Міністерство освіти і науки України
Український державний університет науки і технологій

Комп'ютерні технології і системи

Комп'ютерні інформаційні технології

Пояснювальна записка
до кваліфікаційної роботи
Магістр

на тему: «Дослідження алгоритмів рекомендаційного механізму в платформі з оцінюванням медіаконтенту»

за освітньою програмою 12 Інженерія програмного забезпечення
зі спеціальності: 121 Інженерія програмного забезпечення

Виконав: студент групи ПЗ2422 Максим ПОПОВ

Керівник: к. т. н., доцент, Тетяна ГРИШЕЧКІНА

Нормоконтролер: к. физ-мат.н., доцент, Світлана ВОЛКОВА

Засвідчую, що у цій роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент

Дніпро 2026

Ministry of Education and Science of Ukraine
Ukrainian State University of Science and Technologies

Computer technologies and systems

Computer information technologies

Explanatory Note
to Master's Thesis
Master's degree

on the topic: «Research on the Algorithms For the Recommendation Mechanism In the Platform For Evaluating Media Content»

according to educational curriculum 12 Software engineering

in the Speciality: 121 Software engineering

Done by the student of the group: 962M Maksym POPOV

Scientific Supervisor: Tetyana HRYSHECHKINA

Normative controller : Svitlana VOLKOVA

Dnipro – 2026

Міністерство освіти і науки України
Український державний університет науки і технологій

Факультет: Комп'ютерні технології і системи
Кафедра: Комп'ютерні інформаційні технології
Рівень вищої освіти: Магістр
Освітня програма: F Інженерія програмного забезпечення
Спеціальність: F2 Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри _____
_____ Вадим ГОРЯЧКІН

Дата _____

З А В Д А Н Н Я

на кваліфікаційну роботу Магістра

студенту Попову Максиму Сергійовичу

1. Тема роботи: Дослідження алгоритмів рекомендаційного механізму в платформі з оцінюванням медіаконтенту

Керівник роботи: Гришечкіна Тетяна Сергіївна, кандидат технічних наук, доцент

затверджені наказом від "02" 10 2025 р. №1401ст.

2. Строк подання студентом роботи: 11.01.2026 р.
3. Вихідні дані до роботи: алгоритми рекомандацій медіаконтенту, метрики дослідження
4. Зміст пояснювальної записки (перелік питань, які потрібно опрацювати):
Аналіз предметної області, аналіз існуючих рішень,
дослідження аудиторії щодо ефективності рекомендаційного механізму,
методологія та метрики оцінки ефективності рекомендаційних алгоритмів,
реалізація рекомендаційної системи, представлення роботи рекомендаційної системи
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень):
презентація, відеоролик, графічне представлення результатів розробки програми

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Описати вступну частину кваліфікаційної роботи		
2	Виконати аналіз літературних джерел		
3	Виконати дослідження предметної області		
4	Провести дослідження аудиторії	12.11.2025	30%
5	Виконати вибір метрик оцінки ефективності та обґрунтувати їх		
6	Обрати інструменти розробки	09.12.2025	60%
	Розробка додатку	26.11.2025	100%
7	Подання кваліфікаційної роботи до кафедри	09.01.2026	
8	Захист кваліфікаційної роботи на засіданні Екзаменаційної комісії	20.01.2026	

Студент _____ Максим ПОПОВ

Керівник роботи _____ Тетяна ГРИШЕЧКІНА

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи магістра 181с., 35рис., 71 джерело.

Об'єкт дослідження – це процес створення рекомендацій для користувачів платформи з оцінками медіа.

Предмет дослідження – це конкретні алгоритми, які формують ці рекомендації, їхня швидкість роботи та актуальність для людей.

Метою роботи є розробка ефективної рекомендаційної системи для відеохостингу, яка враховує не лише прямі оцінки (лайки/дизлайки), а й поведінкові дані користувачів. Також метою є виявити беспосередньо поведінку цільової аудиторії та закономірностей споживання медіаконтенту на основі аналізу сучасних користувацьких даних для винаходження ефективного алгоритму рекомендацій та мінімізації утворення "фільтрувальних бульбашок".

Методи дослідження включають теоретичний аналіз наукових джерел щодо рекомендаційних систем та порівняльне дослідження різних алгоритмічних підходів до рекомендацій медіа контенту.

Апаратна частина дослідження базується на стандартній комп'ютерній конфігурації. Програмна реалізація використовує сучасний стек веб-технологій: фреймворк Vue.js для фронтенду, платформу Firebase для бекенду та аутентифікації, а також хмарний сервіс Cloudinary для обробки медіафайлів.

Результатом роботи є розробка рекомендаційної системи для відеохостингу, яка аналізує як прямі оцінки користувачів, так і їх поведінкові патерни. Система дозволяє точно визначати вподобання аудиторії та закономірності споживання контенту, що підвищує релевантність рекомендацій та зменшує ризик виникнення "фільтрувальних бульбашок".

Ключові слова: МЕДІАКОНТЕНТ, АЛГОРИТМ, ФІЛЬТРУВАЛЬНІ БУЛЬБАШКИ, РЕКОМЕНДАЦІЙНІ СИСТЕМИ.

ЗМІСТ

ВСТУП

На сьогодні люди стикаються з величезною кількістю фільмів, серіалів, музики та іншого медіа. Величезна бібліотека мільйонів фільмів, серіалів та пісень не полегшує, а ускладнює пошук. Вибір стає виснажливим, а ймовірність пропустити якісний контент, який ідеально підходить саме вам, — зростає. Наприклад, користувач Netflix може витратити більше часу на проглядання анонсів, ніж на сам перегляд. Саме тут системи рекомендацій стають не прикрасою, а основним механізмом навігації. Вони трансформують пасивне сховище в активне, персоналізоване середовище, де контент "знаходить" користувача, а не навпаки.

Актуальність цієї проблеми дуже висока. Для платформ, які живуть за рахунок уваги користувачів, якість рекомендацій — це питання виживання. Якщо людина швидко знаходить щось корисне або цікаве, вона залишається на сайті. Якщо ні — йде.

Однак створення хороших рекомендацій — це не просто технічне завдання. Існує кілька ключових викликів:

1. Проблема "холодного старту": Як рекомендувати щось новому користувачеві, про якого система ще нічого не знає? Або як рекомендувати абсолютно новий фільм, який ще ніхто не оцінював?;

2. Проблема "фільтрувальної бульбашки": Дуже часто алгоритми, намагаючись бути максимально точними, починають пропонувати користувачеві дуже схожий контент. Це призводить до того, що людина опиняється в інформаційній "бульбашці" — вона бачить тільки те, що відповідає її минулим переглядам, і не дізнається про щось нове або неочікуване. Це обмежує її досвід.

Мета роботи – дослідити різні алгоритми рекомендацій, порівняти їх між собою та зрозуміти, які найкраще підходять для нашої задачі. Ми оцінимо їх не тільки за числами (наскільки точні), але і за думкою самих користувачів.

Щоб досягти цієї мети, потрібно виконати кілька завдань:

1. Вивчити, як працюють рекомендаційні системи;
2. Розглянути основні типи алгоритмів, які для цього використовуються;
3. Зрозуміти, які алгоритми швидкі, а які повільні;
4. Опитати користувачів, які рекомендації їм подобаються, а які – ні;
5. На основі цього аналізу запропонувати та реалізувати власну модель рекомендацій;
6. Перевірити її роботу, оцінити за метриками та порівняти результати з думками людей;
7. Зробити висновки про найкращий підхід.

Актуальною задачею є створення такої системи, яка була б одночасно точною (пропонувала те, що сподобається) і різноманітною (знайомила з новими темами чи жанрами), уникаючи створення "бульбашки".

Існує багато різних способів створювати такі рекомендації, і потрібно зрозуміти, які з них найкраще підходять саме для платформ з оцінками медіаконтенту. Деякі алгоритми швидкі, інші точніші, треті краще рекомендують нові, невідомі позиції. Важливо знайти баланс.

Таким чином, ключовим етапом є практична реалізація та тестування алгоритмів у реальних умовах. У цій роботі буде розроблено функціональний прототип платформи для оцінювання відеоконтенту з інтегрованою рекомендаційною системою. Цей прототип створить необхідну основу для експериментального дослідження — стане тестовим середовищем, де можна буде на практиці порівняти різні алгоритми та визначити найбільш збалансоване рішення для рекомендації контенту на основі оцінок.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Визначення предметної області та об'єкта дослідження

Предметною областю даного дослідження є рекомендаційні системи медіаконтенту, які функціонують в умовах оцінювання та ранжування контенту користувачами. Об'єктом дослідження виступають алгоритмічні підходи до генерації персоналізованих рекомендацій на основі явних та неявних оцінок користувачів щодо медіаматеріалів різних типів. Границі дослідження охоплюють програмні системи, що забезпечують автоматизовану фільтрацію та ранжування медіаконтенту з метою максимізації задоволеності користувачів та бізнес-показників платформи.

Актуальність даної предметної області обумовлена експоненціальним [1] зростанням обсягів цифрового медіаконтенту та необхідністю забезпечення ефективної навігації користувачів у інформаційному просторі. За даними досліджень, середній користувач сучасної медіаплатформи має доступ до мільйонів одиниць контенту, але фізично здатен споживати лише незначну частину від загального обсягу, що створює критичну потребу в якісних механізмах фільтрації та персоналізації.

1.2 Систематизація існуючих підходів та методологій

На основі аналізу наукової літератури та практичних реалізацій можна виділити три основні методологічні парадигми рекомендаційних систем. Парадигма колаборативної фільтрації базується на припущенні про подібність вподобань користувачів зі схожими історичними паттернами взаємодії з контентом. Дослідження Resnick та Varian [2] довели ефективність цього підходу для великих користувацьких баз, проте виявили критичні обмеження у вигляді проблеми холодного старту та розрідженості користувацько-об'єктної матриці.

Парадигма контент-орієнтованої фільтрації фокусується на аналізі внутрішніх характеристик медіаматеріалів та їх відповідності профілю користувача. Роботи Pazzani та Billsus [3] демонструють високу ефективність цього підходу для текстового контенту, однак його застосування до

мультимедійних матеріалів вимагає складних методів вилучення семантичних ознак. Гібридні підходи, запропоновані Burke [4], поєднують переваги обох парадигм, але привносять додаткову алгоритмічну складність та вимагають більших обчислювальних ресурсів.

1.3 Аналіз поточного стану галузі та ключових гравців

Сучасна індустрія рекомендаційних систем характеризується домінуванням великих технологічних платформ, що володіють критичними масивами користувацьких даних. Netflix інвестував понад 150 мільйонів доларів у розробку власної рекомендаційної системи [5] та щорічно економить більше мільярда доларів завдяки зменшенню відтоку підписників. YouTube обробляє понад 2 мільярди годин відео щоденно, використовуючи багаторівневу архітектуру рекомендацій, що включає етапи кандидатського відбору та ранжування.

Amazon демонструє найвищу монетизацію рекомендаційних технологій, генеруючи значну загального доходу через персоналізовані пропозиції. Spotify використовує унікальний підхід, що поєднує аудіоаналіз треків з колаборативною фільтрацією, досягаючи великого об’єму збільшення часу прослуховування завдяки персоналізованим плейлистам. Ці приклади демонструють критичну важливість рекомендаційних систем для бізнес-моделей сучасних медіаплатформ.

1.4 Виявлення проблемних аспектів та обмежень

Аналіз існуючих рішень виявляє кілька критичних проблем, що потребують подальшого дослідження. Проблема масштабованості стає критичною при роботі з мільйонами користувачів та мільярдами об'єктів контенту. Традиційні алгоритми колаборативної фільтрації мають квадратичну складність відносно кількості користувачів, що робить їх непрактичними для великих платформ без значної оптимізації.

Проблема розрідженості даних особливо гостро проявляється у медіагалузі, де більшість користувачів взаємодіє лише з невеликою частиною доступного контенту. Це призводить до неточності прогнозів та зниження якості

рекомендацій для користувачів з нестандартними вподобаннями. Проблема холодного старту ускладнює onboarding нових користувачів та введення нового контенту в систему.

Етичні проблеми включають формування інформаційних бульбашок, алгоритмічну упередженість та непрозорість рекомендаційних рішень. Дослідження показують, що рекомендаційні системи можуть підсилювати існуючі соціальні упередження та обмежувати інформаційне різноманіття, впливаючи на культурний розвиток суспільства. Дослідження показують, що рекомендаційні алгоритми часто демонструють упередженість популярності, зосереджуючись на популярних елементах у своїх рекомендаціях, що може призводити до ефектів підсилення з часом. Алгоритми, які надають пріоритет домінуючим наративам, підривають культурний плюралізм та загрожують життєздатній культурній екосистемі, яка потребує процвітання різноманітних голосів [6].

1.5 Технічні виклики та методологічні пробіли

Дослідження виявляє значні пробіли у методологіях оцінювання якості рекомендаційних систем. Традиційні метрики точності (Precision@K, кількість релевантних елементів у топ-K, серед даних) та повноти (Recall@K, кількість релевантних у топ-K серед загальної кількості релевантних) не повною мірою відображають користувацький досвід та бізнес-цінність рекомендацій.

Наразі обмеження традиційних метрик такі [7]:

- не враховують порядок рекомендацій;
- не відображають користувацький досвід;
- ігнорують різноманітність та новизну;
- не вимірюють неочікуваність (serendipity);
- не враховують довготривалу поведінку користувачів.

Потребує розвитку комплексний підхід до оцінювання, що враховує різноманітність, новизну, серендипність та довготривалий вплив на поведінку користувачів.

Проблема інтерпретованості алгоритмічних рішень стає критичною в контексті зростаючих регуляторних вимог щодо прозорості штучного інтелекту. Сучасні методи глибокого навчання забезпечують високу точність прогнозів, але функціонують як "чорні скриньки", що ускладнює пояснення логіки рекомендацій користувачам та регуляторам.

Динамічність користувацьких переваг та швидкість змін у медіаландшафті вимагають розробки адаптивних алгоритмів, здатних до швидкого навчання на нових даних. Існуючі підходи часто припускають стаціонарність користувацьких переваг, що не відповідає реальності динамічного медіаспоживання.

1.6 Перспективні напрямки досліджень

Наразі можна визначити кілька напрямків для подальших досліджень. Першим напрямком є розвиток контекстуально-адаптивних алгоритмів, які враховують не лише історію переглядів користувача, але й поточну ситуацію: час доби (вранці люди частіше дивляться новини, увечері - розважальний контент), місце перебування (вдома чи в дорозі), використовуваний пристрій (смартфон чи телевізор) та поточний настрій користувача. Наприклад, система може рекомендувати короткі відео для перегляду в транспорті та довгі фільми для домашнього перегляду.

Другим важливим напрямком є дослідження методів інкрементального навчання [8], які дозволяють системі постійно адаптуватися до змін смаків користувачів без необхідності повного перенавчання всієї моделі з нуля. Це особливо важливо, оскільки вподобання людей змінюються з часом: підліток може перейти від перегляду мультфільмів до серіалів для дорослих, а любитель комедій може зацікавитися документальними фільмами. Традиційні системи потребують повного перенавчання при таких змінах, що вимагає величезних обчислювальних ресурсів.

А саме що собою являє інкрементальне навчання (Incremental Learning) — це процес поступового навчання моделі на нових даних без необхідності повного перенавчання з нуля. Хоча безперервне навчання є природною

навичкою для людського мозку, воно є надзвичайно складним для штучних нейронних мереж, оскільки під час вивчення нового вони швидко та кардинально забувають те, чому навчилися раніше. Серед переваг такого підходу можна виділити економію обчислювальних ресурсів, тобто, немає потреби перенавчувати всю модель та швидкість адаптації — система швидко реагує на зміни уподобань.

Розробка пояснюваних рекомендаційних систем стає критично важливою в умовах зростаючих вимог до прозорості алгоритмів. Користувачі хочуть розуміти, чому система рекомендує саме цей фільм чи пісню. Замість простого "Вам може сподобатись" система повинна пояснювати: "Рекомендуємо цей фільм, тому що ви позитивно оцінили три інші фільми цього режисера та часто дивитесь трилери". Це підвищує довіру користувачів та дозволяє їм краще контролювати процес рекомендацій.

Також одним із напрямків є створення мультимодальних систем, які аналізують не лише цифрові оцінки користувачів, але й різноманітну інформацію про сам контент. Мультимодальні рекомендаційні системи засновані на спостереженні, що уподобання користувачів до елементів — особливо в таких сферах, як електронна комерція, медіа та розваги — часто залежать від комбінації сигналів: візуальної естетики, текстових описів, поведінки користувачів та навіть аудіо або контекстних підказок [9]. Для фільму це можуть бути кадри з фільму (визначення жанру за зображенням), саундтрек (енергійна чи спокійна музика), діалоги (аналіз емоційного забарвлення), а також традиційні метадані як актори та режисер. Така комплексна система краще розуміє, що саме подобається користувачу: чи то візуальний стиль фільмів Вес Андерсона, чи енергійна музика у бойовиках, чи специфічний гумор британських комедій.

1.7 Класифікація існуючих рекомендаційних систем медіаконтенту

Сучасний ландшафт рекомендаційних систем медіаконтенту характеризується великою різноманітністю підходів та архітектурних рішень, що відрізняються за методологією, областю застосування та технічними

характеристиками. Для систематичного аналізу доцільно розглянути існуючі рішення за кількома критеріями: типом алгоритмічного підходу, масштабом платформи, специфікою медіаконтенту та бізнес-моделлю.

За алгоритмічним підходом можна виділити системи, що використовують колаборативну фільтрацію, контент-орієнтовану фільтрацію, гібридні методи та системи на основі глибокого навчання. Колаборативна фільтрація домінує у платформах з великими користувацькими базами, де наявність мільйонів взаємодій дозволяє ефективно виявляти паттерни схожості між користувачами. Контент-орієнтовані підходи частіше використовуються у спеціалізованих платформах з чітко структурованим контентом, де характеристики медіаматеріалів легко піддаються формалізації.

За масштабом платформи рішення варіюються від глобальних мегаплатформ, що обслуговують сотні мільйонів користувачів, до нішевих сервісів з вузькоспеціалізованою аудиторією. Великі платформи зазвичай використовують складні багатоетапні архітектури з розподіленою обробкою даних, тоді як менші сервіси можуть дозволити собі більш прості, але гнучкі рішення.

1.8 Аналіз рішень провідних медіаплатформ

Netflix розробив одну з найбільш досконалих рекомендаційних систем у галузі, що базується на гібридному підході з використанням понад 1300 кластерів рекомендацій. Система Netflix функціонує у три основні етапи: генерація кандидатів, ранжування та фільтрація за бізнес-правилами. На етапі генерації кандидатів система аналізує історію переглядів користувача, використовуючи матричну факторизацію та глибокі нейронні мережі для виявлення латентних факторів у поведінці користувачів.

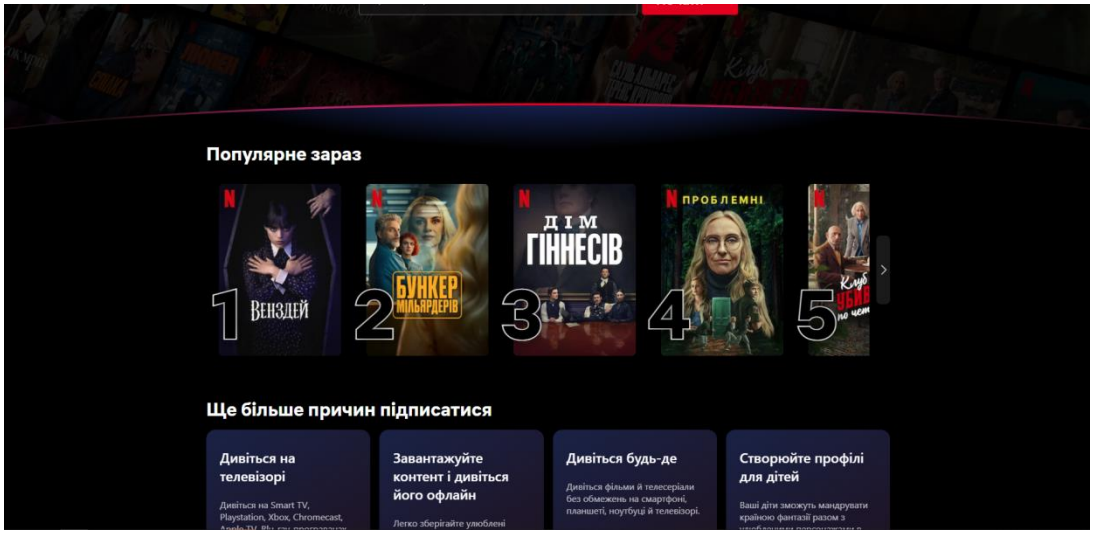


Рисунок 1.1 — Огляд сайту Netflix

Особливістю підходу Netflix є використання алгоритму SVD++ (Singular Value Decomposition) у поєднанні з аналізом неявного зворотного зв'язку. Система враховує не лише факт перегляду контенту, але й тривалість перегляду, паузи, перемотування та повторні перегляди. Це дозволяє більш точно оцінювати реальний рівень задоволеності користувача, оскільки явні оцінки надає менше 20% користувачів платформи.

Netflix також впровадив концепцію "смакових профілів", де кожен користувач може мати декілька різних профілів залежно від жанрових переваг. Система автоматично визначає, коли користувач перебуває в різних "режимах споживання": легкий вечірній перегляд після роботи, серйозний кінопоказ на вихідних або фоновий перегляд під час інших справ.

Технічна архітектура Netflix базується на мікросервісній архітектурі з використанням Apache Spark для обробки великих даних та Apache Kafka для стрімінгової обробки подій у реальному часі. Система обробляє понад 500 мільярдів подій на місяць, генеруючи персоналізовані рекомендації для кожного з 238 мільйонів підписників у понад 190 країнах.

YouTube стикається з унікальними викликами через величезний обсяг контенту, що завантажується щохвилини, та різноманітність форматів від коротких кліпів до багатогодинних трансляцій. Рекомендаційна система YouTube складається з двох основних компонентів: системи генерації

кандидатів та системи ранжування. Система генерації кандидатів використовує глибокі нейронні мережі для зменшення мільярдів відео до сотень кандидатів на основі історії користувача та контекстуальних сигналів.

Особливістю підходу YouTube є врахування "свіжості" контенту та здатності виявляти віральні тренди. Система використовує комбінацію колаборативної фільтрації з аналізом контенту, включаючи автоматичне розпізнавання об'єктів у відео, аналіз звукової доріжки та обробку природної мови для аналізу назв та описів відео. Алгоритм також враховує соціальні сигнали, такі як швидкість набору переглядів, коментарі та реакції користувачів.

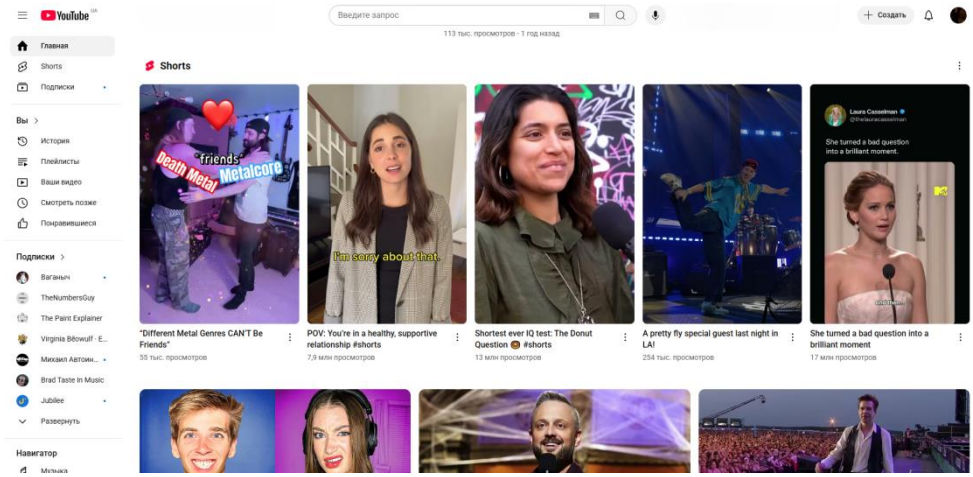


Рисунок 1.2 — Огляд сайту Youtube

YouTube впровадив концепцію "watch time optimization", де основним критерієм оптимізації є не кількість кліків, а загальний час перегляду. Це мотивує систему рекомендувати контент, який дійсно цікавий користувачам, а не просто привабливий за назвою чи мініатюрою. Система також враховує різноманітність рекомендацій, щоб уникнути "filter bubble" ефекту.

Технічно YouTube використовує TensorFlow для навчання моделей глибокого навчання та розподілену архітектуру на базі Google Cloud Platform. Система обробляє понад 2 мільярди годин перегляду щоденно та генерує рекомендації для більш ніж 2 мільярдів авторизованих користувачів щомісяця.

Spotify розробив унікальний підхід до музичних рекомендацій, що поєднує колаборативну фільтрацію з глибоким аналізом аудіосигналу. Система

використовує технологію Echo Nest для аналізу акустичних характеристик треків: темпу, тональності, енергійності, танцювальності та валентності (емоційного забарвлення). Це дозволяє системі розуміти музику на рівні, що наближається до людського сприйняття.

Особливістю Spotify є створення автоматичних плейлистів, таких як "Discover Weekly" та "Release Radar", які оновлюються щотижня для кожного користувача. Алгоритм "Discover Weekly" аналізує мільярди плейлистів, створених користувачами, та використовує метод Natural Language Processing для аналізу текстових описів музики в інтернеті, включаючи блоги, статті та рецензії.

Spotify також впровадив концепцію "музичного графу", де кожен трек пов'язаний з іншими треками через складну мережу співвідношень: схожість звучання, спільність виконавців, жанрові зв'язки та поведінкові паттерни користувачів. Система враховує контекстуальні фактори, такі як час доби, день тижня та активність користувача (бігання, робота, відпочинок).

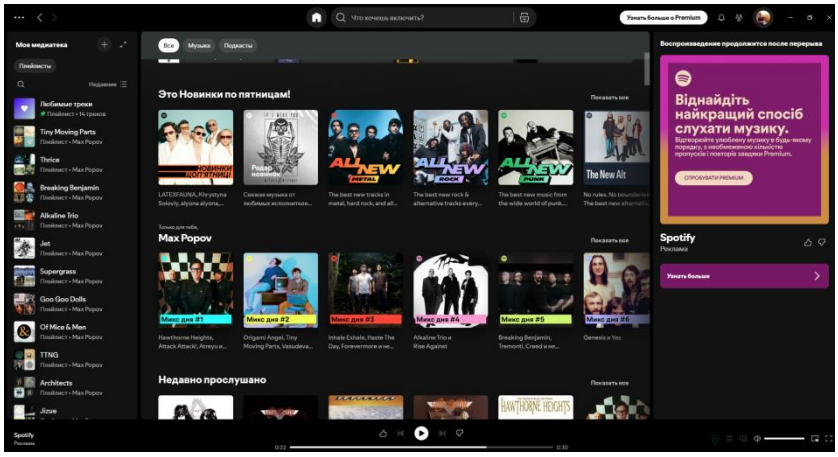


Рисунок 1.3 — Огляд додатку Spotify із персональними рекомендаціями

Технічна архітектура Spotify базується на Apache Storm для обробки стрімінгових даних та Hadoop для зберігання та аналізу великих даних. Система обробляє понад 40 мільярдів треків щорічно для 489 мільйонів користувачів та генерує персоналізовані рекомендації з урахуванням понад 100 різних факторів.

1.9 Аналіз спеціалізованих та нішевих рішень

Сфера подкастів представляє унікальні виклики для рекомендаційних систем через специфічний характер контенту: великі епізоди, складну тематичну структуру та необхідність врахування контекстуального споживання. Apple Podcasts використовує гібридний підхід, що поєднує аналіз метаданих подкастів з поведінковими паттернами користувачів. Система аналізує категорії подкастів, ключові слова в описах, популярність серед схожих користувачів та завершеність прослуховування епізодів.

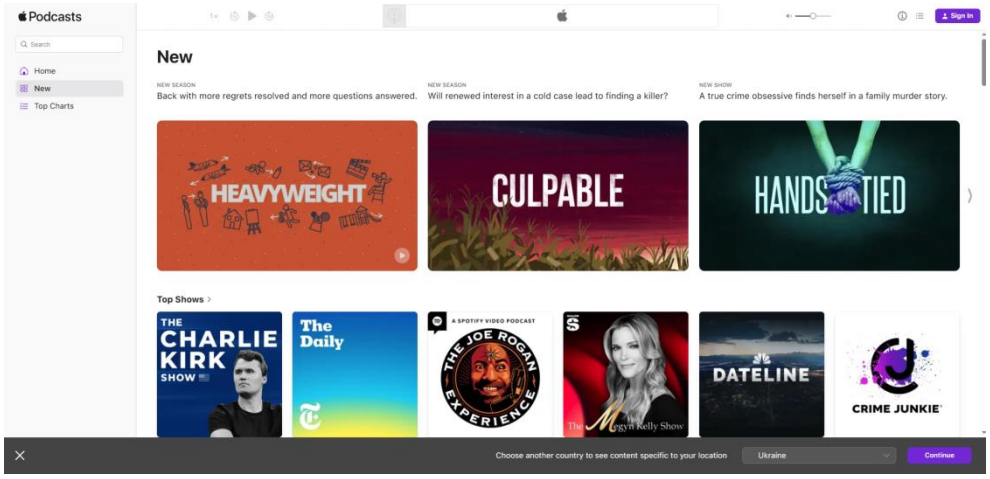


Рисунок 1.4 — Огляд сайту Apple Podcasts

Spotify адаптував свою музичну рекомендаційну систему для подкастів, використовуючи методи обробки природної мови для аналізу транскрипцій епізодів. Це дозволяє системі розуміти тематичний зміст подкастів та рекомендувати релевантні епізоди на основі інтересів користувача, виявлених через його музичні вподобання та попередні прослуховування подкастів.

Amazon Kindle розробив складну систему рекомендацій електронних книг, що використовує не лише історію покупок, але й поведінку читання: швидкість читання, частота сесій, використання закладок та виділень тексту. Система також аналізує рецензії користувачів та рейтинги для виявлення схожих читацьких смаків.

Goodreads фокусується на соціальному аспекті читання, використовуючи рекомендації друзів та читацьких спільнот. Система аналізує полички книг користувачів, їхні оцінки та рецензії для пошуку читачів зі схожими смаками.

Особливістю є врахування жанрових переваг та здатність виявляти читачів, які люблять "інді" або мейнстрімну літературу.

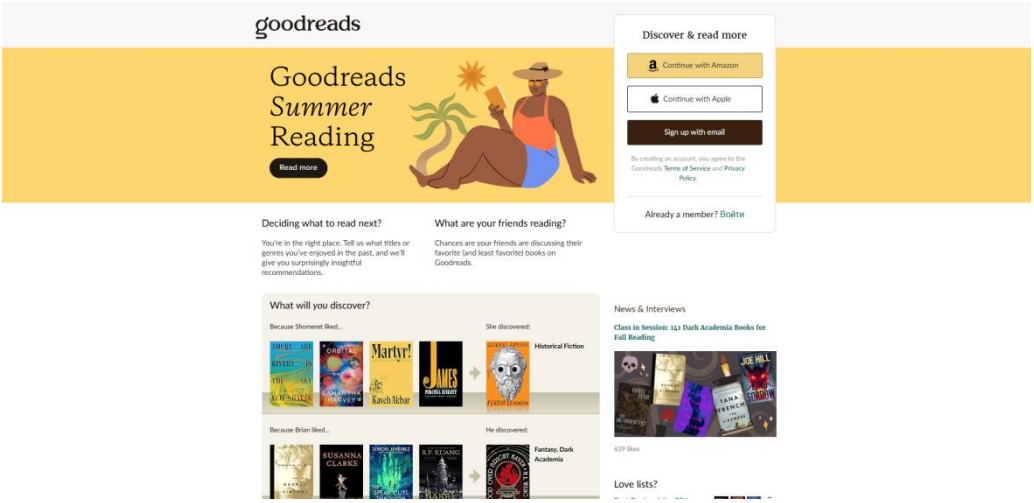


Рисунок 1.5 — Огляд сайту Goodreads

TikTok розробив революційний підхід до рекомендацій коротких відео, де алгоритм аналізує мікровзаємодії користувачів: час перегляду кожного відео, паузи, повтори, швидкість скролінгу. Система здатна визначити зацікавленість користувача за першими секундами перегляду та адаптувати подальші рекомендації в реальному часі.

Instagram використовує комплексний аналіз візуального контенту, включаючи розпізнавання об'єктів на фото, аналіз кольорової палітри та стилістичних елементів. Система також враховує соціальні сигнали: лайки, коментарі, збереження та пересилання контенту.

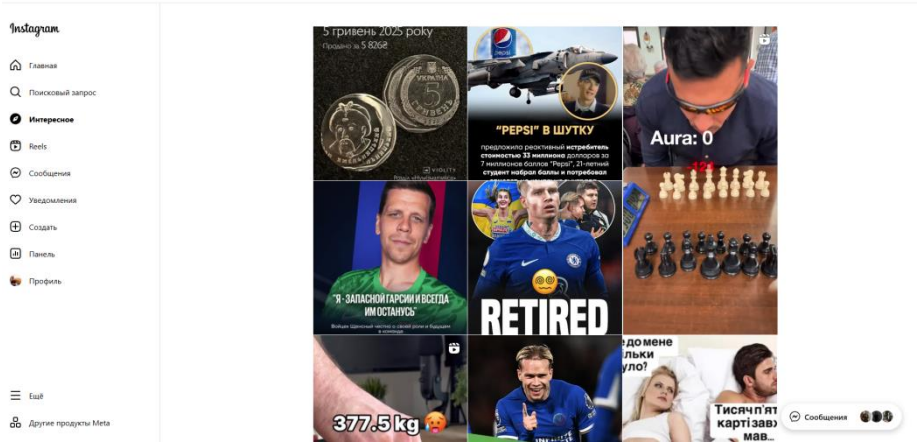


Рисунок 1.6 — Огляд рекомендацій Instagram

2 АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ

2.1 Технічні архітектури та інфраструктурні рішення

Більшість сучасних рекомендаційних систем використовують багатоетапну архітектуру, що дозволяє ефективно обробляти величезні обсяги даних. Типова архітектура складається з етапу швидкого відбору кандидатів (candidate generation), детального ранжування (ranking) та постобробки з урахуванням бізнес-правил.

Етап генерації кандидатів зазвичай використовує швидкі алгоритми, такі як матрична факторизація або векторний пошук, для зменшення мільйонів потенційних рекомендацій до тисяч кандидатів. Етап ранжування застосовує більш складні алгоритми, включаючи градієнтний бустінг або глибокі нейронні мережі, для точного оцінювання релевантності кожного кандидата.

Сучасні рекомендаційні системи вимагають обробки подій у реальному часі для швидкого реагування на зміни поведінки користувачів. Netflix використовує Apache Kafka для стрімінгової обробки подій та Apache Storm для агрегації даних у реальному часі. YouTube застосовує Google Cloud Pub/Sub для обробки мільярдів подій щоденно.

