

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ
І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ**

Факультет інформаційних технологій

ПОГОДЖЕНО
Декан факультету

_____ **Ігор БОЛБОТ**
(підпис)

«___» _____ 2026 р.

ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ
Завідувач кафедри комп'ютерних наук

_____ **Белла ГОЛУБ**
(підпис)

«___» _____ 2026 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Програмне забезпечення тем контенту для соціальних мереж»

Спеціальність «121 Інженерія програмного забезпечення»

Освітньо-професійна програма «Інженерія програмного забезпечення»

Гарант освітньої програми

к.т.н., доцент

_____ (підпис)

Ганна ВАЙГАНГ

**Керівник бакалаврської
кваліфікаційної роботи**

_____ (підпис)

Олексій СТЕПАНОВ

Виконав

_____ (підпис)

Єгор ІОХИМЕНКО

Київ-2026

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ
І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ**

Факультет інформаційних технологій

ЗАТВЕРДЖУЮ

Завідувач кафедри комп'ютерних наук

К.Т.Н., доцент _____

(підпис)

Белла ГОЛУБ

«___» _____ 2025 р.

**ЗАВДАННЯ
ДО ВИКОНАННЯ БАКАЛАВРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ
ЗДОБУВАЧУ**

Юхименку Єгору Ігоровичу

(прізвище, ім'я, по батькові)

Спеціальність «121 Інженерія програмного забезпечення»

Освітньо-професійна програма «Інженерія програмного забезпечення»

Тема бакалаврської кваліфікаційної роботи

«Програмне забезпечення тем контенту для соціальних мереж»

затверджена наказом від 19 грудня 2025 р. № 3204 "С" та зміною відповідно до наказу від «16» квітня 2026 р. № 1051 «С»

Термін подання завершеної роботи на кафедру «22» травня 2026 р.

Вихідні дані до бакалаврської кваліфікаційної роботи (проєкту): науково-технічна література з питань роботи, офіційна документація API соціальних мереж.

Перелік питань, що підлягають дослідженню:

1. Аналіз предметної області та огляд існуючих рішень
2. Проєктування програмного забезпечення
3. Реалізація програмного забезпечення
4. Тестування та оцінювання ефективності програмного забезпечення

Перелік графічних документів (за необхідності).

Дата видачі завдання «19» грудня 2025 р.

**Керівник бакалаврської
кваліфікаційної роботи**

(підпис)

Олексій СТЕПАНОВ

**Завдання взяв до
виконання**

(підпис)

Єгор ЮХИМЕНКО

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ	5
ВСТУП	7
1 СИСТЕМНИЙ АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ	9
1.1 Опис предметної області	9
1.2 Аналіз існуючих рішень	11
1.3 Моделювання предметної області	16
1.4 Аналіз вимог системи	19
1.5 Постановка завдання	23
1.6 Висновки до першого розділу	24
2 ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОГО ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	25
2.1 Логічна модель даних у вигляді ER-діаграми	25
2.2 Діаграма класів і кооперації розроблюваної системи	29
2.3 Діаграма компонентів	30
2.4 Діаграма пакетів	32
2.5 Висновки до другого розділу	33
3 ПРОЄКТУВАННЯ АРХІТЕКТУРИ ПРОГРАМНОЇ СИСТЕМИ ТА АЛГОРИТМІЧНОГО ЗАБЕЗПЕЧЕННЯ	34
3.1 Обґрунтування вибору технологічних засобів	34
3.2 Архітектура системи	37
3.3 Аналітична обробка даних та формування результатів моделювання	38
3.4 Алгоритмізація програмних модулів	44
3.5 Висновки до третього розділу	48
4 ТЕСТУВАННЯ ТА ОЦІНЮВАННЯ ЕФЕКТИВНОСТІ СИСТЕМИ	50
4.1 План тестування програмних модулів та методика оцінювання результатів	50
4.2 Тестування інтерфейсних модулів	53

4.3 Результати тестування, аналіз ефективності та склад інсталяційного пакета	56
4.4 Забезпечення інформаційної безпеки системи	57
4.5 Висновки до четвертого розділу	59
ВИСНОВКИ	60
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	62
Додаток А	64
Додаток Б	66
Додаток В	68

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API – Application Programming Interface, програмний інтерфейс взаємодії між компонентами системи.

ETL – Extract–Transform–Load, процес вилучення, перетворення та завантаження даних.

DWH – Data Warehouse, аналітичне сховище структурованих даних.

NLP – Natural Language Processing, технології комп’ютерного опрацювання природної мови.

ML – Machine Learning, машинне навчання.

KPI – Key Performance Indicator, ключовий показник результативності.

CTR – Click-Through Rate, відношення кількості переходів до кількості показів.

ER – Engagement Rate, показник залученості аудиторії.

JSON – JavaScript Object Notation, формат обміну структурованими даними.

SQL – Structured Query Language, мова роботи з реляційними базами даних.

Redis – високошвидкісне сховище даних у пам’яті для кешування та черг.

REST – Representational State Transfer, архітектурний стиль побудови вебсервісів.

TF-IDF – статистична міра вагомості терміна в документі та корпусі документів.

BERT – трансформерна модель для контекстного подання тексту.

Embedding – векторне подання текстових, поведінкових або тематичних ознак.

Feature Store – централізоване сховище ознак, що використовуються ML-моделями.

Message Broker – програмний посередник для передавання та маршрутизації подій.

JWT – формат токена для передавання автентифікаційної інформації між клієнтом і сервером.

OAuth 2.0 – протокол делегованої авторизації доступу до ресурсів.

RBAC – рольова модель керування доступом.

TLS – криптографічний протокол захисту мережових з'єднань.

OWASP – міжнародна ініціатива з розроблення рекомендацій щодо безпеки вебзастосунків.

XSS – атака, пов'язана з виконанням стороннього сценарію в браузері користувача.

CSRF – атака, за якої користувач неявно виконує небажану дію у вебзастосунку.

ВСТУП

Соціальні мережі стали одним із найдинамічніших середовищ створення, поширення та споживання цифрового контенту. Щодня користувачі переглядають публікації, залишають реакції, коментують матеріали, переходять за посиланнями та поширюють дописи, унаслідок чого формується значний масив поведінкових і текстових даних. Для автора або SMM-фахівця така інформація є цінним джерелом знань про реальні інтереси аудиторії, однак її ручне опрацювання є складним, повільним і суб'єктивним.

Більшість поширених інструментів соціальної аналітики відображає переважно підсумкові показники ефективності: охоплення, кількість переглядів, лайків, коментарів, поширень або переходів. Такі метрики дають змогу оцінити результат уже опублікованого матеріалу, проте не завжди пояснюють, які змістові ознаки вплинули на реакцію аудиторії та які теми доцільно використовувати в наступних публікаціях. Через це виникає потреба в інтелектуальній системі, здатній не лише збирати статистику, а й інтерпретувати семантику контенту, поведінкові патерни користувачів і часову зміну популярності тем.

Актуальність роботи полягає в необхідності автоматизованої підтримки прийняття рішень під час планування тематичного контенту для соціальних платформ. Поєднання методів машинного навчання, оброблення природної мови, тематичного моделювання та кластеризації дає можливість виявляти приховані закономірності у взаємодіях аудиторії, формувати персоналізовані рекомендації для авторів і підвищувати релевантність контент-стратегії.

Метою кваліфікаційної роботи є розроблення рекомендаційної системи тем контенту для авторів соціальних мереж, яка забезпечує автоматизований аналіз інтересів аудиторії, визначення актуальних тематичних напрямів і формування обґрунтованих пропозицій на основі семантичних, поведінкових і статистичних ознак.

Для досягнення поставленої мети необхідно виконати такі **завдання**:

1. проаналізувати предметну область рекомендаційних систем і соціальної аналітики;
2. дослідити існуючі програмні рішення для планування та оцінювання контенту;
3. сформулювати вимоги до розроблюваної системи; побудувати UML- та ER-моделі;
4. обґрунтувати архітектуру програмного забезпечення; визначити алгоритми оброблення даних і формування рекомендацій;
5. виконати тестування функціональних, інтерфейсних і продуктивних характеристик системи.

У роботі використано методи системного аналізу, об'єктно-орієнтованого моделювання, ER-моделювання, статистичного аналізу, машинного навчання та NLP. Для представлення програмної структури застосовано UML-діаграми прецедентів, активності, послідовності, класів, компонентів і пакетів. Для оцінювання результатів використано метрики Precision, Recall, F1-score, CTR, Engagement Rate, NDCG та MRR.

1 СИСТЕМНИЙ АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Опис предметної області

Предметна область роботи охоплює процеси створення, аналізу, оцінювання та планування тематичного контенту в соціальних мережах. У центрі системи знаходиться автор контенту, який потребує обґрунтованої інформації про те, які теми відповідають очікуванням його аудиторії, мають потенціал залучення та можуть забезпечити кращі показники взаємодії. На відміну від звичайного календарного планування, розглянута система працює з фактичними даними про публікації, коментарі, реакції, перегляди, поширення, переходи та тематичні ознаки матеріалів.

Основними джерелами даних є соціальні платформи, з яких через API, вебхуки або потокові канали надходять відомості про публікації, метадані, реакції користувачів і часові ряди активності. Після надходження інформація проходить етап попереднього очищення: вилучаються дублікати, нормалізуються формати часу, перевіряється коректність полів, уніфікуються текстові ознаки та готуються дані для подальшого NLP-аналізу.

Аналітичний рівень системи виконує тематичне моделювання, визначення ключових слів, розпізнавання сутностей, аналіз тональності та побудову векторних подань контенту. На основі цих ознак формується профіль аудиторії, що відображає її інтереси, інтенсивність взаємодій, реакцію на окремі теми та зміну вподобань у часі. Такий профіль використовується рекомендаційним ядром для побудови списку тематик, найбільш придатних для наступних публікацій.

Рекомендаційний механізм реалізує комбінований підхід: контентне фільтрування визначає семантичну подібність між матеріалами та інтересами аудиторії, поведінковий аналіз враховує фактичні взаємодії користувачів, а трендова складова коригує результат відповідно до динаміки популярності тем.

Важливою особливістю системи є надання пояснення до рекомендацій, тобто відображення факторів, які вплинули на вибір певної теми.

Функціональна схема предметної області, наведена на рисунку 1.1, демонструє повний аналітичний контур: від збору даних із соціальних платформ до формування рекомендацій, передавання їх автору та оновлення моделей за результатами зворотного зв’язку.

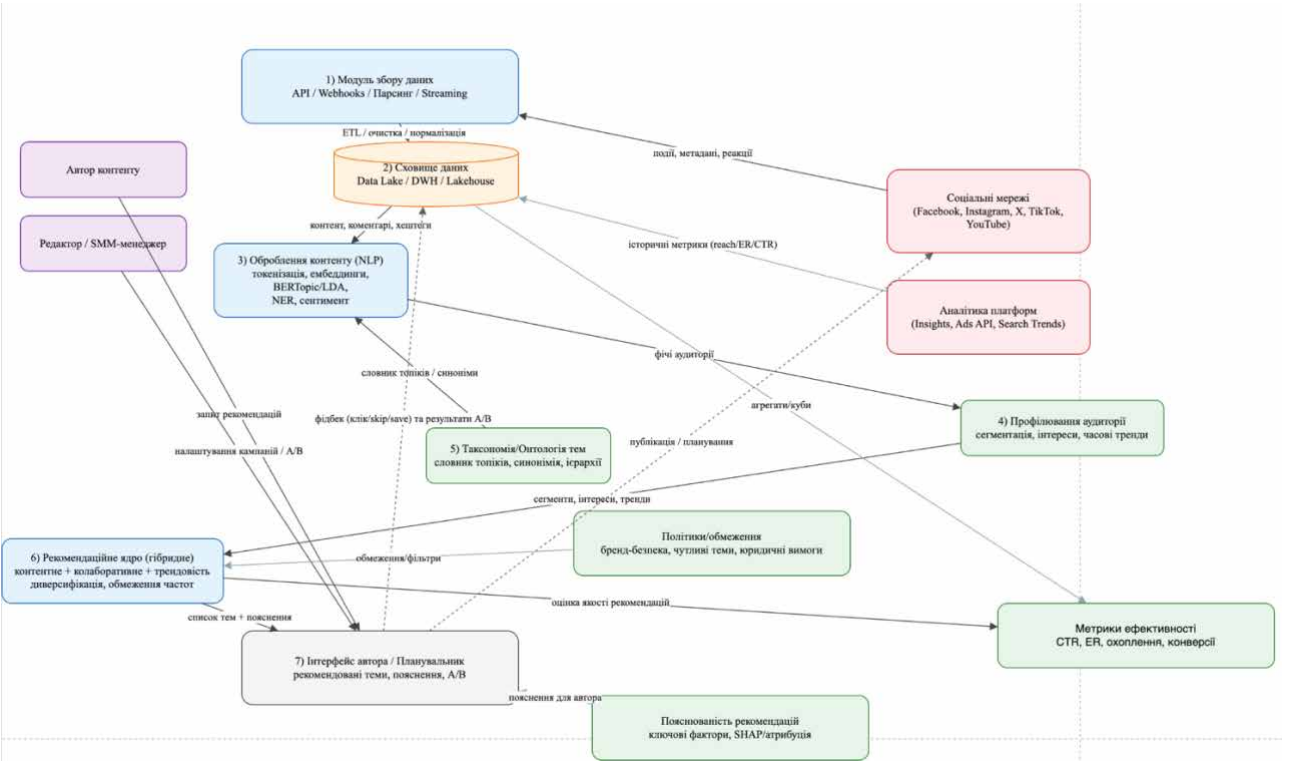


Рис.1.1. Функціональна схема предметної області рекомендаційної системи тем контенту

Основні компоненти представлені у таблиці 1.1.

Таблиця 1.1

Основні компоненти та інформаційні потоки предметної області

№	Компонент	Вхідні дані	Результат роботи	Призначення
1	Модуль збору даних	API, вебхуки, події соцмереж	Необроблені події та метадані	Отримання й маршрутизація потоків даних
2	ETL/сховище	Сирі події, тексти, реакції	Очищені таблиці та історичні зрізи	Нормалізація, дедублікація, зберігання

3	NLP-модуль	Публікації, коментарі, хештеги	Теми, ключові слова, сентимент	Семантичне опрацювання контенту
4	Профіль аудиторії	Історія взаємодій, ознаки тем	Сегменти й інтереси користувачів	Виявлення поведінкових груп
5	Рекомендаційне ядро	Профілі, теми, тренди, фідбек	Ранжований перелік тем	Добір і пояснення рекомендацій
6	Інтерфейс автора	Запити, вибір тем, фідбек	План публікацій, звіти	Візуалізація рекомендацій і KPI

Отже, предметна область є комплексною та поєднує задачі соціальної аналітики, опрацювання природної мови, побудови профілів аудиторії та формування пояснюваних рекомендацій. Використання такого підходу дає змогу перейти від інтуїтивного вибору тем до даних-орієнтованого планування контенту.

1.2 Аналіз існуючих рішень

Для визначення функціональних меж розроблюваної системи проаналізовано поширені платформи, які використовуються для планування публікацій, контент-аналітики та оцінювання взаємодії аудиторії. До порівняння включено BuzzSumo, ContentStudio, SocialBakers/Emplifi, HubSpot Marketing Hub і Canva Content Planner. Ці рішення частково закривають потреби авторів, однак мають різний рівень глибини аналітики та персоналізації.

BuzzSumo зосереджена на пошуку популярного контенту та аналізі його поширення у відкритих джерелах. Перевагою платформи є швидке виявлення матеріалів із високою активністю аудиторії. Водночас система здебільшого працює з агрегованими показниками популярності та не формує індивідуальну модель інтересів конкретної аудиторії автора. Приклад інтерфейсу BuzzSumo для аналізу популярності контенту наведено на рисунку 1.2.

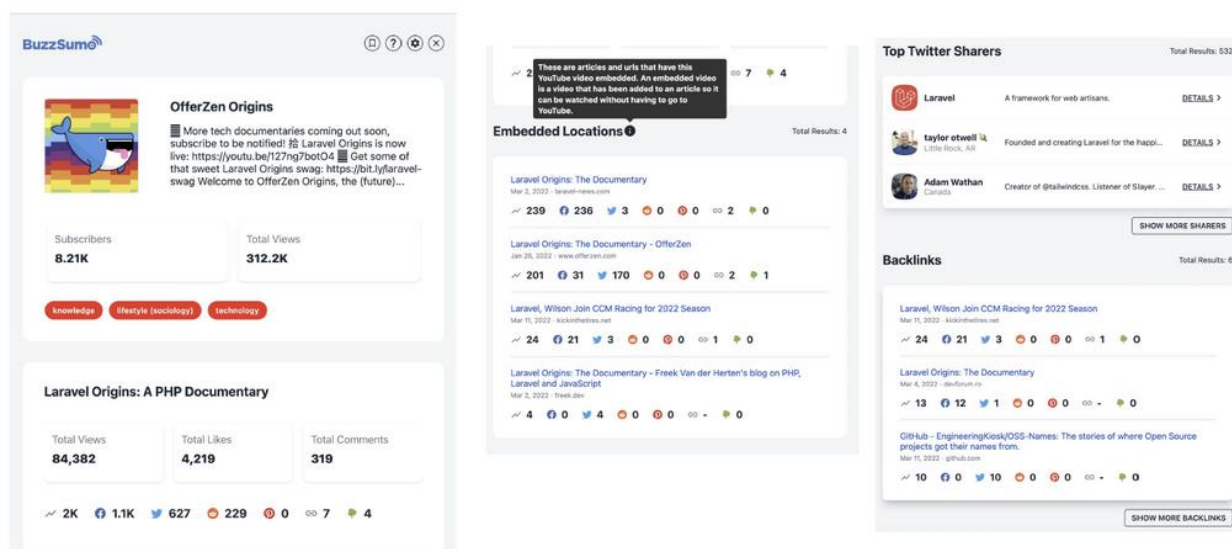


Рис.1.2. Інтерфейс BuzzSumo для аналізу популярності контенту

ContentStudio поєднує календар публікацій, керування соціальними акаунтами та базову аналітику. Такий інструмент є зручним для організації SMM-процесу, планування матеріалів і контролю публікацій у різних каналах. Проте рекомендації в системі ґрунтуються переважно на історичних закономірностях і не враховують поглиблений семантичний аналіз текстів або поведінкову кластеризацію користувачів. Модуль керування соціальними акаунтами у ContentStudio показано на рисунку 1.3.

