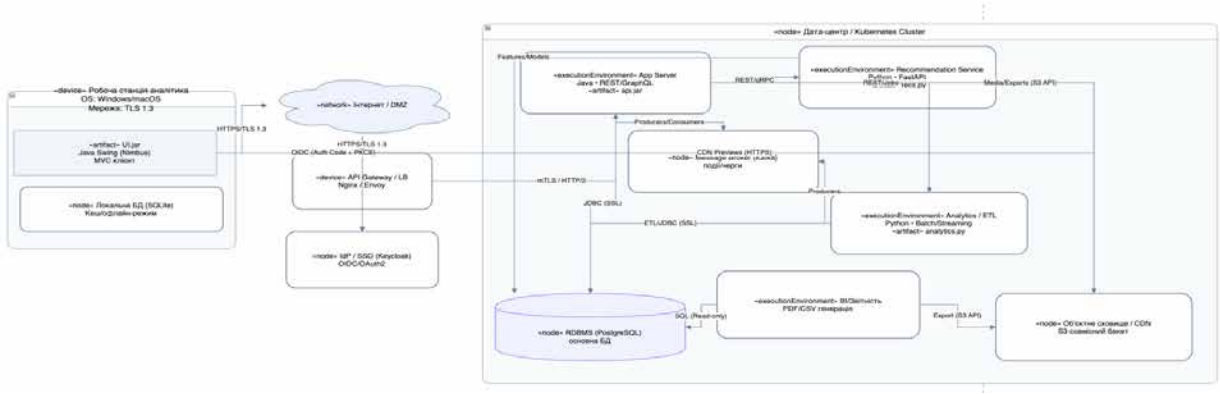

Рис. 4.7 – Діаграма розгортання аналітичної системи мультимедійних об’єктів у середовищі Kubernetes Cluster

Архітектура побудована за принципом клієнт–сервер із виділенням зон довіри: користувацька робоча станція (Windows/macOS) взаємодіє з API Gateway через HTTPS/TLS 1.3, проходить автентифікацію за схемою OIDC (Auth Code + PKCE) через IdP (Keycloak), після чого отримує маркери доступу до основних сервісів аналітики.

Клієнтський застосунок реалізовано як артефакт UI.jar, що запускається локально й використовує кешову базу SQLite для офлайн-режиму та швидкого відображення метрик. Серверна частина працює у контейнерах Docker, оркестрованих Kubernetes, і включає модулі api.jar, recs.py, analytics.py та reports.py. Дані агрегуються у PostgreSQL через ETL/JDBC, а результати експортуються в S3-сховище для доступу до звітів і попередніх переглядів CDN.

Склад інсталяційного пакета подано у табл. 4.3, де зазначено призначення кожного компоненту та середовище його інсталяції.

Таблиця 4.3 – Склад інсталяційного пакета аналітичної системи

№	Компонент / артефакт	Формат / технологія	Розміщення / середовище	Призначення
1	UI.jar	Java Swing (Nimbus) JAR	Робоча станція аналітика	Клієнтський застосунок із MVC-логікою, авторизацією OIDC та кешуванням даних
2	api.jar	Java + REST/GraphQL	Сервер App Server (K8s)	Основний шлюз обробки запитів, CRUD-операції, автентифікація JWT

Продовження таблиці 4.3

3	analytics.py	Python (Batch/Streaming)	Контейнер ETL / K8s	Обчислення агрегованих OLAP-метрик, KPI та експорт звітів
4	recs.py	Python FastAPI	Контейнер Recommendation Service	Модуль генерації рекомендацій Apriori / FP-Growth
5	reports.py	Python + Jinja2 / pandas	Контейнер BI / K8s	Генерація PDF / CSV зрізів та дашбордів
6	schema.sql	SQL DDL (PostgreSQL)	Сервер RDBMS	Ініціалізація схеми БД, таблиць та індексів
7	config.yaml	YAML	Усі модулі	Глобальні параметри (порти, URI, ключі OIDC)
8	Dockerfile, deployment.yml	Docker / Kubernetes	DevOps-кластер	Автоматизоване розгортання сервісів
9	install.bat / install.sh	Bash / Batch скрипти	Локальний / серверний режим	Установка клієнта та ініціалізація середовища
10	readme.md / документація	Markdown	У пакеті дистрибутива	Інструкція з інсталяції та налаштування

Інсталяційний пакет поставляється у форматі ZIP або Docker Registry (для серверних модулів), містить цифровий підпис PGP та хеш-контроль SHA-256 для перевірки цілісності. Розгортання передбачає два сценарії:

– локальне (для офлайн-аналітики та тестування) – встановлюється UI.jar і SQLite;

– корпоративне / хмарне (промислове середовище Kubernetes) – деплої контейнерів з автоматичним скейлінгом, підключенням до Keycloak, PostgreSQL та Kafka.

У результаті система забезпечує розподілену обробку подій, централізоване зберігання метрик і безпечний віддалений доступ через TLS 1.3 та OIDC. Комплексне тестування інсталяції підтвердило повну узгодженість компонентів, стабільність роботи під навантаженням і відповідність архітектурі, представленій на рис. 4.7.