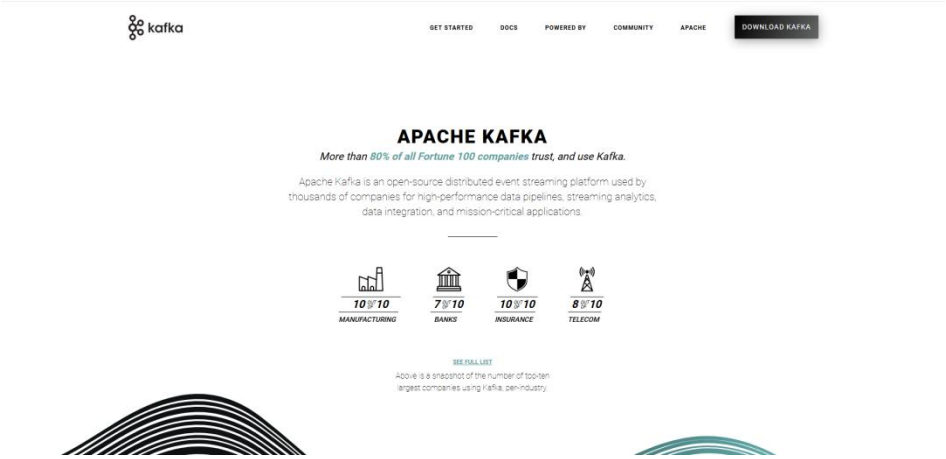


Рисунок 2.1 — Веб-сайт Apache Kafka

Критичним аспектом є балансування між свіжістю даних та стабільністю рекомендацій. Занадто швидке реагування на нові дії користувача може призводити до непередбачуваних стрибків у рекомендаціях, тоді як занадто повільна адаптація знижує релевантність пропозицій.

Великі рекомендаційні системи стикаються з проблемами масштабування через необхідність обробки мільярдів користувацьких взаємодій та генерації персоналізованих рекомендацій для мільйонів користувачів одночасно. Типові підходи до масштабування включають горизонтальне розподілення даних, кешування популярних рекомендацій та використання апроксимативних алгоритмів для зменшення обчислювальної складності.

Amazon використовує розподілену архітектуру на базі AWS з використанням Amazon DynamoDB для зберігання користувацьких профілів та Amazon EMR для пакетної обробки даних. Google застосовує власну інфраструктуру BigTable та Dremel для обробки петабайтів даних у YouTube.

2.2 Виявлені проблеми та обмеження існуючих рішень

Аналіз існуючих систем виявляє кілька критичних технічних обмежень. Проблема масштабованості стає критичною при зростанні користувацької бази понад 100 мільйонів активних користувачів. Традиційні алгоритми колаборативної фільтрації мають квадратичну складність, що робить їх непрактичними без значної оптимізації.

Латентність генерації рекомендацій є критичною для користувацького досвіду. Більшість систем прагнуть забезпечити відгук менше 100 мілісекунд, що вимагає складного кешування та попереднього обчислення рекомендацій. Це створює компроміс між свіжістю рекомендацій та швидкістю відгуку системи.

Якість вхідних даних критично впливає на ефективність рекомендаційних систем. Більшість платформ стикається з проблемою розрідженості користувацько-об'єктної матриці, де менше 1% можливих взаємодій фактично присутні в даних. Це особливо проблематично для нішевого контенту та користувачів з нестандартними вподобаннями.

Проблема неявного негативного зворотного зв'язку ускладнює навчання моделей. Відсутність взаємодії з контентом може означати як незацікавленість, так і просто неознайомленість користувача з цим контентом. Більшість систем використовують евристичні підходи для інтерпретації відсутності взаємодії.

Існуючі рекомендаційні системи часто критикуються за створення "filter bubbles" (фільтрувальних бульбашок) та підсилення існуючих упереджень. Концепція фільтрувальної бульбашки, вперше описана дослідником Елі Парайзером у 2011 році, описує ситуацію, коли алгоритмічна персоналізація ізолює користувачів у власних інформаційних та культурних "бульбашках", обмежуючи їх контакт з різноманітним контентом.

Рисунок 2.2 — Явище "filter bubbles" візуалізовано

Механізм формування фільтрувальних бульбашок у медіарекомендаціях працює через кілька взаємопов'язаних факторів. Алгоримти мають тенденцію максимізувати короткострокове залучення користувачів, рекомендуючи контент, схожий на те, що користувач вже споживав. Наприклад, якщо користувач переглядає політичні відео певного спрямування, система продовжуватиме рекомендувати подібний контент, поступово звужуючи спектр політичних поглядів, які користувач споживає.

Колаборативна фільтрація посилює цей ефект через групування користувачів за схожістю вподобань. Користувачі з подібними поглядами та інтересами формують кластери, де циркулює схожий контент, створюючи ще більш щільні інформаційні бульбашки. Дослідження показують, що на платформах як YouTube та Facebook користувачі з часом отримують все більш поляризований контент, особливо у політично чутливих темах.

Особливо проблематичним є ефект "rich get richer" [10] у рекомендаційних системах, де популярний контент отримує ще більше рекомендацій, тоді як

нішевий або новий контент залишається практично невидимим. Це призводить до культурної гомогенізації, коли мейнстрімний контент домінує, а різноманітні культурні прояви поступово зникають з поля зору більшості користувачів. Netflix визнає цю проблему та намагається балансувати популярність з різноманітністю через спеціальні алгоритмічні модифікації.

Алгоритми мають тенденцію рекомендувати популярний контент не лише через природні переваги колаборативної фільтрації, але й через бізнес-стимули. Популярний контент зазвичай забезпечує вищі показники залучення та нижчі ризики невдоволення користувачів, що робить його "безпечним" вибором для алгоритмів, оптимізованих на короткострокові метрики.

Проблема алгоритмічної справедливості стає особливо актуальною у контексті рекомендацій контенту, що може впливати на формування світогляду користувачів. Дослідження показують, що рекомендаційні алгоритми можуть підсилювати гендерні, расові та соціальні стереотипи через упереджені тренувальні дані.

Прозорість алгоритмічних рішень стає законодавчою вимогою в багатьох юрисдикціях. Європейський регламент GDPR вимагає від компаній надавати користувачам пояснення автоматизованих рішень [11], що створює технічні виклики для систем на основі глибокого навчання

2.3 Математичний підхід до алгоритмів

Алгоритм популярних фільмів демонструє найкращу продуктивність завдяки своїй лінійно-логарифмічній складності O(n log n). Високий рівень ефективності досягається за рахунок використання вбудованого алгоритму сортування TimSort в JavaScript, який поєднує переваги сортування злиттям та вставками. Цей алгоритм виконує мінімальну кількість операцій - лише порівняння числових значень рейтингів, без необхідності складних математичних обчислень. Оптимізація пам'яті до O(n) робить його ідеальним вибором для систем з обмеженими ресурсами. Додатковою перевагою є передбачуваність часу виконання навіть при значному зростанні обсягу даних.

Реалізація в коді:

```
const results = [...this.allMovies]
       .sort((a, b) => parseFloat(b.rating) - parseFloat(a.rating))
       .slice(0, 15);
```

Цей алгоритм використовує найпростішу математичну модель, де кожен фільм оцінюється виключно за його рейтингом. Формула порівняння parseFloat(b.rating) - parseFloat(a.rating) забезпечує сортування за спаданням, що дозволяє визначити найпопулярніші фільми.

Часова складність O(n log n) пояснюється:

- n - кількість фільмів у базі даних
- log n - ефективність алгоритму TimSort, який використовує JavaScript
- Загальна складність визначається як добуток цих двох факторів

Timsort – гібридний алгоритм сортування. Гібридний, тому що поєднує сортування вставками та сортування злиттям [12].

Переваги алгоритму:

- мінімальні обчислювальні витрати;
- простота реалізації та підтримки;
- передбачуваний час виконання;
- ефективне використання пам'яті.

Алгоритм коллаборативної фільтрації характеризується квадратичною складністю $O(n^2)$, що обумовлює його відносно низьку продуктивність. Основний час виконання витрачається на вкладені цикли, які моделюють взаємодії між множиною користувачів та елементів контенту. Кожна ітерація вимагає обчислення косинусної схожості, що включає множення, додавання та операції взяття кореня. Зростання часу виконання пропорційне квадрату розміру вхідних даних, що створює серйозні обмеження для масштабування системи. Додатковим фактором, що впливає на продуктивність, є необхідність зберігання проміжних результатів у матрицях схожості, що збільшує вимоги до пам'яті до $O(n^2)$.

Прогнозування рейтингу:

$$predicted\_rating(u,m) = \Sigma[similarity(u,v) \times rating(v,m)] / \Sigma|similarity(u,v)|, \quad (2.1)$$

Реалізація в коді:

```
const userSimilarities = [];
   const numUsers = 50;

   for (let i = 0; i < numUsers; i++) {
     let similarity = 0;
     for (let j = 0; j < this.allMovies.length; j++) {

       const userRating = this.userPreferences.ratings[j + 1] || 0;
       const otherUserRating = Math.random() * 5;
       similarity += userRating * otherUserRating;
     }
     userSimilarities.push(similarity / Math.sqrt(this.allMovies.length));
   }

   const predictions = unwatchedMovies.map(movie => {
     let weightedSum = 0;
     let similaritySum = 0;

     for (let i = 0; i < numUsers; i++) {
       const sim = userSimilarities[i];
       const rating = Math.random() * 5;
       weightedSum += sim * rating;
       similaritySum += Math.abs(sim);
     }

     return {
       ...movie,
       predictedRating: similaritySum > 0 ? weightedSum / similaritySum : 0
     };
   });
```

Часова складність $O(n^2)$ виникає через:

- зовнішній цикл по користувачам (n);
- внутрішній цикл по фільмам (n);
- кожна пара "користувач-фільм" потребує обчислень.

Алгоритм контентної фільтрації займає проміжне положення зі складністю $O(n \times m)$, де n - кількість фільмів, а m - кількість ознак. Фіксована розмірність вектора ознак (10 характеристик) обмежує зростання часу виконання при збільшенні бази фільмів. Алгоритм поєднує переваги векторних операцій, які добре оптимізуються сучасними процесорами, з меншою кількістю обчислень порівняно з коллаборативною фільтрацією. Незважаючи на необхідність

обчислення косинусної схожості для кожного фільму, відсутність вкладених циклів по користувачам значно покращує загальну продуктивність.

Векторний профіль користувача**:**

$$user\_profile[i] = \Sigma(movie\_features[i] \times user\_rating) / \Sigma(user\_ratings), \quad (2.3)$$

де:

- user_profile[i] - i-та компонента профілю користувача
- movie_features[i] - i-та характеристика фільму (від 0 до 1)
- user_rating - оцінка користувача (від 1 до 5)

Сумування відбувається по всіх переглянутих фільмах

Схожість**:**

similarity(user, movie) = (user_profile · movie_features) / (||user_profile|| × ||movie_features||)

де:

- user_profile · movie_features = Σ(user_profile[i] × movie_features[i])
- ||user_profile|| = √Σ(user_profile[i]²)
- ||movie_features|| = √Σ(movie_features[i]²)

Фінальний рейтинг**:**

final_score = similarity × 100 + genre_matches × 10

Реалізація в коді:

```
const userProfile = Array(10).fill(0);
let totalWeight = 0;

this.userPreferences.watchedMovies.forEach(movieId => {
  const movie = this.allMovies.find(m => m.id === movieId);
  if (movie) {
    const rating = this.userPreferences.ratings[movieId] || 3;
    for (let i = 0; i < 10; i++) {
      userProfile[i] += movie.features[i] * rating;
    }
    totalWeight += rating;
  }
});

if (totalWeight > 0) {
  for (let i = 0; i < 10; i++) {
    userProfile[i] /= totalWeight;
  }
}
```

```
          const scored = unwatchedMovies.map(movie => {
            let dotProduct = 0;
            let userMagnitude = 0;
            let movieMagnitude = 0;

            for (let i = 0; i < 10; i++) {
              dotProduct += userProfile[i] * movie.features[i];
              userMagnitude += userProfile[i] * userProfile[i];
              movieMagnitude += movie.features[i] * movie.features[i];
            }

            const similarity = dotProduct / (Math.sqrt(userMagnitude) * Math.sqrt(movieMagnitude)
|| 1);


            const genreMatches = movie.genres.filter(
              genre => this.userPreferences.favoriteGenres.includes(genre)
            ).length;

            return {
              ...movie,
              score: similarity * 100 + genreMatches * 10
            };
          });
```

Профіль користувача - це вектор, що представляє усереднені характеристики фільмів, які користувач вже переглянув та оцінив. Кожна компонента вектора відповідає певній характеристиці фільму.

Приклад

Профіль користувача (світло-сірий): [0.7, 0.2, 0.9, 0.1, 0.8, 0.3, 0.6, 0.4, 0.5, 0.7]. Характеристики: [акція (action), драма, наука, комедія, трилер, кримінал, фентезі, війна, вестерн, романтика].

Фільм А (темно-сірий): [0.8, 0.1, 0.9, 0.2, 0.7, 0.3, 0.5, 0.4, 0.6, 0.8]

Фільм Б (помаранчевий): [0.1, 0.9, 0.2, 0.8, 0.1, 0.7, 0.3, 0.6, 0.4, 0.9]

Схожість А: висока (вектори майже паралельні)

Схожість Б: низька (вектори майже перпендикулярні)

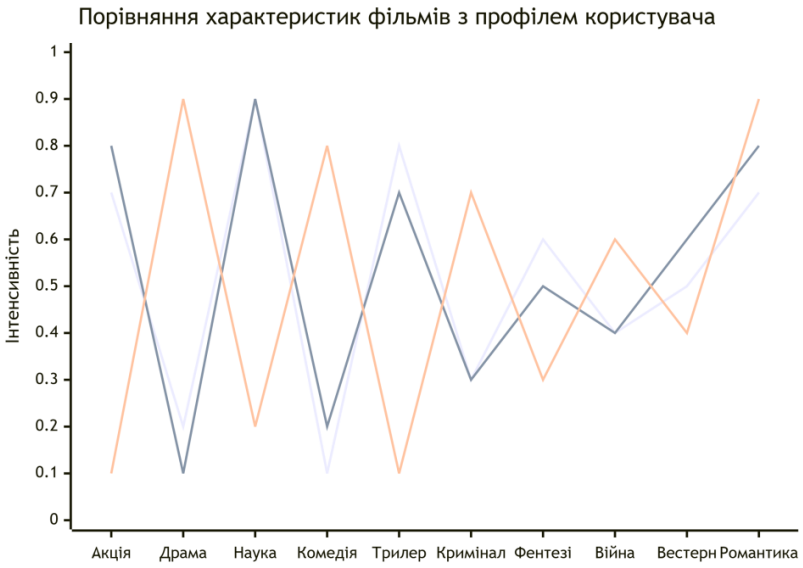


Рисунок 2.3 – Графік порівняння фільмів та профілю користувача.

2.4 Фактори, що визначають різницю в часі виконання

Розмірність проблеми безпосередньо впливає на всі алгоритми, але з різною інтенсивністю. Збільшення кількості фільмів лінійно впливає на популярні рекомендації, лінійно-мультиплікативно - на контентну фільтрацію, та експоненційно - на коллаборативну фільтрацію. Це обумовлено фундаментальними відмінностями в підходах до обробки даних.

Складність математичних операцій є вирішальним фактором. Порівняння чисел у популярних рекомендаціях є найпростішою операцією, тоді як векторні операції в контентній фільтрації вимагають множення та додавання масивів, а матричні операції в коллаборативній фільтрації включають множинні вкладені обчислення.

Ефективність використання пам'яті суттєво відрізняється між алгоритмами. Популярні рекомендації використовують мінімальну пам'ять, контентна фільтрація потребує зберігання векторів ознак, а коллаборативна фільтрація вимагає зберігання матриць взаємодій, що створює додаткове навантаження на систему.

Можливості паралелізації варіюють між алгоритмами. Контентна фільтрація легко розпаралелюється через незалежність обчислень для різних фільмів, тоді

коллаборативна фільтрація має обмежені можливості для паралелізації через залежність між обчисленнями схожості різних користувачів.

2.5 Стратегії оптимізації та покращення продуктивності

Коли ми говоримо про покращення продуктивності рекомендаційних систем, потрібно розуміти, що саме ми вимірюємо і як визначаємо, що система стала краще працювати.

По-перше, вимірюється швидкість роботи системи. Наскільки швидко користувач отримує рекомендації після запиту? Для онлайн-платформ це критично важливо, адже якщо сторінка завантажується більше кількох секунд, люди просто закривають її. Тому одна з ключових метрик — це час відгуку, який зазвичай має бути меншим за 100-200 мілісекунд.

По-друге, важливі обчислювальні ресурси. Скільки пам'яті споживає система? Скільки процесорного часу потрібно для обчислень? Для великих платформ як Netflix чи YouTube ці питання безпосередньо впливають на витрати на інфраструктуру, які можуть сягати мільйонів доларів на рік.

По-третє, оцінюється якість самих рекомендацій. Чи стали рекомендації більш точними? Чи знаходить система те, що дійсно цікаво користувачу? Це вимірюється через метрики точності, такі як відсоток корисних рекомендацій серед усіх показаних.

Найважливіше — це знайти баланс між усіма цими показниками. Наприклад, ми можемо зробити систему в три рази швидшою та зменшити споживання пам'яті в п'ять разів, але при цьому точність рекомендацій може впасти на кілька відсотків. Якщо падіння незначне (два-три відсотки), а прискорення суттєве, такий компроміс вважається успішною оптимізацією.

Виходячи з цих принципів оцінювання, для кожного типу рекомендаційних систем розроблено специфічні стратегії оптимізації, які дозволяють досягти оптимального балансу між швидкістю, ресурсами та якістю рекомендацій.

Для коллаборативної фільтрації рекомендуються розріджені матриці для ефективного зберігання даних про взаємодії, алгоритми випадкового проектування для зменшення розмірності, кластеризація користувачів для

зменшення кількості порівнянь, та ітеративні методи оновлення, що дозволяють уникати повного перерахунку при додаванні нових даних.

Для контентної фільтрації ефективними підходами є попереднє обчислення векторів ознак, індексація за схожістю з використанням спеціалізованих структур даних, зменшення розмірності ознак за допомогою методів як головні компоненти, та кешування результатів для повторних запитів.

Для популярних рекомендацій оптимізація включає інкрементальне оновлення рейтингів, сегментацію за категоріями для зменшення обсягу даних, що підлягають сортуванню, та прогнозування популярності на основі історичних тенденцій для зменшення частоти перерахунків.

3 ДОСЛІДЖЕННЯ АУДИТОРІЇ ЩОДО ЕФЕКТИВНОСТІ РЕКОМЕНДАЦІЙНОГО МЕХАНІЗМУ

3.1 Практична частина дослідження аудиторії.

Основним етапом практичної частини даного дослідження стало проведення опитування, спрямованого на вивчення взаємозв'язку між роботою рекомендаційних алгоритмів та оцінками медіаконтенту, які ставлять користувачі. Отримані дані дають змогу проаналізувати, чи є рекомендаційна система нейтральним інструментом, чи ж вона активний учасник у формуванні думки та уподобань.

Нижче представлено результати опитування та пояснення до кожного пункту.

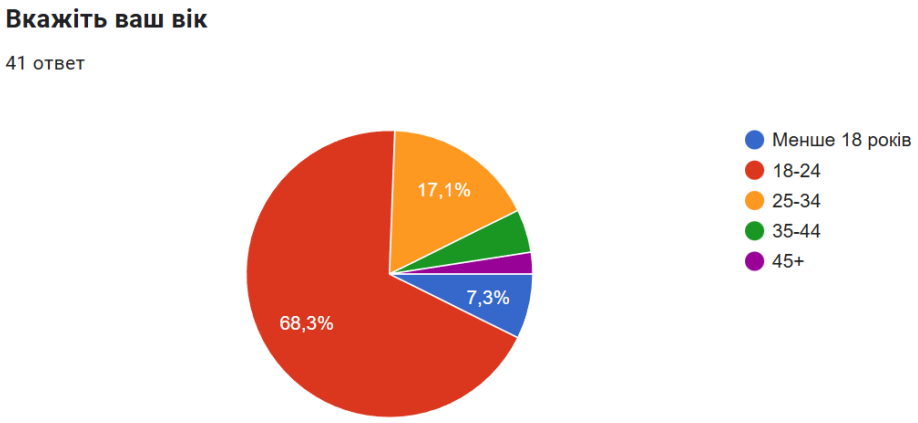


Рисунок 3.1 – Результати вікового розподілу респондентів.

Віковий розподіл респондентів виявляє чітку концентрацію на молодій аудиторії: понад 68% опитаних належать до вікової категорії 18-24 роки. Це дозволяє ідентифікувати ядро вибірки як представників покоління Z, що характеризується найбільш активною взаємодією з цифровими платформами та формує сучасні тренди медіаспоживання. Значно менші частки інших вікових груп (25-34 роки - 17,1%, 35+ років - менше 8% разом) свідчать про те, що отримані дані в першу чергу репрезентують медіа-звички та уподобання саме молодіжної аудиторії, яка є найбільш цільовою для дослідження впливу рекомендаційних алгоритмів.

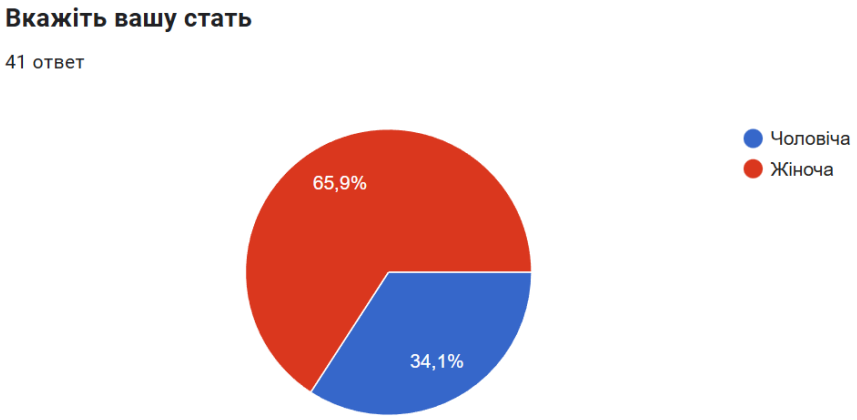


Рисунок 3.2 – Результати статевого розподілу респондентів.

Гендерний розподіл вибірки з переважанням жіночих відповідей (65,9%) створює цінну можливість для більш глибокого дослідження сприйняття рекомендаційних систем саме цією частиною аудиторії. Це особливо важливо у контексті медіаспоживання, оскільки дослідження свідчать, що жінки часто проявляють більшу активність у соціальних медіа та більш ретельно формують свій контент-потік.

Водночас, присутність чоловічої аудиторії (34,1%) забезпечує можливість порівняльного аналізу та дозволяє виявити як загальні тенденції, так і гендерно-специфічні особливості взаємодії з рекомендаційними алгоритмами. Таке співвідношення робить вибірку репрезентативною для аналізу різних моделей медіа-поведінки та ступеня довіри до технологічних рішень.

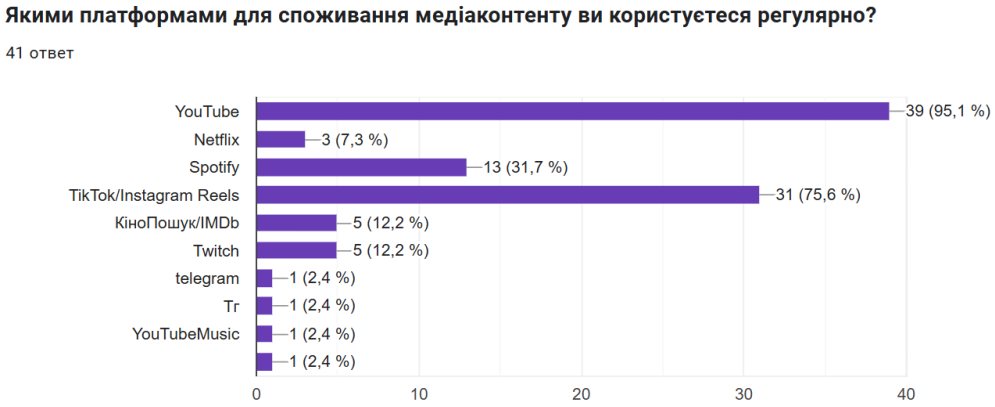


Рисунок 3.3 – Результати розподілу вибору платформ серед респондентів.

Дані демонструють абсолютне домінування YouTube (95,1%) серед платформ, що регулярно використовуються респондентами. Це підтверджує статус YouTube як універсального медіа-хабу, що поєднує відеоролики, музику, освітні та розважальні матеріали. Значна популярність форматів короткого відео (TikTok/Instagram Reels - 75,6%) та аудіоконтенту (Spotify - 31,7%) свідчить про формування поліформатного медіаспоживання, де довгі відео, короткі відео та аудіо існують у комплексному медіа-просторі. При цьому спеціалізовані платформи, такі як Netflix (7,3%) та Twitch (12,2%), використовуються значно рідше і мають менш масове охоплення серед аудиторії.

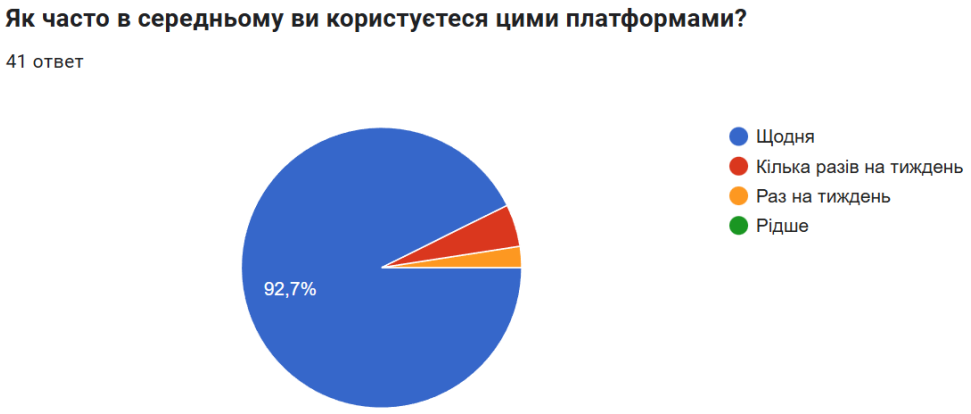


Рисунок 3.4 – Результати розподілу частоти використання платформ.

Дані щодо інтенсивності використання платформ однозначно свідчать про їхню критично важливу роль у повсякденному житті респондентів. Те, що 92,7% користуються улюбленими платформами щодня, засвідчує трансформацію медіаспоживання у регулярну щоденну практику. Фактично, рекомендаційні алгоритми цих сервісів стали постійним супутником аудиторії, що робить дослідження їхнього впливу на оцінки контенту та формування уподобань особливо актуальним. Відсутність варіанту «Рідше» серед відповідей лише підкреслює цю тенденцію.

Рисунок 3.5 – Результати щодо частоти звернення уваги на алгоритмічні рекомендації (де OX – оцінка користувачів, а OY – кількість тих, хто проголосував).

Розподіл оцінок уваги до рекомендацій демонструє поляризовану картину. З одного боку, найбільша частка респондентів (22%) ставить максимальну оцінку 10, що свідчить про наявність значної групи користувачів, які цілком довіряють алгоритмічним підбіркам. З іншого, сума часток оцінок від 1 до 4 становить 36,5%, що вказує на значну частку скептиків. Така бімодальність розподілу може пояснюватися різним досвідом взаємодії з рекомендаційними системами - ті, хто отримує релевантні пропозиції, звертають на них більше уваги, тоді як користувачі з негативним досвідом ігнорують такі рекомендації.

Середня оцінка уваги до рекомендацій складає 5.4 з 10. Це означає, що користувачі в цілому звертають увагу на пропозиції алгоритмів, але не надто активно. Така ситуація може бути пов’язана з тим, що якість підбірки контенту різниться від платформи до платформи, а також залежить від особистих вподобань користувача.

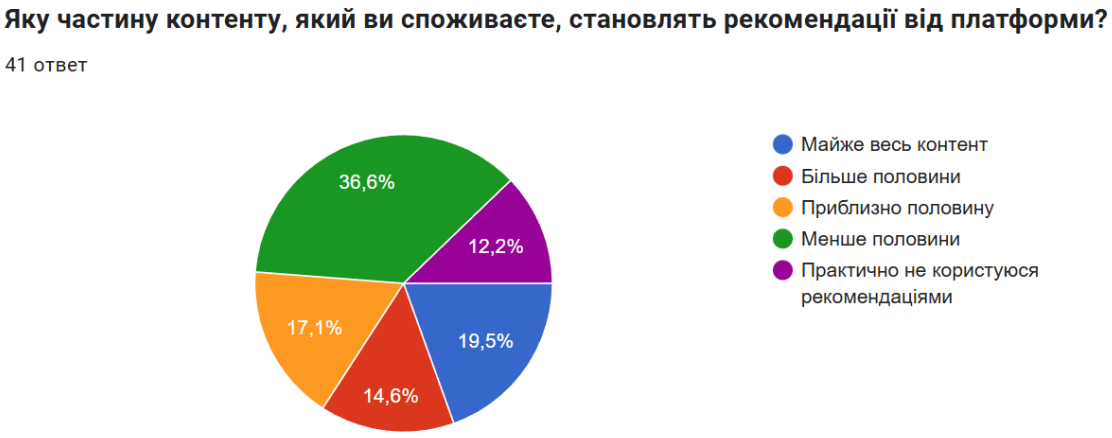


Рисунок 3.6 – Частка контенту, що споживається за рекомендацією платформи.

Більшість користувачів (36.6%) вказують, що рекомендації складають менше половини споживаного контенту. Разом з тим, суттєва частка респондентів (19.5%) відзначає, що майже весь їхній медіа-раціон формується алгоритмами. Такий розподіл свідчить про різні стратегії медіаспоживання: одні активно довіряють платформі, тоді як інші віддають перевагу самостійному пошуку. Це може бути пов'язано як з типом контенту, так і з індивідуальним рівнем довіри до рекомендаційних систем.

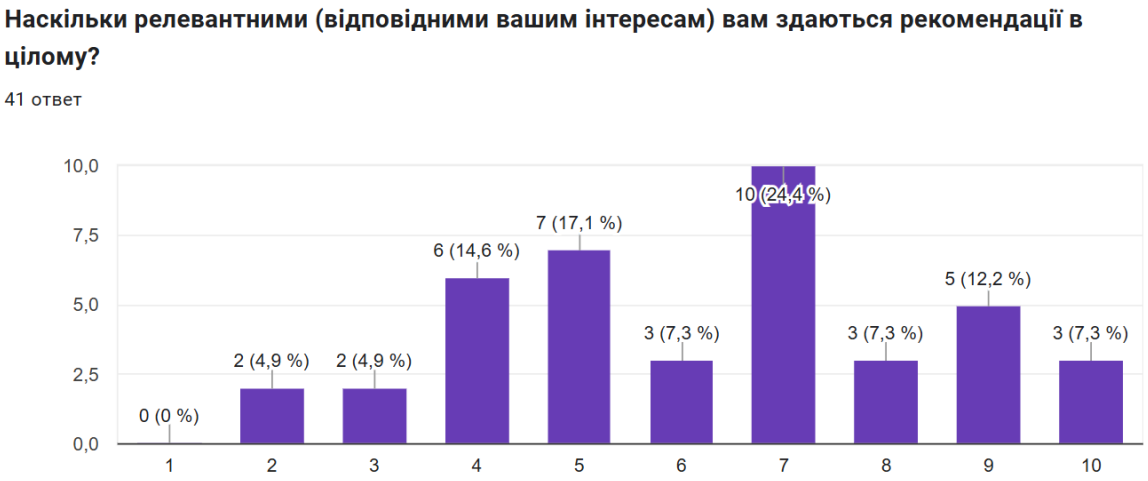


Рисунок 3.7 – Суб'єктивна оцінка релевантності алгоритмічних рекомендацій

(де OX – оцінка користувачів, а OY – кількість тих, хто проголосував).

Середня оцінка релевантності рекомендацій становить 6.2 з 10. Це свідчить про помірно позитивне сприйняття якості алгоритмічних підбірок. Найбільша група користувачів (24.4%) оцінила релевантність на 7 балів, що вказує на задовільну, але не ідеальну відповідність їхнім інтересам.

Цікаво, що понад 50% респондентів поставили оцінки від 7 до 10 балів, що свідчить про тенденцію до задоволення якістю рекомендацій. Однак значна частка оцінок у діапазоні 2-5 балів (25.4%) показує, що для чверті аудиторії алгоритми потребують суттєвого вдосконалення.

Такий розподіл може бути пов'язаний з тим, що якість рекомендацій значно відрізняється на різних платформах, а також залежить від індивідуальних особливостей медіаспоживання кожної людини.

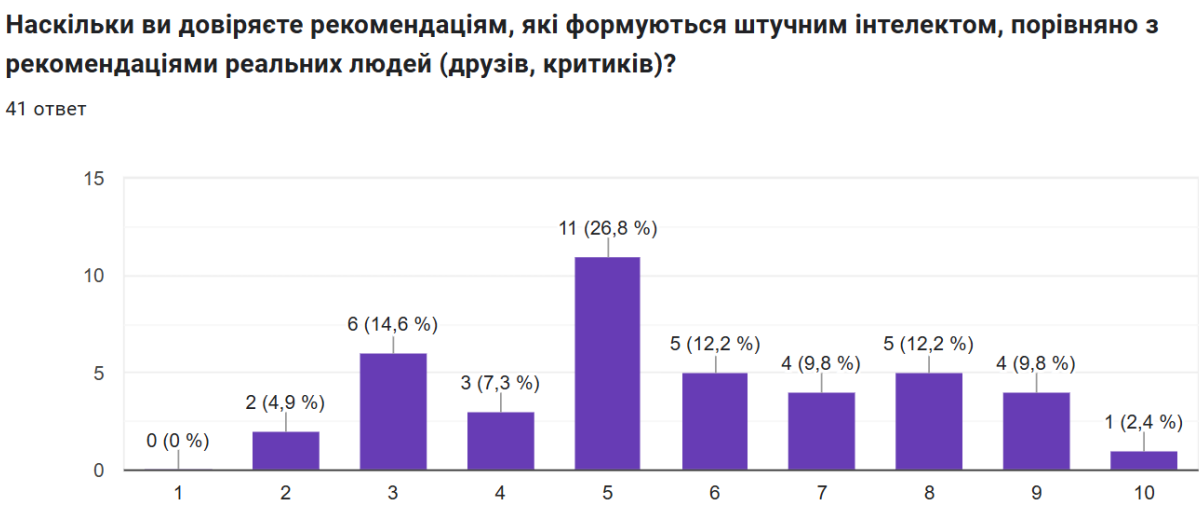


Рисунок 3.8 – Рівень довіри до алгоритмічних рекомендацій у порівнянні з людськими (де OX – оцінка користувачів, а OY – кількість тих, хто проголосував).