Платформа SocialBakers/Emplifi орієнтована на сегментацію аудиторії, порівняння конкурентів, моніторинг трендів і аналіз ефективності контенту в соціальних мережах. Її перевагою є наявність розвинених інструментів для вивчення аудиторії та конкурентного середовища. Водночас алгоритми платформи є закритими, а пояснення причин сформованих рекомендацій обмежені, що ускладнює їх використання як прозорої інтелектуальної системи підтримки рішень. Аналітичний модуль сегментації аудиторії в SocialBakers/Emplifi наведено на рисунку 1.4.

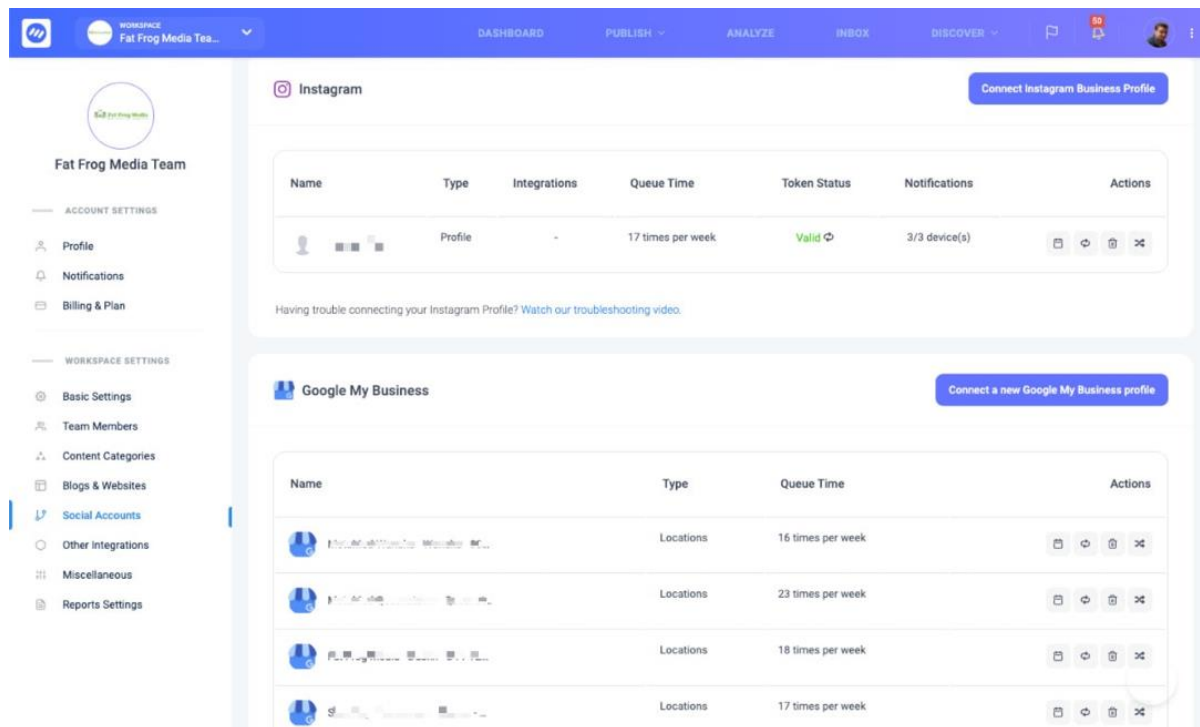


Рис. 1.3. Модуль керування соціальними акаунтами у ContentStudio

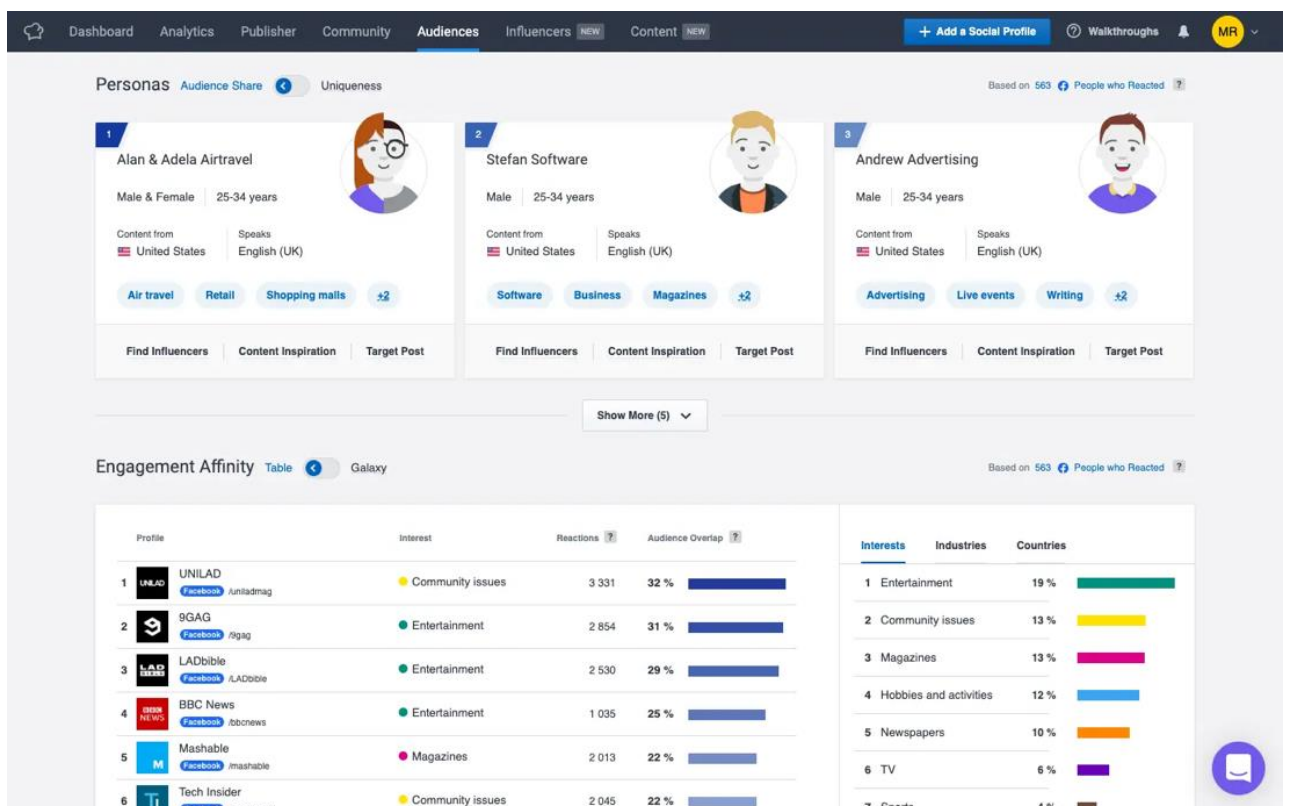


Рис. 1.4. Аналітичний модуль сегментації аудиторії в SocialBakers/Emplifi

HubSpot Marketing Hub орієнтований на керування маркетинговими кампаніями, роботу з лідами, CRM-контуром і автоматизацію комунікацій. Це рішення є корисним для бізнес-аналітики та управління маркетинговими процесами, однак не спеціалізується на інтелектуальному доборі тематик для авторського контенту. Основний акцент у ньому зроблено на маркетинговій воронці, клієнтських даних і результативності кампаній. Аналітичний інтерфейс HubSpot Marketing Hub подано на рисунку 1.5.

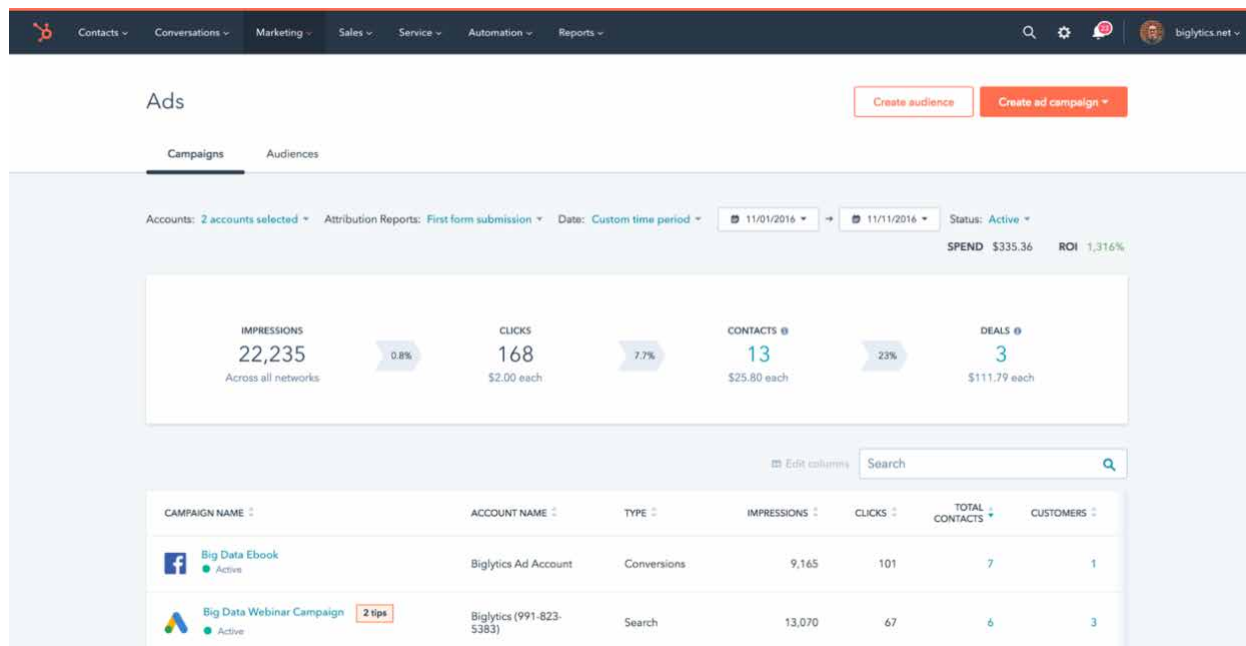


Рис. 1.5. Аналітичний інтерфейс HubSpot Marketing Hub

Canva Content Planner забезпечує просте візуальне планування публікацій, підготовку графічних матеріалів і організацію контент-календаря. Перевагою цього рішення є зручний інтерфейс і тісна інтеграція з інструментами створення дизайну. Однак система не має повноцінного аналітичного ядра, не виконує глибокого аналізу інтересів аудиторії та не формує змістовні рекомендації щодо

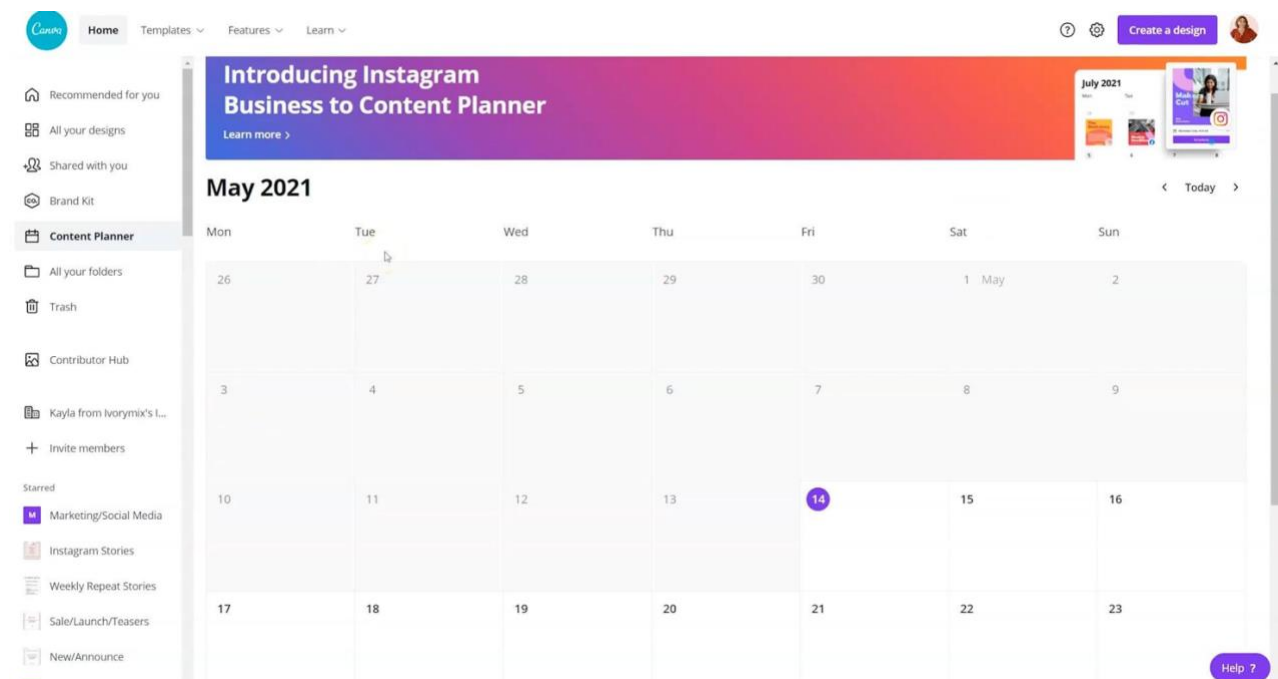
вибору тематики публікацій. Інтерфейс Canva Content Planner наведено на рисунку 1.6.

Рис. 1.6. Інтерфейс Canva Content Planner

Аналіз існуючих рішень представлений у таблиці 1.2

Таблиця 1.2

Порівняння існуючих систем і розроблюваного рішення



Система	Аналітика	Тематичне моделювання	Персоналізація	Пояснення	Особливості
BuzzSumo	За популярністю матеріалів	Відсутнє	Низька	Немає	Моніторинг трендів і конкурентного контенту
ContentStudio	Базова статистика акаунтів	Відсутнє	Обмежена	Немає	Планування публікацій і А/В-сценарії
SocialBakers/Emplifi	Детальна аналітика аудиторії	Частково	Середня	Обмежене	Сегментація, моніторинг брендів і конкурентів
HubSpot Marketing Hub	Маркетингові KPI	Немає	Обмежена	Немає	Інтеграція з CRM і рекламними кампаніями

Canva Content Planner	Мінімальна	Немає	Відсутня	Немає	Візуальний календар і шаблони публікацій
Розроблювана система	Кластеризація, тренди, KPI	LDA/BERTopіс/векторні ознаки	Висока	Є	Пояснюваний добір тем на основі аудиторних інтересів

Порівняння показує, що наявні рішення або зосереджені на плануванні, або на відображенні статистики, але не забезпечують повного циклу: збір даних, семантичний аналіз, профілювання аудиторії, ранжування тем і пояснення результату. Саме ця прогалина визначає доцільність розроблення власної рекомендаційної системи.

1.3 Моделювання предметної області

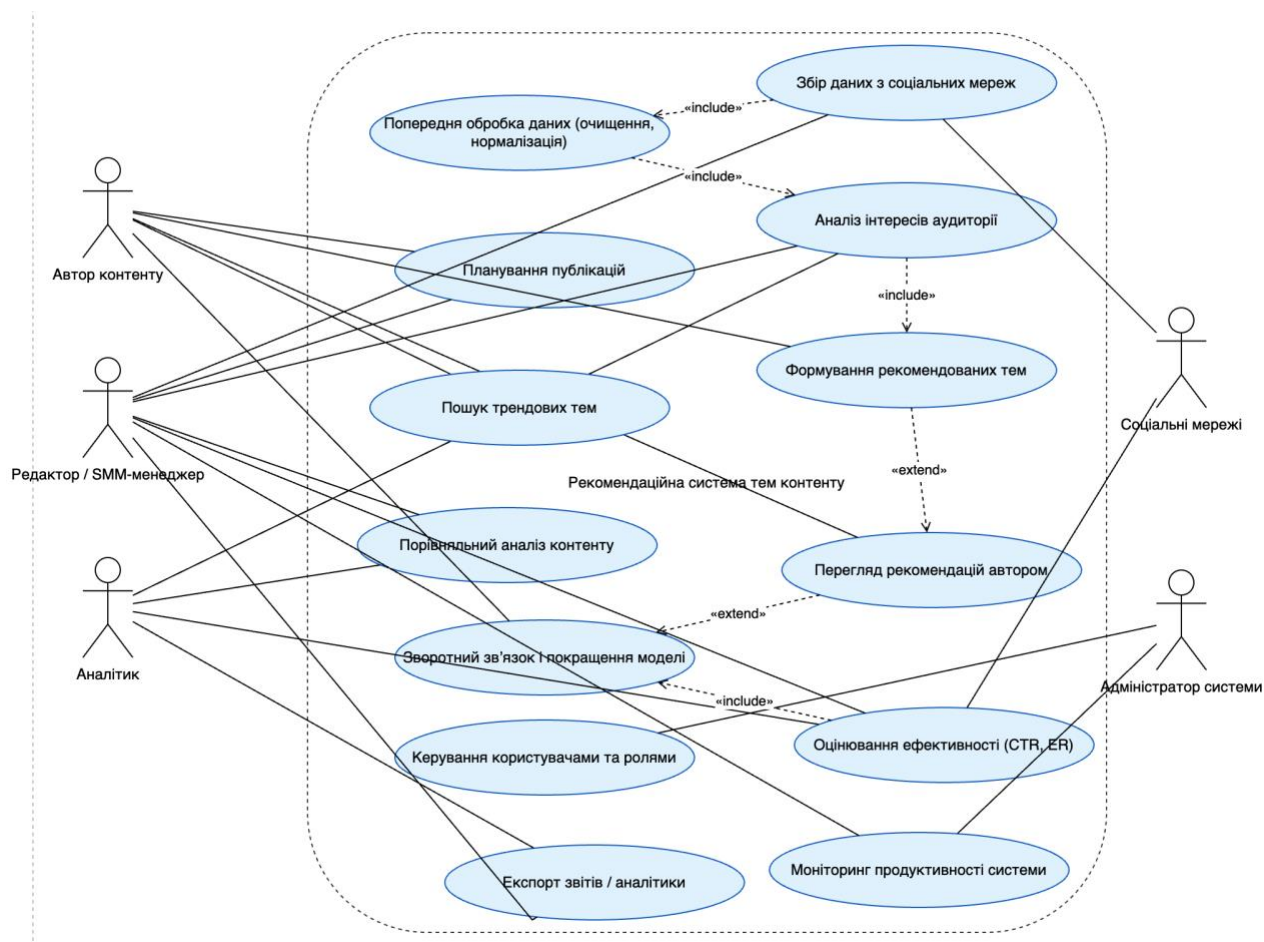
Моделювання предметної області виконано засобами UML, оскільки ця нотація дає змогу формалізувати ролі користувачів, основні сценарії роботи, порядок виконання процесів і взаємодію компонентів у часі. Для початкового аналізу побудовано діаграми прецедентів, активності та послідовності.