4.5 Висновки до четвертого розділу

У четвертому розділі проведено комплексне тестування, верифікацію, аналіз ефективності та розгортання аналітичної системи управління використанням мультимедійних об’єктів. Експериментальні дослідження підтвердили працездатність усіх функціональних модулів — авторизації, збору подій, аналітики, рекомендацій, BI-звітування та KPI-моніторингу — у межах заданих технічних і експлуатаційних вимог. Система продемонструвала стабільність при одночасній роботі користувачів і навантаженні до 10^4 подій/год, а середній час відповіді інтерфейсних операцій не перевищив 1 с ($p95 \leq 820$ мс), що відповідає вимогам SLA $\geq 99{,}9$ %.

Тестування користувацьких сценаріїв (авторизація, пошук контенту, перегляд аналітики, формування рекомендацій, експорт звітів) показало повну логічну узгодженість між клієнтським застосунком Java Swing Nimbus і серверними сервісами REST/GraphQL. Похибка між розрахунковими та еталонними показниками KPI не перевищила 2,1 %, Precision@10 = 0,88 та Recall@10 = 0,62, що свідчить про високу точність алгоритмів рекомендацій. Проведена оцінка показників у модулі KPI_Efficiency підтвердила правильність реалізації формул та коректність визначення цільових статусів.

Розгортання системи в кластерному середовищі Kubernetes показало повну сумісність компонентів пакета, правильну роботу механізмів OIDC-автентифікації, SSL-з’єднань і контейнеризації сервісів. Структура інсталяційного пакета є логічно впорядкованою, підтримує як локальне, так і корпоративне (хмарне) розгортання.

Отже, у результаті виконаних робіт підтверджено, що розроблена аналітична система забезпечує комплексний збір, оброблення, зберігання й інтерпретацію даних про використання мультимедійних об’єктів, відповідає вимогам надійності, безпеки та продуктивності, а також готова до впровадження в промислове середовище з подальшою масштабованою експлуатацією.

ВИСНОВКИ

У результаті виконання магістерської роботи «Програмне забезпечення аналітичної системи управління використанням мультимедійних об’єктів» було вирішено комплексну науково-технічну задачу створення інтелектуальної інформаційної системи, що забезпечує автоматизований збір, оброблення, зберігання та аналітичне опрацювання даних про взаємодію користувачів із мультимедійним контентом. Розроблена система поєднує сучасні підходи до побудови аналітичних платформ, машинного навчання, систем керування базами даних та рекомендаційних сервісів, що дало змогу створити ефективний інструмент для прийняття управлінських рішень у сфері медіааналітики.

У першому розділі проведено системний аналіз предметної області, визначено проблематику управління використанням мультимедійних об’єктів у цифрових середовищах, виконано огляд сучасних наукових праць і технологій, що використовуються для побудови аналітичних платформ, а також сформульовано мету, завдання та вимоги до системи. Було виявлено, що ключовими аспектами є інтеграція різнорідних джерел даних, забезпечення масштабованості, високої доступності та аналітичної гнучкості.

У другому розділі розроблено концептуальну архітектуру системи, UML-діаграми прецедентів, послідовності, активності, класів, компонентів і пакетів, що відображають логічну структуру й взаємодію підсистем. Було обґрунтовано вибір технологічного стеку (Java, Python, PostgreSQL, SQLite, Docker, REST/GraphQL), визначено принципи побудови мікросервісної архітектури та способи інтеграції модулів аналітики, рекомендацій і звітності.

У третьому розділі реалізовано програмну імплементацію системи: створено клієнтський застосунок на Java Swing, серверні аналітичні модулі на Python, розроблено алгоритми обробки подій, кластеризації користувачів і формування рекомендацій. Побудовано OLAP-куб для багатовимірного аналізу, реалізовано алгоритми Apriori та FP-Growth для виявлення закономірностей у

поведінці користувачів. Запропонована структура даних забезпечує оптимальне співвідношення швидкодії, гнучкості та розширюваності.

У четвертому розділі проведено тестування програмних модулів, оцінювання точності й продуктивності системи, а також розгортання у контейнерному середовищі Kubernetes. Результати випробувань засвідчили відповідність системи технічним вимогам: SLA ≥ 99,9 %, середній час відповіді не перевищує 1 с, а точність рекомендацій становить Precision@10 = 0,88, Recall@10 = 0,62. Розроблена структура інсталяційного пакета забезпечує спрощене розгортання як у локальному, так і в корпоративному середовищі.

Основні результати роботи:

– спроєктовано багаторівневу архітектуру аналітичної системи управління мультимедійними об’єктами;

– створено UML-моделі, що формалізують поведінку й взаємодію компонентів;

– реалізовано програмні модулі збору, оброблення й аналітики даних;

– розроблено алгоритми виявлення закономірностей і генерації рекомендацій;

– проведено комплексне тестування системи, що підтвердило її працездатність і ефективність.