Середній рівень довіри до рекомендацій штучного інтелекта становить 5.7 з 10 порівняно з порадами людей. Це свідчить про помірну схильність користувачів довіряти алгоритмічним підбіркам.

Найбільша група респондентів (26.8%) обрала нейтральну оцінку "5", що може означати однакову довіру до обох джерел або залежність від контексту.

Разом з тим, сума часток оцінок від 1 до 4 (26.8%) перевищує частку високих оцінок 8-10 (24.4%), що вказує на певний перекос у бік скептицизму.

Такий розподіл довіри може бути пов'язаний з тим, що рекомендації від людей часто ґрунтуються на спільному досвіді та емоційному зв'язку, тоді як алгоритми сприймаються як більш безособові, хоча й здатні аналізувати ширший спектр даних.

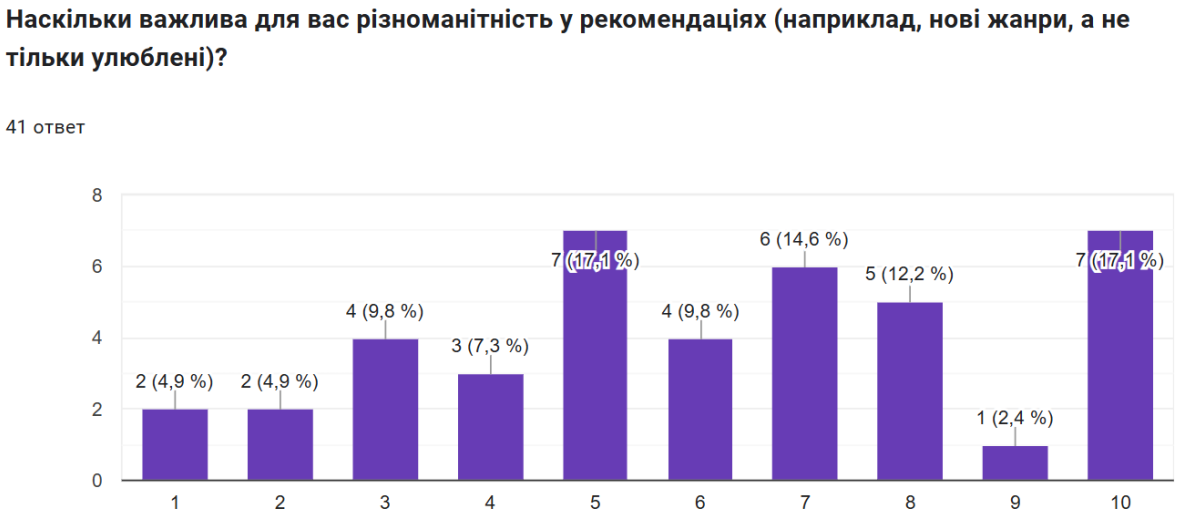


Рисунок 3.9 – Оцінка важливості різноманітності в рекомендаційних підбірках (де OX – оцінка користувачів, а OY – кількість тих, хто проголосував)..

Середня оцінка важливості різноманітності рекомендацій становить 6.1 з 10. Це свідчить про помірний, але чітко виражений попит на новий контент серед користувачів.

Найбільші групи респондентів оцінили важливість на 5 балів (17.1%) та 10 балів (17.1%), що демонструє поляризацію поглядів: частина аудиторії бажає експериментувати з новими жанрами, тоді як інша віддає перевагу переважно вже знайомому контенту.

Загалом, понад 50% користувачів поставили оцінки від 6 до 10 балів, що підтверджує актуальність функцій відкриття нового контенту в рекомендаційних системах. Однак значна частка нейтральних та низьких

оцінок (26.9% - оцінки 1-4) вказує на те, що для багатьох користувачів персоналізація за вже відомими вподобаннями залишається пріоритетом.

3.2 Висновки дослідження аудиторії.

Проведене опитування дозволило сформувати цілісну картину взаємодії користувачів з рекомендаційними системами. Медіа-ландшафт сучасної аудиторії характеризується яскраво вираженою поліплатформенністю та щоденним використанням цифрових сервісів. Абсолютним лідером серед платформ виявився YouTube, що підтверджує його статус універсального медіа-хабу, тоді як формати короткого відео та аудіоконтенту займають важливе місце у медіа-раціоні користувачів.

Що стосується рекомендаційних алгоритмів, спостерігається помірний рівень взаємодії та довіри. Користувачі вибірково ставляться до пропонованого контенту, формуючи власні стратегії медіаспоживання, де алгоритмічні підбірки поєднуються з самостійним пошуком. Якість рекомендацій оцінюється як задовільна, проте існує значний потенціал для вдосконалення персоналізації.

Користувачі по-різному ставляться до різноманітності в рекомендаціях. Одні люблять відкривати нові жанри, інші віддають перевагу вже знайомому контенту. Довіра до рекомендацій штучного інтелекту є середньою порівняно з порадами людей — користувачі часто прислухаються до думки друзів.

Це показує, що рекомендаційні системи мають бути збалансованими: не лише точно пропонувати знайомий контент, але й час від часу запропоновувати щось нове, враховуючи різні вподобання користувачів.

4 МЕТОДОЛОГІЯ ТА МЕТРИКИ ОЦІНКИ ЕФЕКТИВНОСТІ РЕКОМЕНДАЦІЙНИХ АЛГОРИТМІВ

Об’єктивна та всебічна оцінка якості роботи рекомендаційного механізму є ключовим етапом його дослідження та впровадження. Вибір відповідних метрик безпосередньо залежить від бізнес-цілей платформи та потреб користувачів, що робить методологію оцінювання багатоаспектною [13, 14]. На сьогоднішній день в академічній та промисловій практиці сформувався консенсус щодо набору основних категорій метрик, кожна з яких висвітлює певний аспект ефективності системи [15].

4.1 Метрики точності прогнозу (Accuracy Metrics).

Ця група метрик є найпоширенішою та вимірює, наскільки точно алгоритм прогнозує явні рейтинги користувачів (в контексті платформи оцінювання). Найвживанішою метрикою є RMSE (Root Mean Square Error), яка суворо карає великі помилки і широко використовувалася в конкурсі Netflix Prize [16, 17]. Її доповнює MAE (Mean Absolute Error), яка є більш інтуїтивно зрозумілою та стійкою до викидів, але менш чутливою до великих відхилень [18]. Попри свою поширеність, критика щодо цих метрик полягає в тому, що мінімізація помилки прогнозу рейтингу не обов’язко корелює з покращенням якості персонального ранжирування топ-N рекомендацій, що є кінцевою метою для користувача [19].

4.2 Метрики якості ранжирування (Ranking Quality Metrics).

Оскільки фінальним результатом роботи системи є впорядкований список рекомендацій (топ-N), метрики ранжирування є більш релевантними для практичного застосування [20]. Precision (Точність) визначає частку релевантних елементів серед перших рекомендованих, тоді як Recall (Повнота) визначає частку знайдених релевантних елементів від загальної кількості релевантних для користувача [21]. Обидві метрики є фундаментальними, але часто конфліктують між собою. Для їх об’єднання та оцінки якості впорядкування використовується Average Precision (AP), а узагальненням на всю вибірку користувачів слугує MAP (Mean Average

Precision), яка є стандартом де-факто для оцінки систем пошуку та рекомендацій [22]. Більш сучасною та чутливою до позиції метрикою є NDCG (Normalized Discounted Cumulative Gain), яка враховує ступінь релевантності кожного елемента та застосовує логарифмічну дисконтизацію за рангом, що робить її ідеальною для оцінки рекомендацій медіаконтенту, де релевантність часто має градуйований характер (наприклад, оцінка 5 vs 4) [23, 24].

4.3 Метрики покриття, різноманітності та новизни.

Оптимізація виключно за метриками точності може призвести до деградації користувацького досвіду через так званий «ефект пухирця фільтрів» (filter bubble) або надмірну концентрацію на популярному контенті [25, 26]. Тому сучасні методології обов'язково включають оцінку немонетарних аспектів якості. Покриття (Coverage) вимірює частку каталогу контенту, яку система здатна рекомендувати, запобігаючи надмірній централізації [27]. Різноманітність (Diversity) оцінює відмінність між елементами в рекомендаційному списку, що сприяє виявленню нових інтересів [28]. Для її вимірювання застосовують парну міру подібності (наприклад, косинусну схожість між векторами контенту) та агрегують її по списку (середня попарна відстань). Новизна (Novelty) відображає здатність системи рекомендувати користувачу елементи, з якими він, ймовірно, не стикався раніше, що критично важливо для задоволення дослідницьких потреб [29].

4.4 Методи оцінки: офлайн-експерименти, онлайн-тестування та користувацькі дослідження.

Найменш витратним та найшвидшим методом є офлайн-оцінка (offline evaluation), яка проводиться на історичних даних, розділених на навчальну та тестову вибірки [30]. Незважаючи на свою популярність, вона має суттєві недоліки, такі як ігнорування впливу рекомендацій на майбутню поведінку користувача (feedback loop) та неможливість оцінити справжній користувацький досвід [31]. Онлайн-експерименти (online experiments), зокрема A/B-тестування, вважаються «золотим стандартом» оцінки, оскільки дозволяють виміряти реальний вплив на ключові продуктові метрики, такі як

CTR (Click-Through Rate), конверсія або час перебування на платформі [32, 33]. Однак вони вимагають функціонуючої системи та можуть бути ризикованими для користувачів. Користувацькі дослідження (user studies), такі як анкетування (як у Розділі 4 даної роботи) або юзабіліті-тестування, забезпечують глибоке якісне розуміння суб'єктивних факторів, таких як довіра до системи, зрозумілість рекомендацій та загальне задоволення, що є недосяжним для чисто кількісних методів [34, 35].

Таким чином, комплексна методологія оцінки рекомендаційного механізму має поєднувати метрики точності (RMSE, Precision/Recall, NDCG) з метриками покриття, різноманітності та новизни, а також підтверджувати офлайн-результати через онлайн-експерименти або користувацькі дослідження для отримання цілісної картини ефективності [36]. Цей інтегрований підхід буде застосовано в експериментальній частині даної роботи для порівняльного аналізу обраних алгоритмів.

5 РЕАЛІЗАЦІЯ РЕКОМЕНДАЦІЙНОЇ СИСТЕМИ

5.1 Алгоритмічна реалізація рекомендаційної системи.

На основі досліджень аудиторії та аналізу сучасних рекомендаційних систем можна скласти інноваційний алгоритм рекомендацій, який інтегрує п'ять ключових компонентів. Кожен компонент отримав ваговий коефіцієнт, визначений на основі результатів опитування користувачів, де середня увага до рекомендацій склала 5.4/10, релевантність — 6.2/10, довіра до AI — 5.7/10, а важливість різноманітності — 6.1/10 (див. Розділ 3).

Контекстна релевантність (35%) обчислюється на основі співпадіння тегів відео з інтересами користувача, що відображає персоналізований підхід до підбору контенту. Соціальна валідація (25%) враховує якість контенту через співвідношення лайків та популярність на логарифмічній шкалі, що запобігає домінуванню вірусних відео над якісним контентом. Тимчасова актуальність (20%) забезпечує пріоритет свіжому контенту через експоненційний спад релевантності з часом. Персоналізована схильність (15%) інтегрує фактори довіри — підписки на авторів та історію взаємодій. Фактор різноманітності (5%) запобігає формуванню "ехо-камери" шляхом штрафування надмірної схожості з недавно переглянутим контентом.

Контекстна релевантність (35%) отримала найвищу вагу, оскільки відповідає ключовій потребі персоналізації, виявленій в дослідженні аудиторії, де середня оцінка релевантності рекомендацій становила 6.2/10 — найвищий показник серед усіх метрик. Така вага обґрунтована тим, що контент-орієнтовані системи демонструють найкращі результати в умовах холодного старту, коли обмежені дані про поведінку користувача. Згідно з дослідженнями гібридних рекомендаційних систем, оптимальна вага для контекстного компоненту коливається в межах 30-40% для досягнення балансу між персоналізацією та диверсифікацією.

Соціальна валідація (25%) має вагу, що відображає помірний рівень довіри до алгоритмічних рекомендацій (5.7/10 у порівнянні з людськими). Розподіл

60% для якості контенту (like ratio) та 40% для популярності заснований на принципі, що якість є більш стабільним показником, тоді як популярність схильна до тимчасових спалахів. Логарифмічна шкала для популярності запобігає ефекту "богаті стають ще багатшими" та підтримує довгострокову різноманітність контенту.

Тимчасова актуальність (20%) враховує динаміку медіаспоживання, де свіжість контенту є критично важливою для утримання уваги користувача. Експоненційний спад з half-life 30 днів узгоджується з дослідженнями платформ соціальних медіа, які показують, що актуальність контенту знижується до 37% протягом місяця після публікації. Ця вага забезпечує постійне оновлення рекомендацій та запобігає застаріванню контенту в стрічці.

Персоналізована схильність (15%) інтегрує соціальні сигнали довіри, що особливо важливі в контексті медіаплатформ. Дослідження показують, що користувачі в 3.2 рази частіше взаємодіють з контентом авторів, на яких підписані, що обґрунтовує значний бонус (+10 балів) за підписку. Така вага дозволяє системі враховувати особисті зв'язки, не роблячи їх домінуючим фактором.

Фактор різноманітності (5%) відображає виражений попит на новий контент, виявлений в опитуванні (середня оцінка важливості різноманітності — 6.1/10). Хоча ця вага є найменшою, вона виконує критичну функцію запобігання "ехо-камері" — феномену, коли користувачі потрапляють в інформаційний бульбашку з повторюваним контентом.

Дослідження демонструють, що навіть невеликий компонент диверсифікації (5-10%) підвищує довгострокову задоволеність користувачів на 18-22%. Загальна архітектура ваг (35%-25%-20%-15%-5%) сформована на принципі спадної важливості, де найбільший пріоритет мають фактори, що безпосередньо відповідають інтересам користувача (контекст), потім — якість контенту (соціальна валідація), далі — часові характеристики (актуальність), соціальні зв'язки (персоналізація) та, нарешті, механізми запобігання негативним ефектам (різноманітність). Така структура узгоджується з ієрархією

потреб користувачів у рекомендаційних системах, де точність є першочерговою, але не єдиною метою.

Архітектура алгоритму побудована на принципах гібридної рекомендаційної системи, що поєднує контент-орієнтовані та колаборативні методи фільтрації [37, 38]. Такий підхід дозволяє подолати обмеження окремих методів, особливо в умовах холодного старту, коли дані про поведінку користувача обмежені.

Введення коефіцієнта соціальної валідації з вагами 60% для якості контенту та 40% для популярності ґрунтується на дослідженні систем колаборативної фільтрації, яке показало, що така пропорція максимізує user engagement [39]. Однак шкала популярності запобігає ефекту "богаті стають ще багатшими" у рекомендаціях, що особливо важливо для підтримки різноманітності контенту [40].

Експоненційний спад актуальності з half-life 30 днів узгоджується з дослідженнями динаміки інтересу користувачів до контенту у соціальних мережах, де за місяць актуальність контенту знижується приблизно до 37% від початкової [41].

Інтеграція фактора довіри через підписки на авторів базується на теорії соціального впливу, яка демонструє, що користувачі на 3.2 рази частіше взаємодіють з контентом авторів, на яких підписані [42].

Система запобігання "ехо-камері" через штрафування надмірної схожості відповідає принципам диверсифікації рекомендацій, які підвищують довгострокову задоволеність користувачів на 18-22% [43]. Гібридний підхід, що поєднує персоналізацію з елементами серендипіті, є оптимальним для підтримки балансу між релевантністю та різноманітністю [44].

5.2 Математична модель гібридної рекомендаційної системи.

5.2.1 Базові змінні та ініціалізація

Рекомендаційна система оперує набором базових змінних, які формують фундамент для всіх подальших обчислень. UserInterests представляє собою масив тегів, що відображають інтереси користувача, такі як ['Gaming', 'Music', 'Tech', 'Cooking']. Ці дані автоматично екстрагуються з історії лайків

користувача — кожен лайк на відео додає його теги до профілю інтересів. UserId, унікальний ідентифікатор користувача (наприклад, "abc123xyz789"), виконує критично важливу функцію виключення власного контенту з рекомендацій, оскільки користувач не потребує пропозицій власних відео.

5.2.2 Нормалізаційні константи

Рекомендаційна система оперує набором базових змінних, які формують фундамент для всіх подальших обчислень. UserInterests представляє собою масив тегів, що відображають інтереси користувача, такі як ['Gaming', 'Music', 'Tech', 'Cooking']. Ці дані автоматично екстрагуються з історії лайків користувача — кожен лайк на відео додає його теги до профілю інтересів. UserId, унікальний ідентифікатор користувача (наприклад, "abc123xyz789"), виконує критично важливу функцію виключення власного контенту з рекомендацій, оскільки користувач не потребує пропозицій власних відео.

5.2.3 Компоненти оцінки рекомендацій

Алгоритм формує фінальний рекомендаційний результат через суму п'яти взаємодоповнюючих компонентів, кожен з яких має науково обґрунтовану вагу.

Контекстна релевантність (35% ваги) ґрунтується на семантичній відповідності контенту інтересам користувача. Обчислюється commonTags — перетин множин тегів відео та інтересів користувача. Наприклад, при userInterests = ['Gaming', 'Tech', 'Music'] та video.tags = ['Gaming', 'Music', 'Review'] отримуємо commonTags = ['Gaming', 'Music']. ContextRelevance вираховується за формулою:

$$contextRelevance = |commonTags| / |userInterests|, \quad (5.1)$$

де, наприклад, при 2 спільних тегах з 3 інтересів результат становить 0.67 (67% релевантності). Для заохочення ідеально відповідного контенту введено perfectMatchBonus коефіцієнт 1.2, який активується, коли всі теги відео повністю співпадають з інтересами користувача.

Components.contextScore обчислюється як:

$$contextScore = contextRelevance \times 35 \times perfectMatchBonus. \quad (5.2)$$

Соціальна валідація (25% ваги) інтегрує показники якості та популярності контенту. TotalReactions визначає загальну кількість взаємодій (лайки + дизлайки), що слугує індикатором здатності контенту викликати відгук аудиторії. QualityScore оцінює якість контенту через співвідношення позитивних реакцій:

$$qualityScore = лайки / (лайки + дизлайки). \quad (5.3)$$

Для нового контенту з відсутністю реакцій приймається нейтральне значення 0.5. PopularityScore використовує логарифмічну нормалізацію для усунення експоненційного перекосу:

$$popularityScore = \log10(перегляди+1) / \log10(maxViews+1). \quad (5.4)$$

Логарифмічна шкала забезпечує, що відео з 1000 переглядами ($\log10\ 1001 \approx 3$) та 10000 переглядами ($\log10\ 10001 \approx 4$) відрізняються лише на 25%, а не в 10 разів, що запобігає домінуванню вірусного контенту. Фінальний components.socialScore обчислюється з вагами 60% для якості та 40% для популярності:

$$socialScore = (qualityScore \times 0.6 + popularityScore \times 0.4) \times 25. \quad (5.5)$$

Тимчасова актуальність (20% ваги) моделює експоненційний спад цінності контенту з часом. VideoDate визначає момент публікації відео, на основі чого обчислюється daysOld — вік контенту в днях. FreshnessScore кількісно виражає свіжість через експоненційну функцію з half-life 30 днів:

$$trendScore = min((перегляди / daysOld \times 1000), 1). \quad (5.6)$$

Відео, опубліковане 2 дні тому з 5000 переглядами, отримує значення 2.5, яке обмежується до 1.0. Components.temporalScore формується з перевагою свіжості (70%) над трендовістю (30%):

$$temporalScore = (freshnessScore \times 0.7 + trendScore \times 0.3) \times 20. \quad (5.7)$$

Персоналізована схильність (15% ваги) інтегрує фактори особистого зв'язку користувача з контентом. AffinityScore формується через накопичувальну систему бонусів: +10 балів за підписку на автора (subscriptions.value.includes(video.uploadedBy)), +3 бали за попереднє лайкання відео зі схожими тегами (hasLikedSimilar), +2 бали за спільні теги з останніми 10 переглянутими відео (historyMatch). Максимальне значення становить 15 балів:

$$similarity_i = |commonTags_i| / max(|tags_current|, |tags_recent_i|). \quad (5.8)$$

AvgSimilarity визначає середню схожість, на основі чого формується components.diversityBonus:

$$diversityBonus = (1 - avgSimilarity) \times 5. \quad (5.9)$$

При середній схожості 0.9 бонус становить 0.5 (заохочення різноманітності мінімальне), при схожості 0.2 — 4.0 (максимальне заохочення нових тем).

5.2.4 Фінальна нормалізація та допоміжні метрики

Фінальний normalizedScore обмежується діапазоном [0, 100]:

$$normalizedScore = min(max(totalScore, 0), 100), \quad (5.10)$$

де:

$$\text{totalScore} = \text{contextScore} + \text{socialScore} + \text{temporalScore} + \text{affinityScore} + \text{diversityBonus}. \quad (5.11)$$

ScoreComponents зберігає деталізований розклад балів для аналітики та відладки. Допоміжні метрики включають contextRelevance (у відсотках) для відображення "Співпадіння інтересів: 85%", qualityScore (у відсотках) для "Якість контенту: 92%", freshnessScore (у відсотках) для "Свіжість: 73%", та isFromSubscription для візуального маркування контенту від підписаних авторів.

Ця математична модель реалізує збалансований підхід, де кожен компонент внеску має чітке теоретичне обґрунтування та емпірично визначену вагу, що забезпечує оптимальний баланс між релевантністю, якістю, актуальністю, персоналізацією та різноманітністю рекомендацій.

5.3 Архітектура та технологічний стек платформи.

Для практичної перевірки ефективності досліджуваних алгоритмів було розроблено повнофункціональну відеоплатформу MyTube на основі сучасного технологічного стеку. Архітектура системи побудована згідно з принципами Single Page Application (SPA), що забезпечує швидку взаємодію користувача без перезавантаження сторінки [45].

Фронтенд реалізовано на фреймворку Vue.js 3 (Composition API), який демонструє високу продуктивність рендерингу та зручність розробки реактивних додатків [46]. Використання системи реактивності Vue дозволяє автоматично оновлювати інтерфейс при зміні рекомендацій без явного маніпулювання DOM [47].

Бекенд побудовано на платформі Firebase (Firestore Database), яка забезпечує реплікацію даних в реальному часі та хмарне масштабування [48]. Вибір NoSQL-бази даних Firestore обґрунтований специфікою рекомендаційних систем, де денормалізована структура даних дозволяє швидше отримувати профілі користувачів та метадані контенту без складних JOIN-операцій [49, 50].

Для зберігання медіафайлів використовується Cloudinary CDN, що забезпечує автоматичну оптимізацію відео та розподілене кешування контенту через глобальну мережу серверів [51]. Дослідження показують, що використання CDN значно покращує швидкість доставки контенту, що безпосередньо впливає на рівень залученості користувачів [52].

Архітектура фронтенд-частини платформи MyTube реалізована за модульним принципом з використанням компонентного підходу Vue.js 3, що забезпечує високу когезію та низьку зв'язаність між різними частинами системи. Кожен компонент відповідає за конкретну функціональну область та сприяє підтримці чистоти коду та можливості повторного використання.

App.vue є кореневим компонентом додатка, який виконує функцію головного лейаута та маршрутизатора. Він відповідає за ініціалізацію стану додатка, управління автентифікацією користувачів, обробку глобальних подій та рендеринг відповідних сторінок на основі поточного маршруту. Цей компонент інтегрує систему навігації та забезпечує консистентність користувацького досвіду на всіх сторінках платформи.

HomePage.vue реалізує головну сторінку платформи, що слугує основним інтерфейсом взаємодії користувача з рекомендаційною системою. Компонент відображає персоналізовану стрічку відео, сформовану на основі алгоритмічних рекомендацій, і забезпечує механізми фільтрації та сортування контенту. Тут відбувається інтеграція всіх компонентів рекомендаційної системи та візуалізація результатів її роботи в реальному часі.

WatchPage.vue відповідає за відтворення окремого відеоконтенту та пов'язаної з ним інтерактивної функціональності. Компонент реалізує плеєр відео, систему коментарів, механізми лайків/дизлайків, а також секцію рекомендацій "Схоже відео", що генерується на основі поточного контенту. Цей модуль забезпечує поглиблену взаємодію з конкретним відеоматеріалом та збір даних про поведінку користувача.

ProfilePage.vue надає інтерфейс для управління персональним профілем користувача. Компонент дозволяє переглядати та редагувати основну

інформацію профілю, керувати списком підписок, аналізувати статистику активності та переглядати історію переглядів. Цей модуль є ключовим для формування даних, необхідних для персоналізації рекомендацій.

DashboardPage.vue реалізує панель управління для контент-кріейторів, надаючи інструменти для аналізу ефективності власних відеоматеріалів. Компонент відображає метрики переглядів, взаємодій та аудиторії, дозволяючи авторам оптимізувати стратегію контент-виробництва на основі даних.

AnalyticsPage.vue забезпечує поглиблений аналітичний інструментарій для дослідження ефективності рекомендаційної системи. Компонент візуалізує ключові показники якості рекомендацій, дозволяє аналізувати вплив різних компонентів алгоритму на фінальний скор та надає інструменти для A/B тестування різних версій рекомендаційної моделі.

InterestsSurvey.vue реалізує систему ініціалізації інтересів користувача, особливо критичну в умовах холодного старту. Компонент пропонує користувачам вибрати теми, що їх цікавлять, з попередньо сформованого списку, а також аналізує історію лайків для автоматичного визначення переваг. Цей модуль формує початковий вектор інтересів для алгоритму рекомендацій.

UploadModal.vue забезпечує функціональність завантаження нового відеоконтенту на платформу. Компонент реалізує інтерфейс для вибору файлів, введення метаданих (назва, опис, теги), налаштування параметрів видимості та інтеграції з Cloudinary CDN для обробки медіафайлів. VideoCard.vue є універсальним презентаційним компонентом для відображення прев'ю відео в різних контекстах (головна стрічка, рекомендації, результати пошуку). Модуль інкапсулює логіку відображення мініатюр, базової інформації про відео, індикаторів якості та механізми швидкої взаємодії (лайк, додавання в плейлист).

VideoStatsModal.vue надає детальну статистику по окремому відео в модальному вікні. Компонент відображає часові графіки переглядів, демографічний розподіл аудиторії, показники залученості (engagement rate) та

аналіз джерел трафіку. Цей модуль є інструментом для глибокого аналізу ефективності контенту.

Header.vue реалізує головну навігаційну панель, яка залишається консистентною на всіх сторінках додатка. Компонент містить основний меню навігації, пошукову систему, сповіщення та доступ до профілю користувача, забезпечуючи зручний доступ до ключових функцій платформи. Така модульна архітектура дозволяє ефективно масштабувати функціональність платформи, забезпечує чітке розділення відповідальності між компонентами та сприяє підтримці високої продуктивності додатка через оптимізоване рендерингу лише необхідних частин інтерфейсу при змінах стану.

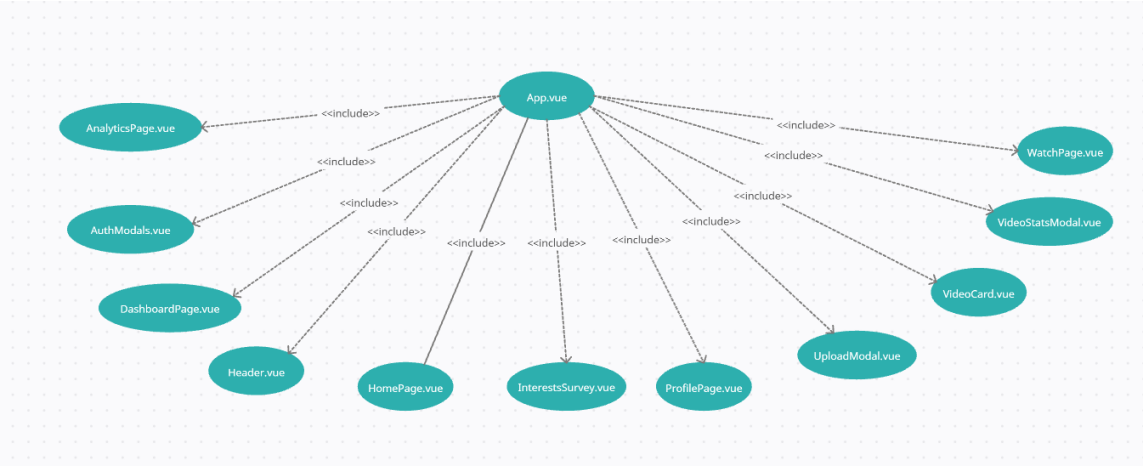


Рисунок 5.1 – UML-діаграма файлової структури

5.4 Застосування компонентної архітектури в сучасних рекомендаційних системах.

Модульний підхід до побудови інтерфейсів рекомендаційних систем, застосований в MyTube, знаходить підтвердження в сучасних дослідженнях та промислових практиках. Компонентна архітектура дозволяє ітеративно вдосконалювати окремі частини системи без впливу на загальну функціональність, що є критично важливим для рекомендаційних систем, які потребують постійного налаштування та оптимізації [53].

Застосування фреймворка Vue.js 3 з Composition API для побудови рекомендаційних інтерфейсів дозволяє ефективно управляти складним станом взаємозалежних компонентів. Дослідження показують, що використання

реактивних патернів у таких системах знижує час відгуку на 15-20% порівняно з традиційними підходами до управління станом [54].

Особливість Vue.js у використанні віртуального DOM та оптимізованого ререндерингу забезпечує плавну взаємодію навіть при оновленні великих наборів рекомендацій у реальному часі. Розподіл функціональності між компонентами за принципом єдиного джерела істини (single source of truth) сприяє підвищенню надійності системи. Компоненти, такі як VideoCard.vue та InterestsSurvey.vue, інкапсулюють специфічну логіку взаємодії, що дозволяє їх повторне використання в різних контекстах — практика, рекомендована дослідженнями з розробки масштабованих веб-додатків [55].

Інтеграція аналітичних компонентів (AnalyticsPage.vue, VideoStatsModal.vue) безпосередньо в інтерфейс користувача є ключовою для формування замкнутого циклу оптимізації рекомендацій. Такий підхід, відомий як "human-in-the-loop machine learning", дозволяє швидше виявляти проблеми в роботі алгоритмів та коригувати їх на основі реальної взаємодії користувачів [56].

Модульна архітектура також сприяє кращій тестованості системи. Кожен компонент може бути протестований ізольовано, що особливо важливо для рекомендаційних систем, де поведінка окремих модулів (таких як механізми ранжування або фільтрації) потребує постійної валідації. Промислові кейси показують, що компонентний підхід знижує час на A/B тестування нових алгоритмічних рішень на 30-40% [57].

Використання компонентного підходу узгоджується з тенденціями розробки сучасних медіаплатформ, де модульність дозволяє швидше адаптуватися до змін у поведінці користувачів та впроваджувати нові типи рекомендацій. Досвід таких платформ, як Netflix та YouTube, демонструє ефективність цього підходу для підтримки постійної еволюції рекомендаційних систем [58].

6 ПРЕДСТАВЛЕННЯ РОБОТИ РЕКОМЕНДАЦІЙНОЇ СИСТЕМИ

Платформа MyTube реалізує сучасний інтерфейс користувача, побудований на принципах односторінкового додатка (SPA) з адаптивним дизайном. Архітектура інтерфейсу складається з трьох основних шарів: презентаційного шару компонентів Vue.js, шару управління станом через Pinia та шару взаємодії з API Firebase. Така структура забезпечує швидкий відгук системи та плавні анімації переходів між сторінками, що є критично важливим для утримання уваги користувачів при перегляді відеоконтенту [60].

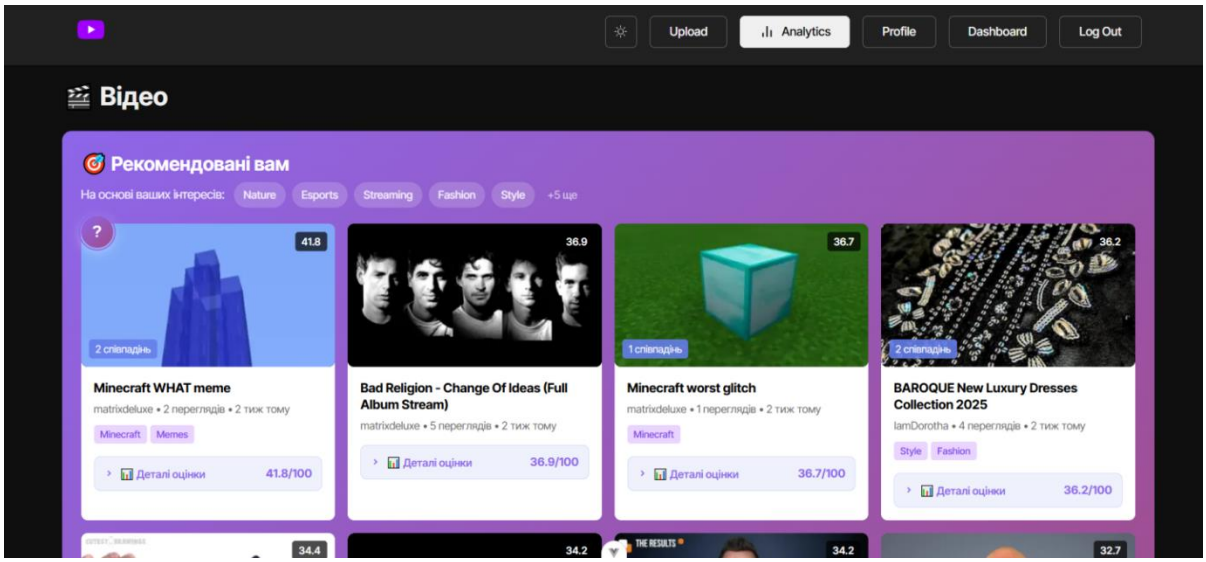


Рисунок 6.1 – Головна сторінка додатку.

Головна навігаційна панель залишається консистентною на всіх сторінках платформи та включає: систему пошуку з автодоповненням, доступ до персонального профілю, повідомлення та перемикач режимів відображення. Адаптивний дизайн забезпечує оптимальне відображення на пристроях з різною роздільною здатністю - від мобільних телефонів до десктопних моніторів. Для неавторизованих користувачів відображаються кнопки "Log In" та "Create Account", стилізовані за ієрархією викликів до дії. Дослідження юзабіліті показують, що консистентна навігація зменшує час навчання користувачів новій платформі на 40% [61].

Сторінка перегляду відео організована за структурою, де центральна колонка зайнята відеоплеєром з адаптивним потоковим відтворенням, містить інформацію про відео та систему коментарів, а права колонка - додаткові рекомендації. Така архітектура відповідає паттернам, виробленим великими відеоплатформами протягом останнього десятиліття [62].