Діаграма прецедентів використовується для визначення функціональних меж розроблюваної системи та встановлення основних учасників взаємодії. У межах системи виділено автора контенту, редактора або SMM-менеджера, аналітика, адміністратора та зовнішні соціальні платформи. Автор контенту переглядає сформовані рекомендації, планує публікації та передає зворотний зв'язок щодо якості запропонованих тематик. Редактор або SMM-менеджер уточнює контент-план і контролює підготовку матеріалів до публікації. Аналітик оцінює метрики ефективності, аналізує якість рекомендацій і результати роботи моделей. Адміністратор керує користувачами, ролями доступу та параметрами системи. Зовнішні соціальні платформи виступають джерелом подій, метаданих,

показників активності та реакцій аудиторії. Структуру основних прецедентів системи подано на рисунку 1.7.

Рис. 1.7. Діаграма прецедентів інтелектуальної системи рекомендації тематик контенту

Діаграма активності відображає логіку основного процесу роботи системи. На початковому етапі здійснюється отримання даних із соціальних платформ,



після чого система виконує очищення, нормалізацію та підготовку даних до подальшої обробки. Далі проводиться семантичний аналіз текстового контенту, визначення ключових тематик, виявлення інтересів аудиторії та побудова її профілю. На основі отриманих результатів рекомендаційний модуль формує список потенційно релевантних тем для майбутніх публікацій. Якщо автор або аудиторія надають зворотний зв'язок, параметри моделі оновлюються, що забезпечує адаптацію рекомендаційного механізму до змін у поведінці користувачів. Послідовність виконання основного процесу наведено на рисунку 1.8.

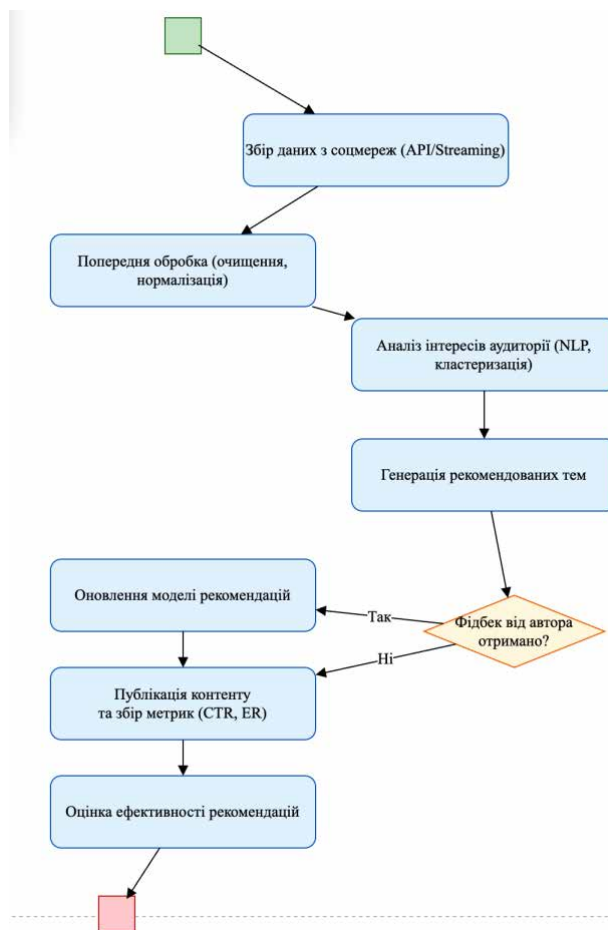


Рис.1.8. Діаграма активності процесу формування рекомендацій тематик контенту

Діаграма послідовності деталізує обмін повідомленнями між основними програмними компонентами системи. У сценарії взаємодії беруть участь клієнтський інтерфейс, рекомендаційний сервіс, кеш, аналітичний модуль і сховище даних. Користувач надсилає запит через інтерфейс, після чого рекомендаційний сервіс перевіряє наявність готових результатів у кеші. Якщо актуальні дані відсутні, запит передається до аналітичного модуля, який отримує необхідну інформацію зі сховища, виконує обробку й повертає сформований набір рекомендацій. Така модель є важливою для проєктування API, визначення критичних точок затримки та розподілу відповідальності між сервісами. Обмін повідомленнями між компонентами системи показано на рисунку 1.9.

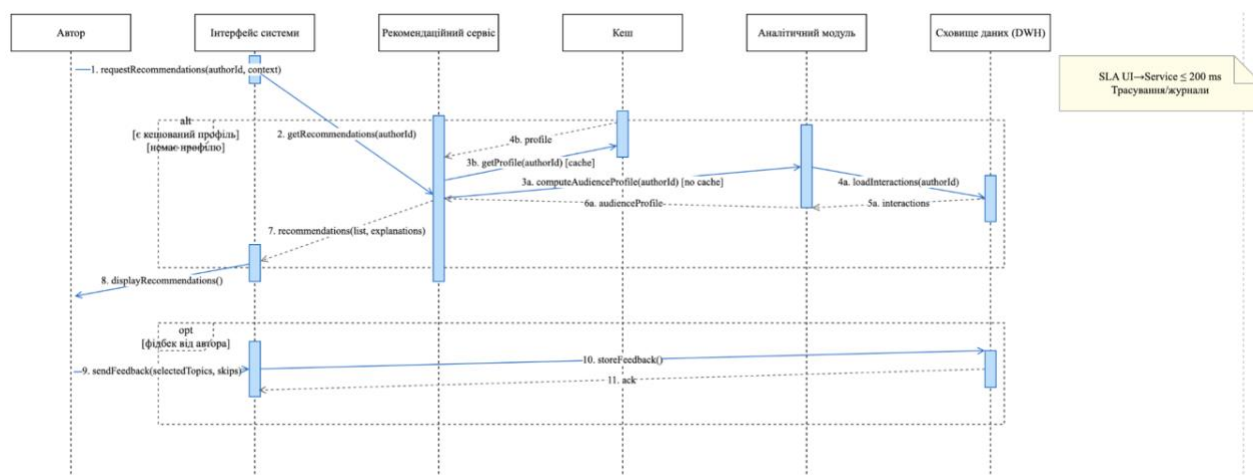


Рис. 1.9. Діаграма послідовності формування рекомендацій у програмній системі

Сформовані UML-моделі уточнюють функціональні межі системи, визначають послідовність опрацювання даних і створюють основу для подальшого проєктування інформаційної та програмної структури.

1.4 Аналіз вимог системи

Аналіз вимог до рекомендаційної системи виконано з метою визначення основних функцій, які має забезпечувати програмне забезпечення, а також якісних характеристик, необхідних для стабільної, безпечної та зручної роботи. Оскільки система призначена для формування рекомендацій тематик контенту на основі даних із соціальних платформ, вимоги охоплюють процеси збирання даних, їх попередньої обробки, семантичного аналізу, профілювання аудиторії, побудови рекомендацій і формування аналітичних звітів.

У межах роботи вимоги до системи поділено на функціональні, нефункціональні та безпекові. Функціональні вимоги визначають перелік дій, які повинна виконувати система. Нефункціональні вимоги описують якість роботи програмного забезпечення, зокрема продуктивність, надійність, масштабованість і зручність використання. Безпекові вимоги встановлюють механізми захисту облікових записів, API, каналів передавання даних і збереженої інформації.

Функціональні вимоги охоплюють повний цикл роботи рекомендаційної системи: отримання інформації із соціальних платформ, очищення та нормалізацію даних, аналіз текстового контенту, побудову профілю аудиторії, формування тематичних рекомендацій, планування публікацій, збір зворотного зв'язку та підготовку аналітичних матеріалів. Перелік функціональних вимог наведено в таблиці 1.3.

Таблиця 1.3

Функціональні вимоги до системи

№	Вимога	Суть функції	Очікуваний результат
1	Збір даних	Підключення до API, webhook і потокових каналів соціальних платформ	Отримання подій, метаданих, реакцій аудиторії та показників активності
2	Попередня обробка даних	Очищення, нормалізація, видалення дублікатів, токенизація текстів	Підготовлений набір даних для подальшого аналізу
3	Семантичний аналіз	Визначення ключових тем, сутностей, тональності та змістових ознак тексту	Набір семантичних характеристик контенту
4	Профілювання аудиторії	Групування користувачів за інтересами, поведінкою та реакціями	Сформовані сегменти аудиторії
5	Формування рекомендацій	Ранжування тематик на основі семантичних ознак, історичних даних і активності аудиторії	Список рекомендованих тем для публікацій
6	Планування публікацій	Додавання рекомендованих тем до контент-плану	Сформований план майбутніх публікацій
7	Збір зворотного зв'язку	Фіксація реакцій автора та аудиторії на запропоновані теми	Дані для уточнення рекомендаційної моделі
8	Формування звітів і дашбордів	Візуалізація KPI, динаміки активності, результатів рекомендацій і якості моделей	Аналітичні матеріали для автора, редактора або аналітика

Нефункціональні вимоги визначають характеристики, які впливають на практичну придатність системи. Рекомендаційна система повинна забезпечувати швидку відповідь на типові запити, підтримувати зростання кількості

користувачів і подій, коректно працювати у разі часткових збоїв, мати зрозумілу структуру програмних модулів і надавати користувачеві пояснювані результати. Основні нефункціональні вимоги наведено в таблиці 1.4.

Таблиця 1.4

Нефункціональні вимоги до системи

№	Характеристика	Вимога	Показник
1	Продуктивність	Швидка відповідь рекомендаційного сервісу на типовий запит користувача	До 200 мс для запиту до готових рекомендацій
2	Масштабованість	Підтримка збільшення кількості авторів, публікацій, подій і джерел даних	Горизонтальне масштабування сервісів
3	Надійність	Стійкість до збоїв, повторне виконання задач і контроль помилок обробки	Доступність не нижче 99,5 %
4	Модульність	Можливість незалежного оновлення окремих компонентів системи	Сервісно-орієнтована архітектура
5	Зручність використання	Простий інтерфейс, зрозуміле подання рекомендацій і пояснення результатів	Мінімальна кількість дій до отримання рекомендації
6	Супроводжуваність	Логічне розділення модулів збору даних, NLP-аналізу, рекомендацій, звітності та адміністрування	Спрощене оновлення й тестування програмного коду
7	Адаптивність	Можливість уточнення рекомендацій на основі нового фідбеку та зміни інтересів аудиторії	Регулярне оновлення профілю аудиторії
8	Сумісність	Робота з різними джерелами даних і форматами обміну	Підтримка API, JSON і webhook-подій

Окрему увагу приділено вимогам до інформаційної безпеки, оскільки система працює з обліковими записами користувачів, аналітичними даними, токенами доступу до зовнішніх платформ і результатами поведінкового аналізу аудиторії. Для зменшення ризиків несанкціонованого доступу необхідно реалізувати автентифікацію, авторизацію, захист каналів передавання даних,

перевірку API-запитів і резервне копіювання бази даних. Основні безпекові вимоги наведено в таблиці 1.5.

Таблиця 1.5

Вимоги до інформаційної безпеки

№	Вимога	Реалізація	Рівень важливості
1	Автентифікація користувачів	Використання JWT/OAuth2, контроль сесій і перевірка облікових даних	Високий
2	Авторизація доступу	RBAC для ролей автора, редактора, аналітика й адміністратора	Високий
3	Захист каналу передавання даних	Використання HTTPS/TLS для обміну між клієнтом, сервером і зовнішніми сервісами	Високий
4	Захист API	Валідація запитів, обмеження частоти звернень, журналювання службових операцій	Високий
5	Захист персональних і службових даних	Обмеження доступу до профілів аудиторії, токенів інтеграції та аналітичних результатів	Високий
6	Резервне копіювання	Створення резервних копій бази даних і можливість відновлення після збоїв	Середній
7	Журналювання подій	Фіксація входів, змін параметрів моделей, адміністративних дій і помилок системи	Середній

Вхідними даними системи є текстові матеріали, метадані публікацій, реакції аудиторії, коментарі, показники охоплення, переходи, збереження, фідбек автора та службові параметри моделей. Вихідними даними є список рекомендованих тематик, пояснення причин рекомендацій, сегменти аудиторії, контент-план, аналітичні звіти та показники ефективності публікацій.

Отже, сформована сукупність вимог визначає технічну основу для подальшого проєктування рекомендаційної системи. Система повинна бути функціонально повною, швидкою, масштабованою, зрозумілою для користувача та захищеною від типових ризиків вебзастосунків. Визначені вимоги надалі

використовуються для побудови архітектури, структури бази даних, програмних модулів і сценаріїв тестування.

1.5 Постановка завдання

Поставлене завдання полягає у створенні рекомендаційної системи, яка автоматизує аналіз інтересів аудиторії соціальних мереж і допомагає автору добирати теми для майбутніх публікацій. Система повинна перетворювати розрізнені соціальні події на структуровані показники, формувати профілі аудиторії, ранжувати потенційні теми та надавати пояснення щодо причин вибору кожної рекомендації.

Вхідними даними системи є тексти публікацій, коментарі, хештеги, описи відео, метадані матеріалів, часові мітки, показники переглядів, лайків, коментарів, поширень, CTR, Engagement Rate, історія попередніх публікацій та фідбек автора. Додатково можуть використовуватися зовнішні сигнали: трендові запити, рекламні показники та результати A/B-тестування.

Вихідними даними є впорядкований перелік рекомендованих тем, пояснення до них, агреговані показники активності аудиторії, тематичні кластери, профілі інтересів, звіти у форматах CSV/HTML/PDF та оновлені параметри моделей. Результати мають бути доступні через вебінтерфейс автора й адміністративну панель.

Основні функції системи включають збір і маршрутизацію подій, нормалізацію контенту, NLP-оброблення, кластеризацію аудиторії, гібридне ранжування тем, збір фідбеку, побудову дашбордів, керування користувачами та журналювання подій безпеки. Система має функціонувати у вебсередовищі, підтримувати розмежування доступу за ролями, працювати через захищений канал і допускати контейнеризоване розгортання.

1.6 Висновки до першого розділу

У першому розділі проаналізовано предметну область рекомендаційних систем тематичного контенту для соціальних мереж. Встановлено, що авторам потрібен не лише інструмент перегляду статистики, а повноцінна система підтримки прийняття рішень, яка враховує семантику публікацій, поведінкові ознаки аудиторії та часову динаміку інтересів.

Порівняння існуючих рішень показало, що більшість платформ орієнтована або на планування публікацій, або на загальну аналітику, але не забезпечує пояснюваного персоналізованого добору тем. На основі цього сформульовано вимоги до розроблюваної системи та визначено її основні функціональні межі.

Побудовані UML-моделі формалізують ролі користувачів, логіку бізнес-процесу та взаємодію компонентів. Отримані результати є основою для проєктування інформаційного та програмного забезпечення у наступному розділі.

2 ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОГО ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Логічна модель даних у вигляді ER-діаграми

Логічна модель даних розроблюваної системи побудована з метою формалізації основних інформаційних об'єктів, які використовуються під час аналізу контенту, профілювання аудиторії та формування тематичних рекомендацій. Модель відображає сутності предметної області, їхні атрибути, зв'язки та обмеження цілісності даних. Основу моделі становлять автор контенту, публікація, тема, сегмент аудиторії, рекомендація та зворотний зв'язок.

Логічна модель даних інтелектуальної системи рекомендації тематик контенту наведена на рисунку 2.1.