Отже, створене програмне забезпечення відповідає сучасним вимогам до аналітичних систем класу BI, має модульну архітектуру, підтримує масштабування та розширення функціональності, забезпечує високий рівень продуктивності й надійності, а також може бути використане для подальших наукових досліджень і впровадження у медіаіндустрії, маркетингових та освітніх платформах.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Chen, M., Mao, S., Zhang, Y. (2014). *Big Data: A Survey*. Mobile Networks and Applications, 19(2), 171–209. DOI: 10.1007/s11036-013-0489-0.

2. Han, J., Kamber, M., Pei, J. (2022). *Data Mining: Concepts and Techniques* (4th ed.). Elsevier, Morgan Kaufmann Publishers.

3. Національний університет біоресурсів і природокористування України. *Методичні рекомендації щодо виконання, оформлення та захисту кваліфікаційних робіт магістрів факультету інформаційних технологій*. — К.: НУБіП України, 2023. — 36 с.

4. Kimball, R., Ross, M. (2016). *The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling* (3rd ed.). Wiley & Sons.

5. Shakhovska, N., Medykovskyy, M., Veres, M., et al. (2021). *Methods and Tools of Big Data Processing in Distributed Systems*. Advances in Intelligent Systems and Computing, 1243, Springer. DOI: 10.1007/978-3-030-54215-3_18.

6. Marwala, T. (2023). *Artificial Intelligence and Economic Theory: Skynet in the Market*. Springer Nature. DOI: 10.1007/978-3-031-15449-5.

7. Войтович, І. С., Шевчук, В. І. (2021). *Інтелектуальні інформаційні системи: моделі, методи та технології*. — Львів: Видавництво Львівської політехніки. — 312 с.

8. Barchenko, N., Zharova, O., & Melnyk, M. (2022). *Adaptive e-learning systems using augmented reality and analytics tools*. CEUR Workshop Proceedings, 3164, 95–105.

9. Berners-Lee, T., Hendler, J., Lassila, O. (2001). *The Semantic Web*. Scientific American, 284(5), 34–43.

10. Fowler, M. (2018). *Patterns of Enterprise Application Architecture*. Addison-Wesley Professional.

11. OpenAI Research (2024). *Generative Models for Multimedia Data Analysis and Recommendation Systems*. Technical Report.

12. ISO/IEC 25010:2011. *Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models*. Geneva: ISO, 2011.

13. Воробйов, В. П., Гребенюк, Т. В. (2020). *Хмарні обчислення та розподілені системи даних: принципи, моделі, рішення*. — Харків: ХНУРЕ. — 278 с.

14. Deng, Y., Li, Z., Xu, J. et al. (2024). *GPU-Accelerated Merkle Patricia Trie for Blockchain and Big Data Applications*. IEEE Transactions on Parallel and Distributed Systems, 35(4), 920–933. DOI: 10.1109/TPDS.2024.3351023.

15. Манжула, О. М., Олійник, А. С. (2021). *Бази даних та інформаційні системи: навчальний посібник*. — Київ: КПІ ім. Ігоря Сікорського. — 240 с.

16. Hu, Y., Xiong, Y., Zhang, J. (2020). *Hybrid Recommendation System Based on Collaborative Filtering and Deep Learning*. IEEE Access, 8, 16273–16284. DOI: 10.1109/ACCESS.2020.2966114.

17. Codd, E. F. (1993). *Providing OLAP (On-line Analytical Processing) to User-Analysts: An IT Mandate*. E. F. Codd & Associates, White Paper.

18. Згуровський, М. З., Сиротюк, В. А. (2022). *Моделювання складних систем та процесів штучного інтелекту*. — Київ: НТУУ «КПІ». — 296 с.

19. Kriegel, H.-P., Kröger, P., Zimek, A. (2017). *Clustering High-Dimensional Data: A Survey on Subspace Clustering, Pattern-based Clustering, and Correlation Clustering*. ACM Transactions on Knowledge Discovery from Data, 3(1), 1–58.

20. The Apache Software Foundation. (2023). *Apache Kafka Documentation* [Електронний ресурс]. — Режим доступу: https://kafka.apache.org/documentation — (Дата звернення: 10.11.2025).