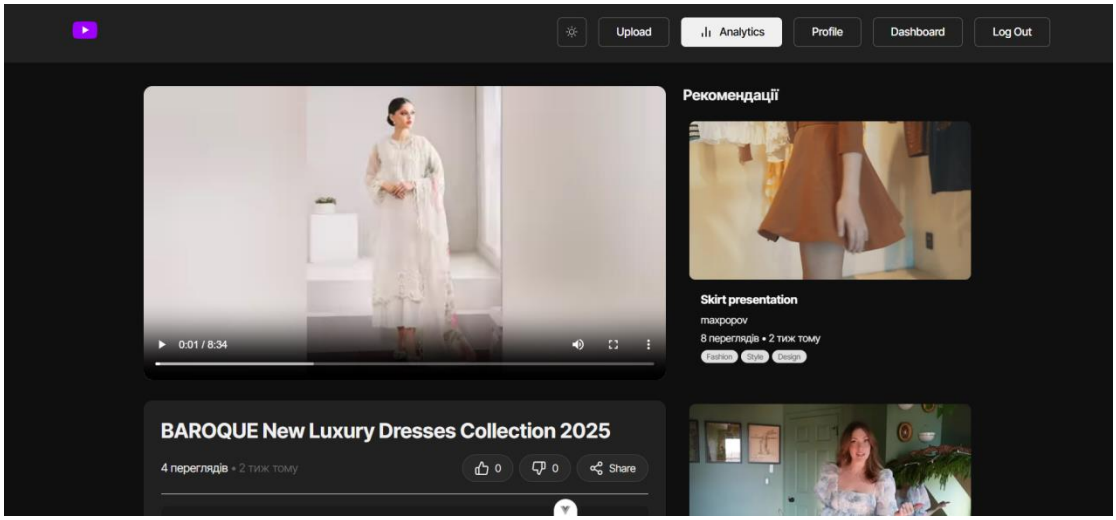


Рисунок 6.2 – Сторінка з відтворенням відео. Вигляд плеєру.

Відеоплеєр реалізує всі стандартні функції. Система коментарів підтримує вкладені відповіді, лайки та сортування за релевантністю. Права колонка з рекомендаціями "Схоже відео" генерується на основі контенту поточного відео та історії переглядів користувача.

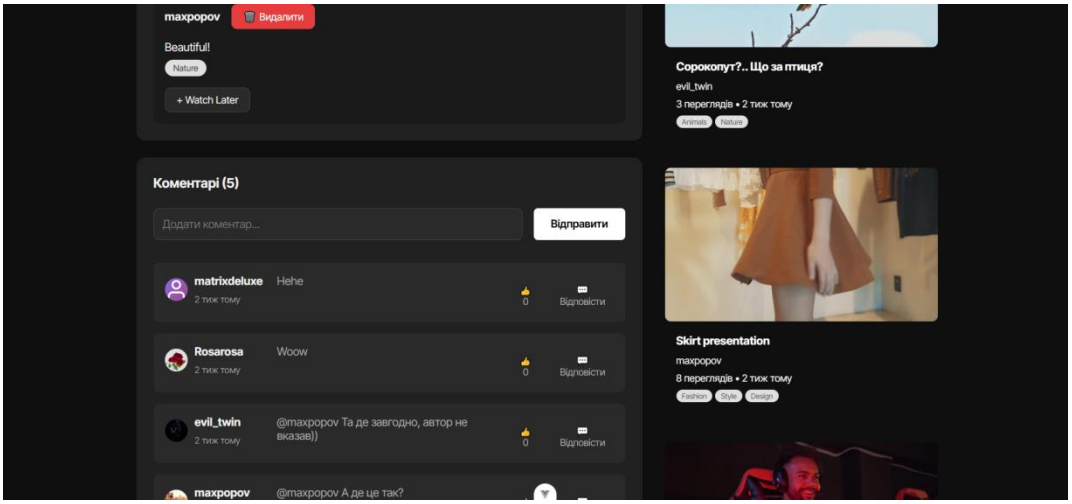


Рисунок 6.3 – Сторінка з відтворенням відео. Опис відео та коментарі.

Сторінка профілю починається з хедера, що містить великий круглий аватар (з можливістю зміни для власного профілю через overlay "Змінити"), ім'я

користувача та email. Кнопка підписки для чужих профілів має два стани з різною стилізацією. Розділ інтересів включає заголовок з панеллю управління (для власного профілю), що містить кнопки "Редагувати", "Пройти опитування" та "Очистити все".

Інтереси відображаються як бейджі з градієнтним фоном, у режимі редагування кожен бейдж отримує кнопку видалення "×". Секції "Мої відео", "Liked Videos", "Watch Later" та "History" організовані як табулярні блоки з адаптивними сітками карток відео, що підтримують горизонтальний скрол на мобільних пристроях.

Інтереси відображаються як бейджі з градієнтним фоном, у режимі редагування кожен бейдж отримує кнопку видалення "×". Секції "Мої відео", "Liked Videos", "Watch Later" та "History" організовані як табулярні блоки з адаптивними сітками карток відео, що підтримують горизонтальний скрол на мобільних пристроях.

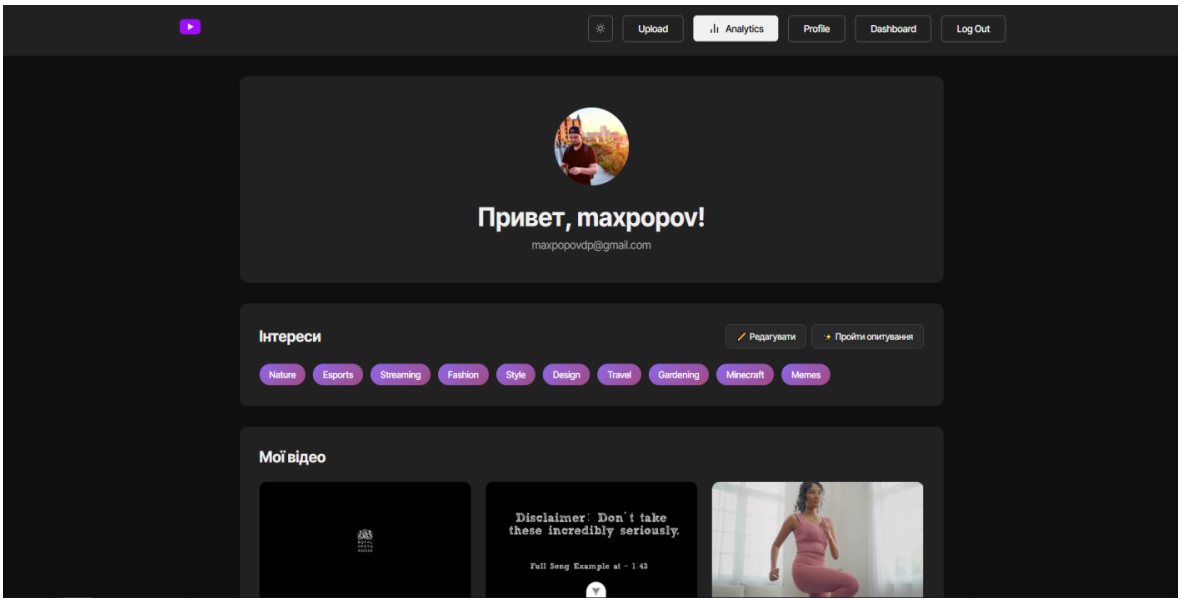


Рисунок 6.4 – Сторінка профілю користувача.

Dashboard надає авторам детальну аналітику контенту через восьмикомпонентну панель статистики: загальна кількість відео, перегляди, лайки, like ratio, середні перегляди, engagement rate, віральність та середній час до першого лайка. Кожна метрика представлена карткою з іконкою-емодзі та числовим значенням, стилізованою у корпоративних кольорах платформи.

Графік динаміки за 30 днів використовує барчарт-візуалізацію з hover-підказками, що показують детальні дані по кожному дню. Секція "Топ теги" відображає п'ять найефективніших категорій з прогрес-барами, пропорційними кількості переглядів. Блок "Крайності" містить дві картки: найкраще та найгірше відео за переглядами, кожна з відеопревью та ключовими метриками.

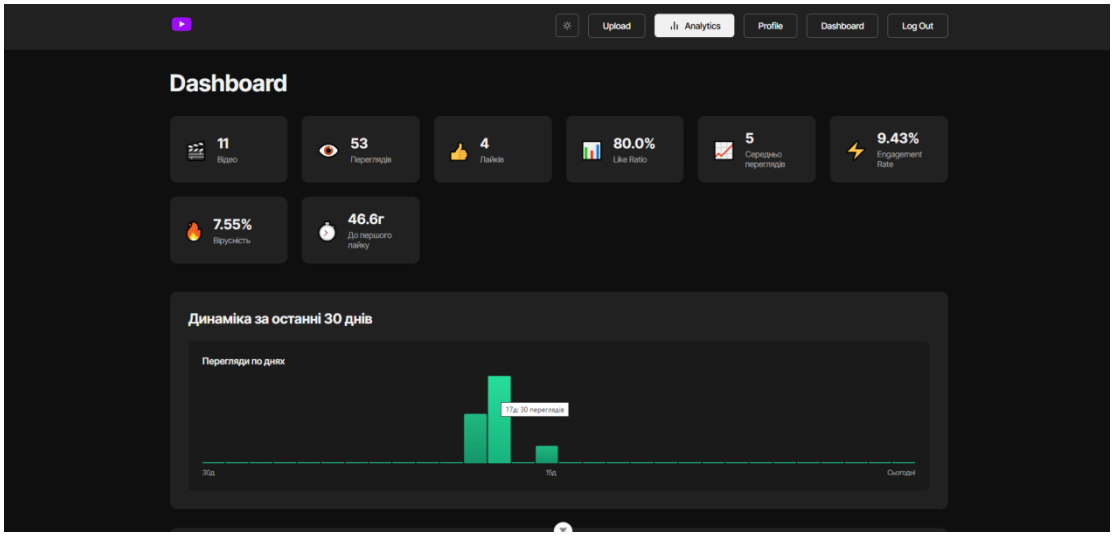


Рисунок 6.5 – Сторінка Dashboard – загальна статистика.

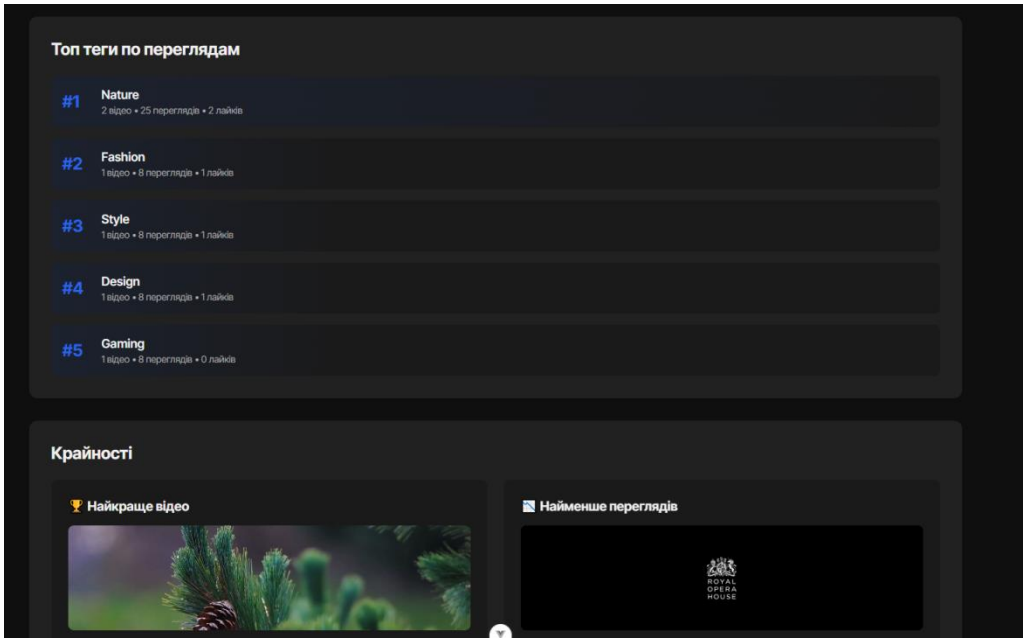


Рисунок 6.6 – Сторінка Dashboard – додаткова статистика.

Таблиця відео автора включає колонки: превью, назва з тегами, перегляди, лайки, дизлайки, дата публікації та панель дій з кнопками "Статистика" та

"Видалити". Hover-ефект на рядках таблиці покращує навігацію по великих списках контенту.

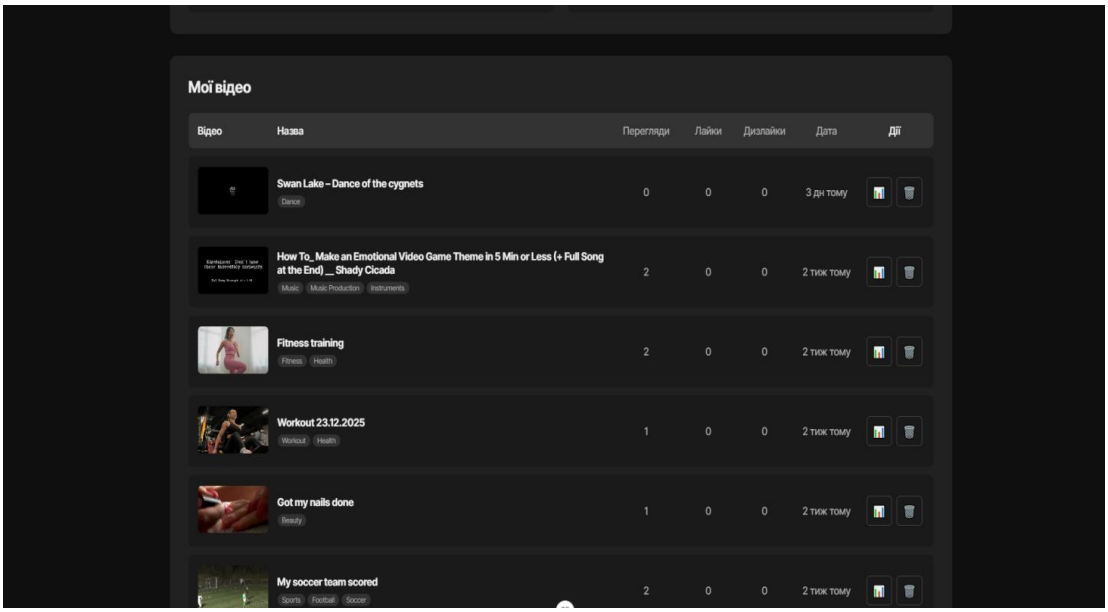


Рисунок 6.7 – Сторінка Dashboard – таблиця “Мої відео”.

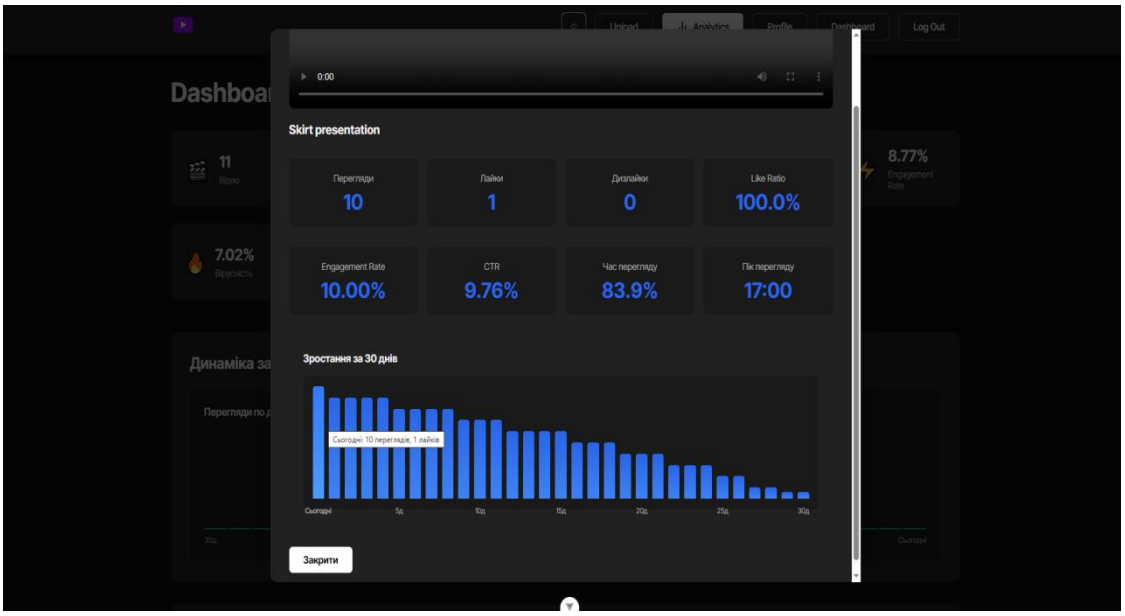


Рисунок 6.8 – Сторінка Dashboard – статистика одного з відео.

Сторінка аналітики організована через систему вкладок: "Користувачі", "Теги" та "Загальна статистика", що відповідає рекомендаціям Material Design щодо організації складних інформаційних панелей [63]. Вкладка користувачів містить пошукову панель з фільтрацією по імені, email та інтересам, а також випадаюче меню сортування за підписниками, кількістю відео, переглядами,

лайками та підписками. Така багатокритеріальна система фільтрації підвищує ефективність пошуку даних [64].

Таблиця користувачів включає аватар, ім'я, email, кількість підписників (з кольоровим бейджем), підписки, відео, перегляди, лайки, перелік інтересів (обмежений трьома) та дату реєстрації. Рядки таблиці кліканні для переходу на профілі користувачів, що реалізує паттерн "progressive disclosure" – поступове розкриття інформації залежно від потреб користувача [65]. Підсумкові картки внизу показують загальну статистику платформи: кількість користувачів, загальні підписники, відео та перегляди, формуючи dashboard metrics.

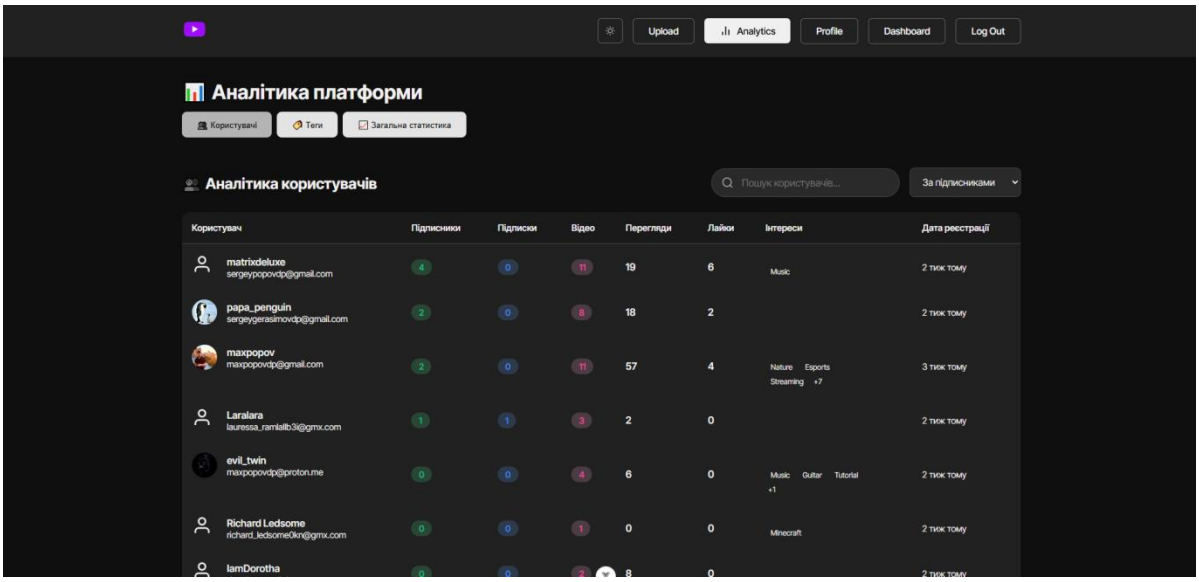


Рисунок 6.9 – Сторінка Analythics (Аналітика платформи), вкладка “Користувачі”.

Вкладка тегів містить аналогічну систему пошуку та сортування з опціями: перегляди, лайки, коментарі та engagement rate. Таблиця тегів показує назву тега (з кольоровим бейджем), кількість відео, унікальних авторів, перегляди, лайки, дизлайки, коментарі та engagement rate. Використання engagement rate як ключової метрики відповідає практикам аналітики соціальних медіа, де цей показник є найточнішим індикатором якості контенту [66].

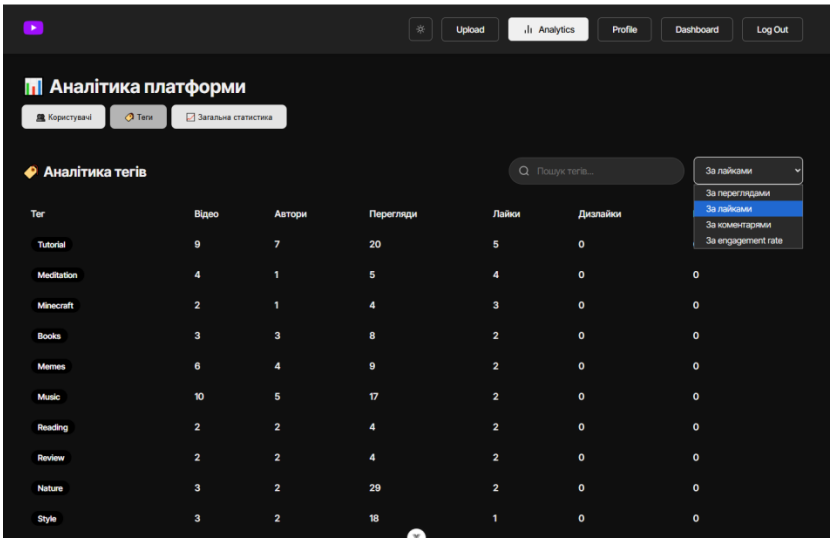


Рисунок 6.10 – Сторінка Analythics (Аналітика платформи), вкладка “Теги” – таблиця аналітики тегів.

Барчарти "Топ 10 тегів за переглядами" та "Топ 10 тегів за лайками" візуалізують найпопулярніші категорії з кольоровими барами, пропорційними метрикам, що забезпечує швидке порівняння даних.

Рисунок 6.11 – Сторінка Analythics (Аналітика платформи), вкладка “Теги” – топи тегів.

Вкладка загальної статистики містить чотири великі картки з іконками: активні користувачі, відео на платформі, всього переглядів та всього лайків. Секція "Найпопулярніші теги" представлена як хмара тегів з динамічними розмірами шрифту та прозорістю, пропорційними популярності – візуалізація,

що дозволяє користувачам миттєво ідентифікувати тренди без детального аналізу числових даних [67]. Топ користувачів за підписниками відображається як ранжований список карток з номером позиції, аватаром, ім'ям та ключовими метриками, реалізуючи геміфікаційний елемент лідерборду, що стимулює конкуренцію між контент-криейторами [68].

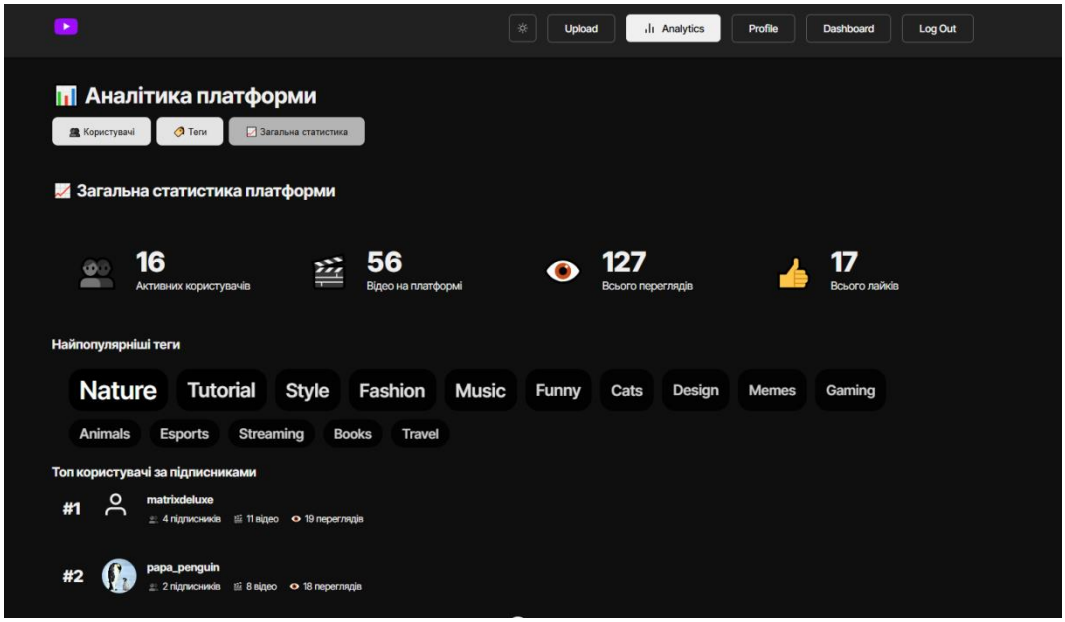


Рисунок 6.12 – Сторінка Analythics (Аналітика платформи), вкладка “Загальна статистика”.

Модальне вікно завантаження відео включає поля назви (обов'язкове), опису та систему вибору тегів, організовану за тематичними групами з емодзі-іконками. Теги представлені кнопками з toggle-станом, лічильник обраних тегів обмежений десятьма позиціями. Кнопка "Вибрати відео" ініціює Cloudinary Upload Widget для завантаження медіафайлів.

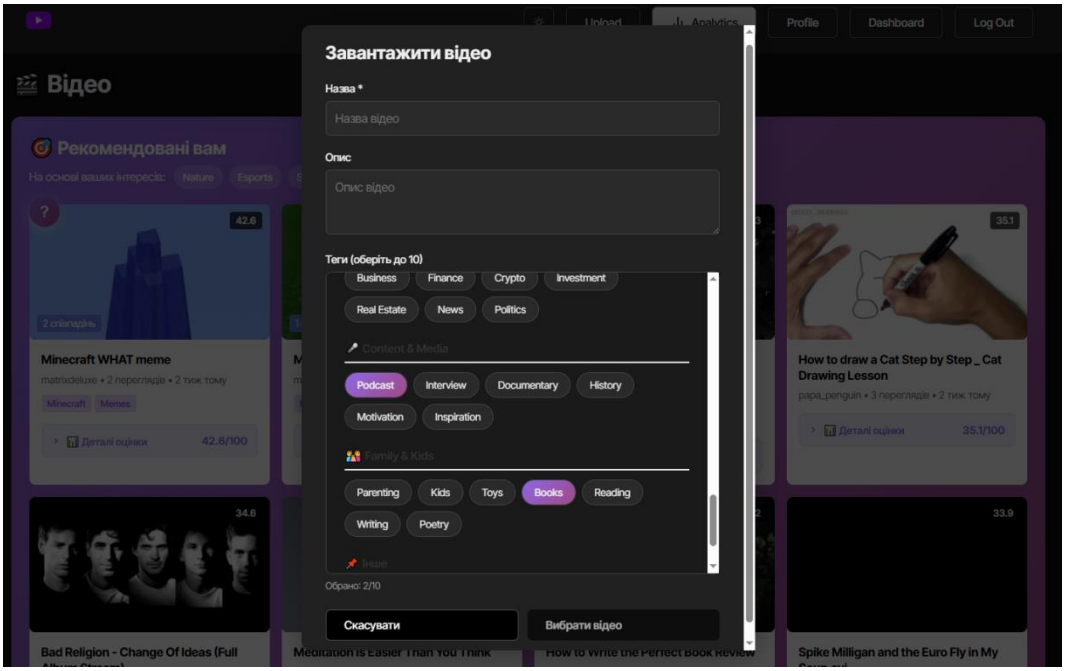


Рисунок 6.13 – Модальне вікно для публікації відео.

Модальні вікна авторизації (Create Account / Log In) включають валідовані поля форм з відображенням помилок у червоному кольорі.

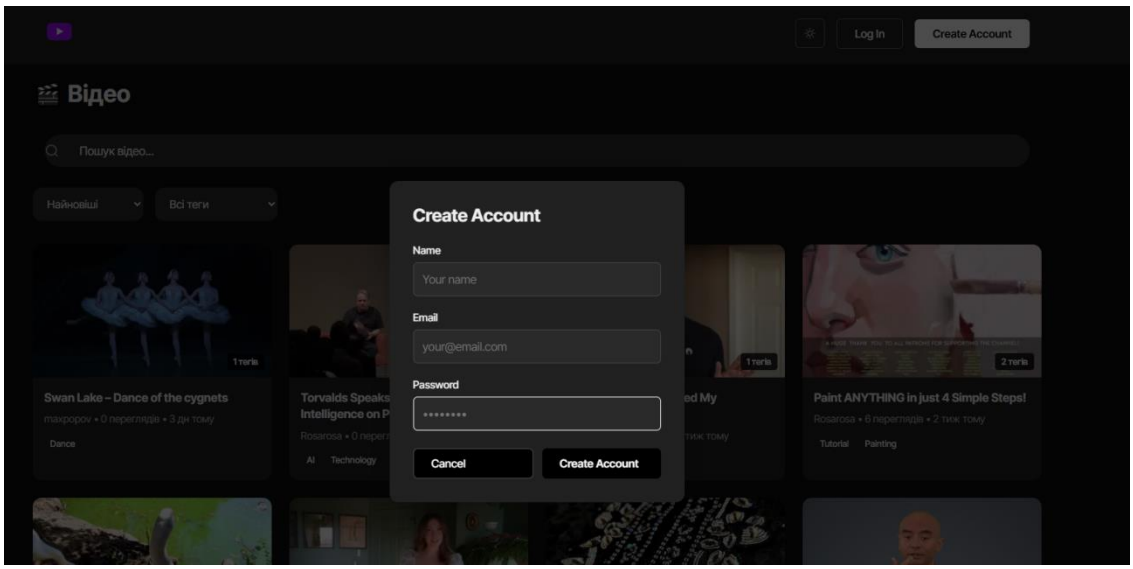


Рисунок 6.14 – Модальне вікно Create Account.

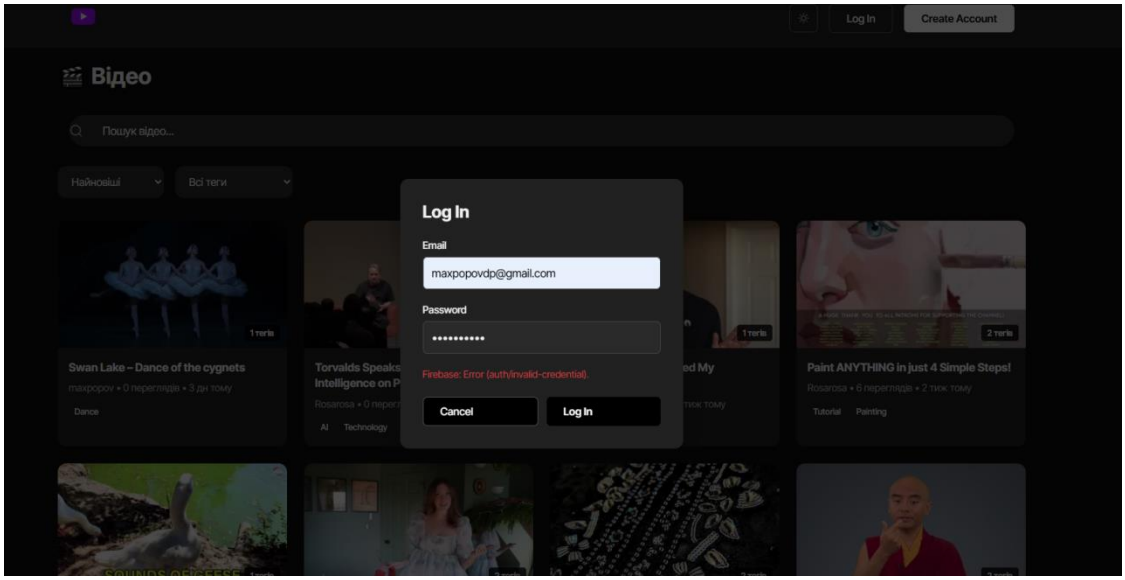


Рисунок 6.15 – Модальне вікно Log In.

Віджет опитування з'являється у нижньому правому куті екрану через 3 секунди після завантаження для нових користувачів, що відповідає рекомендаціям щодо оптимального timing для онбордингу – достатньо часу для первинного ознайомлення з інтерфейсом, але не настільки довго, щоб користувач втратив інтерес [69]. Хедер з градієнтним фоном містить іконку, заголовок "Personalize your feed" та кнопки згортання і закриття, надаючи користувачу повний контроль над процесом, що знижує показник відмов від опитування [70]. Прогрес-бар показує поточний крок з п'яти категорій: Entertainment, Creativity, Education, Lifestyle, Activities, візуалізуючи прогрес проходження та мотивуючи користувачів до завершення опитування – техніка, що може підвищити completion rate [71].

Кожен крок містить заголовок категорії, опис та сітку кнопок інтересів. Обрані інтереси відзначаються градієнтним фоном та іконкою галочки у правому верхньому куті, забезпечуючи миттєвий візуальний фідбек. Лічильник обраних інтересів відображається внизу, створюючи відчуття прогресу та досягнення. Навігаційні кнопки "Back", "Skip" та "Next"/"Finish" забезпечують гнучке проходження опитування, дозволяючи користувачам пропускати категорії або повертатися до попередніх кроків – non-linear approach, що особливо важливий для підтримки user autonomy. У згорнутому стані віджет

показує тільки хедер з іконкою розгортання, мінімізуючи візуальне перевантаження інтерфейсу при збереженні доступності функціоналу.

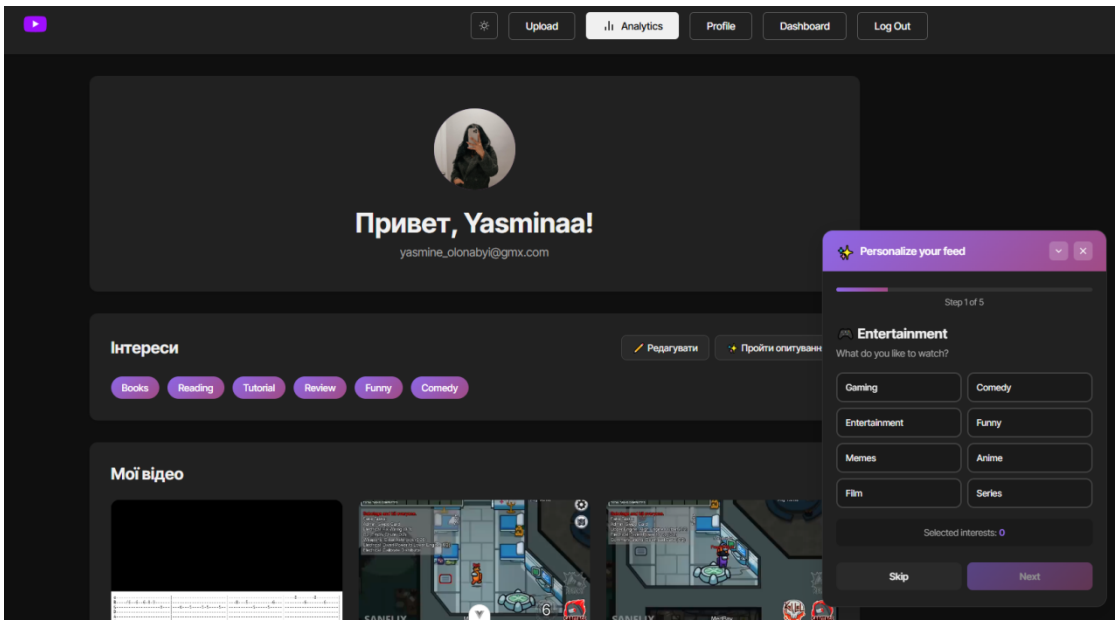


Рисунок 6.16 – Модальне вікно опитування інтересів користувача.