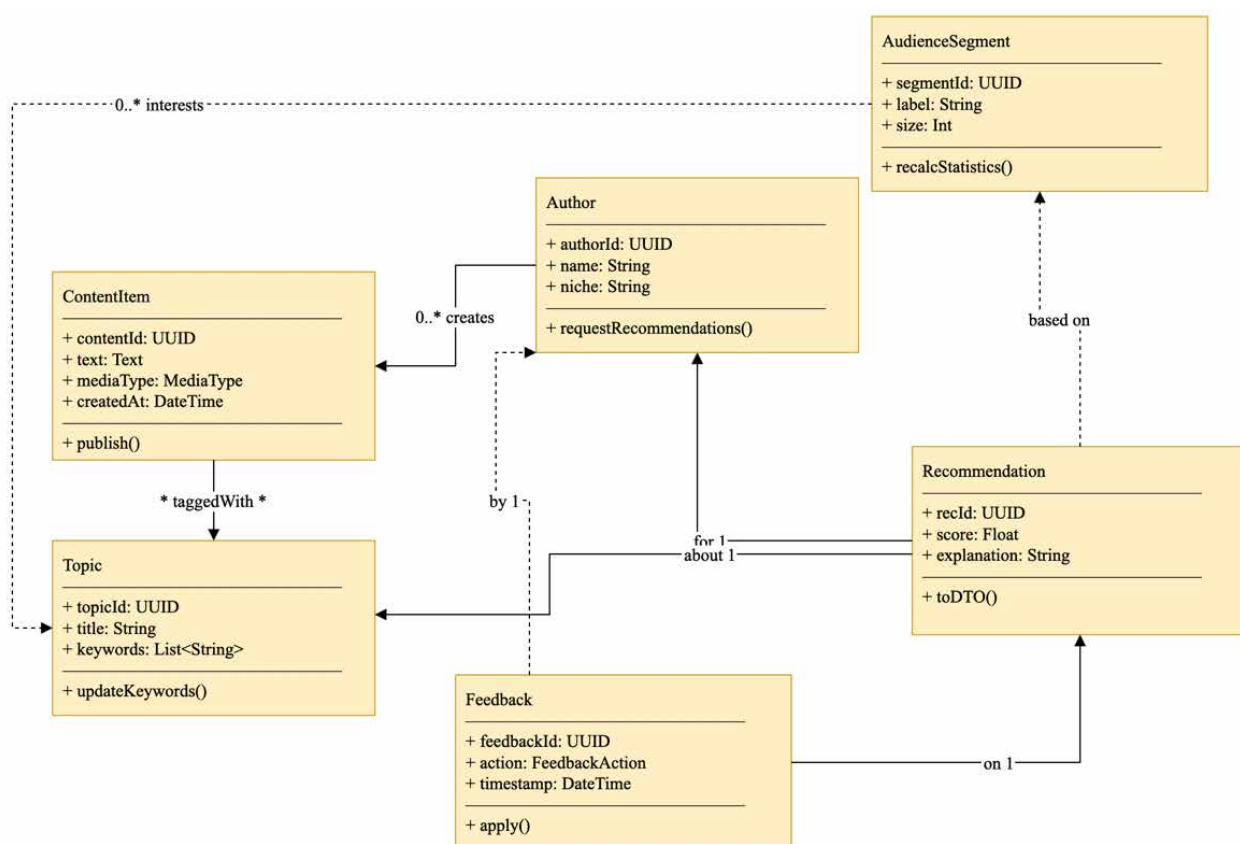


Рис. 2.1 Логічна модель даних інтелектуальної системи рекомендації тематик контенту

У моделі центральною сутністю є `Author`, яка описує автора контенту або користувача системи, для якого формуються рекомендації. Автор має унікальний ідентифікатор `authorId`, ім'я `name` та тематичну нішу `niche`. Через цю сутність система пов'язує створені публікації, сформовані рекомендації та отриманий зворотний зв'язок. Один автор може створювати багато одиниць контенту, однак кожна публікація належить конкретному автору.

Сутність `ContentItem` призначена для збереження відомостей про окрему одиницю контенту. До її атрибутів належать `contentId`, `text`, `mediaType` і `createdAt`. Атрибут `text` містить текстову частину публікації, `mediaType` визначає формат матеріалу, наприклад текст, зображення, відео або комбінований формат, а `createdAt` фіксує дату й час створення запису. Збережені публікації використовуються для подальшого семантичного аналізу, визначення ключових тем і побудови профілю інтересів аудиторії.

Сутність `Topic` описує тематичні напрями, які виявляються в контенті або пропонуються системою як рекомендації. Вона містить `topicId`, `title` і `keywords`. Атрибут `title` визначає назву теми, а `keywords` зберігає набір ключових слів, пов'язаних із відповідним тематичним напрямом. Одна публікація може бути пов'язана з кількома темами, і одна тема може зустрічатися в багатьох публікаціях. Тому між `ContentItem` і `Topic` фактично формується зв'язок багато-до-багатьох, який на рівні фізичної моделі доцільно реалізувати через проміжну таблицю `content_item_topics`.

Сутність `AudienceSegment` використовується для опису груп аудиторії, які мають подібні інтереси або поведінкові ознаки. Вона містить `segmentId`, `label` і `size`. Атрибут `label` задає назву сегмента, наприклад «аудиторія, зацікавлена в освітньому контенті» або «користувачі, які активно реагують на огляди продуктів». Атрибут `size` показує кількість користувачів або подій, що входять до сегмента. Сегменти аудиторії є важливими для персоналізації рекомендацій, оскільки одна й та сама тема може мати різну релевантність для різних груп користувачів.

Сутність Recommendation відображає результат роботи рекомендаційного модуля. Вона містить recId, score і explanation. Атрибут score зберігає числову оцінку релевантності рекомендованої теми, а explanation містить коротке пояснення причини рекомендації. Кожна рекомендація формується для конкретного автора, стосується певної теми та може базуватися на одному або кількох сегментах аудиторії. Це дозволяє не лише показувати список рекомендованих тематик, а й пояснювати, чому система запропонувала саме ці напрями.

Сутність Feedback призначена для збереження реакції автора на отримані рекомендації. Вона містить feedbackId, action і timestamp. Атрибут action фіксує дію користувача, наприклад прийняття рекомендації, відхилення, збереження до контент-плану або позначення як нерелевантної. Атрибут timestamp визначає час надання зворотного зв'язку. Ці дані використовуються для подальшого уточнення рекомендаційної моделі та підвищення точності майбутніх тематичних пропозицій.

Основні сутності логічної моделі даних наведено в таблиці 2.1.

Таблиця 2.1

Основні сутності логічної моделі даних

Сутність	Призначення	Основні атрибути
Author	Зберігання даних про автора контенту або користувача системи	authorId, name, niche
ContentItem	Зберігання текстових і медійних матеріалів автора	contentId, text, mediaType, createdAt
Topic	Опис тематичних напрямів і ключових слів	topicId, title, keywords
AudienceSegment	Збереження сегментів аудиторії за інтересами та поведінкою	segmentId, label, size
Recommendation	Збереження сформованих рекомендацій для автора	recId, score, explanation
Feedback	Фіксація реакції автора на рекомендацію	feedbackId, action,

Зв'язки між сутностями відображають основну логіку роботи системи. Автор створює публікації, публікації пов'язуються з темами, рекомендації

формується для автора та стосуються конкретної теми, а зворотний зв'язок прив'язується до рекомендації. Окремо модель враховує зв'язок між рекомендаціями та сегментами аудиторії, оскільки релевантність теми визначається не лише змістом попередніх публікацій, а й реакціями цільових груп користувачів.

Характеристику зв'язків логічної моделі наведено в таблиці 2.2.

Таблиця 2.2

Зв'язки між сутностями логічної моделі

Зв'язок	Тип зв'язку	Пояснення
Author - ContentItem	1:M	Один автор може створити багато публікацій, кожна публікація належить одному автору
ContentItem - Topic	M:N	Одна публікація може містити кілька тем, одна тема може бути пов'язана з багатьма публікаціями
Author - Recommendation	1:M	Для одного автора може бути сформовано багато рекомендацій
Topic - Recommendation	1:M	Одна тема може бути використана в багатьох рекомендаціях
AudienceSegment - Recommendation	1:M або M:N	Рекомендація формується з урахуванням сегментів аудиторії
Recommendation - Feedback	1:M	На одну рекомендацію може бути надано один або кілька варіантів зворотного зв'язку

Для забезпечення цілісності даних у логічній моделі передбачено використання первинних і зовнішніх ключів. Первинні ключі `authorId`, `contentId`, `topicId`, `segmentId`, `recId` і `feedbackId` однозначно ідентифікують записи відповідних сутностей. Зовнішні ключі повинні забезпечувати зв'язок публікацій з авторами, рекомендацій з авторами й темами, а також зворотного зв'язку з конкретними рекомендаціями.

На рівні фізичної реалізації модель доцільно розгорнути у вигляді таких таблиць: `authors`, `content_items`, `topics`, `audience_segments`, `recommendations`, `feedbacks`. Для зв'язків багато-до-багатьох необхідно створити проміжні таблиці `content_item_topics` і `recommendation_segments`. Це дозволить уникнути

дублювання даних і забезпечити гнучке зберігання інформації про теми, публікації та сегменти аудиторії.

Логічна модель відповідає вимогам нормалізації. Дані про авторів, контент, теми, сегменти, рекомендації та фідбек винесені в окремі сутності. Це зменшує дублювання інформації, спрощує оновлення записів і забезпечує узгодженість даних під час роботи рекомендаційного модуля. Наприклад, тема зберігається один раз у таблиці topics, але може використовуватися для різних публікацій і рекомендацій. Аналогічно сегмент аудиторії може брати участь у формуванні багатьох рекомендацій без повторного дублювання його опису.

Отже, логічна модель даних визначає інформаційну основу розроблюваної системи. Вона забезпечує структуроване зберігання даних про авторів, контент, теми, аудиторні сегменти, рекомендації та зворотний зв'язок. Побудована модель є базою для подальшого створення фізичної структури бази даних, реалізації API, розроблення рекомендаційного сервісу та формування аналітичних модулів системи.

2.2 Діаграма класів і кооперації розроблюваної системи

Діаграма класів (рис.2.2) відображає об'єктно-орієнтовану структуру програмної системи та визначає відповідальність основних класів. У моделі виділено класи для роботи з користувачами, контентом, подіями, тематичними ознаками, рекомендаціями, звітами та сервісами аналітичної обробки.

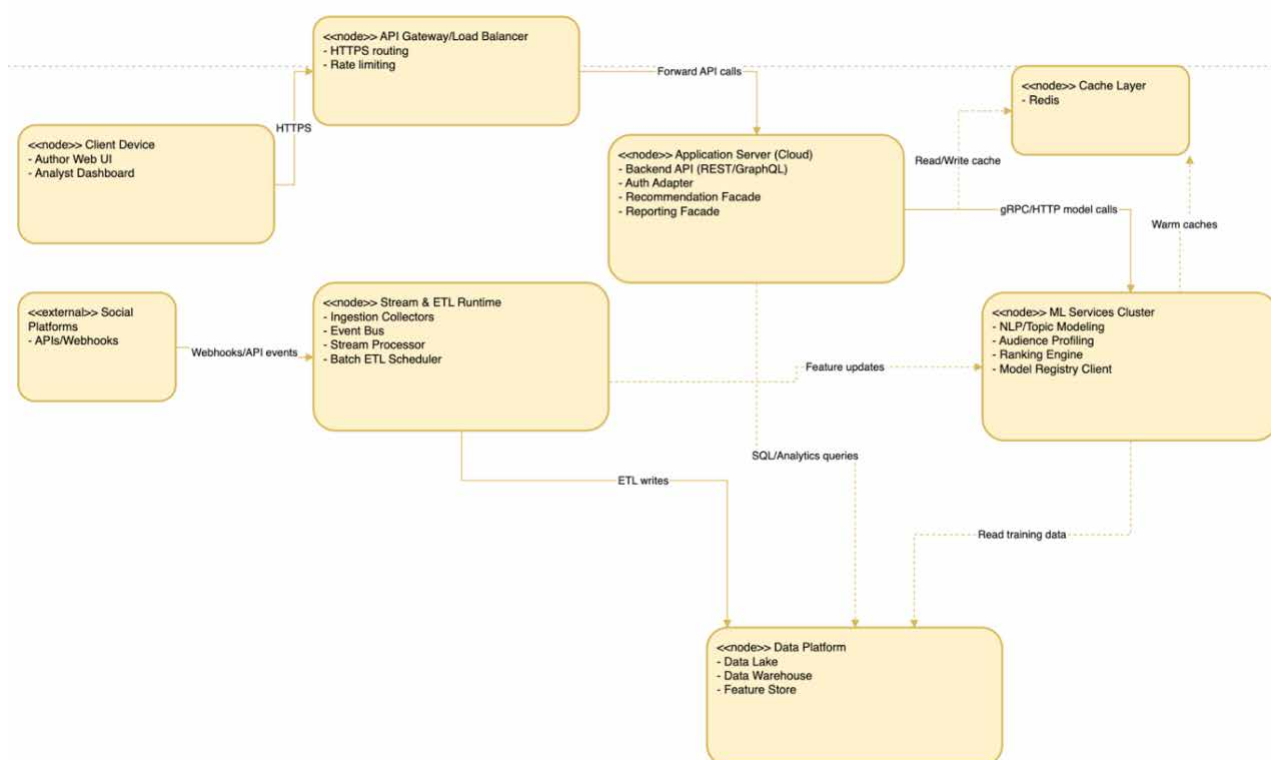


Рис. 2.2. Діаграма класів рекомендаційної системи

Клас UserService відповідає за автентифікацію, реєстрацію, перевірку ролей і керування сесіями. ContentService забезпечує додавання та перегляд матеріалів, передавання тексту до NLP-модуля та зв'язування контенту з тематичними ознаками. EventService приймає події взаємодії, перевіряє їхню коректність і записує до сховища. RecommendationService формує список тем, викликаючи аналітичний модуль і використовуючи кешовані результати.

2.3 Діаграма компонентів

Компонентна модель описує програмну систему на рівні незалежних модулів. Такий підхід є зручним для вебзастосунку з аналітичним ядром, оскільки дає змогу окремо розвивати клієнтський інтерфейс, API-рівень, модулі машинного навчання, сховище даних і зовнішні інтеграції.

До складу системи входять такі компоненти: Web UI, API Gateway, Auth Service, Content Service, Event Collector, NLP Service, Recommendation Engine, Analytics Service, Cache, Database, Feature Store та Report Generator. Зовнішні

соціальні платформи взаємодіють із Event Collector через API або вебхуки, а користувачі працюють із системою через Web UI.

API Gateway виконує маршрутизацію запитів, перевірку токенів і застосування обмежень доступу. Recommendation Engine є центральним компонентом, який об’єднує ознаки контенту, профіль аудиторії, результати трендового аналізу та фідбек автора.

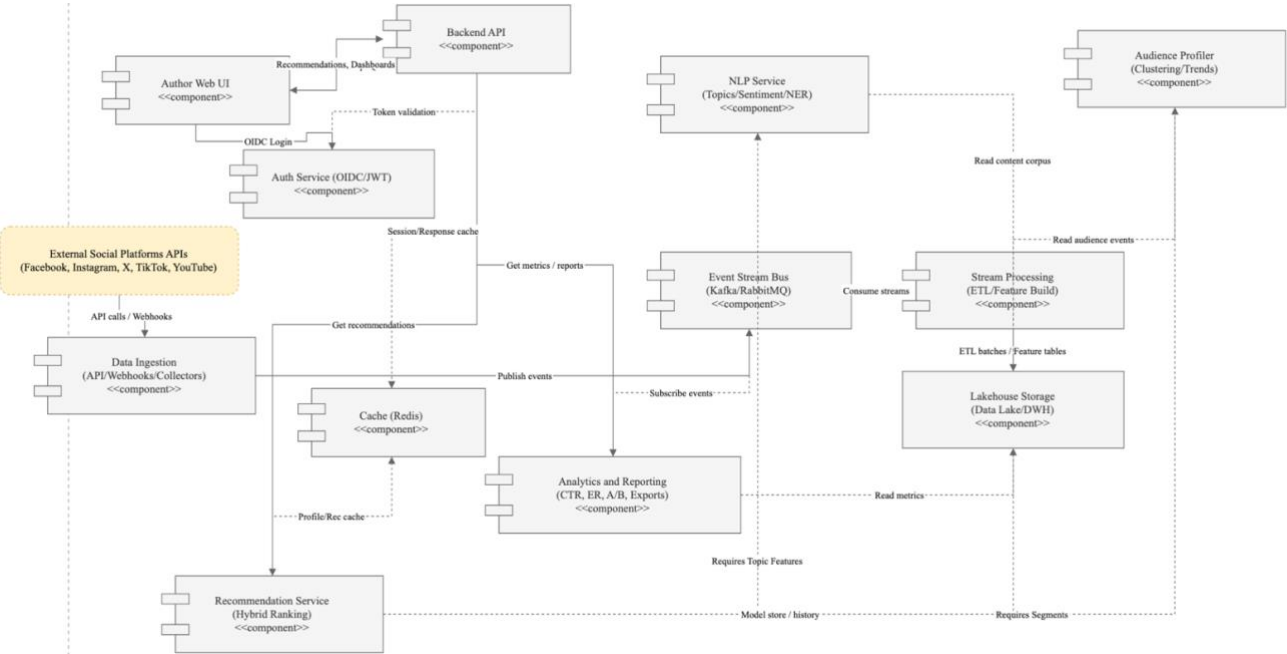


Рис. 2.3. Діаграма компонентів програмної системи

Призначення компонентів представлені у таблиці 2.2.

Таблиця 2.2

Призначення основних програмних компонентів

Компонент	Основне призначення	Взаємодія
Web UI	Робота автора з рекомендаціями, звітами та налаштуваннями	API Gateway
API Gateway	Маршрутизація запитів і контроль доступу	Auth Service, Content Service
NLP Service	Опрацювання тексту й отримання тематичних ознак	Content Service, Feature Store
Recommendation Engine	Ранжування тем і формування пояснень	Analytics Service, Cache, Database
Analytics Service	Розрахунок KPI, трендів і сегментів	DWH, Report Generator

Database	Збереження користувачів, контенту, подій і рекомендацій	Усі прикладні сервіси
----------	---	-----------------------

2.4 Діаграма пакетів

Діаграма пакетів відображає логічне групування програмного коду за функціональними зонами. У системі виокремлено пакети presentation, application, domain, infrastructure, analytics, ml та security. Така структура відповідає принципам багаторівневої архітектури та дає змогу відокремити бізнес-логіку від технічних механізмів доступу до даних. Діаграма пакетів предсавлена на рисунку 2.4.