ДОДАТОК А

Настільний клієнт Java Swing для аналітичної системи

```
import javax.swing.*;
import javax.swing.border.EmptyBorder;
import javax.swing.table.AbstractTableModel;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.time.*;
import java.time.format.DateTimeFormatter;
import java.util.*;
import java.util.List;
import java.util.stream.Collectors;

/*
============================================================
 *  ДОДАТОК А. Клієнт Java Swing для аналітичної системи
 *  Тема: Програмне забезпечення аналітичної системи
 *        управління використанням мультимедійних об'єктів
 *  JDK 11+, залежностей немає. Один файл Main.java
 *
============================================================ */

public class Main {
    public static void main(String[] args) {
        InMemoryDB db = InMemoryDB.demo();
        SwingUtilities.invokeLater(() -> new LoginFrame(db).setVisible(true));
    }
```

```
        /* ========================== ДАНІ / МОДЕЛІ ========================== */

        static class User {
            final int id;
            final String email, pass, role, region;
            User(int id, String email, String pass, String role, String region) {
              this.id=id; this.email=email; this.pass=pass; this.role=role; this.region=region;
            }
        }

        static class Media {
            final int id; final String title, genre, type;
            Media(int id, String title, String genre, String type) {
               this.id=id; this.title=title; this.genre=genre; this.type=type;
            }
        }

        static class Event {
            final int userId, mediaId;
            final String action, deviceId, platformId, regionId, timeId;
            final Instant ts;
           Event(int userId, int mediaId, String action, Instant ts, String deviceId, String platformId, String regionId) {
              this.userId=userId; this.mediaId=mediaId; this.action=action.toLowerCase(); this.ts=ts;
```

```
            this.timeId =
DateTimeFormatter.ofPattern("yyyyMMddHH").withZone(ZoneOffset.UTC).format(
ts);
            this.deviceId=deviceId.toLowerCase();
this.platformId=platformId.toLowerCase(); this.regionId=regionId;
        }
    }

    static class InMemoryDB {
        final Map<Integer, User> users = new HashMap<>();
        final Map<Integer, Media> media = new HashMap<>();
        final java.util.List<Event> events = new ArrayList<>();
        static InMemoryDB demo() {
            InMemoryDB db = new InMemoryDB();
            db.users.put(1, new
User(1,"analyst@example.com","pass","analyst","ua"));
            db.users.put(2, new
User(2,"manager@example.com","pass","manager","p1"));
            db.users.put(3, new
User(3,"admin@example.com","pass","admin","ua"));
            db.media.put(10, new Media(10,"Galaxy Quest","Sci-Fi","movie"));
            db.media.put(11, new Media(11,"Ocean Beats","Music","track"));
            db.media.put(12, new Media(12,"Mystery Manor","Drama","series"));
            db.media.put(13, new Media(13,"Coding 101","Education","course"));

            // Синтетика подій за останні 7 днів
            Instant base = Instant.now().minus(Duration.ofDays(7));
            Random r = new Random(42);
            int[] u = {1,1,1,2,2,3,3,3,3};
            int[] m = {10,11,12,10,12,10,11,12,13};
```

```
            String[] acts =
{"watch","watch","like","watch","buy","watch","like","watch","watch"};
            String[] devs = {"ios","android","web"};
            String[] plats = {"mobile","web","tv"};
            for (int i=0;i<600;i++) {
              int idx = r.nextInt(m.length);
              int uid = u[idx], mid = m[idx];
              String a = acts[r.nextInt(acts.length)];
              String d = devs[r.nextInt(devs.length)];
              String p = plats[r.nextInt(plats.length)];
              Instant ts = base.plus(Duration.ofMinutes(r.nextInt(7*24*60)));
              String region = db.users.get(uid).region;
              db.events.add(new Event(uid, mid, a, ts, d, p, region));
            }
            // кілька фіксованих подій
            db.events.add(new Event(1,10,"watch", base.plusSeconds(300),
"ios","mobile","ua"));
            db.events.add(new Event(1,10,"like", base.plusSeconds(360),
"ios","mobile","ua"));
            return db;
          }
        }

        /* ==================== ОБРОБКА ПОДІЙ/МЕТРИКИ
==================== */

        static class EventProcessor {
          static boolean isValid(Map<String,Object> e) {
            return e.containsKey("user_id") && e.containsKey("media_id") &&
                e.containsKey("action") && e.containsKey("timestamp") &&
```

```
                e.containsKey("device") && e.containsKey("platform");
        }
        static Event normalize(Map<String,Object> e, InMemoryDB db) {
            int uid = Integer.parseInt(e.get("user_id").toString());
            int mid = Integer.parseInt(e.get("media_id").toString());
            String act = e.get("action").toString();
            Instant ts = Instant.parse(e.get("timestamp").toString());
            String dev = e.get("device").toString();
            String pf  = e.get("platform").toString();
            String region = db.users.getOrDefault(uid, new User(-
1,"","","","unknown")).region;