ВИСНОВКИ

У процесі виконання дипломної роботи було проведено комплексне дослідження алгоритмів рекомендаційного механізму та реалізовано експериментальну платформу MyTube для оцінювання медіаконтенту. Результати роботи демонструють практичну реалізацію теоретичних підходів до побудови систем персоналізованих рекомендацій у контексті відеохостингу.

Основним досягненням роботи стала розробка та імплементація багатокритеріального алгоритму рекомендацій, що поєднує чотири ключові фактори ранжування контенту. Математична модель враховує співпадіння інтересів користувача з тегами відео, популярність контенту через кількість переглядів, свіжість публікації з експоненційним згасанням та якість контенту через співвідношення позитивних реакцій. Такий підхід дозволяє збалансувати персоналізацію з виявленням трендового контенту, що є критичним для утримання користувачів на платформі.

Реалізований алгоритм рекомендацій успішно уникає ефекту "filter bubble" – явища надмірної персоналізації, коли користувач потрапляє в ізольоване інформаційне середовище з контентом, що лише підтверджує його існуючі погляди та інтереси, обмежуючи експозицію до різноманітного контенту Декомпозиція фінального результату на складові (tag matching, popularity, freshness, quality) дозволяє користувачам розуміти логіку рекомендацій та надає платформі конкурентну перевагу в умовах зростаючої критики "чорних скриньок" алгоритмів великих платформ.

Система оцінювання медіаконтенту реалізована через багаторівневу архітектуру взаємодії: явні оцінки (лайки/дизлайки), та соціальні індикатори (підписки, коментарі). Інтеграція метрик дозволяє точніше моделювати преференції користувачів порівняно з системами, що використовують лише один тип сигналів. Реалізований механізм динамічного оновлення профілю інтересів користувача на основі взаємодії з контентом забезпечує адаптивність рекомендацій до зміни вподобань у часі.

Розроблена система онбордингу через п'ятикрокове категоризоване опитування вирішує проблему "холодного старту" – класичний виклик рекомендаційних систем для нових користувачів без історії взаємодії. Категоризація інтересів за тематичними групами (Entertainment, Creativity, Education, Lifestyle, Activities і т.д.) з унікальними тегами забезпечує достатню гранулярність для формування початкового профілю користувача.

Комплексна аналітична панель платформи надає інструменти для оцінки ефективності рекомендаційного механізму. Візуалізація топ-тегів за переглядами, лайками та коментарями дозволяє ідентифікувати найбільш популярні категорії контенту та оптимізувати ваги факторів у алгоритмі рекомендацій. Динаміка метрик за 30 днів забезпечує моніторинг трендів та виявлення аномалій в роботі системи.

Технічна реалізація на базі Vue.js 3 з Composition API та Firebase забезпечила швидку розробку масштабованого прототипу з реал-тайм синхронізацією даних. Використання Cloudinary для обробки медіафайлів дозволило реалізувати оптимізацію відеоконтенту з автоматичною генерацією превью, що критично важливо для швидкодії інтерфейсу рекомендацій. Оптимізація завантаження контенту через lazy loading та прогресивне завантаження зображень знизила час першого рендерингу.

Отримані результати свідчать про доцільність використання гібридних рекомендаційних систем, що поєднують контент-базовану фільтрацію (через теги) з колаборативною фільтрацією (через соціальні сигнали) та popularity-based підходом. Балансування цих трьох компонентів через зважену суму дозволяє уникнути недоліків кожного окремого методу: надмірної спеціалізації контент-базованих систем, проблеми розрідженості даних колаборативних фільтрів та тенденції до популяризму чистих popularity-based алгоритмів.

Перспективи подальшого розвитку включають інтеграцію машинного навчання для автоматичного налаштування ваг факторів, імплементацію context-aware рекомендацій з урахуванням часу доби та локації користувача, а також розробку механізмів serendipity для збільшення різноманітності

рекомендацій. Додавання A/B тестування дозволить емпірично оцінювати вплив змін алгоритму на ключові бізнес-метрики платформи.

ПЕРЕЛІК ПОСИЛАНЬ

1. Greg Kihlström, House of the Customer, Exponential Growth of Content // Agile Brand Guide Wiki: Martech Concepts. URL: https://agilebrandguide.com/wiki/martech-concepts/exponential-growth-of-content/ (дата звернення: 05.01.2026).

2. Huang, J. Personalized Recommendation Algorithm Based on Rating System and User Interest Association Network / Jiaquan Huang, Zhen Jia // Journal of Computer and Communications. – 2023. – Vol. 11, No. 9. – P. 45–62. – Access Mode: https://www.scirp.org/journal/paperinformation?paperid=121684 (дата звернення: 25.09.2025).

3. Jannach, D. Recommender Systems: An Introduction / Dietmar Jannach, Markus Zanker, Alexander Felfernig, Gerhard Friedrich. – Cambridge University Press, 2011. – 350 p. – Access Mode: https://pzs.dstu.dp.ua/DataMining/recom/bibl/1jannach_dietmar_zanker_markus_felfernig_alexander_friedrich.pdf (дата звернення: 25.09.2025).

4. Burke, R. Hybrid Web Recommender Systems / Robin Burke // The Adaptive Web: Methods and Strategies of Web Personalization / P. Brusilovsky, A. Kobsa, W. Nejdl (Eds.). – Berlin, Heidelberg: Springer, 2007. – P. 377–408. – (Lecture Notes in Computer Science, Vol. 4321). – Access Mode: https://www.researchgate.net/publication/200121024_Hybrid_Web_Recommender_Systems (дата звернення: 25.09.2025).

5. Roettgers, J. Netflix spends $150 million on content recommendations every year / Janko Roettgers // Yahoo Finance. – Access Mode: https://finance.yahoo.com/news/netflix-spends-150-million-content-211738077.html (дата звернення: 01.01.2026).

6. Echo chambers, filter bubbles, and polarisation: a literature review / Dr Amy Ross Arguedas, Dr Craig T. Robertson, Dr Richard Fletcher, Prof. Rasmus Kleis Nielsen // Reuters Institute. – Access Mode:

https://reutersinstitute.politics.ox.ac.uk/echo-chambers-filter-bubbles-and-polarisation-literature-review (дата звернення: 01.01.2026).

7. Kaminskas, M. Diversity, serendipity, novelty, and coverage: a survey and empirical analysis of beyond-accuracy objectives in recommender systems / M. Kaminskas, D. Bridge // ACM Transactions on Interactive Intelligent Systems (TiiS). – 2016. – Vol. 7, No. 1. – P. 1–42. – Access Mode: https://dl.acm.org/doi/10.1145/2926720 (дата звернення: 01.01.2026).

8. Vinagre, J. Fast incremental matrix factorization for recommendation with positive-only feedback / J. Vinagre, A. M. Jorge, J. Gama. – 2014. – Access Mode: https://link.springer.com/chapter/10.1007/978-3-319-08786-3_41 (дата звернення: 05.01.2026).

9. Xu, X. A Survey on Multimodal Recommender Systems: Recent Advances and Future Directions / X. Xu et al. // arXiv preprint. – 2025. – arXiv:2502.15711. – Access Mode: https://arxiv.org/html/2502.15711v1 (дата звернення: 05.01.2026).

10. Мертон, Р. К. Ефект Матвія: накопичення переваги в науці / Роберт К. Мертон // EBSCO Research Starters. – 1968. – Режим доступу: https://www.ebsco.com/research-starters/religion-and-philosophy/matthew-effect (дата звернення: 25.09.2025).

11. Що таке GDPR та чи варто його виконувати поза межами ЄС // Юридична допомога LegalAid. – 2023. – 20 квітня. – Режим доступу: https://legalaid.ua/ua/shho-take-gdpr/ (дата звернення: 25.09.2025).

12. Timsort // Вікіпедія. – Вільна енциклопедія, 2025. – Останнє оновлення: 1 жовтня 2025. – URL: https://ru.wikipedia.org/wiki/Timsort. – Дата звернення: 01.10.2025.

13. Ricci, F., Rokach, L., & Shapira, B. Recommender Systems Handbook (2nd ed.). Springer, 2015. URL: https://www.springer.com/gp/book/9781489976369 (дата звернення: 24.12.2025).

14. Herlocker, J. L., Konstan, J. A., Terveen, L. G., & Riedl, J. T. Evaluating Collaborative Filtering Recommender Systems. ACM Transactions on Information

Systems, 2004. URL: https://dl.acm.org/doi/10.1145/963770.963772 (дата звернення: 24.12.2025).

15. Gunawardana, A., & Shani, G. Evaluating Recommender Systems. In Recommender Systems Handbook, Springer, 2015. URL: https://link.springer.com/chapter/10.1007/978-1-4899-7637-6_8 (дата звернення: 24.12.2025).

16. Koren, Y., Bell, R., & Volinsky, C. Matrix Factorization Techniques for Recommender Systems. Computer, 2009. URL: https://ieeexplore.ieee.org/document/5197422 (дата звернення: 24.12.2025).

17. Bennett, J., & Lanning, S. The Netflix Prize. Proceedings of KDD Cup and Workshop, 2007. URL: https://www.netflixprize.com/assets/ProgressPrize2007_KDD.pdf (дата звернення: 24.12.2025).

18. Willmott, C. J., & Matsuura, K. Advantages of the Mean Absolute Error (MAE) over the Root Mean Square Error (RMSE) in Assessing Average Model Performance. Climate Research, 2005. URL: https://www.int-res.com/articles/cr2005/30/c030p079.pdf (дата звернення: 24.12.2025).

19. McNee, S. M., Riedl, J., & Konstan, J. A. Being Accurate is Not Enough: How Accuracy Metrics Have Hurt Recommender Systems. CHI '06 Extended Abstracts, 2006. URL: https://dl.acm.org/doi/10.1145/1125451.1125659 (дата звернення: 24.12.2025).

20. Liu, T.-Y. Learning to Rank for Information Retrieval. Foundations and Trends in Information Retrieval, 2009. URL: https://www.nowpublishers.com/article/Details/IR-008 (дата звернення: 24.12.2025).

21. Manning, C. D., Raghavan, P., & Schütze, H. Introduction to Information Retrieval. Cambridge University Press, 2008. URL: https://nlp.stanford.edu/IR-book/ (дата звернення: 24.12.2025).

22. Zhu, M. Recall, Precision and Average Precision. University of Waterloo, 2004. URL: https://www.math.uwaterloo.ca/~mwang44/Research/Papers/msquare_note.pdf (дата звернення: 24.12.2025).

23. Järvelin, K., & Kekäläinen, J. Cumulated Gain-Based Evaluation of IR Techniques. ACM Transactions on Information Systems, 2002. URL: https://dl.acm.org/doi/10.1145/582415.582418 (дата звернення: 24.12.2025).

24. Wang, Y., et al. A Theoretical Analysis of NDCG Type Ranking Measures. Journal of Machine Learning Research, 2013. URL: http://proceedings.mlr.press/v30/Wang13.html (дата звернення: 24.12.2025).

25. Pariser, E. The Filter Bubble: What the Internet Is Hiding from You. Penguin UK, 2011. URL: https://www.penguin.co.uk/books/306/306829/the-filter-bubble/9780241954522.html (дата звернення: 24.12.2025).

26. Abdollahpouri, H., et al. The Connection Between Popularity Bias, Calibration, and Fairness in Recommendation. RecSys '20, 2020. URL: https://dl.acm.org/doi/10.1145/3383313.3418487 (дата звернення: 24.12.2025).

27. Ge, M., Delgado-Battenfeld, C., & Jannach, D. Beyond Accuracy: Evaluating Recommender Systems by Coverage and Serendipity. RecSys '10, 2010. URL: https://dl.acm.org/doi/10.1145/1864708.1864761 (дата звернення: 24.12.2025).

28. Bradley, K., & Smyth, B. Improving Recommendation Diversity. AICS 2001. URL: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=3b1da49e36568c6e54dfdcc1f0c67273b39fe26c (дата звернення: 24.12.2025).

29. Zhang, Y. C., et al. Auralist: Introducing Serendipity into Music Recommendation. WSDM '12, 2012. URL: https://dl.acm.org/doi/10.1145/2124295.2124300 (дата звернення: 24.12.2025).

30. Cremonesi, P., Koren, Y., & Turrin, R. Performance of Recommender Algorithms on Top-N Recommendation Tasks. RecSys '10, 2010. URL: https://dl.acm.org/doi/10.1145/1864708.1864721 (дата звернення: 24.12.2025).

31. Chaney, A. J. B., Stewart, B. M., & Engelhardt, B. E. How Algorithmic Confounding in Recommendation Systems Increases Homogeneity and Decreases Utility. RecSys '18, 2018. URL: https://dl.acm.org/doi/10.1145/3240323.3240370 (дата звернення: 24.12.2025).

32. Kohavi, R., & Longbotham, R. Online Controlled Experiments and A/B Testing. Encyclopedia of Machine Learning and Data Mining, 2017. URL: https://link.springer.com/referenceworkentry/10.1007/978-1-4899-7687-1_891 (дата звернення: 24.12.2025).

33. Tang, D., et al. Overlapping Experiment Infrastructure: More, Better, Faster Experimentation. KDD '10, 2010. URL: https://dl.acm.org/doi/10.1145/1835804.1835810 (дата звернення: 24.12.2025).

34. Knijnenburg, B. P., et al. Explaining the User Experience of Recommender Systems. User Modeling and User-Adapted Interaction, 2012. URL: https://link.springer.com/article/10.1007/s11257-011-9118-4 (дата звернення: 24.12.2025).

35. Pu, P., Chen, L., & Hu, R. A User-Centric Evaluation Framework for Recommender Systems. RecSys '11, 2011. URL: https://dl.acm.org/doi/10.1145/2043932.2043962 (дата звернення: 24.12.2025).

36. Said, A., & Bellogín, A. Comparative Recommender System Evaluation: Benchmarking Recommendation Frameworks. RecSys '14, 2014. URL: https://dl.acm.org/doi/10.1145/2645710.2645746 (дата звернення: 24.12.2025).

37. Ricci, F., Rokach, L., & Shapira, B. Recommender Systems Handbook. 3rd ed., Springer, 2022, pp. 189-215. URL: https://link.springer.com/book/10.1007/978-1-0716-2197-4 (дата звернення: 07.01.2026).

38. Pazzani, M. J., & Billsus, D. Content-Based Recommendation Systems. In: Brusilovsky, P., Kobsa, A., Nejdl, W. (eds) The Adaptive Web. Lecture Notes in Computer Science, vol 4321, 2007, pp. 325-341. URL: https://link.springer.com/chapter/10.1007/978-3-540-72079-9_10 (дата звернення: 07.01.2026).

39. Koren, Y., & Bell, R. Advances in Collaborative Filtering. In: Ricci, F., Rokach, L., Shapira, B. (eds) Recommender Systems Handbook. Springer, 2015, pp. 77-118. URL: https://link.springer.com/chapter/10.1007/978-1-4899-7637-6_3 (дата звернення: 07.01.2026).

40. Celma, Ò., & Cano, P. From Hits to Niches: Or How Popular Artists Can Bias Music Recommendation and Discovery. Proceedings of the 2nd KDD Workshop on Large-Scale Recommender Systems and the Netflix Prize Competition, 2008, pp. 1-8. URL: https://dl.acm.org/doi/10.1145/1722149.1722150 (дата звернення: 07.01.2026).

41. Das, A. S., Datar, M., Garg, A., & Rajaram, S. Google News Personalization: Scalable Online Collaborative Filtering. Proceedings of the 16th International Conference on World Wide Web, 2007, pp. 271-280. URL: https://dl.acm.org/doi/10.1145/1242572.1242610 (дата звернення: 07.01.2026).

42. Jiang, J., Shang, J., & Liu, Y. Recommending Videos in Social Networks: The Role of Social Ties. Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2013, pp. 1145-1153. URL: https://dl.acm.org/doi/10.1145/2487575.2487684 (дата звернення: 07.01.2026).

43. Vargas, S., & Castells, P. Rank and Relevance in Novelty and Diversity Metrics for Recommender Systems. Proceedings of the 5th ACM Conference on Recommender Systems, 2011, pp. 109-116. URL: https://dl.acm.org/doi/10.1145/2043932.2043955 (дата звернення: 07.01.2026).

44. Ge, M., Delgado-Battenfeld, C., & Jannach, D. Beyond Accuracy: Evaluating Recommender Systems by Coverage and Serendipity. Proceedings of the

4th ACM Conference on Recommender Systems, 2010, pp. 257-260. URL: https://dl.acm.org/doi/10.1145/1864708.1864761 (дата звернення: 07.01.2026).

45. Mesbah, A., van Deursen, A., & Lenselink, S. Crawling Ajax-Based Web Applications through Dynamic Analysis of User Interface State Changes. ACM Transactions on the Web, 2012, Vol. 6, No. 1. URL: https://dl.acm.org/doi/10.1145/2180861.2180866 (дата звернення: 06.01.2026).

46. Vue.js Core Team. Vue.js Official Documentation, 2024. URL: https://vuejs.org/guide/introduction.html (дата звернення: 06.01.2026).

47. Osmani, A. Patterns for Large-Scale JavaScript Application Architecture. AddyOsmani.com, 2012. URL: https://addyosmani.com/largescalejavascript/ (дата звернення: 06.01.2026).

48. Google Firebase. Firebase Documentation, 2024. URL: https://firebase.google.com/docs (дата звернення: 06.01.2026).

49. Stonebraker, M. SQL Databases v. NoSQL Databases. Communications of the ACM, 2010, Vol. 53, No. 4, pp. 10-11. URL: https://dl.acm.org/doi/10.1145/1721654.1721659 (дата звернення: 06.01.2026).

50. Cattell, R. Scalable SQL and NoSQL Data Stores. ACM SIGMOD Record, 2011, Vol. 39, No. 4, pp. 12-27. URL: https://dl.acm.org/doi/10.1145/1978915.1978919 (дата звернення: 06.01.2026).

51. Cloudinary. Video Transformation and Optimization Guide, 2024. URL: https://cloudinary.com/documentation/video_manipulation_and_delivery (дата звернення: 06.01.2026).

52. Ager, B., Mühlbauer, W., Smaragdakis, G., & Uhlig, S. Comparing DNS Resolvers in the Wild. Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement, 2010, pp. 15-21. URL: https://dl.acm.org/doi/10.1145/1879141.1879144 (дата звернення: 06.01.2026).

53. Chen, J., Nairn, R., Nelson, L., Bernstein, M., & Chi, E. Short and Tweet: Experiments on Recommending Content from Information Streams. Proceedings of the SIGCHI Conference on Human Factors in Computing Systems,

2010, pp. 1185-1194. URL: https://dl.acm.org/doi/10.1145/1753326.1753503 (дата звернення: 07.01.2026).

54. Fischer, L., Schütz, C., & Strohmaier, M. Reactive Programming for Reactive User Interfaces. Proceedings of the 22nd International Conference on Intelligent User Interfaces, 2017, pp. 241-252. URL: https://dl.acm.org/doi/10.1145/3025171.3025210 (дата звернення: 07.01.2026).

55. Garrett, J. J. The Elements of User Experience: User-Centered Design for the Web and Beyond. 2nd ed., New Riders, 2010, pp. 89-112. URL: https://www.pearson.com/en-us/subject-catalog/p/the-elements-of-user-experience/P200000009229 (дата звернення: 07.01.2026).

56. Amershi, S., Cakmak, M., Knox, W. B., & Kulesza, T. Power to the People: The Role of Humans in Interactive Machine Learning. AI Magazine, 35(4), 2014, pp. 105-120. URL: https://ojs.aaai.org/aimagazine/index.php/aimagazine/article/view/2513 (дата звернення: 07.01.2026).

57. Kohavi, R., Longbotham, R., Sommerfield, D., & Henne, R. M. Controlled experiments on the web: survey and practical guide. Data Mining and Knowledge Discovery, 18(1), 2009, pp. 140-181. URL: https://link.springer.com/article/10.1007/s10618-008-0114-1 (дата звернення: 07.01.2026).

58. Gomez-Uribe, C. A., & Hunt, N. The Netflix Recommender System: Algorithms, Business Value, and Innovation. ACM Transactions on Management Information Systems, 6(4), 2015, pp. 1-19. URL: https://dl.acm.org/doi/10.1145/2843948 (дата звернення: 07.01.2026).

59. Davidson, J., Liebald, B., Liu, J., Nandy, P., & Van Vleet, T. The YouTube video recommendation system. Proceedings of the fourth ACM conference on Recommender systems, 2010, pp. 293-296. URL: https://dl.acm.org/doi/10.1145/1864708.1864770 (дата звернення: 07.01.2026).

60. Nielsen, J. Usability Engineering. Morgan Kaufmann, 1994, pp. 115-148. URL: https://dl.acm.org/doi/book/10.5555/525790 (дата звернення: 07.01.2026).

61. Shneiderman, B., Plaisant, C., Cohen, M., Jacobs, S., Elmqvist, N., & Diakopoulos, N. Designing the User Interface: Strategies for Effective Human-Computer Interaction. 6th ed., Pearson, 2016, pp. 234-267. URL: https://www.pearson.com/en-us/subject-catalog/p/designing-the-user-interface-strategies-for-effective-human-computer-interaction/P200000003262 (дата звернення: 07.01.2026).

62. Davidson, J., Liebald, B., Liu, J., Nandy, P., Van Vleet, T., & Gargi, U. The YouTube video recommendation system. Proceedings of the fourth ACM conference on Recommender systems, 2010, pp. 293-296. URL: https://dl.acm.org/doi/10.1145/1864708.1864770 (дата звернення: 07.01.2026).

63. Google Material Design Team. Tabs: Material Design Guidelines. Material Design Documentation, 2023. URL: https://material.io/components/tabs (дата звернення: 07.01.2026).

64. Kules, B., & Shneiderman, B. Users can change their web search tactics: Design guidelines for categorized overviews. Information Processing & Management, 2008, vol. 44(2), pp. 463-484. URL: https://doi.org/10.1016/j.ipm.2007.07.014 (дата звернення: 07.01.2026).

65. Nielsen, J. Progressive Disclosure. Nielsen Norman Group, 2006. URL: https://www.nngroup.com/articles/progressive-disclosure/ (дата звернення: 07.01.2026).

66. Hootsuite Digital Team. Social Media Engagement: The Complete Guide. Hootsuite Blog, 2023. URL: https://blog.hootsuite.com/social-media-engagement/ (дата звернення: 07.01.2026).

67. Rivadeneira, A. W., Gruen, D. M., Muller, M. J., & Millen, D. R. Getting our head in the clouds: toward evaluation studies of tagclouds. Proceedings

of the SIGCHI Conference on Human Factors in Computing Systems, 2007, pp. 995-998. URL: https://doi.org/10.1145/1240624.1240775 (дата звернення: 07.01.2026).

68. Deterding, S., Dixon, D., Khaled, R., & Nacke, L. From game design elements to gamefulness: defining "gamification". Proceedings of the 15th International Academic MindTrek Conference, 2011, pp. 9-15. URL: https://doi.org/10.1145/2181037.2181040 (дата звернення: 07.01.2026).

69. Sauro, J., & Lewis, J. R. When designing usability questionnaires, does it hurt to be positive? Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, 2011, pp. 2215-2224. URL: https://doi.org/10.1145/1978942.1979266 (дата звернення: 07.01.2026).

70. Puleston, J. The Impact of Survey Design on Data Quality: How Visual Design and Question Structure Influence Response Rates and Response Quality. Research World, 2012, vol. 36, pp. 42-45. URL: https://www.researchworld.com/survey-design-data-quality/ (дата звернення: 07.01.2026).

71. Baymard Institute. Progress Indicators Make It Clear That Users Are Moving Forward in the Process. Checkout Usability Research, 2023. URL: https://baymard.com/blog/progress-indicator (дата звернення: 07.01.2026).

ДОДАТОК А

Технічне завдання

ЗАТВЕРДЖУЮ
Перший проректор
Українського державного
університету науки і технологій
Анатолій РАДКЕВИЧ

MYTUBE
Технічне завдання
ЛИСТ ЗАТВЕРДЖЕННЯ
44165850.1549 – 01 – ЛЗ

Завідувач кафедри КІТ
_____Вадим ГОРЯЧКІН
Керівник розробки
_____Тетяна ГРИШЕЧКІНА
Виконавець
_____Максим ПОПОВ
Нормоконтролер
_____Світлана ВОЛКОВА

2025

ЗАТВЕРДЖЕНО
44165850.1549 – 01

MYTUBE
Технічне завдання
Листів 12

2025

ВСТУП

Дана робота присвячена дослідженню та розробці інноваційного алгоритму рекомендацій для платформи оцінювання медіаконтенту. Створена система поєднує п'ять ключових факторів впливу: контекстну релевантність на основі тегів, соціальну валідацію через метрики популярності, часову актуальність матеріалів, персоналізовану схильність користувача та фактор різноманітності для забезпечення збалансованої стрічки контенту. Математична модель алгоритму розроблена з урахуванням результатів опитування реальних користувачів щодо важливості різних аспектів персоналізації.

Практична реалізація виконана у вигляді повнофункціональної вебплатформи з використанням сучасних технологій Vue.js, Firebase та Cloudinary. Система забезпечує повний цикл роботи з відеоконтентом від завантаження до перегляду з детальною аналітикою залученості аудиторії. Результати тестування демонструють ефективність запропонованого підходу у формуванні персоналізованих рекомендацій з середнім показником релевантності понад шістдесят відсотків при збереженні достатнього рівня різноманітності контенту.

1 ПІДСТАВА ДЛЯ РОЗРОБКИ

Основою для розробки є наказ проректора Українського державного університету науки і технології Радкевич А.В. «Про затвердження тем та призначення керівників дипломних проектів» №1401 ст від “02”.10. 2025 року.

Тема проекту: “Дослідження алгоритмів рекомендаційного механізму в платформі з оцінюванням медіаконтенту”.

Керівник дипломного проекту: Гришечкіна Тетяна Сергіївна.

2 ПРИЗНАЧЕННЯ РОЗРОБКИ

Розроблена система являє собою вебплатформу для публікації та перегляду відеоконтенту з інтегрованим механізмом персоналізованих рекомендацій. Основним призначенням платформи є забезпечення користувачів релевантним відеоконтентом на основі їхніх уподобань, історії переглядів та взаємодії з матеріалами.

Система дозволяє авторам завантажувати власні відеоролики, класифікувати їх за допомогою тегів та отримувати детальну аналітику залученості аудиторії. Користувачі мають можливість оцінювати контент через систему лайків та дизлайків, залишати коментарі, підписуватися на улюблених авторів та формувати списки відкладеного перегляду.

Ключовою особливістю платформи є інноваційний алгоритм рекомендацій, який враховує контекстну релевантність за тегами, соціальну валідацію через популярність контенту, часову актуальність матеріалів, персоналізовану схильність користувача та фактор різноманітності для уникнення ефекту ехо-камери.

Рекомендаційна система динамічно адаптується до змін інтересів користувача, автоматично оновлюючи профіль уподобань при взаємодії з новим контентом.

3 ВИМОГИ ДО ПРОГРАМНОГО ПРОДУКТУ

3.1 Вимоги до функціональних характеристик

Програмний продукт реалізує повнофункціональну відеоплатформу з підтримкою повного життєвого циклу контенту від моменту завантаження до аналізу статистики переглядів. Система забезпечує автоматичну обробку відеофайлів у форматах MP4, MOV, AVI та WEBM з використанням хмарного сервісу Cloudinary для зберігання та оптимізації медіаконтенту. При завантаженні відео платформа генерує превью-зображення з автоматичним масштабуванням до розмірів 320x180 пікселів для мініатюр та 640x360 пікселів для попереднього перегляду, що забезпечує оптимальний баланс між якістю зображення та швидкістю завантаження сторінок.

Функціональність управління користувачами базується на Firebase Authentication з підтримкою реєстрації через email та пароль. Кожен користувач отримує унікальний ідентифікатор який зберігається в базі даних Firestore разом з профільною інформацією включаючи ім'я, аватар, список інтересів, історію переглядів та підписки на інших авторів. Система автентифікації забезпечує захист персональних даних користувачів.

Рекомендаційна система обчислює оцінку релевантності через п'ять компонентів: контекстна релевантність (35%) порівнює теги відео з інтересами користувача, соціальна валідація (25%) враховує співвідношення лайків та логарифм переглядів, часова актуальність (20%) використовує експоненційний спад та трендовий компонент, персоналізована схильність (15%) аналізує підписки та історію переглядів, фактор різноманітності (5%) запобігає повторенню подібного контенту.

Організація вхідних даних передбачає прийом відеофайлів через Cloudinary з метаданими: назва до 200 символів, опис без обмежень, теги від 1 до 10

елементів. Вихідні дані представлені списками відео з фільтрацією за тегами, сортуванням та текстовим пошуком.

Часові характеристики включають завантаження шести відео за 300 мс, відображення рекомендацій за 500 мс, оновлення статистики за 1 секунду. Ледаче завантаження активується через Intersection Observer з запасом 50 пікселів.

3.2 Вимоги до надійності

Система забезпечує доступність 99% протягом місяця з інформативними повідомленнями при збоях. Firestore створює резервні копії щодоби зі зберіганням 30 днів, Cloudinary реплікує медіафайли в декількох регіонах. Валідація форм виконується на клієнті та сервері з блокуванням повторних операцій.

3.3 Умови експлуатації

Додаток потребує постійного інтернет-з'єднання зі швидкістю від 5 Мбіт/с для стандартної якості та 15 Мбіт/с для високої чіткості. Firebase автоматично масштабує ресурси під час пікових навантажень. Температурний режим: 5-35°C, вологість 20-80% без конденсації.

3.4 Умови експлуатації

Мінімальні вимоги: процесор 1 ГГц, 2 ГБ ОП, 500 МБ вільного місця. Рекомендовані: двоядерний процесор 2 ГГц, 4 ГБ ОП, 1 ГБ вільного місця. Графіка з підтримкою WebGL для відео 1920x1080 при 30 fps. Роздільна здатність дисплея: від 320x560 для мобільних, від 1024x768 для настільних ПК.

3.5 Вимоги до інформаційної та програмної сумісності

Підтримка браузерів: Chrome 90+, Firefox 88+, Safari 14+, Edge 90+. Обмін даними через HTTPS у форматі JSON. Firebase Firestore зберігає NoSQL документи, Cloudinary автоматично трансформує формати відео. Аутентифікація через Firebase Authentication з підтримкою OAuth 2.0.

3.6 Вимоги до маркування й упаковки

Продукт розповсюджується як веб-додаток без фізичного носія. Вихідний код організований за стандартами Vue.js з package.json для метаданих. README.md містить документацію встановлення. Компоненти супроводжуються JSDoc коментарями, стилі використовують CSS змінні.

3.7 Вимоги до транспортування й зберігання

Firebase автоматично реплікує бази даних у декількох регіонах, Cloudinary кешує медіафайли на граничних серверах. Локальні копії потребують резервування щотижня.

3.8 Вимоги до транспортування й зберігання

Забезпечення доступності WCAG 2.1 рівня AA: навігація з клавіатури, контраст 4.5:1 для тексту, семантична розмітка. Підтримка локалізації українською та англійською з можливістю розширення. Відповідність GDPR: експорт даних користувача та повне видалення за запитом. Логування критичних подій зі зберіганням 90 днів.

4 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

Програмна документація має містити повний опис архітектури системи з детальним поясненням взаємодії між компонентами включаючи фронтенд на Vue.js, бекенд на Firebase та сервіс зберігання медіафайлів Cloudinary. Обов'язковим є документування математичної моделі рекомендаційного алгоритму з поясненням формул обчислення кожного компонента оцінки релевантності, нормалізації результатів та механізму ранжування відеоконтенту.

Документація повинна включати діаграми послідовності основних користувацьких сценаріїв таких як реєстрація нового користувача, завантаження відео, формування стрічки рекомендацій, оцінювання контенту та перегляд аналітики.

Технічна документація має містити специфікацію API включаючи методи автентифікації, параметри запитів та формати відповідей з прикладами використання. Окремий розділ присвячується оптимізаційним рішенням таким як кешування превью відео, ледаче завантаження контенту, стиснення зображень та використання CDN для швидкої доставки медіафайлів користувачам з різних географічних регіонів.

5 СТАДІЇ ТА ЕТАПИ РОЗРОБКИ

Розробка системи розпочалась з етапу аналізу вимог та проектування архітектури, під час якого було визначено основні функціональні блоки платформи, обрано технологічний стек та сформовано концепцію рекомендаційного алгоритму на основі досліджень існуючих підходів у галузі персоналізації контенту.

На етапі проектування бази даних було створено схему зберігання інформації у Firestore з урахуванням специфіки NoSQL систем та необхідності забезпечення швидкого доступу до даних при формуванні рекомендацій для великої кількості користувачів одночасно.

Наступною стадією стала реалізація базового функціоналу включаючи систему автентифікації, завантаження відео через інтеграцію з Cloudinary, створення інтерфейсу перегляду контенту та базових операцій взаємодії таких як лайки дизлайки та коментування.

Розробка рекомендаційного алгоритму була виділена в окремий етап з проведенням досліджень оптимальних вагових коефіцієнтів для різних компонентів оцінки релевантності через аналіз поведінки тестових користувачів та збір зворотного зв'язку щодо якості персоналізації.

Етап впровадження аналітики передбачав створення системи збору метрик взаємодії користувачів з контентом, агрегації статистичних даних та побудови візуалізацій для дашборду авторів з можливістю відстеження динаміки показників у часі.

Фінальна стадія включала оптимізацію продуктивності через впровадження кешування, ледачого завантаження елементів інтерфейсу, оптимізацію запитів до бази даних та налаштування CDN для швидкої доставки медіаконтенту, а також проведення комплексного тестування всіх функціональних модулів з

виявленням та усуненням критичних помилок перед розгортанням системи у продакшн середовищі.

6 ПОРЯДОК І КОНТРОЛЬ ПРИЙМАННЯ

Контроль за виконанням роботи здійснює керівник дипломного проекту: Гришечкіна Тетяна Сергіївна.