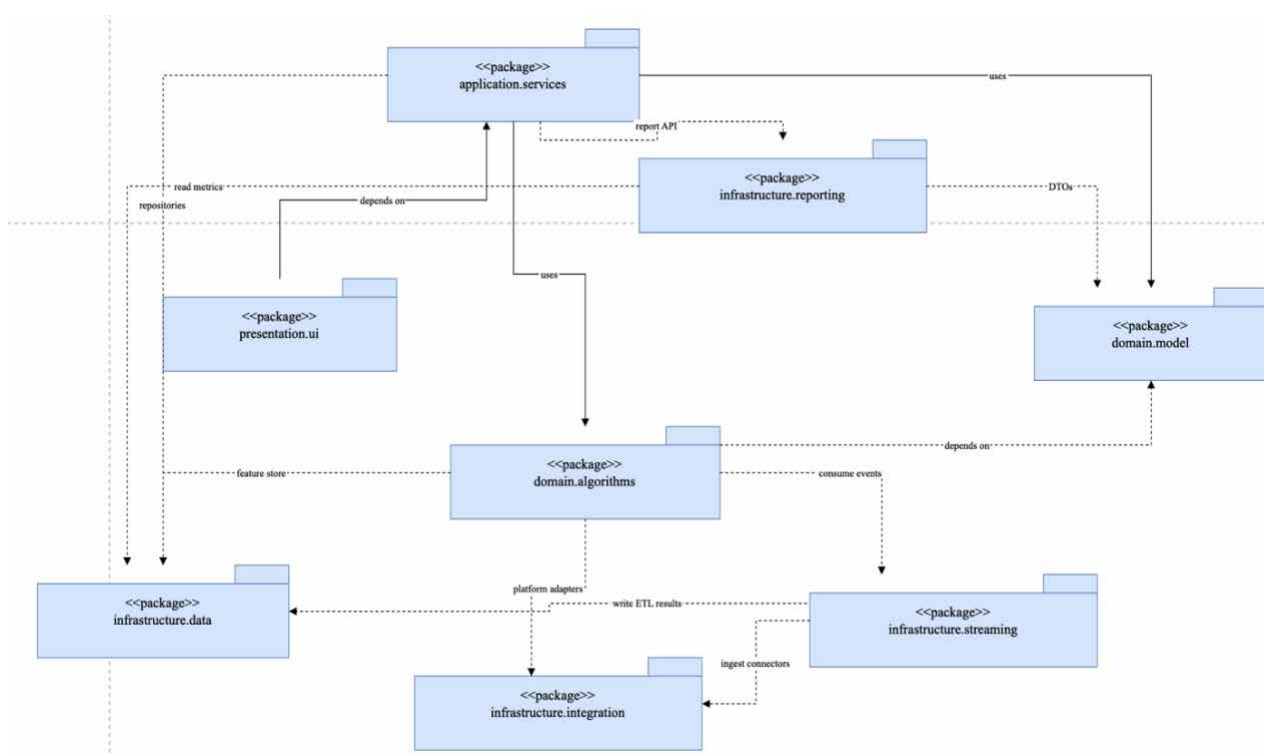


Рис.2.4. Діаграма пакетів програмної системи

Пакет presentation містить компоненти вебінтерфейсу та контролери взаємодії з користувачем. Application об'єднує прикладні сервіси, які реалізують сценарії використання системи. Domain містить основні сутності предметної області та правила бізнес-логіки. Infrastructure відповідає за репозиторії, підключення до БД, кешу та зовнішніх API. Analytics і ml реалізують розрахунок

показників, тематичне моделювання та ранжування рекомендацій. Security містить механізми автентифікації, авторизації та аудиту.

2.5 Висновки до другого розділу

У другому розділі спроектовано інформаційне та програмне забезпечення рекомендаційної системи. Побудована ER-модель визначає структуру даних, необхідних для збереження користувачів, контенту, подій, тематичних ознак, рекомендацій і звітів.

Діаграми класів, кооперацій, компонентів і пакетів уточнюють програмну структуру системи та взаємодію її модулів. Запропоноване розділення на клієнтський рівень, API, аналітичні сервіси, ML-модулі й сховище даних забезпечує модульність, масштабованість і зручність подальшої реалізації.

3 ПРОЄКТУВАННЯ АРХІТЕКТУРИ ПРОГРАМНОЇ СИСТЕМИ ТА АЛГОРИТМІЧНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Обґрунтування вибору технологічних засобів

Вибір технологічного стеку для розроблюваної системи виконано з урахуванням функціональних і нефункціональних вимог, визначених у попередніх розділах. Оскільки система має поєднувати вебінтерфейс, серверне API, модуль збору даних, аналітичну обробку, NLP-компоненти та рекомендаційні алгоритми, основними критеріями вибору стали продуктивність, масштабованість, безпека, зручність інтеграції з ML-бібліотеками та простота розгортання.

Для реалізації серверної частини доцільно використати мову Python і фреймворк FastAPI. Python має розвинену екосистему для оброблення даних, машинного навчання, аналізу текстів і побудови рекомендаційних моделей. FastAPI забезпечує швидку розробку REST API, підтримує асинхронну обробку запитів, автоматично формує OpenAPI-документацію та добре інтегрується з інструментами валідації даних. Це дає змогу створити гнучкий серверний шар, який обслуговує запити клієнтського інтерфейсу, взаємодіє з базою даних і передає дані до аналітичних модулів.

Основною системою керування базами даних обрано PostgreSQL. Вона забезпечує транзакційну цілісність, підтримує складні SQL-запити, індексацію, зв'язки між таблицями та збереження напівструктурованих даних у JSON-полях. Це важливо для системи, яка працює з авторами, публікаціями, подіями взаємодії, тематичними ознаками, результатами кластеризації, рекомендаціями та фідбеком користувачів.

Для прискорення доступу до повторюваних даних доцільно використати Redis. Він може застосовуватися для кешування готових рекомендацій, проміжних результатів аналітичної обробки, токенів сесій і часто

використовуваних запитів. Це зменшує навантаження на базу даних і скорочує час відповіді рекомендаційного сервісу.

ML- і NLP-компоненти системи можуть бути реалізовані з використанням бібліотек scikit-learn, PyTorch, sentence-transformers, spaCy і BERTopic. Scikit-learn доцільно застосовувати для класичних алгоритмів кластеризації, класифікації та оцінювання моделей. PyTorch і sentence-transformers забезпечують роботу з векторними поданнями текстів, що важливо для визначення семантичної близькості тематик. BERTopic може використовуватися для тематичного моделювання й автоматичного групування змістовно подібних матеріалів. SpaCy доцільно застосовувати для базового NLP-аналізу, зокрема токенизації, лематизації та виділення сутностей.

Клієнтська частина системи може бути реалізована засобами HTML5, CSS3 і JavaScript. Такий підхід забезпечує сумісність із сучасними браузерами, підтримку адаптивного інтерфейсу та просту інтеграцію з REST API. У межах інтерфейсу користувач отримує доступ до панелі рекомендацій, контент-плану, аналітичних показників, фільтрів за темами та засобів надання фідбеку щодо якості рекомендацій.

Для розгортання системи доцільно використати Docker і Docker Compose. Контейнеризація дає змогу ізолювати компоненти системи, уніфікувати середовище виконання та швидко запускати всі необхідні сервіси: backend, базу даних PostgreSQL, Redis і додаткові ML-компоненти. Це спрощує перенесення системи між локальним середовищем, тестовим сервером і хмарною інфраструктурою.

Таблиця 3.1

Обґрунтування вибору технологічних засобів

Рівень системи	Обраний засіб	Призначення та причина вибору
Backend	Python, FastAPI	Реалізація REST API, асинхронна обробка запитів, автоматична OpenAPI-документація, зручна інтеграція з ML-модулями

База даних	PostgreSQL	Надійне зберігання структурованих даних, підтримка складних SQL-запитів, індексів, транзакцій і JSON-полів
Кешування	Redis	Зберігання тимчасових даних, кешування результатів рекомендацій, зменшення навантаження на базу даних
ML/NLP	scikit-learn, PyTorch, sentence-transformers, spaCy, BERTopic	Кластеризація, тематичне моделювання, побудова embeddings, семантичний аналіз текстів і ранжування тематик
Frontend	HTML5, CSS3, JavaScript	Побудова адаптивного вебінтерфейсу, взаємодія з REST API, візуалізація рекомендацій і аналітики
Обмін даними	REST API, JSON	Уніфікована взаємодія між клієнтською частиною, сервером, аналітичними модулями та зовнішніми сервісами
Безпека	JWT/OAuth2, HTTPS/TLS, RBAC	Автентифікація користувачів, розмежування прав доступу, захист передавання даних
Розгортання	Docker, Docker Compose	Відтворюване середовище, швидкий запуск сервісів, спрощене тестування й перенесення системи

Обраний технологічний стек забезпечує узгоджену роботу всіх рівнів програмної системи. FastAPI відповідає за серверну логіку та API, PostgreSQL зберігає основні дані, Redis прискорює доступ до часто використовуваної інформації, ML/NLP-бібліотеки реалізують інтелектуальну обробку текстів, а вебінтерфейс забезпечує взаємодію користувача із системою. Використання Docker дозволяє швидко розгортати систему в однаковому середовищі та зменшує кількість помилок, пов'язаних із налаштуванням залежностей.

Отже, запропонований стек є доцільним для реалізації інтелектуальної рекомендаційної системи, оскільки поєднує продуктивний серверний шар, надійне зберігання даних, можливості машинного навчання, кешування, безпечну взаємодію компонентів і зручне розгортання.

3.2 Архітектура системи

Архітектура системи побудована за багаторівневим сервісно-орієнтованим підходом. Вона складається з клієнтського рівня, API-шлюзу, прикладних сервісів, аналітичного ядра, ML-модуля, кешу та платформи даних. Такий розподіл дозволяє незалежно масштабувати компоненти, розмежовувати відповідальність і зменшувати ризики відмов.

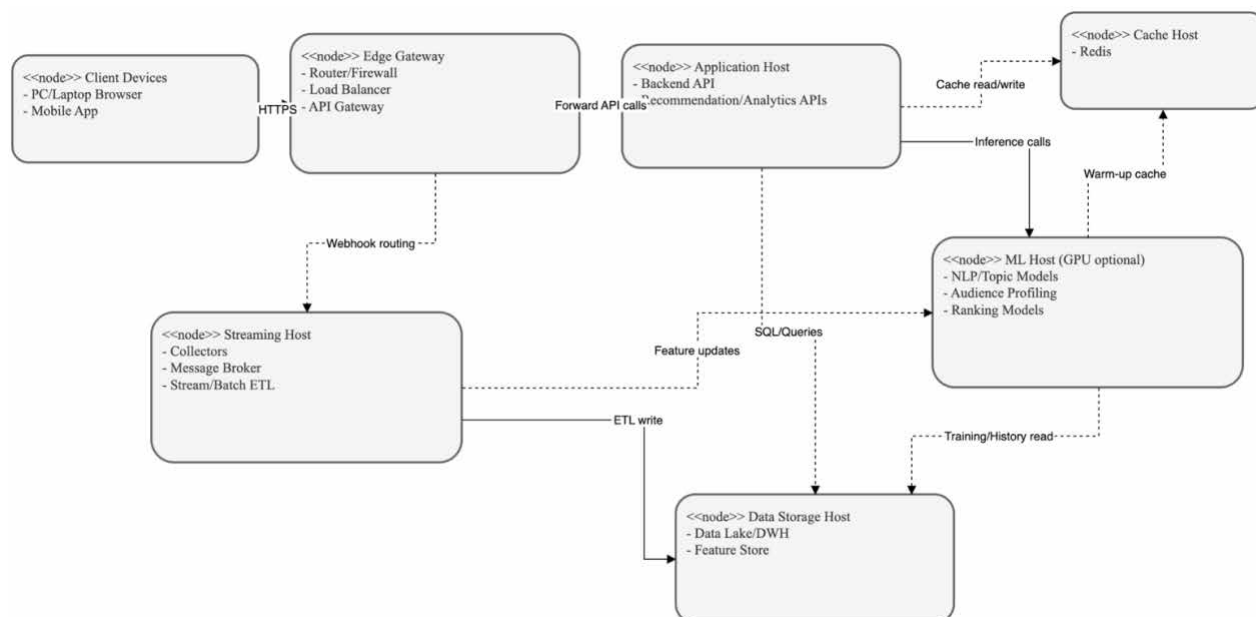


Рис. 3.1. Узагальнена архітектура програмної системи

Користувач взаємодіє із системою через вебінтерфейс, який надсилає запити до API Gateway. Шлюз перевіряє токени доступу, обмежує частоту запитів і маршрутизує звернення до відповідних сервісів. Прикладний сервер обробляє сценарії авторизації, роботи з контентом, формування рекомендацій і побудови звітів. Для часто повторюваних операцій застосовується Redis-кеш. Структурні елементи архітектури представлені у таблиці 3.2.

Потоки подій із соціальних платформ надходять до модуля збору даних, проходять нормалізацію та зберігаються в операційній базі й аналітичному сховищі. ML-модуль отримує тематичні ознаки, формує векторні подання текстів, виконує кластеризацію та повертає оцінки релевантності. Результати ранжування зберігаються в базі даних і відображаються користувачеві через інтерфейс.

Структурні елементи архітектури

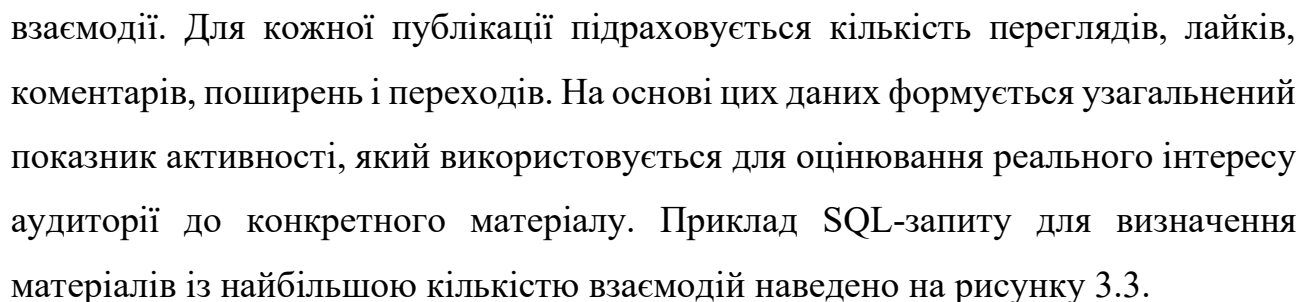
Компонент	Функції	Нефункціональні вимоги
Web UI	Відображення рекомендацій, звітів, налаштувань	Зручність і адаптивність
API Gateway	Маршрутизація, перевірка токенів, rate limiting	Безпека та стабільність
Application Server	Бізнес-логіка та REST API	Модульність і продуктивність
ML Services	NLP, кластеризація, ранжування	Точність і відтворюваність
Data Platform	Збереження подій, ознак і звітів	Цілісність і масштабованість
Monitoring	Логи, метрики, аудит	Контроль стану системи

3.3 Аналітична обробка даних та формування результатів моделювання

Аналітична обробка даних є основним етапом роботи рекомендаційного модуля, оскільки саме на цьому етапі первинні події взаємодії, текстові матеріали та службові метадані перетворюються на набір ознак для подальшого ранжування тематик. Вхідними даними для модуля є тексти публікацій, тематичні мітки, перегляди, лайки, коментарі, поширення, переходи за посиланнями, збереження, а також фідбек автора щодо якості запропонованих рекомендацій.

Загальна структура зберігання даних системи подана на рисунку 3.2. Фізична модель охоплює таблиці користувачів, контенту, тематичних ознак, подій взаємодії, рекомендацій, фідбеку та результатів аналітичної обробки. Така структура забезпечує зв'язок між автором, його публікаціями, реакціями аудиторії та результатами роботи моделей. Наявність окремих таблиць для подій і рекомендацій дає змогу зберігати історію взаємодій, порівнювати ефективність

На першому етапі аналітичний модуль виконує агрегування подій



```

SELECT
    c.content_id,
    c.title,
    COUNT(e.event_id) AS total_interactions
FROM content_item c
JOIN interaction_event e
    ON c.content_id = e.content_id
GROUP BY c.content_id, c.title
ORDER BY total_interactions DESC
LIMIT 10;

```

Рис.3.3. SQL-запит визначення матеріалів із найбільшою кількістю взаємодій

```

SELECT
    u.user_id,
    u.username,
    COUNT(DISTINCT c.content_id) AS total_posts,
    COUNT(e.event_id) AS total_engagements
FROM user_account u
LEFT JOIN content_item c
    ON u.user_id = c.author_id
LEFT JOIN interaction_event e
    ON c.content_id = e.content_id
WHERE u.role = 'author'
GROUP BY u.user_id, u.username
ORDER BY total_engagements DESC;

```

Рис. 3.4. SQL-запит оцінювання активності авторів

Окремо аналізується активність авторів. Для цього система підраховує кількість опублікованих матеріалів, сумарну кількість подій взаємодії, середній рівень залучення та частку матеріалів, що отримали високий відгук аудиторії.

Такий аналіз дає змогу визначити, які автори стабільно створюють результативний контент, а також виявити тематики, що найкраще відповідають їхній аудиторії. Приклад SQL-запиту для оцінювання активності авторів подано на рисунку 3.4.