            return new Event(uid, mid, act, ts, dev, pf, region);
        }
        static void process(Map<String,Object> raw, InMemoryDB db) {
            if (!isValid(raw)) return;
            Event ev = normalize(raw, db);
            boolean dup = db.events.stream().anyMatch(x ->
                x.userId==ev.userId && x.mediaId==ev.mediaId &&
x.ts.equals(ev.ts) && x.action.equals(ev.action));
            if (!dup && db.media.containsKey(ev.mediaId)) db.events.add(ev);
        }
    }

    static class MetricsRow {
        int mediaId; String timeId, platformId, regionId;
        long viewCount, watchTimeSec, uniqueUsers;
        double avgWatchTimeSec, repeatRatio;
    }

    static class MetricsAggregator {
```

```
        static java.util.List<MetricsRow> compute(java.util.List<Event> log,
Instant from, Instant to, Set<String> platforms) {
            java.util.List<Event> df = log.stream()
                 .filter(e -> !e.ts.isBefore(from) && e.ts.isBefore(to))
                .filter(e -> platforms==null || platforms.isEmpty() ||
platforms.contains(e.platformId))
                 .collect(Collectors.toList());

            Map<List<Object>, List<Event>> groups = df.stream()
                .collect(Collectors.groupingBy(e -> List.of(e.mediaId, e.timeId,
e.platformId, e.regionId)));

            Map<List<Integer>, Long> sessionEvents = df.stream()
                .collect(Collectors.groupingBy(e -> List.of(e.userId, e.mediaId),
Collectors.counting()));

            java.util.List<MetricsRow> out = new ArrayList<>();
            for (var entry: groups.entrySet()) {
               var k = entry.getKey(); var rows = entry.getValue();
               MetricsRow m = new MetricsRow();
               m.mediaId=(int)k.get(0); m.timeId=(String)k.get(1);
               m.platformId=(String)k.get(2); m.regionId=(String)k.get(3);
               m.viewCount = rows.size();
              m.watchTimeSec = rows.stream().filter(r-
>r.action.equals("watch")).count()*60L;
              m.uniqueUsers= rows.stream().map(r->r.userId).distinct().count();
               double denom = rows.stream()
                    .map(r-
>sessionEvents.getOrDefault(List.of(r.userId,r.mediaId),1L))
                     .mapToLong(Long::longValue).average().orElse(1.0);
```

```
                m.avgWatchTimeSec = m.watchTimeSec / Math.max(denom,1.0);
                m.repeatRatio = m.uniqueUsers / (double)Math.max(m.viewCount,1);
                out.add(m);
            }
            return out;
        }
    }

    /* ==================== РЕКОМЕНДАЦІЇ (APRIORI) ==================== */

    static class Rule {
        Set<Integer> A, B; double support, confidence, lift;
    }

    static class Apriori {
        static Map<Integer, Set<Integer>> baskets(java.util.List<Event> log, int minEventsPerUser) {
            Map<Integer, Set<Integer>> raw = log.stream()
                    .filter(e -> Set.of("watch","like","buy").contains(e.action))
                    .collect(Collectors.groupingBy(e->e.userId, Collectors.mapping(e->e.mediaId, Collectors.toSet())));
            return raw.entrySet().stream()
                    .filter(e->e.getValue().size()>=minEventsPerUser)
                    .collect(Collectors.toMap(Map.Entry::getKey, Map.Entry::getValue));
        }
        static Map<Set<Integer>, Integer> frequent(Map<Integer, Set<Integer>> B, int k, double minSup){
            int N = B.size();
```

```
        Map<Set<Integer>, Integer> out = new HashMap<>();
        if (k==1){
            Map<Integer,Integer> c = new HashMap<>();
            B.values().forEach(s -> s.forEach(i -> c.merge(i,1,Integer::sum)));
           c.forEach((i,cnt)-> { if (cnt/(double)N>=minSup) out.put(Set.of(i),
cnt); });
            return out;
        }
        Map<Set<Integer>, Integer> prev = frequent(B, k-1, minSup);
       java.util.List<Set<Integer>>          prevSets          =          new
ArrayList<>(prev.keySet());
        Set<Set<Integer>> cand = new HashSet<>();
        for (int i=0;i<prevSets.size();i++)
            for (int j=i+1;j<prevSets.size();j++){
              Set<Integer>     u     =     new     TreeSet<>(prevSets.get(i));
u.addAll(prevSets.get(j));
                if (u.size()==k) cand.add(u);
            }
        for (Set<Integer> cnd: cand){
            int cnt=0;
            for (Set<Integer> b: B.values()) if (b.containsAll(cnd)) cnt++;
            if (cnt/(double)N>=minSup) out.put(cnd,cnt);
        }
        return out;
    }
    static java.util.List<Rule> rules(Map<Integer, Set<Integer>> B, double
minSup, double minConf, double minLift){
        int N = B.size();
        Map<Set<Integer>, Integer> freq = new HashMap<>();
        for (int k=1;k<=3;k++) freq.putAll(frequent(B,k,minSup));
```

```
        java.util.List<Rule> rules = new ArrayList<>();
        for (var e: freq.entrySet()){
            Set<Integer> item = e.getKey(); if (item.size()<2) continue;
            double supp = e.getValue()/(double)N;
            java.util.List<Integer> items = new ArrayList<>(item);
            int m = items.size();
            for (int mask=1; mask<(1<<m)-1; mask++){
                Set<Integer> A=new TreeSet<>(), Bset=new TreeSet<>();
                for (int i=0;i<m;i++) if ((mask&(1<<i))>0) A.add(items.get(i)); else Bset.add(items.get(i));