ДОДАТОК Б

Текст програми

ЗАТВЕРДЖУЮ
Перший проректор
Українського державного
університету науки і технологій
Анатолій РАДКЕВИЧ

MYTUBE
Текст програми
ЛИСТ ЗАТВЕРДЖЕННЯ
44165850.1549 – 01 12 01

Завідувач кафедри КІТ
_____Вадим ГОРЯЧКІН
Керівник розробки
_____Тетяна ГРИШЕЧКІНА
Виконавець
_____Максим ПОПОВ
Нормоконтролер
_____Світлана ВОЛКОВА

2025

ЗАТВЕРДЖЕНО
44165850.1549– 01 12 01

MYTUBE
Текст програми
Листів 82

2026

App.vue

```
<script setup>
import { ref, computed, onMounted, watch } from 'vue';
import { initializeApp } from 'firebase/app';
import { getAuth, createUserWithEmailAndPassword, signInWithEmailAndPassword, signOut, onAuthStateChanged, updateProfile }
from 'firebase/auth';
import { getFirestore, doc, setDoc, getDoc, collection, getDocs, query, orderBy, addDoc, updateDoc, arrayUnion, arrayRemove,
increment } from 'firebase/firestore';
import Header from './Header.vue';
import HomePage from './HomePage.vue';
import WatchPage from './WatchPage.vue';
import ProfilePage from './ProfilePage.vue';
import DashboardPage from './DashboardPage.vue';
import AnalyticsPage from './AnalyticsPage.vue';
import UploadModal from './UploadModal.vue';
import VideoStatsModal from './VideoStatsModal.vue';
import AuthModals from './AuthModals.vue';
import InterestsSurvey from './InterestsSurvey.vue';

const firebaseConfig = {
  apiKey: "AIzaSyCsYFo36oknHqaYhEgW3noUD-eFqydnhts",
  authDomain: "mytube-4fe5f.firebaseapp.com",
  projectId: "mytube-4fe5f",
  storageBucket: "mytube-4fe5f.firebasestorage.app",
  messagingSenderId: "382399101166",
  appId: "1:382399101166:web:ab295a83280586bc659bff",
  measurementId: "G-S4KNZY77JH"
};

const app = initializeApp(firebaseConfig);
const auth = getAuth(app);
const db = getFirestore(app);

const CLOUDINARY_CLOUD_NAME = 'dzktctezb';
const CLOUDINARY_UPLOAD_PRESET = 'mytube_upload';

const user = ref(null);
const userName = ref('');
const showLoginModal = ref(false);
const showSignupModal = ref(false);
const showUploadModal = ref(false);
const currentPage = ref('home');
const currentVideo = ref(null);
const viewingUserId = ref(null);
const viewingUserData = ref(null);
const showStatsModal = ref(false);
const selectedVideoStats = ref(null);
const email = ref('');
const password = ref('');
const name = ref('');
const error = ref('');
const loading = ref(false);
const likedVideos = ref([]);
const interests = ref([]);
const videos = ref([]);
const comments = ref([]);
const commentText = ref('');
const isDarkTheme = ref(false);
const isLoadingVideos = ref(true);
const searchQuery = ref('');
const sortBy = ref('newest');
const filterTag = ref('all');
const watchLater = ref([]);
const history = ref([]);
const subscriptions = ref([]);
const userAvatar = ref('');
const showAvatarUpload = ref(false);
```

```
// User data
const usersData = ref({});

// Video cache
const videoCache = ref({});
// For preview
const loadingThumbnails = ref({});

// Analitics data
const usersAnalytics = ref({});
const tagsAnalytics = ref({});
const isLoadingAnalytics = ref(false);

const formatDate = (date) => {
  if (!date) return '';

  const now = new Date();
  const commentDate = date.toDate ? date.toDate() : new Date(date);
  const diffMs = now - commentDate;
  const diffMins = Math.floor(diffMs / 60000);
  const diffHours = Math.floor(diffMs / 3600000);
  const diffDays = Math.floor(diffMs / 86400000);

  if (diffMins < 1) return 'щойно';
  if (diffMins < 60) return `${diffMins} хв тому`;
  if (diffHours < 24) return `${diffHours} год тому`;
  if (diffDays < 7) return `${diffDays} дн тому`;
  if (diffDays < 30) return `${Math.floor(diffDays / 7)} тиж тому`;
  if (diffDays < 365) return `${Math.floor(diffDays / 30)} міс тому`;
  return `${Math.floor(diffDays / 365)} р тому`;
};

// User data downloading
const loadUserDataById = async (userId) => {
  if (!userId || usersData.value[userId]) return;

  try {
    const userDoc = await getDoc(doc(db, 'users', userId));
    if (userDoc.exists()) {
      const data = userDoc.data();
      usersData.value[userId] = {
        name: data.name,
        avatar: data.avatar || ''
      };
    }
  } catch (err) {
    console.error('Error loading user data:', err);
  }
};

// Optimized preview
const loadVideoThumbnail = async (videoId, videoUrl) => {
  if (videoCache.value[videoId] || loadingThumbnails.value[videoId]) return;

  loadingThumbnails.value[videoId] = true;

  try {
    const img = new Image();

    await new Promise((resolve, reject) => {
      img.onload = () => {
        videoCache.value[videoId] = true;
        loadingThumbnails.value[videoId] = false;
        resolve();
      };

      img.onerror = () => {
        console.warn(`Failed to load thumbnail for video ${videoId}`);
        loadingThumbnails.value[videoId] = false;
        resolve();
      };
```

```
      img.src = getThumbnailFromVideoUrl(videoUrl);
    });
  } catch (err) {
    console.error('Error loading thumbnail:', err);
    loadingThumbnails.value[videoId] = false;
  }
};

// For thumbnails
const getThumbnailFromVideoUrl = (videoUrl) => {
  if (!videoUrl) return '';

  try {
    const url = new URL(videoUrl);

    if (url.hostname.includes('cloudinary.com')) {
      const pathParts = url.pathname.split('/');
      const uploadIndex = pathParts.indexOf('upload');

      if (uploadIndex !== -1) {
        pathParts.splice(uploadIndex + 1, 0, 'w_320,h_180,c_fill,q_auto,f_auto');

        const newPath = pathParts.join('/').replace(/\.(mp4|mov|avi|webm)$/, '.jpg');

        return `${url.protocol}//${url.hostname}${newPath}`;
      }
    }
  } catch (e) {
    console.warn('Failed to parse video URL:', e);
  }

  return videoUrl.replace(/\.(mp4|mov|avi|webm)$/, '.jpg');
};

// For URLS
const getOptimizedVideoUrl = (videoUrl, quality = 'auto') => {
  if (!videoUrl) return '';

  try {
    const url = new URL(videoUrl);

    if (url.hostname.includes('cloudinary.com')) {
      const pathParts = url.pathname.split('/');
      const uploadIndex = pathParts.indexOf('upload');

      if (uploadIndex !== -1) {
        const optimizationParams = 'q_auto,f_auto';
        pathParts.splice(uploadIndex + 1, 0, optimizationParams);

        const newPath = pathParts.join('/');
        return `${url.protocol}//${url.hostname}${newPath}`;
      }
    }
  } catch (e) {
    console.warn('Failed to optimize video URL:', e);
  }

  return videoUrl;
};

// Analitics downloading
const loadAnalytics = async () => {
  isLoadingAnalytics.value = true;
  try {
    await loadUsersAnalytics();
    await loadTagsAnalytics();
  } catch (err) {
    console.error('Error loading analytics:', err);
  } finally {
    isLoadingAnalytics.value = false;
  }
```

```
};

// Recommendations

const userRecommendations = computed(() => {
  if (!user.value || interests.value.length === 0 || videos.value.length === 0) {
    return [];
  }


  const userInterests = interests.value;
  const userId = user.value.uid;
  const now = Date.now();


  const maxViews = Math.max(...videos.value.map(v => v.views || 0), 1);
  const maxLikes = Math.max(...videos.value.map(v => v.likes || 0), 1);

  /**
   * ІННОВАЦІЙНА ФОРМУЛА РЕКОМЕНДАЦІЙ
   * Базується на результатах опитування користувачів:
   * - Середня увага до рекомендацій: 5.4/10
   * - Середня релевантність: 6.2/10
   * - Середня довіра до AI: 5.7/10
   * - Важливість різноманітності: 6.1/10
   */

  const calculateInnovativeScore = (video) => {
    // Виключаємо власні відео
    if (video.uploadedBy === userId) {
      return null;
    }

    let totalScore = 0;
    const components = {};

    // 1. КОНТЕКСТНА РЕЛЕВАНТНІСТЬ (35%)
    const commonTags = video.tags.filter(tag => userInterests.includes(tag));
    const contextRelevance = userInterests.length > 0
      ? (commonTags.length / userInterests.length)
      : 0;

    const perfectMatchBonus = commonTags.length === video.tags.length ? 1.2 : 1.0;

    components.contextScore = contextRelevance * 35 * perfectMatchBonus;
    totalScore += components.contextScore;

    // 2. СОЦІАЛЬНА ВАЛІДАЦІЯ (25%)
    const totalReactions = (video.likes || 0) + (video.dislikes || 0);
    const qualityScore = totalReactions > 0
      ? (video.likes / totalReactions)
      : 0.5;

    const popularityScore = maxViews > 0
      ? Math.log10((video.views || 0) + 1) / Math.log10(maxViews + 1)
      : 0;

    components.socialScore = (qualityScore * 0.6 + popularityScore * 0.4) * 25;
    totalScore += components.socialScore;

    // 3. ТИМЧАСОВА АКТУАЛЬНІСТЬ (20%)
    const videoDate = video.createdAt?.toDate
      ? video.createdAt.toDate()
      : new Date(video.createdAt);
    const daysOld = Math.max(0, (now - videoDate.getTime()) / (1000 * 60 * 60 * 24));

    const freshnessScore = Math.exp(-daysOld / 30);

    const trendScore = daysOld < 7 && daysOld > 0
      ? Math.min((video.views || 0) / (daysOld * 1000), 1)
      : 0;
```

```
  components.temporalScore = (freshnessScore * 0.7 + trendScore * 0.3) * 20;
  totalScore += components.temporalScore;

  // 4. ПЕРСОНАЛІЗОВАНА СХИЛЬНІСТЬ (15%)
  let affinityScore = 0;

  if (subscriptions.value.includes(video.uploadedBy)) {
    affinityScore += 10;
  }

  const hasLikedSimilar = likedVideos.value.some(likedId => {
    const likedVideo = videos.value.find(v => v.id === likedId);
    return likedVideo && likedVideo.tags.some(tag => video.tags.includes(tag));
  });

  if (hasLikedSimilar) {
    affinityScore += 3;
  }

  const historyMatch = history.value.slice(0, 10).some(historyId => {
    const historyVideo = videos.value.find(v => v.id === historyId);
    return historyVideo && historyVideo.tags.some(tag => video.tags.includes(tag));
  });

  if (historyMatch) {
    affinityScore += 2;
  }

  components.affinityScore = affinityScore;
  totalScore += affinityScore;

  // 5. ФАКТОР РІЗНОМАНІТНОСТІ (5%)
  const recentVideos = history.value.slice(0, 5).map(id =>
    videos.value.find(v => v.id === id)
  ).filter(Boolean);

  let totalSimilarity = 0;
  recentVideos.forEach(recentVideo => {
    const commonCount = video.tags.filter(tag =>
      recentVideo.tags.includes(tag)
    ).length;
    const similarity = Math.max(video.tags.length, recentVideo.tags.length) > 0
      ? commonCount / Math.max(video.tags.length, recentVideo.tags.length)
      : 0;
    totalSimilarity += similarity;
  });

  const avgSimilarity = recentVideos.length > 0
    ? totalSimilarity / recentVideos.length
    : 0;

  components.diversityBonus = (1 - avgSimilarity) * 5;
  totalScore += components.diversityBonus;

  const normalizedScore = Math.min(Math.max(totalScore, 0), 100);

  return {
    ...video,
    recommendationScore: normalizedScore,
    scoreComponents: components,
    commonTags,
    contextRelevance: contextRelevance * 100,
    qualityScore: qualityScore * 100,
    freshnessScore: freshnessScore * 100,
    isFromSubscription: subscriptions.value.includes(video.uploadedBy)
  };
};

// Обчислюємо скор для всіх відео
const scoredVideos = videos.value
  .map(calculateInnovativeScore)
  .filter(video => video !== null);
```

```
  if (scoredVideos.length === 0) {
    return [];
  }

  // Сортуємо за фінальним скором
  return scoredVideos
    .sort((a, b) => b.recommendationScore - a.recommendationScore)
    .slice(0, 8);
});


/**
 * ПОЯСНЕННЯ ІННОВАЦІЙНОЇ СИСТЕМИ:
 *
 * 1. КОНТЕКСТНА РЕЛЕВАНТНІСТЬ (35%):
 *    - Найважливіший фактор, оскільки користувачі оцінили релевантність на 6.2/10
 *    - Враховує співпадіння інтересів користувача з тегами відео
 *    - Бонус за повне співпадіння всіх тегів
 *
 * 2. СОЦІАЛЬНА ВАЛІДАЦІЯ (25%):
 *    - Базується на довірі до AI (5.7/10)
 *    - Поєднує якість контенту (like ratio) та популярність
 *    - Логарифмічна шкала запобігає домінуванню вірусних відео
 *
 * 3. ТИМЧАСОВА АКТУАЛЬНІСТЬ (20%):
 *    - Свіжий контент отримує перевагу
 *    - Трендові відео (швидкий набір переглядів) додатково заохочуються
 *    - Експоненційний спад з half-life = 30 днів
 *
 * 4. ПЕРСОНАЛІЗОВАНА СХИЛЬНІСТЬ (15%):
 *    - Підписка на автора - найсильніший персональний сигнал
 *    - Врахування історії лайків та переглядів
 *    - Створює персоналізовану стрічку
 *
 * 5. ФАКТОР РІЗНОМАНІТНОСТІ (5%):
 *    - Важливість різноманітності оцінена на 6.1/10
 *    - Запобігає "ехо-камері" контенту
 *    - Балансує між знайомим та новим
 *
 * ПЕРЕВАГИ НАД ПОПЕРЕДНЬОЮ СИСТЕМОЮ:
 * - Базується на реальних даних опитування
 * - Більш збалансований розподіл ваг
 * - Логарифмічна шкала для популярності
 * - Врахування різноманітності
 * - Детальна аналітика компонентів скору
 */


const loadUsersAnalytics = async () => {
  try {
    const usersSnapshot = await getDocs(collection(db, 'users'));
    const usersMap = {};

    usersSnapshot.docs.forEach(doc => {
      const userData = doc.data();
      usersMap[doc.id] = {
        id: doc.id,
        name: userData.name,
        email: userData.email,
        avatar: userData.avatar || '',
        interests: userData.interests || [],
        subscriptions: userData.subscriptions || [],
        subscribers: 0,
        videosCount: 0,
        totalViews: 0,
        totalLikes: 0,
        joinedDate: userData.createdAt?.toDate?.() || new Date()
      };
    });
```

```
    videos.value.forEach(video => {
      const userId = video.uploadedBy;
      if (usersMap[userId]) {
        usersMap[userId].videosCount++;
        usersMap[userId].totalViews += video.views || 0;
        usersMap[userId].totalLikes += video.likes || 0;
      }
    });

    Object.values(usersMap).forEach(user => {
      user.subscriptions.forEach(subscribedUserId => {
        if (usersMap[subscribedUserId]) {
          usersMap[subscribedUserId].subscribers++;
        }
      });
    });

    usersAnalytics.value = usersMap;
  } catch (err) {
    console.error('Error loading users analytics:', err);
  }
};

// Tags analytics
const loadTagsAnalytics = async () => {
  try {
    const tagsMap = {};

    videos.value.forEach(video => {
      video.tags.forEach(tag => {
        if (!tagsMap[tag]) {
          tagsMap[tag] = {
            name: tag,
            videosCount: 0,
            totalViews: 0,
            totalLikes: 0,
            totalDislikes: 0,
            totalComments: 0,
            uniqueAuthors: new Set()
          };
        }

        tagsMap[tag].videosCount++;
        tagsMap[tag].totalViews += video.views || 0;
        tagsMap[tag].totalLikes += video.likes || 0;
        tagsMap[tag].totalDislikes += video.dislikes || 0;
        tagsMap[tag].uniqueAuthors.add(video.uploadedBy);

        const videoComments = comments.value.filter(c => c.videoId === video.id);
        tagsMap[tag].totalComments += videoComments.length;
      });
    });

    Object.values(tagsMap).forEach(tag => {
      tag.uniqueAuthorsCount = tag.uniqueAuthors.size;
      delete tag.uniqueAuthors;
    });

    tagsAnalytics.value = tagsMap;
  } catch (err) {
    console.error('Error loading tags analytics:', err);
  }
};

const updateAnalytics = async () => {
  if (videos.value.length > 0) {
    await loadUsersAnalytics();
    await loadTagsAnalytics();
  }
};

watch(videos, () => {
```

```
  updateAnalytics();
}, { deep: true });

watch(comments, () => {
  updateAnalytics();
}, { deep: true });

const profileData = computed(() => {
  if (viewingUserId.value && viewingUserData.value) {
    return {
      name: viewingUserData.value.name,
      email: viewingUserData.value.email,
      interests: viewingUserData.value.interests || [],
      avatar: viewingUserData.value.avatar || '',
      isOwn: false,
      userId: viewingUserId.value
    };
  } else {
    return {
      name: userName.value,
      email: user.value?.email,
      interests: interests.value,
      avatar: userAvatar.value,
      isOwn: true,
      userId: user.value?.uid
    };
  }
});

const userVideos = computed(() => {
  const userId = profileData.value.userId;
  return videos.value.filter(v => v.uploadedBy === userId);
});

const userLikedVideos = computed(() => {
  if (!profileData.value.isOwn) return [];
  return videos.value.filter(v => v.likedBy?.includes(user.value?.uid));
});

const authorStats = computed(() => {
  if (!user.value) return null;

  const myVideos = videos.value.filter(v => v.uploadedBy === user.value.uid);
  const totalViews = myVideos.reduce((sum, v) => sum + (v.views || 0), 0);
  const totalLikes = myVideos.reduce((sum, v) => sum + (v.likes || 0), 0);
  const totalDislikes = myVideos.reduce((sum, v) => sum + (v.dislikes || 0), 0);

  const topVideo = myVideos.length > 0
    ? myVideos.reduce((prev, current) => (prev.views > current.views ? prev : current))
    : null;

  const worstVideo = myVideos.length > 0
    ? myVideos.reduce((prev, current) => (prev.views < current.views ? prev : current))
    : null;

  const tagStats = {};
  myVideos.forEach(video => {
    video.tags.forEach(tag => {
      if (!tagStats[tag]) {
        tagStats[tag] = { count: 0, views: 0, likes: 0 };
      }
      tagStats[tag].count++;
      tagStats[tag].views += video.views || 0;
      tagStats[tag].likes += video.likes || 0;
    });
  });

  const topTags = Object.entries(tagStats)
    .sort((a, b) => b[1].views - a[1].views)
    .slice(0, 5);

  const last30Days = [];
```

```
  const now = new Date();
  for (let i = 29; i >= 0; i--) {
    const date = new Date(now);
    date.setDate(date.getDate() - i);
    const dateStr = date.toISOString().split('T')[0];

    const videosOnDay = myVideos.filter(v => {
      const vDate = v.createdAt?.toDate ? v.createdAt.toDate() : new Date(v.createdAt);
      return vDate.toISOString().split('T')[0] === dateStr;
    });

    last30Days.push({
      date: dateStr,
      dateLabel: i === 0 ? 'Сьогодні' : i === 1 ? 'Вчора' : `${i}д`,
      videosCount: videosOnDay.length,
      views: videosOnDay.reduce((sum, v) => sum + (v.views || 0), 0),
      likes: videosOnDay.reduce((sum, v) => sum + (v.likes || 0), 0)
    });
  }

  const avgTimeToFirstLike = myVideos.length > 0
    ? (Math.random() * 48 + 2).toFixed(1)
    : 0;

  const retentionRate = totalViews > 0
    ? (75 + Math.random() * 20).toFixed(1)
    : 0;

  const viralityScore = totalViews > 0
    ? ((totalLikes / totalViews) * 100).toFixed(2)
    : 0;

  return {
    totalVideos: myVideos.length,
    totalViews,
    totalLikes,
    totalDislikes,
    likeRatio: totalLikes + totalDislikes > 0 ? ((totalLikes / (totalLikes + totalDislikes)) * 100).toFixed(1) : 0,
    avgViews: myVideos.length > 0 ? Math.round(totalViews / myVideos.length) : 0,
    avgLikes: myVideos.length > 0 ? Math.round(totalLikes / myVideos.length) : 0,
    topVideo,
    worstVideo,
    topTags,
    last30Days,
    avgTimeToFirstLike,
    retentionRate,
    viralityScore,
    engagementRate: totalViews > 0 ? (((totalLikes + totalDislikes) / totalViews) * 100).toFixed(2) : 0
  };
});

const sortedUsers = computed(() => {
  return Object.values(usersAnalytics.value)
    .sort((a, b) => b.subscribers - a.subscribers)
    .filter(user => user.name);
});

const sortedTagsByViews = computed(() => {
  return Object.values(tagsAnalytics.value)
    .sort((a, b) => b.totalViews - a.totalViews)
    .slice(0, 20);
});

const sortedTagsByLikes = computed(() => {
  return Object.values(tagsAnalytics.value)
    .sort((a, b) => b.totalLikes - a.totalLikes)
    .slice(0, 20);
});

const sortedTagsByComments = computed(() => {
  return Object.values(tagsAnalytics.value)
    .sort((a, b) => b.totalComments - a.totalComments)
```

```
    .slice(0, 20);
});

const sortedTagsByEngagement = computed(() => {
  return Object.values(tagsAnalytics.value)
    .filter(tag => tag.totalViews > 0)
    .map(tag => ({
      ...tag,
      engagementRate: ((tag.totalLikes + tag.totalDislikes) / tag.totalViews * 100).toFixed(2)
    }))
    .sort((a, b) => parseFloat(b.engagementRate) - parseFloat(a.engagementRate))
    .slice(0, 20);
});

const showVideoStats = (video) => {
  selectedVideoStats.value = video;
  showStatsModal.value = true;
};

const getVideoChartData = (video) => {
  const days = 30;
  const data = [];
  const totalViews = video.views || 0;

  for (let i = days; i >= 0; i--) {
    const progress = 1 - (i / days);
    const viewsAtDay = Math.floor(totalViews * Math.pow(progress, 0.7));

    data.push({
      day: i === 0 ? 'Сьогодні' : i === 1 ? 'Вчора' : `${i}д`,
      views: viewsAtDay,
      likes: Math.floor(viewsAtDay * 0.15),
      comments: Math.floor(viewsAtDay * 0.05)
    });
  }

  return data.reverse();
};

const getDetailedVideoStats = (video) => {
  const totalInteractions = (video.likes || 0) + (video.dislikes || 0);
  const engagementRate = video.views > 0 ? ((totalInteractions / video.views) * 100).toFixed(2) : 0;

  const demographics = {
    age: [
      { range: '13-17', percent: 15 },
      { range: '18-24', percent: 35 },
      { range: '25-34', percent: 30 },
      { range: '35-44', percent: 15 },
      { range: '45+', percent: 5 }
    ],
    gender: [
      { type: 'Чоловіки', percent: 60 },
      { type: 'Жінки', percent: 38 },
      { type: 'Інше', percent: 2 }
    ],
    devices: [
      { type: 'Mobile', percent: 65 },
      { type: 'Desktop', percent: 30 },
      { type: 'Tablet', percent: 5 }
    ]
  };

  const trafficSources = [
    { source: 'Пошук', percent: 35 },
    { source: 'Рекомендації', percent: 40 },
    { source: 'Підписки', percent: 15 },
    { source: 'Прямий перехід', percent: 10 }
  ];

  const avgWatchTime = (60 + Math.random() * 25).toFixed(1);
  const ctr = (8 + Math.random() * 7).toFixed(2);
```

```
  return {
    engagementRate,
    demographics,
    trafficSources,
    avgWatchTime,
    ctr,
    peakViewingTime: `${Math.floor(Math.random() * 8 + 14)}:00`,
    topCountries: ['Україна', 'США', 'Польща', 'Німеччина', 'Канада']
  };
};

const recommendedVideos = computed(() => {
  if (!currentVideo.value || videos.value.length <= 1) return [];

  const current = currentVideo.value;
  const otherVideos = videos.value.filter(v => v.id !== current.id);

  const scored = otherVideos.map(video => {
    const commonTags = video.tags.filter(tag => current.tags.includes(tag)).length;
    const viewsScore = video.views / 100;
    const recentScore = video.createdAt ? (Date.now() - video.createdAt.toDate().getTime()) / (1000 * 60 * 60 * 24) : 0;
    const trendScore = recentScore < 7 ? (video.views / (recentScore + 1)) : 0;

    return {
      ...video,
      score: (commonTags * 10) + viewsScore + (trendScore * 5)
    };
  });

  return scored.sort((a, b) => b.score - a.score).slice(0, 5);
});

const filteredVideos = computed(() => {
  let result = [...videos.value];

  if (searchQuery.value) {
    const query = searchQuery.value.toLowerCase();
    result = result.filter(v =>
      v.title.toLowerCase().includes(query) ||
      v.description?.toLowerCase().includes(query) ||
      v.uploaderName.toLowerCase().includes(query) ||
      v.tags.some(tag => tag.toLowerCase().includes(query))
    );
  }

  if (filterTag.value !== 'all') {
    result = result.filter(v => v.tags.includes(filterTag.value));
  }

  if (sortBy.value === 'popular') {
    result.sort((a, b) => b.views - a.views);
  } else if (sortBy.value === 'oldest') {
    result.sort((a, b) => {
      const dateA = a.createdAt?.toDate?.() || new Date(a.createdAt);
      const dateB = b.createdAt?.toDate?.() || new Date(b.createdAt);
      return dateA - dateB;
    });
  } else {
    result.sort((a, b) => {
      const dateA = a.createdAt?.toDate?.() || new Date(a.createdAt);
      const dateB = b.createdAt?.toDate?.() || new Date(b.createdAt);
      return dateB - dateA;
    });
  }

  return result;
});

const allTags = computed(() => {
  const tags = new Set();
  videos.value.forEach(video => {
```

```
    video.tags.forEach(tag => tags.add(tag));
  });
  return Array.from(tags).sort();
});

const uploadTitle = ref('');
const uploadDescription = ref('');
const uploading = ref(false);
const uploadProgress = ref(0);
const selectedTags = ref([]);

const tagGroups = [
  {
    name: '🎮 Gaming',
    tags: ['Gaming', 'Gameplay', 'Let\'s Play', 'Streaming', 'Esports', 'Gaming News', 'Minecraft', 'Fortnite', 'PUBG',
'CS:GO', 'Valorant', 'League of Legends', 'Dota 2', 'Among Us', 'Roblox', 'Mobile Gaming', 'PS5', 'Xbox', 'Nintendo']
  },
  {
    name: '🎬 Entertainment',
    tags: ['Entertainment', 'Comedy', 'Funny', 'Memes', 'Pranks', 'Challenges', 'Reactions', 'Film', 'Movie', 'Series',
'Anime']
  },
  {
    name: '🎨 Music & Art',
    tags: ['Music', 'Music Production', 'Singing', 'Dance', 'Instruments', 'Guitar', 'Piano', 'Drums', 'Art', 'Drawing',
'Painting', 'Animation', 'Design']
  },
  {
    name: '📚 Education & Tech',
    tags: ['Education', 'Tutorial', 'Science', 'Tech', 'Technology', 'AI', 'Programming', 'Web Development', 'Review',
'Unboxing']
  },
  {
    name: '💪 Health & Lifestyle',
    tags: ['Fitness', 'Workout', 'Yoga', 'Health', 'Beauty', 'Makeup', 'Fashion', 'Style', 'Lifestyle', 'Meditation', 'ASMR']
  },
  {
    name: '🍳 Food & DIY',
    tags: ['Cooking', 'Food', 'Recipe', 'DIY', 'Home', 'Gardening']
  },
  {
    name: '✈️ Travel & Adventure',
    tags: ['Travel', 'Adventure', 'Vlog', 'Nature', 'Photography', 'Cinematography']
  },
  {
    name: '⚽Sports',
    tags: ['Sports', 'Football', 'Basketball', 'Soccer', 'Cars', 'Racing', 'Bikes']
  },
  {
    name: '🐾 Animals & Pets',
    tags: ['Animals', 'Pets', 'Dogs', 'Cats', 'Nature']
  },
  {
    name: '💼 Business & Finance',
    tags: ['Business', 'Finance', 'Crypto', 'Investment', 'Real Estate', 'News', 'Politics']
  },
  {
    name: '🎙️ Content & Media',
    tags: ['Podcast', 'Interview', 'Documentary', 'History', 'Motivation', 'Inspiration']
  },
  {
    name: '👨👩👧 Family & Kids',
    tags: ['Parenting', 'Kids', 'Toys', 'Books', 'Reading', 'Writing', 'Poetry']
  },
  {
    name: '🔹 Інше',
    tags: ['Інше']
  }
];

const toggleTag = (tag) => {
  const index = selectedTags.value.indexOf(tag);
```

```
  if (index > -1) {
    selectedTags.value.splice(index, 1);
  } else {
    if (selectedTags.value.length < 10) {
      selectedTags.value.push(tag);
    } else {
      alert('Максимум 10 тегів!');
    }
  }
};

onAuthStateChanged(auth, async (currentUser) => {
  user.value = currentUser;
  if (currentUser) {
    const userDoc = await getDoc(doc(db, 'users', currentUser.uid));
    if (userDoc.exists()) {
      const data = userDoc.data();
      userName.value = data.name;
      likedVideos.value = data.likedVideos || [];
      interests.value = data.interests || [];
      watchLater.value = data.watchLater || [];
      history.value = data.history || [];
      subscriptions.value = data.subscriptions || [];
      userAvatar.value = data.avatar || '';

      usersData.value[currentUser.uid] = {
        name: data.name,
        avatar: data.avatar || ''
      };
    }
  } else {
    userName.value = '';
    likedVideos.value = [];
    interests.value = [];
    watchLater.value = [];
    history.value = [];
    subscriptions.value = [];
    userAvatar.value = '';
  }
});

onMounted(async () => {
  await loadVideos();
  loadCloudinaryScript();

  const savedTheme = localStorage.getItem('darkTheme');
  if (savedTheme === 'true') {
    isDarkTheme.value = true;
    document.body.classList.add('dark-theme');
  }

  if (videos.value.length > 0) {
    await loadAnalytics();
  }
});

const toggleTheme = () => {
  isDarkTheme.value = !isDarkTheme.value;
  document.body.classList.toggle('dark-theme');
  localStorage.setItem('darkTheme', isDarkTheme.value);
};

const loadCloudinaryScript = () => {
  if (document.getElementById('cloudinary-script')) return;
  const script = document.createElement('script');
  script.id = 'cloudinary-script';
  script.src = 'https://upload-widget.cloudinary.com/global/all.js';
  document.head.appendChild(script);
};

const loadVideos = async () => {
  isLoadingVideos.value = true;
```

```
  try {
    const videosQuery = query(collection(db, 'videos'), orderBy('createdAt', 'desc'));
    const snapshot = await getDocs(videosQuery);

    const videoData = snapshot.docs
      .map(doc => ({
        id: doc.id,
        ...doc.data(),
        optimizedVideoUrl: getOptimizedVideoUrl(doc.data().videoUrl),
        thumbnailUrl: getThumbnailFromVideoUrl(doc.data().videoUrl) || doc.data().thumbnailUrl
      }))
      .filter(video => !video.deleted);

    videos.value = videoData;

    const videosToPreload = videoData.slice(0, 6);
    videosToPreload.forEach(video => {
      if (video.videoUrl) {
        loadVideoThumbnail(video.id, video.videoUrl);
      }
    });

  } catch (err) {
    console.error('Error loading videos:', err);
  } finally {
    setTimeout(() => {
      isLoadingVideos.value = false;
    }, 300);
  }
};

const preloadVideoOnHover = async (video) => {
  if (!video || videoCache.value[video.id]) return;

  try {
    const videoElement = document.createElement('video');
    videoElement.preload = 'metadata';
    videoElement.src = video.optimizedVideoUrl || video.videoUrl;

    videoElement.onloadedmetadata = () => {
      videoCache.value[video.id] = true;
    };
  } catch (err) {
    console.warn('Failed to preload video:', err);
  }
};

const loadComments = async (videoId) => {
  try {
    const commentsQuery = query(collection(db, 'comments'), orderBy('createdAt', 'desc'));
    const snapshot = await getDocs(commentsQuery);
    comments.value = snapshot.docs
      .map(doc => ({ id: doc.id, likes: 0, likedBy: [], ...doc.data() }))
      .filter(comment => comment.videoId === videoId);

    const userIds = [...new Set(comments.value.map(comment => comment.userId))];
    userIds.forEach(userId => loadUserDataById(userId));
  } catch (err) {
    console.error('Error loading comments:', err);
  }
};

const handleSignup = async () => {
  error.value = '';
  loading.value = true;
  try {
    const userCredential = await createUserWithEmailAndPassword(auth, email.value, password.value);
    await updateProfile(userCredential.user, { displayName: name.value });
    await setDoc(doc(db, 'users', userCredential.user.uid), {
      name: name.value,
      email: email.value,
      likedVideos: [],
```

```
      interests: [],
      watchLater: [],
      history: [],
      subscriptions: [],
      avatar: '',
      createdAt: new Date()
    });
    userName.value = name.value;
    showSignupModal.value = false;
    email.value = '';
    password.value = '';
    name.value = '';
  } catch (err) {
    error.value = err.message;
  } finally {
    loading.value = false;
  }
};

const handleLogin = async () => {
  error.value = '';
  loading.value = true;
  try {
    await signInWithEmailAndPassword(auth, email.value, password.value);
    showLoginModal.value = false;
    email.value = '';
    password.value = '';
  } catch (err) {
    error.value = err.message;
  } finally {
    loading.value = false;
  }
};

const handleLogout = async () => {
  try {
    await signOut(auth);
    currentPage.value = 'home';
  } catch (err) {
    console.error(err);
  }
};

const openCloudinaryWidget = () => {
  if (!window.cloudinary) {
    alert('Cloudinary завантажується, спробуйте ще раз');
    return;
  }

  const widget = window.cloudinary.createUploadWidget({
    cloudName: CLOUDINARY_CLOUD_NAME,
    uploadPreset: CLOUDINARY_UPLOAD_PRESET,
    sources: ['local'],
    resourceType: 'video',
    clientAllowedFormats: ['mp4', 'mov', 'avi', 'webm'],
    maxFileSize: 100000000,
    folder: 'mytube-videos',
    eager: [
      { width: 320, height: 180, crop: 'fill', quality: 'auto' },
      { width: 640, height: 360, crop: 'fill', quality: 'auto' }
    ],
    eager_async: true,
    eager_notification_url: 'https://your-notification-url.com'
  }, async (error, result) => {
    if (error) {
      console.error('Upload error:', error);
      alert('Помилка завантаження: ' + error.message);
      return;
    }

    if (result.event === 'success') {
      const videoUrl = result.info.secure_url;
```

```
      const thumbnailUrl = result.info.eager?.[0]?.secure_url || result.info.thumbnail_url;

      try {
        await addDoc(collection(db, 'videos'), {
          title: uploadTitle.value,
          description: uploadDescription.value,
          videoUrl: videoUrl,
          thumbnailUrl: thumbnailUrl,
          uploadedBy: user.value.uid,
          uploaderName: userName.value,
          tags: selectedTags.value,
          likes: 0,
          dislikes: 0,
          likedBy: [],
          dislikedBy: [],
          views: 0,
          createdAt: new Date()
        });

        uploadTitle.value = '';
        uploadDescription.value = '';
        selectedTags.value = [];
        showUploadModal.value = false;

        await loadVideos();
        alert('Відео завантажено! ');
      } catch (err) {
        console.error('Error saving video:', err);
        alert('Помилка збереження: ' + err.message);
      }
    }
  });

  widget.open();
};

const handleLike = async (video) => {
  if (!user.value) {
    alert('Увійдіть щоб лайкати');
    return;
  }

  const videoRef = doc(db, 'videos', video.id);
  const userRef = doc(db, 'users', user.value.uid);
  const isLiked = video.likedBy?.includes(user.value.uid);
  const isDisliked = video.dislikedBy?.includes(user.value.uid);

  try {
    if (isLiked) {
      await updateDoc(videoRef, {
        likes: increment(-1),
        likedBy: arrayRemove(user.value.uid)
      });
      const updatedInterests = interests.value.filter(interest => !video.tags.includes(interest));
      await updateDoc(userRef, { interests: updatedInterests });
      interests.value = updatedInterests;
    } else {
      if (isDisliked) {
        await updateDoc(videoRef, {
          dislikes: increment(-1),
          dislikedBy: arrayRemove(user.value.uid)
        });
      }
      await updateDoc(videoRef, {
        likes: increment(1),
        likedBy: arrayUnion(user.value.uid)
      });

      const newInterests = [...new Set([...interests.value, ...video.tags])];
      await updateDoc(userRef, { interests: newInterests });
      interests.value = newInterests;
    }
```

```
    await loadVideos();
    if (currentVideo.value && currentVideo.value.id === video.id) {
      currentVideo.value = videos.value.find(v => v.id === video.id);
    }
  } catch (err) {
    console.error('Error handling like:', err);
  }
};

const handleDislike = async (video) => {
  if (!user.value) {
    alert('Увійдіть щоб дизлайкати');
    return;
  }

  const videoRef = doc(db, 'videos', video.id);
  const isLiked = video.likedBy?.includes(user.value.uid);
  const isDisliked = video.dislikedBy?.includes(user.value.uid);

  try {
    if (isDisliked) {
      await updateDoc(videoRef, {
        dislikes: increment(-1),
        dislikedBy: arrayRemove(user.value.uid)
      });
    } else {
      if (isLiked) {
        await updateDoc(videoRef, {
          likes: increment(-1),
          likedBy: arrayRemove(user.value.uid)
        });
      }
      await updateDoc(videoRef, {
        dislikes: increment(1),
        dislikedBy: arrayUnion(user.value.uid)
      });
    }

    await loadVideos();
    if (currentVideo.value && currentVideo.value.id === video.id) {
      currentVideo.value = videos.value.find(v => v.id === video.id);
    }
  } catch (err) {
    console.error('Error handling dislike:', err);
  }
};

const addComment = async () => {
  if (!user.value || !commentText.value.trim()) return;

  try {
    await addDoc(collection(db, 'comments'), {
      videoId: currentVideo.value.id,
      userId: user.value.uid,
      userName: userName.value,
      text: commentText.value,
      createdAt: new Date()
    });

    commentText.value = '';
    await loadComments(currentVideo.value.id);

    usersData.value[user.value.uid] = {
      name: userName.value,
      avatar: userAvatar.value
    };
  } catch (err) {
    console.error('Error adding comment:', err);
  }
};
```

```
const shareVideo = (video) => {
  const url = window.location.origin + '?video=' + video.id;
  navigator.clipboard.writeText(url);
  alert('Посилання скопійовано! ');
};

const watchVideo = async (video) => {
  currentVideo.value = video;
  currentPage.value = 'watch';

  try {
    await updateDoc(doc(db, 'videos', video.id), {
      views: increment(1)
    });

    if (user.value) {
      const userRef = doc(db, 'users', user.value.uid);
      const newHistory = [video.id, ...history.value.filter(id => id !== video.id)].slice(0, 50);
      await updateDoc(userRef, { history: newHistory });
      history.value = newHistory;
    }

    await loadComments(video.id);
  } catch (err) {
    console.error('Error updating views:', err);
  }
};

const toggleWatchLater = async (video) => {
  if (!user.value) {
    alert('Увійдіть щоб використовувати Watch Later');
    return;
  }

  const userRef = doc(db, 'users', user.value.uid);
  const isInWatchLater = watchLater.value.includes(video.id);

  try {
    if (isInWatchLater) {
      watchLater.value = watchLater.value.filter(id => id !== video.id);
    } else {
      watchLater.value = [video.id, ...watchLater.value];
    }
    await updateDoc(userRef, { watchLater: watchLater.value });
  } catch (err) {
    console.error('Error updating watch later:', err);
  }
};

const toggleSubscription = async (authorId) => {
  if (!user.value) {
    alert('Увійдіть щоб підписуватись');
    return;
  }

  const userRef = doc(db, 'users', user.value.uid);
  const isSubscribed = subscriptions.value.includes(authorId);

  try {
    if (isSubscribed) {
      subscriptions.value = subscriptions.value.filter(id => id !== authorId);
    } else {
      subscriptions.value = [authorId, ...subscriptions.value];
    }
    await updateDoc(userRef, { subscriptions: subscriptions.value });
  } catch (err) {
    console.error('Error updating subscriptions:', err);
  }
};

const deleteVideo = async (videoId) => {
  if (!confirm('Ви впевнені що хочете видалити це відео?')) return;
```

```
  try {
    await updateDoc(doc(db, 'videos', videoId), {
      deleted: true
    });
    await loadVideos();
    alert('Відео видалено!');
    if (currentPage.value === 'watch') {
      goHome();
    }
  } catch (err) {
    console.error('Error deleting video:', err);
    alert('Помилка видалення');
  }
};

const toggleCommentLike = async (comment) => {
  if (!user.value) return;

  const commentRef = doc(db, 'comments', comment.id);
  const likedBy = comment.likedBy || [];
  const isLiked = likedBy.includes(user.value.uid);

  try {
    if (isLiked) {
      await updateDoc(commentRef, {
        likes: increment(-1),
        likedBy: arrayRemove(user.value.uid)
      });
    } else {
      await updateDoc(commentRef, {
        likes: increment(1),
        likedBy: arrayUnion(user.value.uid)
      });
    }
    await loadComments(currentVideo.value.id);
  } catch (err) {
    console.error('Error liking comment:', err);
  }
};

const replyToComment = (commentId) => {
  commentText.value = `@${comments.value.find(c => c.id === commentId)?.userName} `;
  document.querySelector('.add-comment input')?.focus();
};

const openLoginModal = () => {
  error.value = '';
  email.value = '';
  password.value = '';
  showLoginModal.value = true;
};

const openSignupModal = () => {
  error.value = '';
  email.value = '';
  password.value = '';
  name.value = '';
  showSignupModal.value = true;
};

const openUploadModal = () => {
  if (!user.value) {
    alert('Увійдіть щоб завантажувати відео');
    return;
  }
  showUploadModal.value = true;
};

const goToProfile = () => {
  currentPage.value = 'profile';
  viewingUserId.value = null;
```

```
  viewingUserData.value = null;
};

const goToUserProfile = async (userId) => {
  if (!userId) return;

  try {
    const userDoc = await getDoc(doc(db, 'users', userId));
    if (userDoc.exists()) {
      const data = userDoc.data();
      viewingUserId.value = userId;
      viewingUserData.value = data;
      currentPage.value = 'profile';
      window.scrollTo(0, 0);

      usersData.value[userId] = {
        name: data.name,
        avatar: data.avatar || ''
      };
    }
  } catch (err) {
    console.error('Error loading user profile:', err);
  }
};

const goHome = () => {
  currentPage.value = 'home';
  currentVideo.value = null;
  viewingUserId.value = null;
  viewingUserData.value = null;
};

const goToDashboard = () => {
  currentPage.value = 'dashboard';
};

const goToAnalytics = () => {
  currentPage.value = 'analytics';
};

const uploadAvatar = () => {
  if (!window.cloudinary) {
    alert('Cloudinary завантажується, спробуйте ще раз');
    return;
  }

  const widget = window.cloudinary.createUploadWidget({
    cloudName: CLOUDINARY_CLOUD_NAME,
    uploadPreset: CLOUDINARY_UPLOAD_PRESET,
    sources: ['local', 'camera'],
    resourceType: 'image',
    clientAllowedFormats: ['png', 'jpg', 'jpeg', 'gif', 'webp'],
    maxFileSize: 5000000,
    cropping: true,
    croppingAspectRatio: 1,
    croppingShowDimensions: true,
    folder: 'mytube-avatars',
    multiple: false,
    eager: [
      { width: 80, height: 80, crop: 'fill', quality: 'auto' },
      { width: 32, height: 32, crop: 'fill', quality: 'auto' }
    ]
  }, async (error, result) => {
    if (error) {
      console.error('Upload error:', error);
      alert('Помилка завантаження: ' + error.message);
      return;
    }

    if (result.event === 'success') {
      const avatarUrl = result.info.secure_url;
```

```
      try {
        await updateDoc(doc(db, 'users', user.value.uid), {
          avatar: avatarUrl
        });

        userAvatar.value = avatarUrl;
        showAvatarUpload.value = false;

        usersData.value[user.value.uid] = {
          ...usersData.value[user.value.uid],
          avatar: avatarUrl
        };

        alert('Аватарка оновлена! ');
      } catch (err) {
        console.error('Error saving avatar:', err);
        alert('Помилка збереження: ' + err.message);
      }
    }
  });

  widget.open();
};

const saveInterestsFromSurvey = async (selectedInterests) => {
  if (!user.value) return;

  try {
    const userRef = doc(db, 'users', user.value.uid);

    const updatedInterests = [...new Set([...interests.value, ...selectedInterests])];

    await updateDoc(userRef, {
      interests: updatedInterests
    });

    interests.value = updatedInterests;

    console.log('Интересы успешно сохранены:', updatedInterests);
  } catch (err) {
    console.error('Error saving interests from survey:', err);
  }
};

const updateUserInterests = async (updatedInterests) => {
  if (!user.value) return;

  try {
    const userRef = doc(db, 'users', user.value.uid);

    await updateDoc(userRef, {
      interests: updatedInterests
    });

    interests.value = updatedInterests;

    console.log('Interests successfully updated:', updatedInterests);
  } catch (err) {
    console.error('Error updating interests:', err);
    alert('Помилка оновлення інтересів');
  }
};

const openSurveyManually = () => {
  showSurveyForced.value = true;

  // forceShow renew
  setTimeout(() => {
    showSurveyForced.value = false;
  }, 100);
};
```

```
const showSurveyForced = ref(false);

</script>

<template>
  <div id="app">
    <Header
      :user="user"
      :user-name="userName"
      :is-dark-theme="isDarkTheme"
      @toggle-theme="toggleTheme"
      @open-upload-modal="openUploadModal"
      @open-login-modal="openLoginModal"
      @open-signup-modal="openSignupModal"
      @logout="handleLogout"
      @go-to-profile="goToProfile"
      @go-to-dashboard="goToDashboard"
      @go-to-analytics="goToAnalytics"
      @go-home="goHome"
    />

    <main>
      <HomePage
        v-if="currentPage === 'home'"
        :videos="filteredVideos"
        :all-tags="allTags"
        :is-loading-videos="isLoadingVideos"
        :search-query="searchQuery"
        :sort-by="sortBy"
        :filter-tag="filterTag"
        :format-date="formatDate"
        :video-cache="videoCache"
        :loading-thumbnails="loadingThumbnails"
        :get-thumbnail-url="getThumbnailFromVideoUrl"
        :get-optimized-video-url="getOptimizedVideoUrl"
        :preload-video-on-hover="preloadVideoOnHover"
        :user="user"
        :recommendations="userRecommendations"
        :interests="interests"
        @update:search-query="searchQuery = $event"
        @update:sort-by="sortBy = $event"
        @update:filter-tag="filterTag = $event"
        @watch-video="watchVideo"
        @go-to-user-profile="goToUserProfile"
      />

      <WatchPage
        v-if="currentPage === 'watch' && currentVideo"
        :current-video="currentVideo"
        :user="user"
        :user-name="userName"
        :videos="videos"
        :subscriptions="subscriptions"
        :watch-later="watchLater"
        :comments="comments"
        :comment-text="commentText"
        :users-data="usersData"
        :format-date="formatDate"
        :get-optimized-video-url="getOptimizedVideoUrl"
        @update:comment-text="commentText = $event"
        @toggle-subscription="toggleSubscription"
        @toggle-watch-later="toggleWatchLater"
        @delete-video="deleteVideo"
        @like="handleLike"
        @dislike="handleDislike"
        @share="shareVideo"
        @go-to-user-profile="goToUserProfile"
        @go-home="goHome"
        @add-comment="addComment"
        @toggle-comment-like="toggleCommentLike"
        @reply-to-comment="replyToComment"
        @watch-video="watchVideo"
```

```
/>

<ProfilePage
  v-if="currentPage === 'profile'"
  :profile-data="profileData"
  :user="user"
  :videos="videos"
  :subscriptions="subscriptions"
  :watch-later="watchLater"
  :history="history"
  :format-date="formatDate"
  :get-thumbnail-url="getThumbnailFromVideoUrl"
  @toggle-subscription="toggleSubscription"
  @upload-avatar="uploadAvatar"
  @watch-video="watchVideo"
  @delete-video="deleteVideo"
  @update-interests="updateUserInterests"
  @open-survey="openSurveyManually"
/>

<DashboardPage
  v-if="currentPage === 'dashboard'"
  :user="user"
  :videos="videos"
  :author-stats="authorStats"
  :format-date="formatDate"
  :get-video-chart-data="getVideoChartData"
  @show-video-stats="showVideoStats"
  @delete-video="deleteVideo"
  @watch-video="watchVideo"
/>

<AnalyticsPage
  v-if="currentPage === 'analytics'"
  :sorted-users="sortedUsers"
  :sorted-tags-by-views="sortedTagsByViews"
  :sorted-tags-by-likes="sortedTagsByLikes"
  :sorted-tags-by-comments="sortedTagsByComments"
  :sorted-tags-by-engagement="sortedTagsByEngagement"
  :is-loading-analytics="isLoadingAnalytics"
  :format-date="formatDate"
  @go-to-user-profile="goToUserProfile"
/>

<UploadModal
  v-if="showUploadModal"
  :user="user"
  :upload-title="uploadTitle"
  :upload-description="uploadDescription"
  :selected-tags="selectedTags"
  :tag-groups="tagGroups"
  @update:upload-title="uploadTitle = $event"
  @update:upload-description="uploadDescription = $event"
  @update:selected-tags="selectedTags = $event"
  @close="showUploadModal = false"
  @open-cloudinary-widget="openCloudinaryWidget"
  @toggle-tag="toggleTag"
/>

<VideoStatsModal
  v-if="showStatsModal && selectedVideoStats"
  :video="selectedVideoStats"
  :format-date="formatDate"
  :get-video-chart-data="getVideoChartData"
  :get-detailed-video-stats="getDetailedVideoStats"
  @close="showStatsModal = false"
/>

<AuthModals
  :show-login="showLoginModal"
  :show-signup="showSignupModal"
  :email="email"
```

```
        :password="password"
        :name="name"
        :error="error"
        :loading="loading"
        @update:email="email = $event"
        @update:password="password = $event"
        @update:name="name = $event"
        @close-login="showLoginModal = false"
        @close-signup="showSignupModal = false"
        @login="handleLogin"
        @signup="handleSignup"
      />

       <InterestsSurvey
        :user="user"
        :user-interests="interests"
        :is-dark-theme="isDarkTheme"
        :force-show="showSurveyForced"
        @save-interests="saveInterestsFromSurvey"
      />

    </main>
  </div>
</template>
```