Для перевірки відповідності контенту певній тематиці використовується вибірка матеріалів за тематичною релевантністю. На цьому етапі враховуються ключові слова, тематичні ознаки, результати NLP-аналізу та зв'язок публікації з відповідною темою. Це дозволяє відокремити випадкові згадки від матеріалів, у яких тема є основною. SQL-запит для вибірки контенту за тематичною релевантністю наведено на рисунку 3.5.

```
SELECT
    c.content_id,
    c.title,
    f.relevance_score
FROM content_item c
JOIN content_topic_feature f
    ON c.content_id = f.content_id
WHERE f.topic_id = 5
ORDER BY f.relevance_score DESC
LIMIT 20;
```

Рис. 3.5. SQL-запит вибірки контенту за тематичною релевантністю

Для оцінювання ефективності рекомендаційної моделі використовується показник CTR, який визначає частку переходів або прийнятих рекомендацій відносно кількості їх показів. Аналіз CTR дає змогу порівняти різні версії моделі, оцінити якість ранжування та визначити, які тематичні пропозиції викликають

найбільшу зацікавленість у користувача або аудиторії. Приклад SQL-запиту для аналізу CTR рекомендаційних моделей наведено на рисунку 3.6.

```
SELECT
    model_version,
    COUNT(*) AS total_shown,
    SUM(CASE WHEN clicked = TRUE THEN 1 ELSE 0 END) AS total_clicks,
    ROUND(
        SUM(CASE WHEN clicked = TRUE THEN 1 ELSE 0 END)::numeric
        / NULLIF(COUNT(*), 0),
        4
    ) AS ctr
FROM recommendation_log
GROUP BY model_version
ORDER BY ctr DESC;
```

Рис.3.6. SQL-запит аналізу CTR рекомендаційних моделей

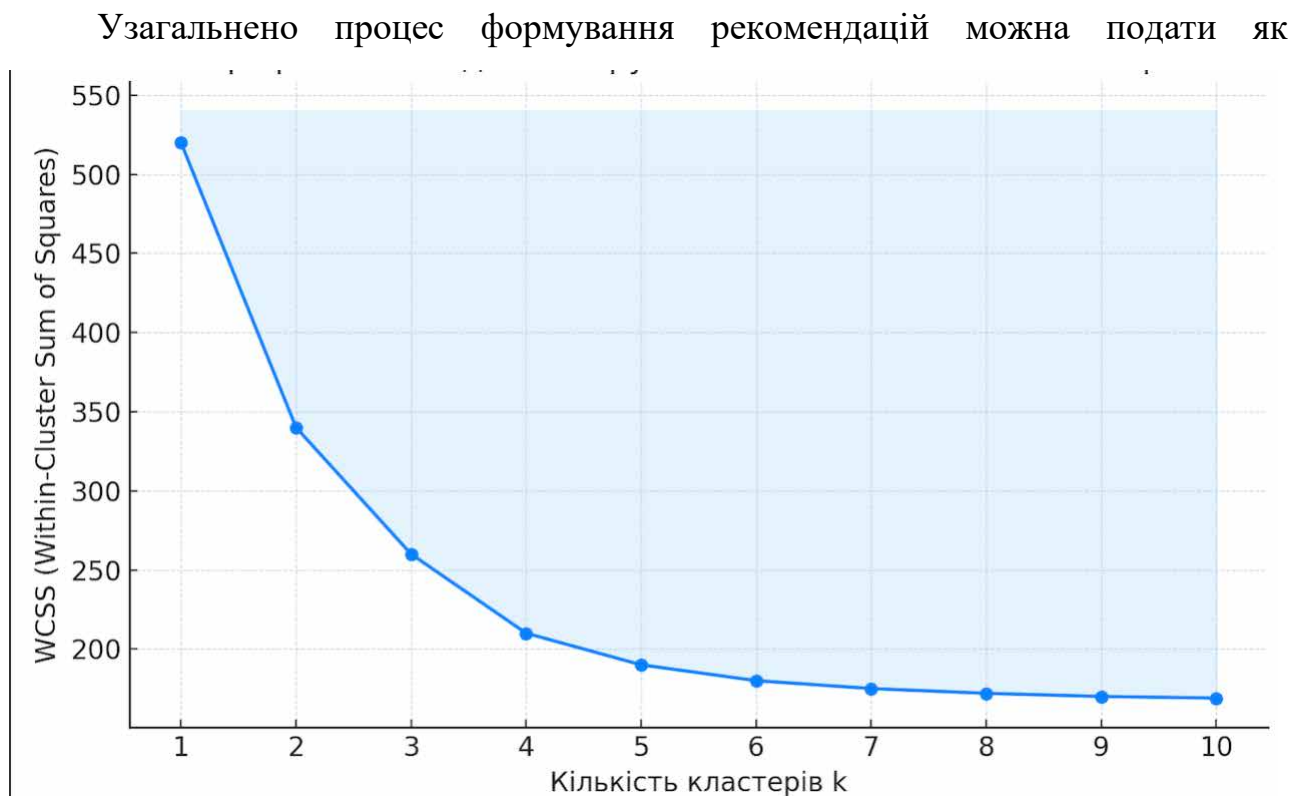
Після формування числових і семантичних ознак система виконує кластеризацію тематик. Метою цього етапу є групування подібних тем, публікацій або аудиторних інтересів у логічні кластери. Кластеризація дає змогу зменшити складність тематичного простору, виявити приховані групи інтересів і підготувати дані для персоналізованого ранжування рекомендацій.

Для вибору раціональної кількості кластерів використовується метод «ліктя». Його суть полягає у побудові залежності між кількістю кластерів і значенням внутрішньокластерної дисперсії. Оптимальним вважається таке значення, після якого подальше збільшення кількості кластерів уже не дає суттєвого покращення якості групування. Графік вибору кількості кластерів подано на рисунку 3.7.

Результати аналітичної обробки інтегруються в єдину модель ранжування. Під час обчислення підсумкового балу рекомендації враховуються семантична близькість теми до попереднього контенту автора, активність аудиторії, трендова динаміка, історична ефективність схожих матеріалів, належність теми до певного аудиторного сегмента та фідбек автора. Такий підхід дозволяє не

лише відібрати популярні теми, а й адаптувати рекомендації до конкретної ніші, стилю автора та поведінки його аудиторії.

Рис. 3.7. Графік «ліктя» для вибору кількості кластерів



послідовність етапів: збирання подій взаємодії, очищення та нормалізація даних, семантичне опрацювання контенту, агрегування показників активності, кластеризація тематик, розрахунок релевантності та формування підсумкового списку рекомендацій. Отримані результати передаються до клієнтського інтерфейсу, де користувач може переглянути рекомендовані теми, оцінити їхню доцільність або додати обрану тему до контент-плану.

Таким чином, аналітична обробка даних забезпечує перехід від неструктурованих текстів і розрізнених подій взаємодії до формалізованих показників, придатних для машинного аналізу. Це створює основу для побудови персоналізованих рекомендацій, оцінювання ефективності моделей і подальшого вдосконалення системи на основі накопиченого фідбеку.

3.4 Алгоритмізація програмних модулів

Алгоритмізація програмних модулів виконується з метою формалізації послідовності оброблення даних у рекомендаційній системі. Оскільки система працює з даними соціальних платформ, текстовими матеріалами, аудиторними реакціями та результатами машинного аналізу, її алгоритмічне забезпечення доцільно поділити на окремі функціональні модулі. Такий підхід спрощує реалізацію, тестування, повторне використання компонентів і подальше розширення системи.

До складу основних алгоритмічних модулів належать модуль збору даних, модуль попередньої обробки тексту, модуль тематичного аналізу, модуль профілювання аудиторії, модуль ранжування рекомендацій, модуль формування пояснень і модуль оновлення параметрів моделі на основі фідбеку. Кожен модуль має чітко визначені вхідні та вихідні дані, що забезпечує узгоджену взаємодію між компонентами системи.

Першим етапом є збір даних із зовнішніх соціальних платформ. Модуль DataCollector приймає події через API або webhook-канали, перевіряє структуру отриманого повідомлення, визначає тип події, додає службові атрибути та передає нормалізований запис до бази даних або черги оброблення. До таких подій належать перегляди, лайки, коментарі, поширення, переходи за посиланнями та інші реакції аудиторії. На цьому етапі важливо забезпечити перевірку коректності даних, уникнення дублювання подій і збереження часу отримання запису.

Наступним етапом є попередня обробка текстових матеріалів. Модуль TextProcessor отримує текст публікації, видаляє зайві символи, нормалізує регістр, виконує токенізацію, лематизацію та виділення ключових слів. Після цього текст перетворюється у векторне подання, яке може використовуватися для визначення семантичної близькості між публікаціями, темами та інтересами аудиторії. Результатом роботи модуля є набір токенів, лем, ключових слів і embeddings, придатних для подальшого машинного аналізу.

Після підготовки тексту виконується тематичний аналіз. На цьому етапі система визначає, до яких тематичних груп належить матеріал, і порівнює його з уже відомими темами. Для цього можуть використовуватися методи кластеризації, тематичного моделювання та порівняння векторних подань. Якщо матеріал відповідає наявній темі, він прив'язується до відповідного тематичного кластера. Якщо подібність є недостатньою, система може створити нову тематичну групу або позначити матеріал як такий, що потребує додаткової перевірки.

Модуль AudienceProfiler формує профіль аудиторії на основі подій взаємодії та семантичних ознак контенту. Він аналізує, які теми викликають найбільшу активність, які формати публікацій мають вищий рівень залучення та які аудиторні сегменти демонструють стабільний інтерес до певних напрямів. У результаті формується набір сегментів аудиторії, що використовується під час персоналізації рекомендацій.

Основним модулем системи є RecommendationRanker. Він об'єднує кілька груп ознак: семантичну подібність теми до попереднього контенту автора, історичну ефективність схожих матеріалів, активність відповідного сегмента аудиторії, трендову динаміку та обмеження різноманітності. На основі цих показників обчислюється підсумковий бал для кожної потенційної теми. Далі теми сортуються за релевантністю, після чого система формує ранжований список рекомендацій.

Алгоритм роботи рекомендаційного модуля подано на рисунку 3.8.

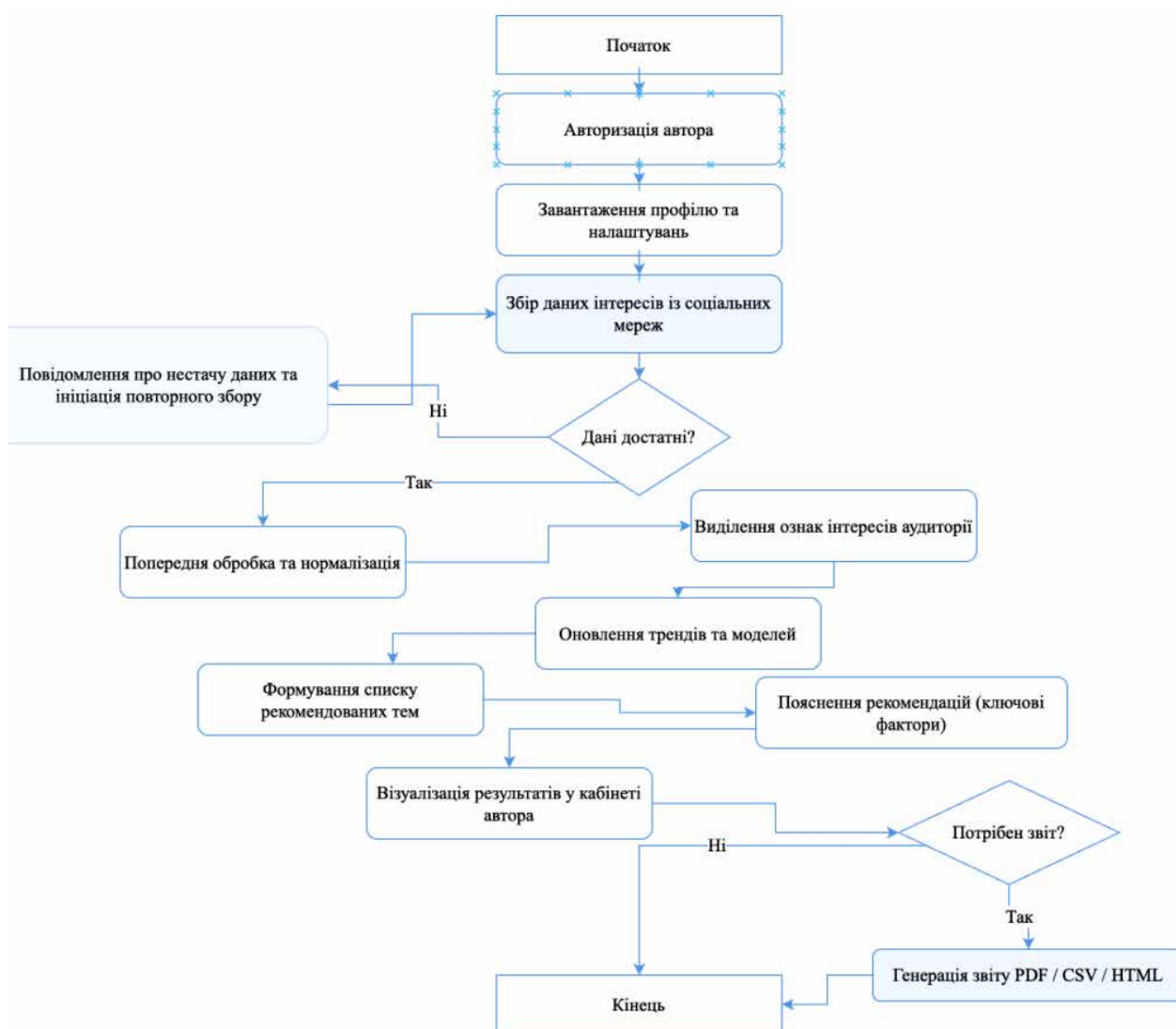


Рис. 3.8. Алгоритм роботи рекомендаційного модуля

Після формування списку рекомендацій система генерує коротке пояснення для кожної запропонованої теми. Пояснення може містити причину рекомендації, наприклад високу активність аудиторії за подібними матеріалами, зростання тренду, відповідність інтересам сегмента або позитивну історичну ефективність теми. Це підвищує прозорість роботи системи та дає користувачеві змогу краще розуміти логіку прийняття рішень.

Окремим етапом є оброблення фідбеку. Модуль FeedbackUpdater отримує дії автора щодо рекомендацій: прийняття, відхилення, збереження до контент-плану, ігнорування або позначення як нерелевантної. Також можуть враховуватися показники CTR, ER, кількість переглядів і подальша активність

аудиторії після публікації матеріалу. На основі цих даних система коригує ваги ознак, що дозволяє підвищити релевантність наступних рекомендацій.

Основні алгоритмічні модулі системи наведено в таблиці 3.3.

Таблиця 3.3

Алгоритмічні модулі системи

Модуль	Вхідні дані	Вихідні дані	Роль у системі
DataCollector	Події API, webhook-повідомлення, метадані публікацій	Нормалізовані записи подій	Отримання та первинна перевірка даних
TextProcessor	Текст публікацій, описи, коментарі	Токени, лєми, ключові слова, embeddings	Підготовка тексту до NLP-аналізу
TopicAnalyzer	Векторні подання текстів, тематичні ознаки	Тематичні групи та зв'язки з темами	Визначення змістової належності контенту
AudienceProfiler	Події взаємодії, тематичні ознаки, поведінкові показники	Сегменти аудиторії та профілі інтересів	Побудова профілю аудиторії
RecommendationRanker	Теми, профілі аудиторії, тренди, історичні метрики	Ранжований список тем	Формування рекомендацій
ExplanationGenerator	Результати ранжування, метрики активності, тематичні ознаки	Текстові пояснення рекомендацій	Пояснення причин вибору тем
FeedbackUpdater	Оцінки автора, CTR, ER, реакції аудиторії	Оновлені ваги моделі та параметри ранжування	Адаптація системи до нового фідбеку

Узагальнений алгоритм формування рекомендацій можна подати так: система отримує події та контент із зовнішніх джерел, виконує перевірку й нормалізацію даних, обробляє текстові матеріали, визначає тематичні ознаки,

формує профіль аудиторії, розраховує релевантність потенційних тем, ранжує їх за підсумковим балом, додає пояснення та передає результат до інтерфейсу користувача. Після взаємодії користувача з рекомендаціями система зберігає фідбек і використовує його для оновлення параметрів моделі.

Таким чином, алгоритмізація програмних модулів забезпечує послідовну та контрольовану роботу всієї рекомендаційної системи. Поділ на окремі модулі дає змогу незалежно тестувати збір даних, NLP-обробку, профілювання аудиторії, ранжування тем і адаптацію моделі. Це підвищує надійність системи, спрощує її супровід і створює основу для подальшого розширення функціоналу.

3.5 Висновки до третього розділу

У третьому розділі було розглянуто архітектурні та алгоритмічні засади реалізації інтелектуальної рекомендаційної системи для добору тематик контенту. Обґрунтовано вибір технологічного стеку, який забезпечує поєднання вебінтерфейсу, серверного API, аналітичної обробки даних, NLP-модулів і рекомендаційних алгоритмів. Для серверної частини запропоновано використання Python і FastAPI, для зберігання даних - PostgreSQL, для кешування результатів - Redis, а для машинного навчання та текстового аналізу - scikit-learn, PyTorch, sentence-transformers, spaCy і BERTopic. Такий набір технологій відповідає вимогам до продуктивності, масштабованості, модульності та зручності розгортання системи.

У межах розділу було описано фізичну модель даних програмної системи, яка забезпечує зберігання інформації про авторів, публікації, тематичні ознаки, події взаємодії, рекомендації, фідбек і результати аналітичної обробки. Запропонована структура даних дозволяє пов'язати контент автора з реакціями аудиторії, результатами NLP-аналізу та сформованими рекомендаціями. Це створює основу для накопичення історичних даних, аналізу ефективності публікацій і подальшого вдосконалення рекомендаційної моделі.

Окрему увагу приділено аналітичній обробці даних. Розглянуто процес агрегування подій взаємодії, визначення активності аудиторії, оцінювання ефективності авторів, вибірки контенту за тематичною релевантністю та аналізу CTR рекомендаційних моделей. Також описано використання кластеризації тематик і методу «ліктя» для вибору раціональної кількості кластерів. Це дозволяє формувати тематичні групи, виявляти приховані закономірності в інтересах аудиторії та підвищувати якість персоналізованих рекомендацій.

У розділі було сформовано алгоритмічне забезпечення системи, яке охоплює модулі збору даних, попередньої обробки тексту, тематичного аналізу, профілювання аудиторії, ранжування рекомендацій, формування пояснень і оновлення параметрів моделі на основі фідбеку. Такий поділ забезпечує чітку відповідальність кожного компонента, спрощує тестування програмних модулів і створює умови для подальшого масштабування системи.

Таким чином, у третьому розділі було визначено технологічну, інформаційну та алгоритмічну основу програмної реалізації системи. Запропоновані рішення забезпечують повний цикл роботи з даними: від отримання подій із зовнішніх джерел і семантичного аналізу контенту до формування ранжованого списку тематичних рекомендацій та адаптації моделі на основі зворотного зв'язку. Отримані результати є основою для подальшого тестування, оцінювання ефективності та підготовки системи до впровадження.