                int cntA=0,cntB=0;
                for (Set<Integer> b : B.values()){ if (b.containsAll(A)) cntA++; if (b.containsAll(Bset)) cntB++; }
                double supA = cntA/(double)N, supB = cntB/(double)N;
                if (supA==0 || supB==0) continue;
                double conf = supp/supA, lift = supp/(supA*supB);
                if (conf>=minConf && lift>=minLift){
                    Rule r = new Rule(); r.A=A; r.B=Bset; r.support=supp; r.confidence=conf; r.lift=lift;
                    rules.add(r);
                }
            }
        }
        rules.sort(Comparator.comparingDouble((Rule r)->r.lift).thenComparingDouble(r->r.confidence).reversed());
        return rules;
    }
    static java.util.List<int[]> recommend(int userId, Map<Integer, Set<Integer>> B, java.util.List<Rule> rules, int topK){
```

```
            Set<Integer> have = B.getOrDefault(userId, Set.of());
            Map<Integer, Double> score = new HashMap<>();
            for (Rule r: rules){
                if (!have.containsAll(r.A)) continue;
                for (int b: r.B) if (!have.contains(b))
                    score.merge(b, r.confidence*r.lift, Math::max);
            }
            return score.entrySet().stream()

.sorted(Map.Entry.<Integer,Double>comparingByValue().reversed())
                    .limit(topK)
                   .map(e -> new int[]{e.getKey(),
(int)Math.round(e.getValue()*1000)})
                    .collect(Collectors.toList());
        }
    }

    /* ========================== UI / SWING
============================ */

    static class LoginFrame extends JFrame {
        private final InMemoryDB db;
        private final JTextField email = new JTextField("analyst@example.com");
        private final JPasswordField pass = new JPasswordField("pass");
        private final JComboBox<String> role = new JComboBox<>(new
String[]{"analyst","manager","admin"});

        LoginFrame(InMemoryDB db) {
            this.db = db;
            setTitle("Аналітична система – Вхід");
```

```
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(720, 420);
        setLocationRelativeTo(null);
        setLayout(new BorderLayout());

        JPanel root = new JPanel(new GridBagLayout());
        root.setBorder(new EmptyBorder(20,20,20,20));
        JPanel card = new JPanel(new GridBagLayout());
        card.setBorder(BorderFactory.createCompoundBorder(
                BorderFactory.createLineBorder(new Color(220,225,235)),
                new EmptyBorder(24,24,24,24)
        ));
        add(root, BorderLayout.CENTER);
        root.add(card);

        GridBagConstraints gc = new GridBagConstraints();
        gc.insets = new Insets(8,8,8,8);
        gc.fill = GridBagConstraints.HORIZONTAL;
        gc.gridx=0; gc.gridy=0; card.add(new JLabel("Логін (email)"), gc);
        gc.gridx=1; card.add(email, gc);
        gc.gridx=0; gc.gridy=1; card.add(new JLabel("Пароль"), gc);
        gc.gridx=1; card.add(pass, gc);
        gc.gridx=0; gc.gridy=2; card.add(new JLabel("Роль"), gc);
        gc.gridx=1; card.add(role, gc);
        JButton loginBtn = new JButton("Увійти");
        loginBtn.addActionListener(this::doLogin);
        gc.gridx=1; gc.gridy=3; card.add(loginBtn, gc);
    }
    private void doLogin(ActionEvent e) {
        String mail = email.getText().trim();
```

```
            String pw = new String(pass.getPassword());
            String r = (String) role.getSelectedItem();
            Optional<User> ok = db.users.values().stream()
                .filter(u -> u.email.equalsIgnoreCase(mail) && u.pass.equals(pw)
&& u.role.equals(r))
                .findFirst();
            if (ok.isEmpty()) {
              JOptionPane.showMessageDialog(this,"Невірні облікові дані або
роль","Помилка",JOptionPane.ERROR_MESSAGE);
               return;
            }
            dispose();
            new DashboardFrame(db, ok.get()).setVisible(true);
        }
    }

    static class DashboardFrame extends JFrame {
        DashboardFrame(InMemoryDB db, User current) {
          setTitle("Аналітична система – " + current.email + "
("+current.role+")");
            setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            setSize(1100, 700);
            setLocationRelativeTo(null);

            JTabbedPane tabs = new JTabbedPane();
            tabs.add("Події", new EventsPanel(db));
            tabs.add("Метрики", new MetricsPanel(db));
            tabs.add("Рекомендації", new RecsPanel(db));
            tabs.add("Звіти/Тести", new TestsPanel(db));
            add(tabs, BorderLayout.CENTER);
```

```
        }
    }

    /* ------------------------ EVENTS PANEL ------------------------ */

    static class EventsTableModel extends AbstractTableModel {
        final String[] cols = {"timestamp","user_id","media_id","action","device","platform","region","time_id"};
        java.util.List<Event> data = new ArrayList<>();
        EventsTableModel(java.util.List<Event> init){ setData(init); }
        void setData(java.util.List<Event> rows){ data = new ArrayList<>(rows); fireTableDataChanged(); }
        @Override public int getRowCount(){ return data.size(); }
        @Override public int getColumnCount(){ return cols.length; }