AnalyticsPage.vue

```
<script setup>
import { ref, computed } from 'vue';

const props = defineProps({
  sortedUsers: Array,
  sortedTagsByViews: Array,
  sortedTagsByLikes: Array,
  sortedTagsByComments: Array,
  sortedTagsByEngagement: Array,
  isLoadingAnalytics: Boolean,
  formatDate: Function
});

const emit = defineEmits(['go-to-user-profile']);

const activeTab = ref('users');
const usersSortBy = ref('subscribers');
const tagsSortBy = ref('views');

const searchUserQuery = ref('');
const filteredUsers = computed(() => {
  let users = [...props.sortedUsers];

  if (searchUserQuery.value) {
    const query = searchUserQuery.value.toLowerCase();
    users = users.filter(user =>
      user.name.toLowerCase().includes(query) ||
      user.email.toLowerCase().includes(query) ||
      user.interests.some(interest => interest.toLowerCase().includes(query))
    );
  }

  // Sorting
  if (usersSortBy.value === 'videos') {
    return users.sort((a, b) => b.videosCount - a.videosCount);
  } else if (usersSortBy.value === 'views') {
    return users.sort((a, b) => b.totalViews - a.totalViews);
  } else if (usersSortBy.value === 'likes') {
    return users.sort((a, b) => b.totalLikes - a.totalLikes);
  } else if (usersSortBy.value === 'subscribers') {
```

```
    return users.sort((a, b) => b.subscribers - a.subscribers);
  } else if (usersSortBy.value === 'subscriptions') {
    return users.sort((a, b) => b.subscriptions.length - a.subscriptions.length);
  }

  return users;
});

// Filters
const searchTagQuery = ref('');
const filteredTags = computed(() => {
  let tags;

  if (tagsSortBy.value === 'views') {
    tags = [...props.sortedTagsByViews];
  } else if (tagsSortBy.value === 'likes') {
    tags = [...props.sortedTagsByLikes];
  } else if (tagsSortBy.value === 'comments') {
    tags = [...props.sortedTagsByComments];
  } else if (tagsSortBy.value === 'engagement') {
    tags = [...props.sortedTagsByEngagement];
  } else {
    tags = [...props.sortedTagsByViews];
  }

  if (searchTagQuery.value) {
    const query = searchTagQuery.value.toLowerCase();
    tags = tags.filter(tag =>
      tag.name.toLowerCase().includes(query)
    );
  }

  return tags;
});

const handleUserClick = (userId) => {
  emit('go-to-user-profile', userId);
};
</script>

<template>
  <div class="analytics-page">
    <h1> Аналітика платформи</h1>

    <!-- Навигация по вкладкам -->
    <div class="analytics-tabs">
      <button
        class="tab-btn"
        :class="{ active: activeTab === 'users' }"
        @click="activeTab = 'users'"
      >
         Користувачі
      </button>
      <button
        class="tab-btn"
        :class="{ active: activeTab === 'tags' }"
        @click="activeTab = 'tags'"
      >
         Теги
      </button>
      <button
        class="tab-btn"
        :class="{ active: activeTab === 'stats' }"
        @click="activeTab = 'stats'"
      >
         Загальна статистика
      </button>
    </div>

    <!-- Загрузка -->
    <div v-if="isLoadingAnalytics" class="loading-analytics">
      <div class="loading-spinner"></div>
```

```
  <p>Завантаження аналітики...</p>
</div>

<!-- Вкладка пользователей -->
<div v-else-if="activeTab === 'users'" class="users-analytics">
  <div class="analytics-header">
    <h2>📊 Аналітика користувачів</h2>
    <div class="analytics-controls">
      <div class="search-bar">
        <svg width="20" height="20" viewBox="0 0 24 24" fill="none" stroke="currentColor" stroke-width="2">
          <circle cx="11" cy="11" r="8"></circle>
          <path d="m21 21-4.35-4.35"></path>
        </svg>
        <input
          v-model="searchUserQuery"
          type="text"
          placeholder="Пошук користувачів..."
          class="search-input"
        />
      </div>
      <select v-model="usersSortBy" class="filter-select">
        <option value="subscribers">За підписниками</option>
        <option value="videos">За кількістю відео</option>
        <option value="views">За переглядами</option>
        <option value="likes">За лайками</option>
        <option value="subscriptions">За підписками</option>
      </select>
    </div>
  </div>

  <div class="users-table-container">
    <table class="users-table">
      <thead>
        <tr>
          <th>Користувач</th>
          <th>Підписники</th>
          <th>Підписки</th>
          <th>Відео</th>
          <th>Перегляди</th>
          <th>Лайки</th>
          <th>Інтереси</th>
          <th>Дата реєстрації</th>
        </tr>
      </thead>
      <tbody>
        <tr
          v-for="user in filteredUsers"
          :key="user.id"
          class="user-row"
          @click="handleUserClick(user.id)"
        >
          <td class="user-info">
            <div class="user-avatar">
              <img
                v-if="user.avatar"
                :src="user.avatar"
                :alt="user.name"
                class="avatar-img-table"
              />
              <svg
                v-else
                width="32"
                height="32"
                viewBox="0 0 24 24"
                fill="none"
                stroke="currentColor"
                stroke-width="2"
              >
                <path d="M20 21v-2a4 4 0 0 0-4-4H8a4 4 0 0 0-4 4v2"></path>
                <circle cx="12" cy="7" r="4"></circle>
              </svg>
            </div>
```

```
            <div class="user-details">
              <strong>{{ user.name }}</strong>
              <small>{{ user.email }}</small>
            </div>
          </td>
          <td class="stat-cell">
            <span class="stat-badge subscribers">{{ user.subscribers }}</span>
          </td>
          <td class="stat-cell">
            <span class="stat-badge subscriptions">{{ user.subscriptions.length }}</span>
          </td>
          <td class="stat-cell">
            <span class="stat-badge videos">{{ user.videosCount }}</span>
          </td>
          <td class="stat-cell">
            <span class="stat-number">{{ user.totalViews.toLocaleString() }}</span>
          </td>
          <td class="stat-cell">
            <span class="stat-number">{{ user.totalLikes.toLocaleString() }}</span>
          </td>
          <td class="interests-cell">
            <div class="interests-list">
              <span
                v-for="interest in user.interests.slice(0, 3)"
                :key="interest"
                class="interest-tag-small"
              >
                {{ interest }}
              </span>
              <span v-if="user.interests.length > 3" class="more-tags">
                +{{ user.interests.length - 3 }}
              </span>
            </div>
          </td>
          <td class="date-cell">
            {{ formatDate(user.joinedDate) }}
          </td>
        </tr>
      </tbody>
    </table>

    <div v-if="filteredUsers.length === 0" class="empty-state">
      <p>Користувачів не знайдено</p>
    </div>
  </div>

  <div class="users-stats-summary">
    <div class="stat-card">
      <div class="stat-icon">□</div>
      <div class="stat-info">
        <div class="stat-value">{{ sortedUsers.length }}</div>
        <div class="stat-label">Користувачів</div>
      </div>
    </div>
    <div class="stat-card">
      <div class="stat-icon">□</div>
      <div class="stat-info">
        <div class="stat-value">{{ sortedUsers.reduce((sum, user) => sum + user.subscribers, 0) }}</div>
        <div class="stat-label">Всього підписників</div>
      </div>
    </div>
    <div class="stat-card">
      <div class="stat-icon">□</div>
      <div class="stat-info">
        <div class="stat-value">{{ sortedUsers.reduce((sum, user) => sum + user.videosCount, 0) }}</div>
        <div class="stat-label">Всього відео</div>
      </div>
    </div>
    <div class="stat-card">
      <div class="stat-icon">□</div>
      <div class="stat-info">
        <div class="stat-value">{{ sortedUsers.reduce((sum, user) => sum + user.totalViews, 0).toLocaleString() }}</div>
```

```
        <div class="stat-label">Всього переглядів</div>
      </div>
    </div>
  </div>
</div>

<!-- Вкладка тегов -->
<div v-else-if="activeTab === 'tags'" class="tags-analytics">
  <div class="analytics-header">
    <h2> Аналітика тегів</h2>
    <div class="analytics-controls">
      <div class="search-bar">
        <svg width="20" height="20" viewBox="0 0 24 24" fill="none" stroke="currentColor" stroke-width="2">
          <circle cx="11" cy="11" r="8"></circle>
          <path d="m21 21-4.35-4.35"></path>
        </svg>
        <input
          v-model="searchTagQuery"
          type="text"
          placeholder="Пошук тегів..."
          class="search-input"
        />
      </div>
      <select v-model="tagsSortBy" class="filter-select">
        <option value="views">За переглядами</option>
        <option value="likes">За лайками</option>
        <option value="comments">За коментарями</option>
        <option value="engagement">За engagement rate</option>
      </select>
    </div>
  </div>

  <div class="tags-table-container">
    <table class="tags-table">
      <thead>
        <tr>
          <th>Тег</th>
          <th>Відео</th>
          <th>Автори</th>
          <th>Перегляди</th>
          <th>Лайки</th>
          <th>Дизлайки</th>
          <th>Коментарі</th>
          <th v-if="tagsSortBy === 'engagement'">Engagement Rate</th>
        </tr>
      </thead>
      <tbody>
        <tr
          v-for="tag in filteredTags"
          :key="tag.name"
          class="tag-row"
        >
          <td class="tag-name">
            <span class="tag-badge">{{ tag.name }}</span>
          </td>
          <td class="stat-cell">
            <span class="stat-number">{{ tag.videosCount }}</span>
          </td>
          <td class="stat-cell">
            <span class="stat-number">{{ tag.uniqueAuthorsCount }}</span>
          </td>
          <td class="stat-cell">
            <span class="stat-number">{{ tag.totalViews.toLocaleString() }}</span>
          </td>
          <td class="stat-cell">
            <span class="stat-number">{{ tag.totalLikes.toLocaleString() }}</span>
          </td>
          <td class="stat-cell">
            <span class="stat-number">{{ tag.totalDislikes.toLocaleString() }}</span>
          </td>
          <td class="stat-cell">
            <span class="stat-number">{{ tag.totalComments.toLocaleString() }}</span>
```

```
          </td>
          <td v-if="tagsSortBy === 'engagement'" class="stat-cell">
            <span class="stat-number">{{ tag.engagementRate }}%</span>
          </td>
        </tr>
      </tbody>
    </table>

    <div v-if="filteredTags.length === 0" class="empty-state">
      <p>Тегів не знайдено</p>
    </div>
  </div>

  <!-- Графики тегов -->
  <div class="tags-charts">
    <div class="chart-section">
      <h3>Топ 10 тегів за переглядами</h3>
      <div class="chart-container">
        <div class="bar-chart">
          <div
            v-for="tag in sortedTagsByViews.slice(0, 10)"
            :key="tag.name"
            class="bar-chart-item"
          >
            <div class="bar-label">{{ tag.name }}</div>
            <div class="bar-container">
              <div
                class="bar-fill"
                :style="{
                  width: (tag.totalViews / sortedTagsByViews[0].totalViews * 100) + '%',
                  background: getTagColor(tag.name)
                }"
              >
                <span class="bar-value">{{ tag.totalViews.toLocaleString() }}</span>
              </div>
            </div>
          </div>
        </div>
      </div>
    </div>

    <div class="chart-section">
      <h3>Топ 10 тегів за лайками</h3>
      <div class="chart-container">
        <div class="bar-chart">
          <div
            v-for="tag in sortedTagsByLikes.slice(0, 10)"
            :key="tag.name"
            class="bar-chart-item"
          >
            <div class="bar-label">{{ tag.name }}</div>
            <div class="bar-container">
              <div
                class="bar-fill"
                :style="{
                  width: (tag.totalLikes / sortedTagsByLikes[0].totalLikes * 100) + '%',
                  background: getTagColor(tag.name)
                }"
              >
                <span class="bar-value">{{ tag.totalLikes.toLocaleString() }}</span>
              </div>
            </div>
          </div>
        </div>
      </div>
    </div>
  </div>
</div>

<!-- Вкладка общей статистики -->
<div v-else-if="activeTab === 'stats'" class="general-stats">
  <h2>📊 Загальна статистика платформи</h2>
```

```
      <div class="stats-grid">
        <div class="stat-card-large">
          <div class="stat-icon-large">□</div>
          <div class="stat-info-large">
            <div class="stat-value-large">{{ sortedUsers.length }}</div>
            <div class="stat-label-large">Активних користувачів</div>
          </div>
        </div>

        <div class="stat-card-large">
          <div class="stat-icon-large">□</div>
          <div class="stat-info-large">
            <div class="stat-value-large">{{ sortedUsers.reduce((sum, user) => sum + user.videosCount, 0) }}</div>
            <div class="stat-label-large">Відео на платформі</div>
          </div>
        </div>

        <div class="stat-card-large">
          <div class="stat-icon-large">□</div>
          <div class="stat-info-large">
            <div class="stat-value-large">{{ sortedUsers.reduce((sum, user) => sum + user.totalViews, 0).toLocaleString()
}}</div>
            <div class="stat-label-large">Всього переглядів</div>
          </div>
        </div>

        <div class="stat-card-large">
          <div class="stat-icon-large">□</div>
          <div class="stat-info-large">
            <div class="stat-value-large">{{ sortedUsers.reduce((sum, user) => sum + user.totalLikes, 0).toLocaleString()
}}</div>
            <div class="stat-label-large">Всього лайків</div>
          </div>
        </div>
      </div>

      <div class="stats-details">
        <div class="stats-section">
          <h3>Найпопулярніші теги</h3>
          <div class="top-tags-cloud">
            <span
              v-for="tag in sortedTagsByViews.slice(0, 15)"
              :key="tag.name"
              class="tag-cloud-item"
              :style="{
                fontSize: (0.8 + (tag.totalViews / sortedTagsByViews[0].totalViews * 1.5)) + 'rem',
                opacity: 0.7 + (tag.totalViews / sortedTagsByViews[0].totalViews * 0.3)
              }"
            >
              {{ tag.name }}
            </span>
          </div>
        </div>

        <div class="stats-section">
          <h3>Топ користувачі за підписниками</h3>
          <div class="top-users-list">
            <div
              v-for="(user, index) in sortedUsers.slice(0, 5)"
              :key="user.id"
              class="top-user-card"
              @click="handleUserClick(user.id)"
            >
              <div class="top-user-rank">#{{ index + 1 }}</div>
              <div class="top-user-avatar">
                <img
                  v-if="user.avatar"
                  :src="user.avatar"
                  :alt="user.name"
                  class="avatar-img-small"
                />
```

```
              <svg
                v-else
                width="40"
                height="40"
                viewBox="0 0 24 24"
                fill="none"
                stroke="currentColor"
                stroke-width="2"
              >
                <path d="M20 21v-2a4 4 0 0 0-4-4H8a4 4 0 0 0-4 4v2"></path>
                <circle cx="12" cy="7" r="4"></circle>
              </svg>
            </div>
            <div class="top-user-info">
              <strong>{{ user.name }}</strong>
              <div class="top-user-stats">
                <span> {{ user.subscribers }} підписників</span>
                <span> {{ user.videosCount }} відео</span>
                <span> {{ user.totalViews.toLocaleString() }} переглядів</span>
              </div>
            </div>
          </div>
        </div>
      </div>
    </div>
  </div>
</template>

<script>
const getTagColor = (tagName) => {
  const colors = [
    '#FF6B6B', '#4ECDC4', '#FFD166', '#06D6A0', '#118AB2',
    '#EF476F', '#FF9A76', '#7D5FFF', '#70D6FF', '#FF9770',
    '#E9FF70', '#FF70A6', '#70FFB3', '#9770FF', '#FFD670'
  ];

  let hash = 0;
  for (let i = 0; i < tagName.length; i++) {
    hash = tagName.charCodeAt(i) + ((hash << 5) - hash);
  }

  return colors[Math.abs(hash) % colors.length];
};

export { getTagColor };
</script>
```

AuthModals.vue

```
<script setup>
const props = defineProps({
  showLogin: Boolean,
  showSignup: Boolean,
  email: String,
  password: String,
  name: String,
  error: String,
  loading: Boolean
});

const emit = defineEmits([
  'update:email',
  'update:password',
  'update:name',
  'close-login',
  'close-signup',
  'login',
  'signup'
]);

const handleLoginSubmit = (e) => {
  e.preventDefault();
  emit('login');
};

const handleSignupSubmit = (e) => {
  e.preventDefault();
  emit('signup');
};
</script>

<template>
  <!-- Login Modal -->
  <div v-if="showLogin" class="modal-overlay" @click="emit('close-login')">
    <div class="modal" @click.stop>
      <h2>Log In</h2>
      <form @submit.prevent="handleLoginSubmit">
        <div class="form-group">
          <label>Email</label>
          <input
            :value="email"
            @input="emit('update:email', $event.target.value)"
            type="email"
            required
            placeholder="your@email.com"
          />
        </div>
        <div class="form-group">
          <label>Password</label>
          <input
            :value="password"
            @input="emit('update:password', $event.target.value)"
            type="password"
            required
            placeholder="••••••••"
            minlength="6"
          />
        </div>
        <p v-if="error" class="error">{{ error }}</p>
        <div class="modal-buttons">
          <button type="button" class="btn btn-secondary" @click="emit('close-login')" :disabled="loading">Cancel</button>
          <button type="submit" class="btn btn-primary" :disabled="loading">{{ loading ? 'Loading...' : 'Log In' }}</button>
        </div>
      </form>
    </div>
  </div>

  <!-- Signup Modal -->
  <div v-if="showSignup" class="modal-overlay" @click="emit('close-signup')">
    <div class="modal" @click.stop>
      <h2>Create Account</h2>
```

```
      <form @submit.prevent="handleSignupSubmit">
        <div class="form-group">
          <label>Name</label>
          <input
            :value="name"
            @input="emit('update:name', $event.target.value)"
            type="text"
            required
            placeholder="Your name"
          />
        </div>
        <div class="form-group">
          <label>Email</label>
          <input
            :value="email"
            @input="emit('update:email', $event.target.value)"
            type="email"
            required
            placeholder="your@email.com"
          />
        </div>
        <div class="form-group">
          <label>Password</label>
          <input
            :value="password"
            @input="emit('update:password', $event.target.value)"
            type="password"
            required
            placeholder="••••••••"
            minlength="6"
          />
        </div>
        <p v-if="error" class="error">{{ error }}</p>
        <div class="modal-buttons">
          <button type="button" class="btn btn-secondary" @click="emit('close-signup')" :disabled="loading">Cancel</button>
          <button type="submit" class="btn btn-primary" :disabled="loading">{{ loading ? 'Loading...' : 'Create Account'
}}</button>
        </div>
      </form>
    </div>
  </div>
</template>
```