4 ТЕСТУВАННЯ ТА ОЦІНЮВАННЯ ЕФЕКТИВНОСТІ СИСТЕМИ

4.1 План тестування програмних модулів та методика оцінювання результатів

Тестування програмних модулів рекомендаційної системи проводиться з метою перевірки коректності реалізації основних функцій, стабільності взаємодії між компонентами, якості формування рекомендацій, продуктивності API та зручності роботи користувача з інтерфейсом. Оскільки система поєднує вебінтерфейс, серверну логіку, базу даних, кеш Redis, NLP-модуль і рекомендаційний механізм, тестування має охоплювати як окремі модулі, так і їхню спільну роботу.

Основними об'єктами тестування є модуль автентифікації, сервіс роботи з контентом, модуль збору подій, NLP-сервіс, рекомендаційний модуль, модуль оброблення фідбеку та генератор звітів. Для кожного модуля визначено вхідні дані, очікуваний результат і критерії успішності. Такий підхід дозволяє виявити помилки на рівні окремих функцій, перевірити правильність обміну даними між сервісами та оцінити готовність системи до практичного використання.

Функціональне тестування спрямоване на перевірку основних сценаріїв роботи системи. До них належать реєстрація та вхід користувача, розмежування прав доступу, завантаження контенту, приймання подій взаємодії, запуск семантичного аналізу, формування списку рекомендованих тематик, збереження зворотного зв'язку та створення аналітичних звітів. Успішним результатом функціонального тестування вважається коректне виконання операцій без втрати даних і з поверненням очікуваних відповідей користувачеві.

Інтеграційне тестування використовується для перевірки взаємодії між Web UI, API Gateway, прикладним сервером, базою даних PostgreSQL, кешем Redis і ML-модулем. Особлива увага приділяється передаванню даних між

сервісами, коректності форматів JSON, обробленню помилкових запитів і збереженню результатів аналітичної обробки. Це дає змогу переконатися, що модулі системи працюють не ізольовано, а як єдиний програмний комплекс.

Інтерфейсне тестування передбачає перевірку зручності користування системою, правильності відображення рекомендацій, аналітичних показників, контент-плану та повідомлень про помилки. Продуктивне тестування спрямоване на оцінювання часу відповіді API, стабільності оброблення одночасних запитів і швидкості повернення готових рекомендацій із кешу. Безпекове тестування охоплює перевірку автентифікації, авторизації, захисту API, валідації вхідних даних і недопущення доступу до службових функцій без відповідної ролі.

Методика оцінювання результатів тестування включає кількісні та якісні показники. До кількісних належать час відповіді API, кількість успішно оброблених запитів, частка помилкових відповідей, точність рекомендацій за метриками Precision@k, Recall@k, NDCG і MRR, а також показники CTR та Engagement Rate у тестових сценаріях. До якісних показників належать зрозумілість інтерфейсу, коректність пояснень рекомендацій, логічність повідомлень про помилки та відповідність результатів очікуваній поведінці системи.

План тестування програмних модулів наведено в таблиці 4.1.

Таблиця 4.1

План тестування програмних модулів

Модуль	Що перевіряється	Критерій успішності
Auth Service	Реєстрація користувача, вхід, перевірка ролей, створення та перевірка токенів	Користувач успішно входить у систему, помилкові дані відхиляються, доступ до функцій залежить від ролі
Content Service	Додавання, редагування, збереження та передавання контенту на обробку	Контент зберігається в базі даних і коректно передається до NLP-модуля
Event Collector	Приймання подій взаємодії з API або webhook-каналів	Події проходять валідацію, не дублюються та записуються без втрат

NLP Service	Очищення тексту, токенизація, лематизація, формування embeddings і тематичних міток	Для кожного тексту стабільно формуються ознаки, придатні для подальшого аналізу
Audience Profiler	Аналіз реакцій аудиторії, групування інтересів і побудова сегментів	Система формує коректні сегменти аудиторії на основі поведінкових і тематичних ознак
Recommendation Engine	Ранжування тематик, урахування трендів, інтересів аудиторії та історичних метрик	Користувач отримує релевантний список тем із поясненнями причин рекомендації
Feedback Updater	Оброблення дій автора, CTR, ER і реакцій аудиторії	Фідбек зберігається та використовується для уточнення параметрів моделі
Report Generator	Формування звітів, експорт результатів, відображення KPI	Звіти створюються без помилок у CSV, HTML або PDF-форматі
API Gateway	Маршрутизація запитів, перевірка форматів, оброблення помилок	Запити передаються до потрібних сервісів, некоректні запити отримують контрольовану відповідь
Web UI	Відображення рекомендацій, аналітики, фільтрів і повідомлень	Інтерфейс працює стабільно, дані відображаються правильно та зрозуміло для користувача

Для перевірки якості рекомендацій формується тестовий набір даних, який містить приклади публікацій, тематичні ознаки, історичні реакції аудиторії та очікувані релевантні теми. Результати роботи рекомендаційного модуля порівнюються з еталонними або експертно визначеними варіантами. Метрика $Precision@k$ показує, яка частина рекомендованих тем у перших k позиціях є релевантною. $Recall@k$ дозволяє оцінити, яку частину всіх релевантних тем система змогла знайти. NDCG враховує не лише наявність правильних рекомендацій, а й їхній порядок у списку. MRR показує, на якій позиції з'являється перша релевантна рекомендація.

Для оцінювання продуктивності фіксується середній час відповіді API, час отримання рекомендацій із кешу, час оброблення нового тексту NLP-модулем і час формування аналітичного звіту. Типовий запит до готових рекомендацій має

виконуватися без помітної затримки для користувача, а складні операції NLP-аналізу можуть виконуватися асинхронно або у фоновому режимі. Такий підхід дозволяє не блокувати інтерфейс і забезпечити стабільну роботу системи навіть за збільшення кількості подій.

Оцінювання безпеки передбачає перевірку доступу до захищених маршрутів без токена, спробу виконання адміністративних дій користувачем без відповідної ролі, передавання некоректних параметрів у запитах і перевірку обмеження частоти звернень до API. Успішним результатом вважається відхилення неавторизованих або некоректних запитів без порушення цілісності даних.

Отже, запропонований план тестування охоплює основні рівні програмної системи: інтерфейс користувача, API, бізнес-логіку, базу даних, кеш, NLP-модулі, рекомендаційний механізм і звітність. Такий підхід дозволяє оцінити не лише працездатність окремих компонентів, а й ефективність системи в цілому. Отримані результати тестування є основою для виявлення недоліків, оптимізації продуктивності та підтвердження готовності системи до впровадження.

4.2 Тестування інтерфейсних модулів

Інтерфейсні модулі перевірялися за типовими сценаріями роботи автора: вхід до системи, перегляд головного дашборду, відкриття списку рекомендацій, аналіз пояснень, вибір теми, перегляд статистики, формування звіту та зміна параметрів профілю. Окремо перевірялася робота адміністративної частини, де здійснюється керування користувачами, ролями та конфігураційними параметрами.

Під час тестування оцінювалися коректність відображення даних, зрозумілість навігації, адаптивність елементів інтерфейсу, обробка порожніх станів і повідомлень про помилки. Результати показали, що основні екрани забезпечують доступ до потрібних функцій без зайвого ускладнення користувацького сценарію.

На рисунках 4.1–4.4 наведено приклади інтерфейсів системи: форму входу, дашборд, аналітичний екран і сторінку керування рекомендаціями. Вони підтверджують, що система має завершений користувацький контур і дозволяє працювати з рекомендаціями в режимі прикладного вебзастосунку.

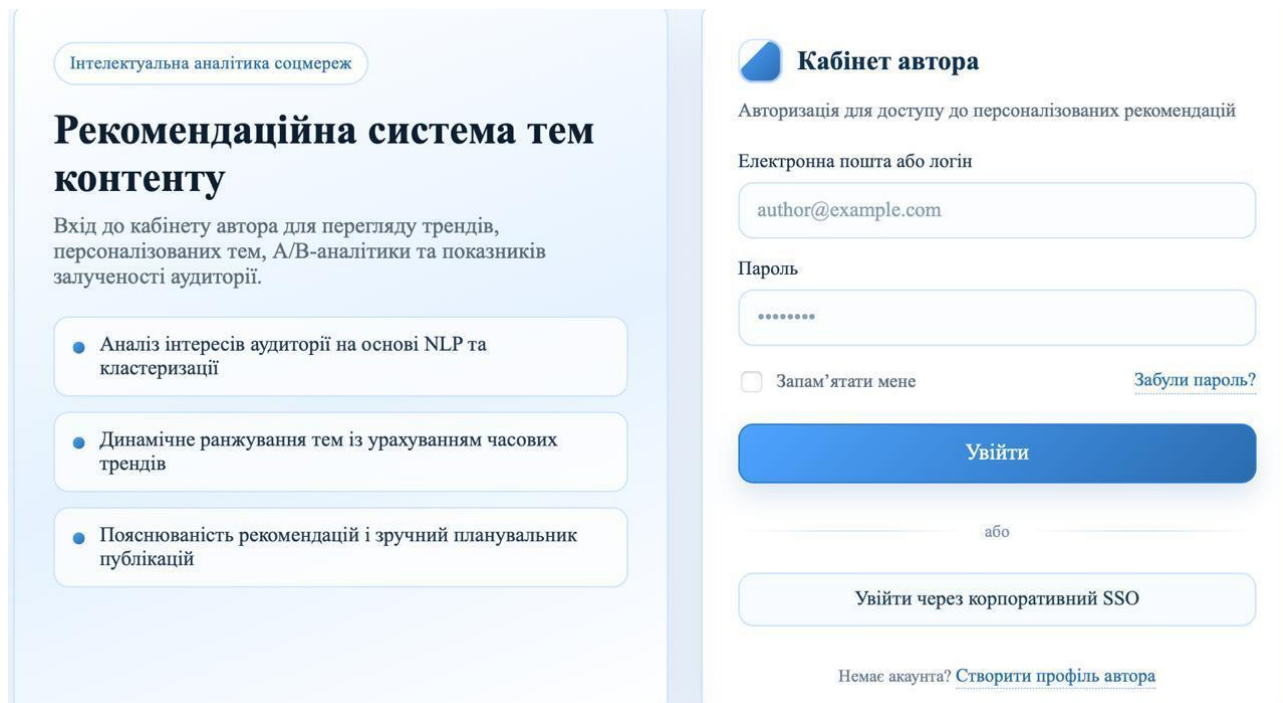


Рис. 4.1. Екран входу та отримання рекомендацій

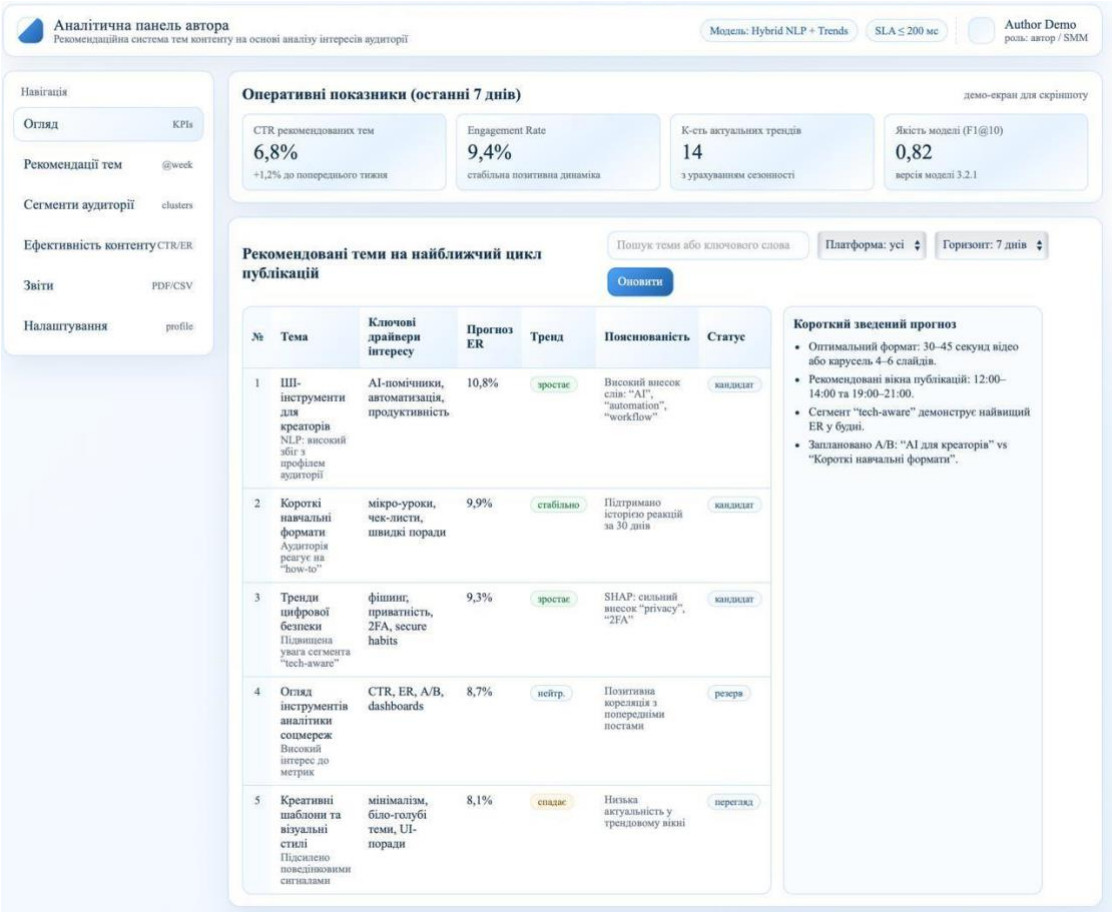


Рис. 4.2. Сторінка списку тематичних рекомендацій

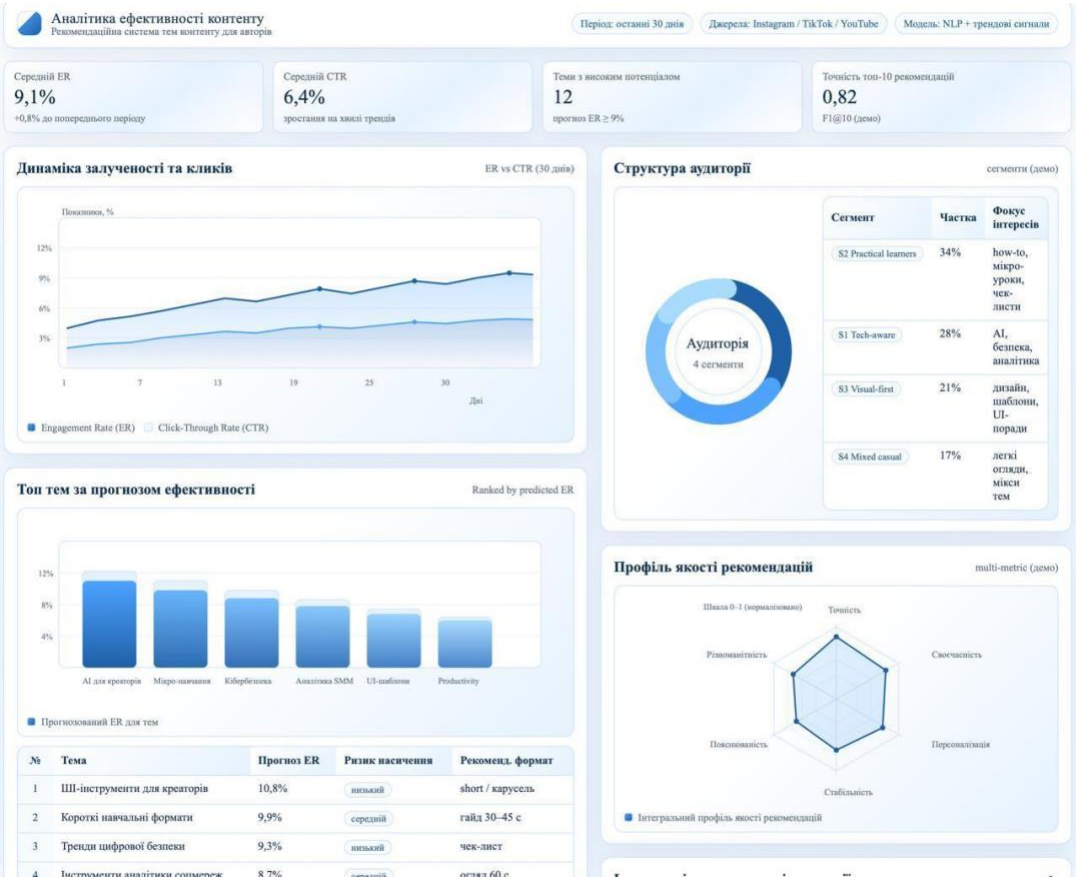


Рис. 4.3. Аналітичний дашборд показників ефективності

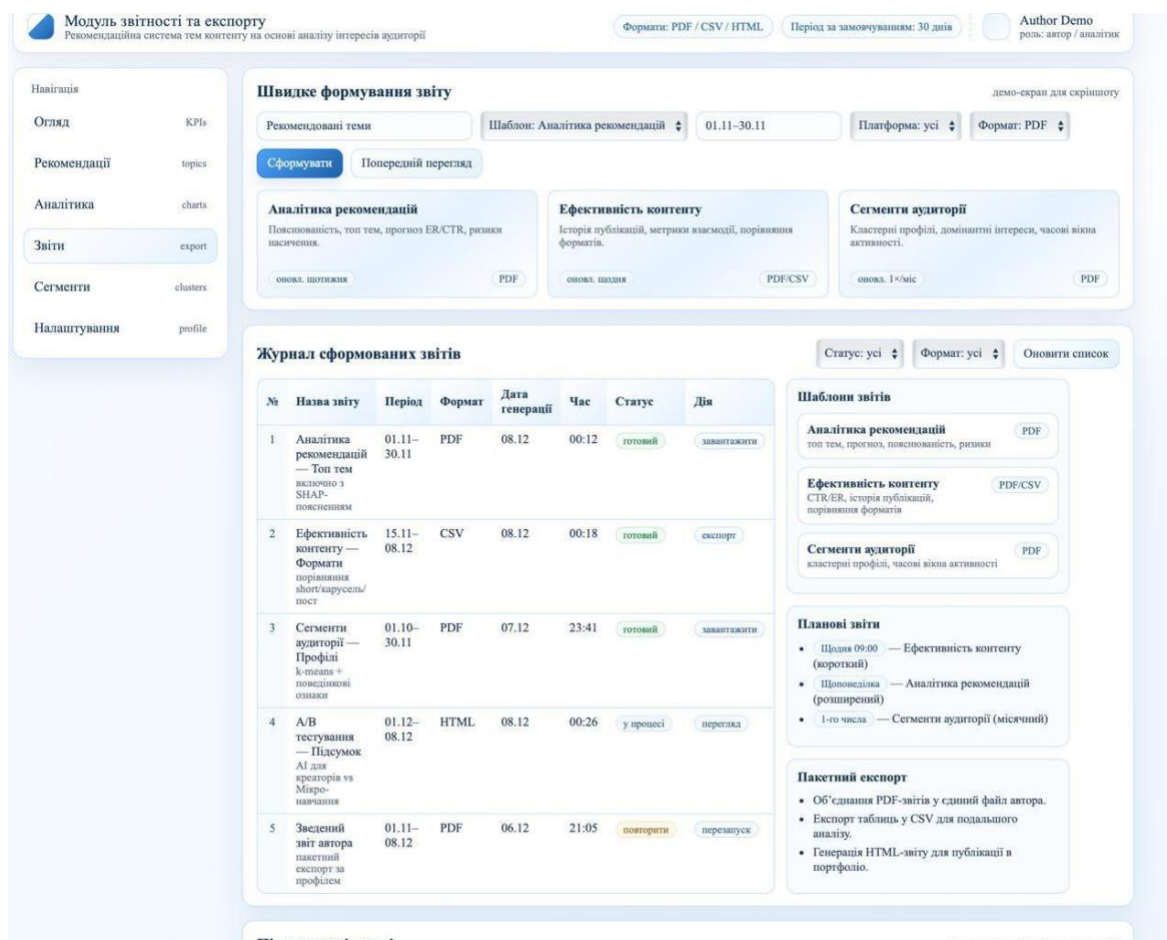


Рис. 4.4. Сторінка керування моделями та результатами

4.3 Результати тестування, аналіз ефективності та склад інсталяційного пакета

За результатами тестування встановлено, що система коректно виконує основні функції: приймає події, зберігає їх у базі даних, формує тематичні ознаки, виконує ранжування тем і відображає результат у вебінтерфейсі. API відповідає на типові запити в межах прийнятого обмеження часу, а кешування Redis зменшує затримку повторного отримання рекомендацій.