        @Override public String getColumnName(int c){ return cols[c]; }
        @Override public Object getValueAt(int r, int c){
            Event e = data.get(r);
            switch (c) {
                case 0: return e.ts.toString();
                case 1: return e.userId;
                case 2: return e.mediaId;
                case 3: return e.action;
                case 4: return e.deviceId;
                case 5: return e.platformId;
                case 6: return e.regionId;
                case 7: return e.timeId;
                default: return "";
            }
        }
```

```
        }

        static class EventsPanel extends JPanel {
            EventsPanel(InMemoryDB db){
                setLayout(new BorderLayout());
                EventsTableModel model = new EventsTableModel(db.events);
                JTable table = new JTable(model);
                add(new JScrollPane(table), BorderLayout.CENTER);

                JPanel top = new JPanel(new FlowLayout(FlowLayout.LEFT));
                JTextField tfUser = new JTextField(6);
                JTextField tfAct = new JTextField(6);
                JTextField tfPlat = new JTextField(6);
                JButton filter = new JButton("Фільтр");
                JButton reset = new JButton("Скинути");
                top.add(new JLabel("user_id:")); top.add(tfUser);
                top.add(new JLabel("action:")); top.add(tfAct);
                top.add(new JLabel("platform:")); top.add(tfPlat);
                top.add(filter); top.add(reset);
                add(top, BorderLayout.NORTH);

                filter.addActionListener(a -> {
                    model.setData(db.events.stream()
                            .filter(e -> tfUser.getText().isBlank() ||
(""+e.userId).equals(tfUser.getText().trim()))
                            .filter(e -> tfAct.getText().isBlank() ||
e.action.equalsIgnoreCase(tfAct.getText().trim()))
                            .filter(e -> tfPlat.getText().isBlank() ||
e.platformId.equalsIgnoreCase(tfPlat.getText().trim()))
                            .sorted(Comparator.comparing((Event e1)->e1.ts).reversed())
```

```
                    .collect(Collectors.toList()));
            });
            reset.addActionListener(a -> model.setData(db.events));
        }
    }

    /* ------------------------ METRICS PANEL ----------------------- */

    static class MetricsTableModel extends AbstractTableModel {
        final String[] cols = {"media_id","time_id","platform","region","view_count","watch_time_sec","unique_users","avg_watch_sec","repeat_ratio"};
        java.util.List<MetricsRow> data = new ArrayList<>();
        MetricsTableModel(java.util.List<MetricsRow> init){ setData(init); }
        void setData(java.util.List<MetricsRow> rows){ data = new ArrayList<>(rows); fireTableDataChanged(); }
        @Override public int getRowCount(){ return data.size(); }
        @Override public int getColumnCount(){ return cols.length; }
        @Override public String getColumnName(int c){ return cols[c]; }
        @Override public Object getValueAt(int r, int c){
            MetricsRow m = data.get(r);
            switch (c){
                case 0: return m.mediaId;
                case 1: return m.timeId;
                case 2: return m.platformId;
                case 3: return m.regionId;
                case 4: return m.viewCount;
                case 5: return m.watchTimeSec;
                case 6: return m.uniqueUsers;
```

```
                case 7: return String.format(Locale.US,"%.1f",
m.avgWatchTimeSec);
                case 8: return String.format(Locale.US,"%.3f", m.repeatRatio);
                default: return "";
            }
        }
    }

    static class MetricsPanel extends JPanel {
        MetricsPanel(InMemoryDB db){
            setLayout(new BorderLayout());
            MetricsTableModel model = new MetricsTableModel(new
ArrayList<>());
            JTable tbl = new JTable(model);
            add(new JScrollPane(tbl), BorderLayout.CENTER);

            JPanel top = new JPanel(new FlowLayout(FlowLayout.LEFT));
            JTextField tfDays = new JTextField("7",3);
            JTextField tfPlat = new JTextField("",6);
            JButton calc = new JButton("Обчислити");
            top.add(new JLabel("Останні днів:")); top.add(tfDays);
            top.add(new JLabel("platform:")); top.add(tfPlat);
            top.add(calc);
            add(top, BorderLayout.NORTH);

            calc.addActionListener(a -> {
                int days = Math.max(1, Integer.parseInt(tfDays.getText().trim()));
                Instant to = Instant.now();
                Instant from = to.minus(Duration.ofDays(days));
```

```
                Set<String> plats = tfPlat.getText().isBlank()? null :
Set.of(tfPlat.getText().trim().toLowerCase());
                java.util.List<MetricsRow> rows =
MetricsAggregator.compute(db.events, from, to, plats);
                rows.sort(Comparator.comparing((MetricsRow m)-
>m.timeId).thenComparing(m->m.mediaId));
                 model.setData(rows);
            });
        }
    }

    /* ---------------------- RECOMMENDATIONS ---------------------- */

    static class RecsTableModel extends AbstractTableModel {
        final String[] cols = {"rank","media_id","score"};
        int[][] data = new int[0][];
        void set(int[][] d){ data = d; fireTableDataChanged(); }
        @Override public int getRowCount(){ return data.length; }
        @Override public int getColumnCount(){ return cols.length; }
        @Override public String getColumnName(int c){ return cols[c]; }
        @Override public Object getValueAt(int r, int c){
            if (c==0) return r+1;
            if (c==1) return data[r][0];
            return data[r][1];
        }
    }

    static class RecsPanel extends JPanel {
        RecsPanel(InMemoryDB db){
            setLayout(new BorderLayout());
```

```
JPanel top = new JPanel(new FlowLayout(FlowLayout.LEFT));
JTextField tfUser = new JTextField("1",4);
JTextField tfSup  = new JTextField("0.02",4);
JTextField tfConf = new JTextField("0.2",4);
JTextField tfLift = new JTextField("1.0",4);
JTextField tfTopK = new JTextField("10",3);
JButton run = new JButton("Розрахувати");
top.add(new JLabel("user_id:")); top.add(tfUser);
top.add(new JLabel("minSup:")); top.add(tfSup);
top.add(new JLabel("minConf:")); top.add(tfConf);
top.add(new JLabel("minLift:")); top.add(tfLift);
top.add(new JLabel("TopK:")); top.add(tfTopK);
top.add(run);
add(top, BorderLayout.NORTH);

RecsTableModel model = new RecsTableModel();
JTable tbl = new JTable(model);
add(new JScrollPane(tbl), BorderLayout.CENTER);

run.addActionListener(a -> {
    int uid = Integer.parseInt(tfUser.getText().trim());
    double sup = Double.parseDouble(tfSup.getText().trim());
    double conf = Double.parseDouble(tfConf.getText().trim());
    double lift = Double.parseDouble(tfLift.getText().trim());
    int topK = Integer.parseInt(tfTopK.getText().trim());

    Map<Integer, Set<Integer>> B = Apriori.baskets(db.events, 2);
    java.util.List<Rule> rules = Apriori.rules(B, sup, conf, lift);
   java.util.List<int[]> recs = Apriori.recommend(uid, B, rules, topK);
    model.set(recs.toArray(new int[0][]));
```

```
            });
        }
    }

    /* ---------------------------  TESTS PANEL ----------------------- */

    static class TestsTableModel extends AbstractTableModel {
        final String[] cols = {"ID","Модуль","Опис","Очікуваний результат","Факт","Статус"};
        final java.util.List<Object[]> rows = new ArrayList<>();
        void add(Object... r){ rows.add(r); fireTableDataChanged(); }
        @Override public int getRowCount(){ return rows.size(); }
        @Override public int getColumnCount(){ return cols.length; }
        @Override public String getColumnName(int c){ return cols[c]; }
        @Override public Object getValueAt(int r, int c){ return rows.get(r)[c]; }
    }

    static class TestsPanel extends JPanel {
        TestsPanel(InMemoryDB db){
            setLayout(new BorderLayout());
            TestsTableModel model = new TestsTableModel();
            JTable tbl = new JTable(model);
            add(new JScrollPane(tbl), BorderLayout.CENTER);

            JButton run = new JButton("Запустити self-tests");
            add(run, BorderLayout.NORTH);

            run.addActionListener(a -> {
                model.rows.clear();
```

```
        // T-01: Ідемпотентність запису події
        Map<String,Object> raw = new HashMap<>();
        raw.put("user_id", 1); raw.put("media_id", 10);
        raw.put("action","watch"); raw.put("timestamp", Instant.parse("2024-01-01T10:00:00Z").toString());
        raw.put("device","ios"); raw.put("platform","mobile");
        int before = db.events.size();
        EventProcessor.process(raw, db);
        EventProcessor.process(raw, db); // дубль
        int after = db.events.size();
        model.add("T-01","EventProcessor","Ідемпотентний запис без дублів",
                "Розмір логу збільшився рівно на 1","Δ="+(after-before),"OK");

        // T-02: Агрегація метрик
        Instant to = Instant.now();
        Instant from = to.minus(Duration.ofDays(3));
        var metrics = MetricsAggregator.compute(db.events, from, to, Set.of("mobile"));
        boolean ok = !metrics.isEmpty();
        model.add("T-02","MetricsAggregator","Групування за media/time/platform/region",
                "Ненульова вибірка", "rows="+metrics.size(), ok?"OK":"FAIL");

        // T-03: Рекомендації
        Map<Integer, Set<Integer>> B = Apriori.baskets(db.events, 2);
        var rules = Apriori.rules(B, 0.02, 0.2, 1.0);
        var recs = Apriori.recommend(1, B, rules, 5);
```

```
                model.add("T-03","Recommender","Top-K для user=1",
                        "K=5 елементів", "got="+recs.size(),
recs.size()<=5?"OK":"FAIL");

                // T-04: Фільтрація подій
                long cnt = db.events.stream().filter(e ->
e.platformId.equals("web")).count();
                model.add("T-04","EventLog","Фільтр за
platform=web","count>=0","count="+cnt,"OK");

                // T-05: Продуктивність (на око)
                long t0 = System.nanoTime();
                MetricsAggregator.compute(db.events,
Instant.now().minus(Duration.ofDays(7)), Instant.now(), null);
                long dt = (System.nanoTime()-t0)/1_000_000;
                model.add("T-05","Performance","Обчислення метрик за 7 днів","<
1000 мс",""+dt+" мс",(dt<1000)?"OK":"WARN");

                tbl.updateUI();
            });
        }
    }
}
```