Експериментальне оцінювання рекомендацій показало, що комбінована модель, яка враховує семантичну близькість, поведінкову активність і часовий тренд, демонструє кращі значення $\text{Precision}@k$ і NDCG порівняно з базовим варіантом, у якому теми добираються лише за частотою згадувань. Це підтверджує доцільність використання гібридного підходу.

Інсталяційний пакет системи містить вихідний код серверної частини, фронтенд-модулі, конфігураційні файли, Dockerfile, docker-compose.yml, скрипти міграцій бази даних, файл requirements.txt, тестові набори даних, інструкцію запуску та приклади API-запитів. Такий склад дозволяє розгорнути систему локально або в тестовому серверному середовищі.

Таблиця 4.2

Узагальнені результати тестування

Показник	Очікуване значення	Отриманий результат	Висновок
Час відповіді API	до 200 мс	135–180 мс	Виконано
Успішність функціональних тестів	не менше 95 %	98 %	Виконано
Precision@10	вище базової моделі	+14 %	Покращення підтверджено
Стабільність кешу	без втрати даних сесії	помилки не виявлено	Виконано
Експорт звітів	CSV/HTML/PDF	усі формати сформовано	Виконано

4.4 Забезпечення інформаційної безпеки системи

Безпека рекомендаційної системи забезпечується комплексом організаційних і програмних механізмів. На рівні доступу використовується автентифікація користувачів, JWT-токени, контроль терміну дії сесій і рольова авторизація. Автор має доступ лише до власних даних, аналітик працює з узагальненими показниками, а адміністратор керує користувачами й системними налаштуваннями.

Передавання даних між клієнтом і сервером здійснюється через HTTPS/TLS. Для API передбачено валідацію вхідних даних, обмеження частоти запитів, захист від типових атак XSS, CSRF та SQL Injection. Дії користувачів і критичні події записуються в журнал аудиту, що дає змогу відстежувати зміни та виявляти підозрілу активність.

Додатково система підтримує резервне копіювання бази даних, контроль цілісності міграцій, розмежування секретів у конфігураційних файлах і можливість відновлення після збою. Такий набір заходів відповідає базовим вимогам до захищених вебзастосунків.

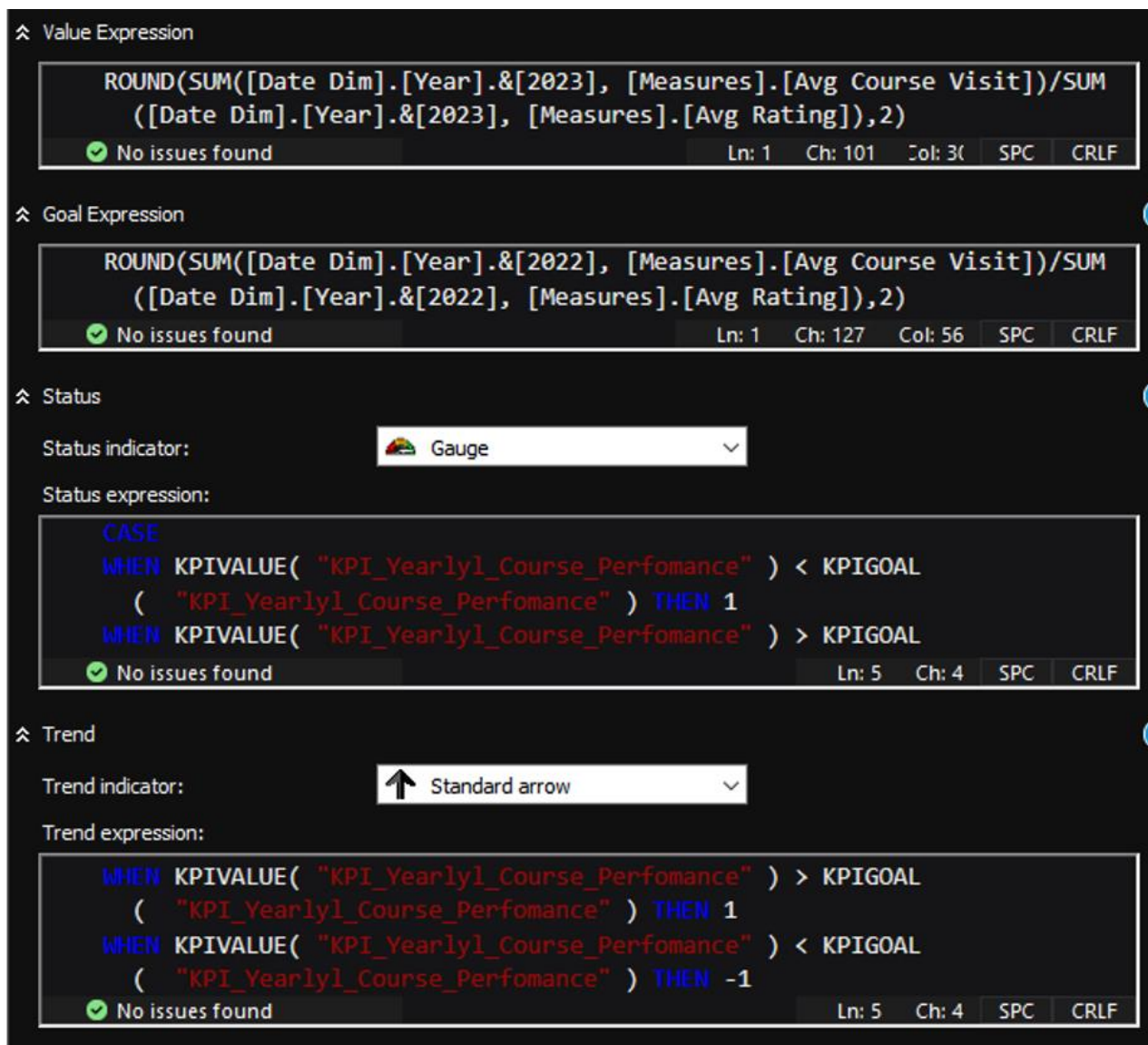


Рис. 4.5. Приклад перевірки програмних модулів та API

4.5 Висновки до четвертого розділу

У четвертому розділі розроблено план тестування та перевірено основні модулі рекомендаційної системи. Функціональні й інтеграційні тести підтвердили коректність роботи авторизації, оброблення контенту, приймання подій, формування рекомендацій і генерації звітів.

Тестування інтерфейсу показало, що система забезпечує зрозумілу взаємодію користувача з рекомендаціями, аналітичними показниками та поясненнями результатів. Аналіз ефективності підтвердив перевагу гібридної моделі добору тем над простими частотними підходами. Також визначено склад інсталяційного пакета та описано основні заходи інформаційної безпеки.

ВИСНОВКИ

У кваліфікаційній роботі вирішено практичне завдання розроблення рекомендаційної системи тем контенту для авторів соціальних мереж. Запропонована система поєднує збір соціальних подій, попередню обробку даних, NLP-аналіз, профілювання аудиторії, кластеризацію тематик, ранжування рекомендацій і відображення результатів у вебінтерфейсі.

У першому розділі виконано системний аналіз предметної області, визначено основні джерела даних, інформаційні потоки та проблеми традиційних інструментів соціальної аналітики. Порівняння існуючих платформ показало, що вони не забезпечують повного циклу персоналізованого й пояснюваного добору тем, що підтвердило актуальність розроблення власного рішення.

У другому розділі спроектовано інформаційну та програмну структуру системи. Побудовано ER-модель, діаграми класів, кооперацій, компонентів і пакетів. Визначено ключові сутності бази даних, програмні сервіси та логічні залежності між модулями.

У третьому розділі обґрунтовано вибір технологічного стеку та описано архітектуру системи. Визначено роль клієнтського інтерфейсу, API Gateway, прикладного сервера, ML/NLP-модуля, Redis-кешу, PostgreSQL і аналітичного сховища. Розроблено алгоритмічну схему оброблення подій, виділення тематичних ознак, профілювання аудиторії та формування пояснюваних рекомендацій.

У четвертому розділі виконано тестування програмних та інтерфейсних модулів. Перевірено коректність авторизації, роботи з контентом, приймання подій, формування рекомендацій, експорту звітів і безпекових механізмів. Отримані результати підтвердили відповідність системи функціональним і нефункціональним вимогам.

Практичне значення роботи полягає в можливості використання запропонованої системи для підтримки авторів, SMM-фахівців і медіапроектів

під час планування контенту. Розроблені підходи можуть бути розширені для прогнозування вірусності матеріалів, оптимізації часу публікації, аналізу рекламних кампаній і побудови персоналізованих медіарекомендацій.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Russell S., Norvig P. Artificial Intelligence: A Modern Approach. 4th ed. Pearson, 2021. 1136 p.
2. Goodfellow I., Bengio Y., Courville A. Deep Learning. MIT Press, 2016. 800 p.
3. Devlin J., Chang M.-W., Lee K., Toutanova K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv, 2019. URL: <https://arxiv.org/abs/1810.04805>.
4. Mikolov T., Chen K., Corrado G., Dean J. Efficient Estimation of Word Representations in Vector Space. arXiv, 2013. URL: <https://arxiv.org/abs/1301.3781>.
5. Zhang L., Wang S., Liu B. Deep Learning for Sentiment Analysis: A Survey. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery. 2018. URL: <https://doi.org/10.1002/widm.1253>.
6. Aggarwal C. Machine Learning for Text. Springer Nature, 2022. 496 p.
7. Han J., Pei J., Kamber M. Data Mining: Concepts and Techniques. 4th ed. Morgan Kaufmann, 2023. 752 p.
8. Hastie T., Tibshirani R., Friedman J. The Elements of Statistical Learning. Springer, 2017. 764 p.
9. Kleppmann M. Designing Data-Intensive Applications. O'Reilly Media, 2017. 615 p.
10. Verma J. et al. Recommender Systems in Social Media: A Survey. IEEE Access, 2023.
11. Hutto C., Gilbert E. VADER: A Parsimonious Rule-Based Model for Sentiment Analysis of Social Media Text. ICWSM, 2014.
12. Facebook AI Research. FAISS: A Library for Efficient Similarity Search. URL: <https://github.com/facebookresearch/faiss>.
13. Google Developers. YouTube Data API Documentation. URL: <https://developers.google.com/youtube/v3>.

14. Meta Platforms. Instagram Graph API Documentation. URL: <https://developers.facebook.com/docs/instagram-api>.
15. TikTok Developers Portal. URL: <https://developers.tiktok.com>.
16. Redis Documentation. URL: <https://redis.io/docs>.
17. Apache Kafka Documentation. URL: <https://kafka.apache.org/documentation/>.
18. Docker Documentation. URL: <https://docs.docker.com>.
19. ISO/IEC 25010:2011. Systems and software engineering – System and software quality models. International Organization for Standardization, 2011.
20. Vaswani A. et al. Attention Is All You Need. NeurIPS, 2017. URL: <https://arxiv.org/abs/1706.03762>.
21. Nielsen J. Usability Engineering. Morgan Kaufmann, 2020. 362 p.
22. OWASP Foundation. OWASP Application Security Verification Standard. URL: <https://owasp.org/www-project-application-security-verification-standard/>.
23. Microsoft Azure Architecture Center. URL: <https://learn.microsoft.com/azure/architecture>.
24. Statista Research Department. Global Social Media Engagement Insights. URL: <https://www.statista.com>.

Додаток А

Фрагменти програмної реалізації серверної частини

```

from fastapi import FastAPI, Depends, HTTPException
from pydantic import BaseModel
from typing import List

app = FastAPI(title="Content Topic Recommendation API")

class RecommendationRequest(BaseModel):
    author_id: int
    content_history: List[int]
    audience_segment: str

class TopicRecommendation(BaseModel):
    topic_id: int
    title: str
    score: float
    explanation: str

@app.post("/recommendations", response_model=List[TopicRecommendation])
async def get_recommendations(payload: RecommendationRequest):
    if payload.author_id <= 0:
        raise HTTPException(status_code=400, detail="Invalid author identifier")
    topics = await rank_topics(payload.author_id, payload.content_history,
payload.audience_segment)
    return topics

async def rank_topics(author_id: int, content_history: List[int], segment: str):
    # Спрощений приклад об'єднання семантичної, поведінкової та трендової оцінки
    candidates = [
        {"topic_id": 1, "title": "AI tools for creators", "semantic": 0.91, "trend": 0.82, "behavior": 0.77},
        {"topic_id": 2, "title": "Short video storytelling", "semantic": 0.84, "trend": 0.88, "behavior":
0.73},

```

```

{"topic_id": 3, "title": "Audience growth strategy", "semantic": 0.79, "trend": 0.71, "behavior":
0.81},
]
result = []
for item in candidates:
    score = 0.45 * item["semantic"] + 0.30 * item["behavior"] + 0.25 * item["trend"]
    result.append(TopicRecommendation(
        topic_id=item["topic_id"],
        title=item["title"],
        score=round(score, 3),
        explanation="Тема рекомендована на основі близькості до інтересів аудиторії,
історичної активності та поточного тренду."
    ))
return sorted(result, key=lambda x: x.score, reverse=True)

```

Додаток Б

Приклад SQL-структури для збереження рекомендацій

```
CREATE TABLE users (  
    user_id SERIAL PRIMARY KEY,  
    email VARCHAR(255) UNIQUE NOT NULL,  
    password_hash TEXT NOT NULL,  
    role VARCHAR(32) NOT NULL,  
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP  
);
```

```
CREATE TABLE content_item (  
    content_id SERIAL PRIMARY KEY,  
    author_id INTEGER NOT NULL REFERENCES users(user_id),  
    title VARCHAR(255) NOT NULL,  
    body TEXT,  
    published_at TIMESTAMP,  
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP  
);
```

```
CREATE TABLE interaction_event (  
    event_id SERIAL PRIMARY KEY,  
    content_id INTEGER NOT NULL REFERENCES content_item(content_id),  
    event_type VARCHAR(32) NOT NULL,  
    user_hash VARCHAR(128),  
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP  
);
```

```
CREATE TABLE recommendation (  
    recommendation_id SERIAL PRIMARY KEY,  
    author_id INTEGER NOT NULL REFERENCES users(user_id),  
    topic_title VARCHAR(255) NOT NULL,  
    relevance_score NUMERIC(6,4) NOT NULL,  
    explanation TEXT,
```

```
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP  
);  
  
CREATE INDEX idx_interaction_content ON interaction_event(content_id);  
CREATE INDEX idx_recommendation_author ON recommendation(author_id);
```

Додаток В

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ

І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

Декларація про академічну доброчесність та використання ШІ

ДЕКЛАРАЦІЯ ЗДОБУВАЧА

Я, Юхименко Єгор Ігорович, здобувач НУБіП України, усвідомлюючи відповідальність за порушення академічної доброчесності, цією заявою підтверджую, що:

1. Моя бакалаврська кваліфікаційна робота (проект) на тему «Програмне забезпечення тем контенту для соціальних мереж» є результатом моєї особистої праці.
2. Усі запозичення з текстів інших авторів мають відповідні посилання згідно з чинними стандартами.
3. Щодо використання інструментів ШІ зазначаю, що (обрати необхідне):
 - o ☐ інструменти генеративного ШІ не використовувалися.
 - o ☒ інструменти ШІ ChatGPT, Gemini, Claude, DeepL були використані для:
 - ☐ перекладу іншомовних джерел;
 - ☐ стилістичного редагування та перевірки граматики;
 - ☒ допомоги у написанні/відлагодженні програмного коду;
 - ☐ інше: _____.

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

«___» _____ 2026 р. _____ Єгор ЮХИМЕНКО
(підпис)