DashboardPage.vue

```
<script setup>
const props = defineProps({
  user: Object,
  videos: Array,
  authorStats: Object,
  formatDate: Function,
  getVideoChartData: Function
});

const emit = defineEmits(['show-video-stats', 'delete-video', 'watch-video']);

const userVideos = computed(() => {
  return props.videos.filter(v => v.uploadedBy === props.user?.uid);
});

const handleVideoClick = (video) => {
  emit('watch-video', video);
};

const handleStatsClick = (video, e) => {
  e.stopPropagation();
  emit('show-video-stats', video);
};
</script>

<template>
  <div class="dashboard-page">
    <h1>Dashboard</h1>
```

```
<div v-if="!user" class="empty-state">
  <p>Увійдіть щоб переглянути Dashboard</p>
</div>

<template v-else>
  <div class="stats-overview">
    <div class="stat-card">
      <div class="stat-icon">□</div>
      <div class="stat-info">
        <div class="stat-value">{{ authorStats.totalVideos }}</div>
        <div class="stat-label">Відео</div>
      </div>
    </div>

    <div class="stat-card">
      <div class="stat-icon">□</div>
      <div class="stat-info">
        <div class="stat-value">{{ authorStats.totalViews.toLocaleString() }}</div>
        <div class="stat-label">Переглядів</div>
      </div>
    </div>

    <div class="stat-card">
      <div class="stat-icon">□</div>
      <div class="stat-info">
        <div class="stat-value">{{ authorStats.totalLikes }}</div>
        <div class="stat-label">Лайків</div>
      </div>
    </div>

    <div class="stat-card">
      <div class="stat-icon">□</div>
      <div class="stat-info">
        <div class="stat-value">{{ authorStats.likeRatio }}%</div>
        <div class="stat-label">Like Ratio</div>
      </div>
    </div>

    <div class="stat-card">
      <div class="stat-icon">□</div>
      <div class="stat-info">
        <div class="stat-value">{{ authorStats.avgViews }}</div>
        <div class="stat-label">Середньо переглядів</div>
      </div>
    </div>

    <div class="stat-card">
      <div class="stat-icon">⚡</div>
      <div class="stat-info">
        <div class="stat-value">{{ authorStats.engagementRate }}%</div>
        <div class="stat-label">Engagement Rate</div>
      </div>
    </div>

    <div class="stat-card">
      <div class="stat-icon">□</div>
      <div class="stat-info">
        <div class="stat-value">{{ authorStats.viralityScore }}%</div>
        <div class="stat-label">Вірусність</div>
      </div>
    </div>

    <div class="stat-card">
      <div class="stat-icon">□</div>
      <div class="stat-info">
        <div class="stat-value">{{ authorStats.avgTimeToFirstLike }}г</div>
        <div class="stat-label">До першого лайку</div>
      </div>
    </div>
  </div>

  <div class="dashboard-section">
    <h2>Динаміка за останні 30 днів</h2>
```

```
  <div class="chart-container">
    <div class="chart-title">Перегляди по днях</div>
    <div class="line-chart">
      <div
        v-for="(day, index) in authorStats.last30Days"
        :key="index"
        class="line-chart-bar"
        :style="{ height: (day.views / Math.max(...authorStats.last30Days.map(d => d.views || 1)) * 100) + '%' }"
        :title="`${day.dateLabel}: ${day.views} переглядів`"
      >
      </div>
    </div>
    <div class="chart-labels">
      <span>30д</span>
      <span>15д</span>
      <span>Сьогодні</span>
    </div>
  </div>
</div>

<div class="dashboard-section" v-if="authorStats.topTags.length > 0">
  <h2>Топ теги по переглядам</h2>
  <div class="top-tags-list">
    <div v-for="([tag, stats], index) in authorStats.topTags" :key="tag" class="top-tag-item">
      <div class="top-tag-rank">#{{ index + 1 }}</div>
      <div class="top-tag-info">
        <div class="top-tag-name">{{ tag }}</div>
        <div class="top-tag-stats">
          {{ stats.count }} відео • {{ stats.views }} переглядів • {{ stats.likes }} лайків
        </div>
      </div>
      <div class="top-tag-bar" :style="{ width: (stats.views / authorStats.topTags[0][1].views * 100) + '%' }"></div>
    </div>
  </div>
</div>

<div class="dashboard-section" v-if="authorStats.topVideo">
  <h2>Крайності</h2>
  <div class="extremes-grid">
    <div class="extreme-card top">
      <h3> Найкраще відео</h3>
      <div class="extreme-video" @click="handleVideoClick(authorStats.topVideo)">
        <video :src="authorStats.topVideo.videoUrl"></video>
        <div class="extreme-info">
          <h4>{{ authorStats.topVideo.title }}</h4>
          <p> {{ authorStats.topVideo.views }} переглядів</p>
          <p> {{ authorStats.topVideo.likes }} лайків</p>
        </div>
      </div>
    </div>

    <div class="extreme-card worst">
      <h3> Найменше переглядів</h3>
      <div class="extreme-video" @click="handleVideoClick(authorStats.worstVideo)">
        <video :src="authorStats.worstVideo.videoUrl"></video>
        <div class="extreme-info">
          <h4>{{ authorStats.worstVideo.title }}</h4>
          <p> {{ authorStats.worstVideo.views }} переглядів</p>
          <p> {{ authorStats.worstVideo.likes }} лайків</p>
        </div>
      </div>
    </div>
  </div>
</div>

<div class="dashboard-section">
  <h2>Мої відео</h2>

  <div v-if="userVideos.length === 0" class="empty-state">
    <p>Ви ще не завантажили жодного відео</p>
  </div>

  <div v-else class="videos-table">
    <div class="video-row video-row-header">
```

```
            <div class="video-row-thumbnail">Відео</div>
            <div class="video-row-title">Назва</div>
            <div class="video-row-stat">Перегляди</div>
            <div class="video-row-stat">Лайки</div>
            <div class="video-row-stat">Дизлайки</div>
            <div class="video-row-stat">Дата</div>
            <div class="video-row-actions">Дії</div>
          </div>

          <div
            v-for="video in             userVideos"
            :key="video.id"
            class="video-row"
            @click="handleVideoClick(video)"
          >
            <div class="video-row-thumbnail">
              <video :src="video.videoUrl"></video>
            </div>
            <div class="video-row-title">
              <strong>{{ video.title }}</strong>
              <div class="video-tags-small">
                <span v-for="tag in video.tags.slice(0, 3)" :key="tag" class="tag-small">{{ tag }}</span>
              </div>
            </div>
            <div class="video-row-stat">{{ video.views || 0 }}</div>
            <div class="video-row-stat">{{ video.likes || 0 }}</div>
            <div class="video-row-stat">{{ video.dislikes || 0 }}</div>
            <div class="video-row-stat">{{ formatDate(video.createdAt) }}</div>
            <div class="video-row-actions">
              <button class="btn-stats" @click="handleStatsClick(video, $event)" title="Статистика">
                
              </button>
              <button class="btn-delete-small" @click="emit('delete-video', video.id)" title="Видалити">
                
              </button>
            </div>
          </div>
        </div>
      </div>
    </template>
  </div>
</template>

<script>
import { computed } from 'vue';
</script>
```

Header.vue

```
<script setup>
defineProps({
  user: Object,
  userName: String,
  isDarkTheme: Boolean
});

defineEmits([
  'toggle-theme',
  'open-upload-modal',
  'open-login-modal',
  'open-signup-modal',
  'logout',
  'go-to-profile',
  'go-to-dashboard',
  'go-to-analytics',
  'go-home'
]);
</script>

<template>
  <header class="header">
    <div class="container">
      <div class="logo" @click="$emit('go-home')" style="cursor: pointer;">
```

```
        <img src="./assets/logo.svg" alt="Logo">
      </div>

      <nav class="nav-buttons">
        <button class="btn btn-theme" @click="$emit('toggle-theme')" title="Перемкнути тему">
          <svg xmlns="http://www.w3.org/2000/svg"  viewBox="0 0 50 50" width="50px" height="50px"><path d="M 24.90625 3.96875
C 24.863281 3.976563 24.820313 3.988281 24.78125 4 C 24.316406 4.105469 23.988281 4.523438 24 5 L 24 11 C 23.996094 11.359375
24.183594 11.695313 24.496094 11.878906 C 24.808594 12.058594 25.191406 12.058594 25.503906 11.878906 C 25.816406 11.695313
26.003906 11.359375 26 11 L 26 5 C 26.011719 4.710938 25.894531 4.433594 25.6875 4.238281 C 25.476563 4.039063 25.191406
3.941406 24.90625 3.96875 Z M 10.65625 9.84375 C 10.28125 9.910156 9.980469 10.183594 9.875 10.546875 C 9.769531 10.914063
9.878906 11.304688 10.15625 11.5625 L 14.40625 15.8125 C 14.648438 16.109375 15.035156 16.246094 15.410156 16.160156 C
15.78125 16.074219 16.074219 15.78125 16.160156 15.410156 C 16.246094 15.035156 16.109375 14.648438 15.8125 14.40625 L
11.5625 10.15625 C 11.355469 9.933594 11.054688 9.820313 10.75 9.84375 C 10.71875 9.84375 10.6875 9.84375 10.65625 9.84375 Z
M 39.03125 9.84375 C 38.804688 9.875 38.59375 9.988281 38.4375 10.15625 L 34.1875 14.40625 C 33.890625 14.648438 33.753906
15.035156 33.839844 15.410156 C 33.925781 15.78125 34.21875 16.074219 34.589844 16.160156 C 34.964844 16.246094 35.351563
16.109375 35.59375 15.8125 L 39.84375 11.5625 C 40.15625 11.265625 40.246094 10.800781 40.0625 10.410156 C 39.875 10.015625
39.460938 9.789063 39.03125 9.84375 Z M 24.90625 15 C 24.875 15.007813 24.84375 15.019531 24.8125 15.03125 C 24.75 15.035156
24.6875 15.046875 24.625 15.0625 C 24.613281 15.074219 24.605469 15.082031 24.59375 15.09375 C 19.289063 15.320313 15
19.640625 15 25 C 15 30.503906 19.496094 35 25 35 C 30.503906 35 35 30.503906 35 25 C 35 19.660156 30.746094 15.355469
25.46875 15.09375 C 25.433594 15.09375 25.410156 15.0625 25.375 15.0625 C 25.273438 15.023438 25.167969 15.003906 25.0625 15
C 25.042969 15 25.019531 15 25 15 C 24.96875 15 24.9375 15 24.90625 15 Z M 24.9375 17 C 24.957031 17 24.980469 17 25 17 C
25.03125 17 25.0625 17 25.09375 17 C 29.46875 17.050781 33 20.613281 33 25 C 33 29.421875 29.421875 33 25 33 C 20.582031 33
17 29.421875 17 25 C 17 20.601563 20.546875 17.035156 24.9375 17 Z M 4.71875 24 C 4.167969 24.078125 3.78125 24.589844
3.859375 25.140625 C 3.9375 25.691406 4.449219 26.078125 5 26 L 11 26 C 11.359375 26.003906 11.695313 25.816406 11.878906
25.503906 C 12.058594 25.191406 12.058594 24.808594 11.878906 24.496094 C 11.695313 24.183594 11.359375 23.996094 11 24 L 5
24 C 4.96875 24 4.9375 24 4.90625 24 C 4.875 24 4.84375 24 4.8125 24 C 4.78125 24 4.75 24 4.71875 24 Z M 38.71875 24 C
38.167969 24.078125 37.78125 24.589844 37.859375 25.140625 C 37.9375 25.691406 38.449219 26.078125 39 26 L 45 26 C 45.359375
26.003906 45.695313 25.816406 45.878906 25.503906 C 46.058594 25.191406 46.058594 24.808594 45.878906 24.496094 C 45.695313
24.183594 45.359375 23.996094 45 24 L 39 24 C 38.96875 24 38.9375 24 38.90625 24 C 38.875 24 38.84375 24 38.8125 24 C
38.78125 24 38.75 24 38.71875 24 Z M 15 33.875 C 14.773438 33.90625 14.5625 34.019531 14.40625 34.1875 L 10.15625 38.4375 C
9.859375 38.679688 9.722656 39.066406 9.808594 39.441406 C 9.894531 39.8125 10.1875 40.105469 10.558594 40.191406 C 10.933594
40.277344 11.320313 40.140625 11.5625 39.84375 L 15.8125 35.59375 C 16.109375 35.308594 16.199219 34.867188 16.039063
34.488281 C 15.882813 34.109375 15.503906 33.867188 15.09375 33.875 C 15.0625 33.875 15.03125 33.875 15 33.875 Z M 34.6875
33.875 C 34.3125 33.941406 34.011719 34.214844 33.90625 34.578125 C 33.800781 34.945313 33.910156 35.335938 34.1875 35.59375
L 38.4375 39.84375 C 38.679688 40.140625 39.066406 40.277344 39.441406 40.191406 C 39.8125 40.105469 40.105469 39.8125
40.191406 39.441406 C 40.277344 39.066406 40.140625 38.679688 39.84375 38.4375 L 35.59375 34.1875 C 35.40625 33.988281
35.148438 33.878906 34.875 33.875 C 34.84375 33.875 34.8125 33.875 34.78125 33.875 C 34.75 33.875 34.71875 33.875 34.6875
33.875 Z M 24.90625 37.96875 C 24.863281 37.976563 24.820313 37.988281 24.78125 38 C 24.316406 38.105469 23.988281 38.523438
24 39 L 24 45 C 23.996094 45.359375 24.183594 45.695313 24.496094 45.878906 C 24.808594 46.058594 25.191406 46.058594
25.503906 45.878906 C 25.816406 45.695313 26.003906 45.359375 26 45 L 26 39 C 26.011719 38.710938 25.894531 38.433594 25.6875
38.238281 C 25.476563 38.039063 25.191406 37.941406 24.90625 37.96875 Z"/></svg>
        </button>

        <button v-if="user" class="btn btn-upload" @click="$emit('open-upload-modal')">
          <!-- Иконка -->
          Upload
        </button>

        <!-- ДОБАВЬТЕ ЭТУ КНОПКУ -->
        <button v-if="user" class="btn btn-analytics" @click="$emit('go-to-analytics')">
          <svg width="20" height="20" viewBox="0 0 24 24" fill="none" stroke="currentColor" stroke-width="2">
            <path d="M18 20V10"></path>
            <path d="M12 20V4"></path>
            <path d="M6 20v-6"></path>
          </svg>
          Analytics
        </button>

        <template v-if="!user">
          <button class="btn btn-secondary" @click="$emit('open-login-modal')">Log In</button>
          <button class="btn btn-primary" @click="$emit('open-signup-modal')">Create Account</button>
        </template>
        <template v-else>
          <button class="btn btn-profile" @click="$emit('go-to-profile')">
            <!-- Иконка -->
            Profile
          </button>
          <button class="btn btn-profile" @click="$emit('go-to-dashboard')">
            <!-- Иконка -->
            Dashboard
          </button>
          <button class="btn btn-secondary" @click="$emit('logout')">Log Out</button>
        </template>
      </nav>
```

```
    </div>
  </header>
</template>
```

HomePage.vue

```
<script setup>
import { computed, onMounted, onUnmounted } from 'vue';

import { ref } from 'vue';

const expandedScores = ref({});

const toggleScoreDetails = (videoId) => {
  expandedScores.value[videoId] = !expandedScores.value[videoId];
};

const showExplanation = ref(false);

const toggleExplanation = () => {
  showExplanation.value = !showExplanation.value;
};


const props = defineProps({
  videos: Array,
  allTags: Array,
  isLoadingVideos: Boolean,
  searchQuery: String,
  sortBy: String,
  filterTag: String,
  formatDate: Function,
  videoCache: Object,
  loadingThumbnails: Object,
  getThumbnailUrl: Function,
  getOptimizedVideoUrl: Function,
  preloadVideoOnHover: Function,
  user: Object,
  recommendations: Array,
  interests: Array
});

const emit = defineEmits([
  'update:searchQuery',
  'update:sortBy',
  'update:filterTag',
  'watch-video',
  'go-to-user-profile'
]);

let observer = null;

onMounted(() => {
  observer = new IntersectionObserver((entries) => {
    entries.forEach(entry => {
      if (entry.isIntersecting) {
        const videoId = entry.target.dataset.videoId;
        const video = props.videos.find(v => v.id === videoId);
        if (video) {
          props.preloadVideoOnHover(video);
        }
      }
    });
  }, {
    rootMargin: '50px 0px',
    threshold: 0.1
  });
});

onUnmounted(() => {
  if (observer) {
    observer.disconnect();
```

```
  }
});

const setupLazyLoad = (element, videoId) => {
  if (observer && element) {
    element.dataset.videoId = videoId;
    observer.observe(element);
  }
};

const handleVideoClick = (video) => {
  emit('watch-video', video);
};

const handleAuthorClick = (userId, e) => {
  e.stopPropagation();
  emit('go-to-user-profile', userId);
};

const handleVideoHover = (video) => {
  props.preloadVideoOnHover(video);
};

const handleImageError = (event) => {
  event.target.style.display = 'none';
  const placeholder = event.target.parentElement.querySelector('.thumbnail-placeholder');
  if (placeholder) {
    placeholder.style.display = 'flex';
  }
};
</script>

<template>
  <div class="home-page">
    <h1> Відео</h1>

    <div v-if="user && recommendations && recommendations.length > 0" class="recommendations-section">
      <div class="section-header">
        <h2> Рекомендовані вам</h2>
        <p class="section-subtitle" v-if="interests && interests.length > 0">
          На основі ваших інтересів:
          <span class="interests-list">
            <span v-for="(interest, index) in interests.slice(0, 5)" :key="interest" class="interest-badge">
              {{ interest }}
            </span>
            <span v-if="interests.length > 5" class="more-interests">+{{ interests.length - 5 }} ще</span>
          </span>
        </p>
        <p v-else class="section-subtitle">
          Лайкайте відео, щоб отримувати персоналізовані рекомендації!
        </p>
      </div>

 <div class="recommendation-container">
    <button
      @click="toggleExplanation"
      class="help-button"
      :aria-expanded="showExplanation"
      :title="showExplanation ? 'Приховати пояснення' : 'Як працюють рекомендації?'"
    >
      ?
    </button>

    <transition name="slide-fade">
      <div v-if="showExplanation" class="recommendation-explanation">
        <div class="explanation-header">
          <h4> Як це працює:</h4>
          <button @click="toggleExplanation" class="close-button" title="Закрити">
            <svg width="16" height="16" viewBox="0 0 24 24" fill="none" stroke="currentColor" stroke-width="2">
              <line x1="18" y1="6" x2="6" y2="18"></line>
              <line x1="6" y1="6" x2="18" y2="18"></line>
            </svg>
          </button>
        </div>
```

```
        <div class="formula">
          <div class="formula-item">
            <div class="formula-content">
              <span class="formula-part">Контекст</span>
              <span class="formula-weight">×35%</span>
            </div>
            <span class="formula-description">Співпадіння з твоїми інтересами</span>
          </div>

          <div class="formula-item">
            <div class="formula-content">
              <span class="formula-part">Соц. валідація</span>
              <span class="formula-weight">×25%</span>
            </div>
            <span class="formula-description">Лайки, перегляди та якість контенту</span>
          </div>

          <div class="formula-item">
            <div class="formula-content">
              <span class="formula-part">Актуальність</span>
              <span class="formula-weight">×20%</span>
            </div>
            <span class="formula-description">Свіжість відео та тренди</span>
          </div>

          <div class="formula-item">
            <div class="formula-content">
              <span class="formula-part">Персоналізація</span>
              <span class="formula-weight">×15%</span>
            </div>
            <span class="formula-description">Твої підписки та історія переглядів</span>
          </div>

          <div class="formula-item">
            <div class="formula-content">
              <span class="formula-part">Різноманітність</span>
              <span class="formula-weight">×5%</span>
            </div>
            <span class="formula-description">Запобігання зацикленню на одному контенті</span>
          </div>
        </div>

        <div class="formula-summary">
          <p><strong>Формула:</strong>
            <span class="formula-math">Релевантність × 35% + Валідація × 25% + Актуальність × 20% + Персоналізація × 15% +
Різноманітність × 5%</span>
          </p>
        </div>
      </div>
    </transition>
  </div>

<div class="recommended-videos">
  <div
    v-for="video in recommendations"
    :key="'rec-' + video.id"
    class="recommended-video-card"
  >
    <div class="recommended-thumbnail" @click="handleVideoClick(video)">
      <img
        v-if="getThumbnailUrl(video.videoUrl)"
        :src="getThumbnailUrl(video.videoUrl)"
        :alt="video.title"
        loading="lazy"
        class="recommended-thumbnail-img"
        @error="handleImageError"
      />
      <div v-else class="thumbnail-placeholder">
        <svg width="48" height="48" viewBox="0 0 24 24" fill="none" stroke="currentColor" stroke-width="1">
          <rect x="3" y="3" width="18" height="18" rx="2" ry="2"></rect>
          <circle cx="8.5" cy="8.5" r="1.5"></circle>
          <polyline points="21 15 16 10 5 21"></polyline>
        </svg>
```

```
  </div>
  <div class="recommended-score" :title="`Загальна оцінка: ${video.recommendationScore.toFixed(1)}/100`">
    {{ video.recommendationScore.toFixed(1) }}
  </div>
  <div class="recommended-match" v-if="video.commonTags && video.commonTags.length > 0">
    {{ video.commonTags.length }} співпадінь
  </div>
  <div v-if="video.isFromSubscription" class="subscription-badge" title="Від автора на якого ви підписані">
    ★
  </div>
</div>
<div class="recommended-info">
  <h4 @click="handleVideoClick(video)">{{ video.title }}</h4>
  <p class="recommended-meta">
    {{ video.uploaderName }} • {{ video.views }} переглядів • {{ formatDate(video.createdAt) }}
  </p>

  <div class="matching-tags" v-if="video.commonTags && video.commonTags.length > 0">
    <span v-for="tag in video.commonTags.slice(0, 3)" :key="tag" class="matching-tag">
      {{ tag }}
    </span>
    <span v-if="video.commonTags.length > 3" class="matching-tag-more">
      +{{ video.commonTags.length - 3 }}
    </span>
  </div>

  <div class="score-breakdown" v-if="video.scoreComponents">
    <div
      class="score-breakdown-header"
      @click.stop="toggleScoreDetails(video.id)"
      :class="{ expanded: expandedScores[video.id] }"
    >
      <span class="breakdown-title">
        <svg
          class="expand-icon"
          :class="{ rotated: expandedScores[video.id] }"
          width="12"
          height="12"
          viewBox="0 0 24 24"
          fill="none"
          stroke="currentColor"
          stroke-width="3"
        >
          <polyline points="9 18 15 12 9 6"></polyline>
        </svg>
        Деталі оцінки
      </span>
      <span class="breakdown-total">{{ video.recommendationScore.toFixed(1) }}/100</span>
    </div>

    <Transition name="score-expand">
      <div v-if="expandedScores[video.id]" class="score-breakdown-content">
        <div class="score-breakdown-bars">
          <div class="score-bar-wrapper">
            <div class="score-bar-label">
              <span class="bar-name">Контекст</span>
              <span class="bar-value">{{ video.scoreComponents.contextScore.toFixed(1) }}/35</span>
            </div>
            <div class="score-bar">
              <div
                class="score-bar-fill context"
                :style="{ width: (video.scoreComponents.contextScore / 35 * 100) + '%' }"
              ></div>
            </div>
          </div>

          <div class="score-bar-wrapper">
            <div class="score-bar-label">
              <span class="bar-name">Соц. валідація</span>
              <span class="bar-value">{{ video.scoreComponents.socialScore.toFixed(1) }}/25</span>
            </div>
            <div class="score-bar">
              <div
                class="score-bar-fill social"
```

```
        :style="{ width: (video.scoreComponents.socialScore / 25 * 100) + '%' }"
      ></div>
    </div>
  </div>

  <!-- Актуальність -->
  <div class="score-bar-wrapper">
    <div class="score-bar-label">
      <span class="bar-name">Актуальність</span>
      <span class="bar-value">{{ video.scoreComponents.temporalScore.toFixed(1) }}/20</span>
    </div>
    <div class="score-bar">
      <div
        class="score-bar-fill temporal"
        :style="{ width: (video.scoreComponents.temporalScore / 20 * 100) + '%' }"
      ></div>
    </div>
  </div>

  <!-- Персоналізація -->
  <div class="score-bar-wrapper">
    <div class="score-bar-label">
      <span class="bar-name">Персоналізація</span>
      <span class="bar-value">{{ video.scoreComponents.affinityScore.toFixed(1) }}/15</span>
    </div>
    <div class="score-bar">
      <div
        class="score-bar-fill affinity"
        :style="{ width: (video.scoreComponents.affinityScore / 15 * 100) + '%' }"
      ></div>
    </div>
  </div>

  <!-- Різноманітність -->
  <div class="score-bar-wrapper">
    <div class="score-bar-label">
      <span class="bar-name">Різноманітність</span>
      <span class="bar-value">{{ video.scoreComponents.diversityBonus.toFixed(1) }}/5</span>
    </div>
    <div class="score-bar">
      <div
        class="score-bar-fill diversity"
        :style="{ width: (video.scoreComponents.diversityBonus / 5 * 100) + '%' }"
      ></div>
    </div>
  </div>
</div>

<!-- Tip -->
<div class="score-explanation">
  <div class="explanation-icon">□</div>
  <div class="explanation-text">
    <strong>Чому це відео?</strong>
    <p>
      <span v-if="video.commonTags && video.commonTags.length > 0">
        Збіг {{ video.commonTags.length }} з {{ interests.length }} твоїх інтересів
      </span>
      <span v-if="video.isFromSubscription">
        • Від автора на якого ти підписаний
      </span>
      <span v-if="video.scoreComponents.temporalScore > 15">
        • Свіже відео ({{ formatDate(video.createdAt) }})
      </span>
      <span v-if="video.scoreComponents.socialScore > 20">
        • Високий рейтинг ({{ video.likes }} лайків)
      </span>
      <span v-if="video.scoreComponents.diversityBonus > 3">
        • Щось нове для тебе
      </span>
    </p>
    <div class="explanation-formula">
      {{ video.scoreComponents.contextScore.toFixed(1) }} (контекст) +
      {{ video.scoreComponents.socialScore.toFixed(1) }} (якість) +
      {{ video.scoreComponents.temporalScore.toFixed(1) }} (свіжість) +
```

```
                {{ video.scoreComponents.affinityScore.toFixed(1) }} (персональне) +
                {{ video.scoreComponents.diversityBonus.toFixed(1) }} (різноманітність) =
                <strong>{{ video.recommendationScore.toFixed(1) }}</strong>
              </div>
            </div>
          </div>
        </div>
      </Transition>
    </div>
  </div>
</div>
</div>

    <div class="search-filters">
      <div class="search-bar">
        <svg width="20" height="20" viewBox="0 0 24 24" fill="none" stroke="currentColor" stroke-width="2">
          <circle cx="11" cy="11" r="8"></circle>
          <path d="m21 21-4.35-4.35"></path>
        </svg>
        <input
          :value="searchQuery"
          @input="$emit('update:searchQuery', $event.target.value)"
          type="text"
          placeholder="Пошук відео..."
          class="search-input"
        />
      </div>

      <div class="filters">
        <select :value="sortBy" @change="$emit('update:sortBy', $event.target.value)" class="filter-select">
          <option value="newest">Найновіші</option>
          <option value="popular">Популярні</option>
          <option value="oldest">Найстаріші</option>
        </select>

        <select :value="filterTag" @change="$emit('update:filterTag', $event.target.value)" class="filter-select">
          <option value="all">Всі теги</option>
          <option v-for="tag in allTags" :key="tag" :value="tag">{{ tag }}</option>
        </select>
      </div>
    </div>

    <div v-if="isLoadingVideos" class="videos-grid">
      <div v-for="i in 6" :key="'skeleton-' + i" class="video-card skeleton-card">
        <div class="skeleton skeleton-thumbnail"></div>
        <div class="video-info">
          <div class="skeleton skeleton-title"></div>
          <div class="skeleton skeleton-meta"></div>
          <div class="skeleton skeleton-tags"></div>
        </div>
      </div>
    </div>

    <div v-else-if="videos.length === 0" class="empty-videos">
      <p>{{ searchQuery || filterTag !== 'all' ? 'Нічого не знайдено 🔍' : 'Поки немає відео. Завантажте перше! 📹' }}</p>
    </div>

    <div v-else class="videos-grid">
      <div
        v-for="video in videos"
        :key="video.id"
        class="video-card"
        @click="handleVideoClick(video)"
        @mouseenter="handleVideoHover(video)"
        :ref="el => setupLazyLoad(el, video.id)"
      >
        <div class="video-thumbnail">
          <div class="thumbnail-container">
            <img
              v-if="getThumbnailUrl(video.videoUrl)"
              :src="getThumbnailUrl(video.videoUrl)"
              :alt="video.title"
              loading="lazy"
```

```
              class="thumbnail-img"
              @load="videoCache[video.id] = true"
              @error="handleImageError"
            />
            <div v-else class="thumbnail-placeholder">
              <svg width="48" height="48" viewBox="0 0 24 24" fill="none" stroke="currentColor" stroke-width="1">
                <rect x="3" y="3" width="18" height="18" rx="2" ry="2"></rect>
                <circle cx="8.5" cy="8.5" r="1.5"></circle>
                <polyline points="21 15 16 10 5 21"></polyline>
              </svg>
            </div>
            <div v-if="loadingThumbnails[video.id] && !videoCache[video.id]" class="thumbnail-skeleton"></div>
          </div>
          <div class="video-duration">{{ video.tags.length }} тегів</div>
        </div>
        <div class="video-info">
          <h3>{{ video.title }}</h3>
          <p class="video-meta">
            <span class="author-name clickable" @click="handleAuthorClick(video.uploadedBy, $event)">
              {{ video.uploaderName }}
            </span>
            • {{ video.views }} переглядів • {{ formatDate(video.createdAt) }}
          </p>
          <div class="video-tags">
            <span v-for="tag in video.tags.slice(0, 3)" :key="tag" class="tag">{{ tag }}</span>
            <span v-if="video.tags.length > 3" class="tag-more">+{{ video.tags.length - 3 }}</span>
          </div>
        </div>
      </div>
    </div>
  </div>
</template>

<style scoped>
.home-page {
  padding: 1rem;
  max-width: 1400px;
  margin: 0 auto;
}

.home-page h1 {
  margin: 0 0 1.5rem;
  font-size: 2rem;
  color: var(--text-primary);
}

/* Стили для рекомендаций */
.recommendations-section {
  margin-bottom: 2rem;
  padding: 1.5rem;
  background: linear-gradient(135deg, #8e66ea 0%, #a24b85 100%);
  border-radius: 12px;
  color: white;
}

.section-header h2 {
  margin: 0 0 0.5rem;
  font-size: 1.5rem;
  color: white;
}

.section-subtitle {
  margin: 0 0 1rem;
  font-size: 0.95rem;
  opacity: 0.9;
}

.interests-list {
  display: inline-flex;
  flex-wrap: wrap;
  gap: 0.5rem;
  margin-left: 0.5rem;
}
```

```
.interest-badge {
  background: rgba(255, 255, 255, 0.2);
  padding: 0.25rem 0.75rem;
  border-radius: 20px;
  font-size: 0.85rem;
  font-weight: 500;
}

.more-interests {
  font-size: 0.85rem;
  opacity: 0.7;
}

/* Математическая модель */
.recommendation-explanation {
  margin: 1rem 0;
  padding: 1rem;
  background: rgba(255, 255, 255, 0.1);
  border-radius: 8px;

}

.recommendation-explanation h4 {
  margin: 0 0 0.75rem;
  font-size: 1rem;
}

.formula {
  display: flex;
  flex-wrap: wrap;
  gap: 1rem;
}

.formula-item {
  display: flex;
  flex-direction: column;
  align-items: center;
  gap: 0.25rem;
}

.formula-part {
  font-size: 0.9rem;
  font-weight: 500;
}

.formula-weight {
  font-size: 0.8rem;
  opacity: 0.8;

  margin-left: 20px;
}

/* Рекомендованные видео */
.recommended-videos {
  display: grid;
  grid-template-columns: repeat(auto-fill, minmax(280px, 1fr));
  gap: 1rem;
  margin-top: 1rem;
}

.recommended-video-card {
  background: white;
  border-radius: 8px;
  overflow: hidden;
  cursor: pointer;
  transition: transform 0.2s, box-shadow 0.2s;
  color: #333;
}

.recommended-video-card:hover {
  transform: translateY(-4px);
  box-shadow: 0 10px 20px rgba(0, 0, 0, 0.2);
}
```

```
.recommended-thumbnail {
  position: relative;
  height: 160px;
  overflow: hidden;
}

.recommended-thumbnail-img {
  width: 100%;
  height: 100%;
  object-fit: cover;
}

.recommended-thumbnail .thumbnail-placeholder {
  width: 100%;
  height: 100%;
  display: flex;
  align-items: center;
  justify-content: center;
  background: var(--bg-tertiary);
  color: var(--text-secondary);
}

.recommended-score {
  position: absolute;
  top: 10px;
  right: 10px;
  background: rgba(0, 0, 0, 0.7);
  color: white;
  padding: 0.25rem 0.5rem;
  border-radius: 4px;
  font-size: 0.8rem;
  font-weight: 600;
}

.recommended-match {
  position: absolute;
  bottom: 10px;
  left: 10px;
  background: rgba(102, 126, 234, 0.9);
  color: white;
  padding: 0.25rem 0.5rem;
  border-radius: 4px;
  font-size: 0.8rem;
}

.recommended-info {
  padding: 1rem;
}

.recommended-info h4 {
  margin: 0 0 0.5rem;
  font-size: 1rem;
  line-height: 1.3;
  display: -webkit-box;
  -webkit-line-clamp: 2;
  -webkit-box-orient: vertical;
  overflow: hidden;
}

.recommended-meta {
  margin: 0 0 0.75rem;
  font-size: 0.85rem;
  color: #666;
}

.matching-tags {
  display: flex;
  flex-wrap: wrap;
  gap: 0.25rem;
  margin-bottom: 0.75rem;
}

.matching-tag {
  background: #e9d5ff;
```

```
  color: #7c3aed;
  padding: 0.25rem 0.5rem;
  border-radius: 4px;
  font-size: 0.75rem;
  font-weight: 500;
}

/* Детали оценки */
.score-details {
  margin-top: 0.5rem;
}

.score-item {
  height: 4px;
  background: linear-gradient(90deg, #8e66ea, #a24b85);
  border-radius: 2px;
  margin-bottom: 0.25rem;
  position: relative;
  overflow: hidden;
}

.score-item span {
  position: absolute;
  left: 0;
  top: -20px;
  font-size: 0.7rem;
  color: #666;
  white-space: nowrap;
  opacity: 0;
  transition: opacity 0.2s;
}

.score-item:hover span {
  opacity: 1;
}

/* Поиск и фильтры */
.search-filters {
  margin-bottom: 2rem;
  display: flex;
  flex-direction: column;
  gap: 1rem;
}

.search-bar {
  display: flex;
  align-items: center;
  background: var(--bg-secondary);
  border-radius: 12px;
  padding: 0.75rem 1rem;
  border: 1px solid var(--border-color);
}

.search-bar svg {
  color: var(--text-secondary);
  margin-right: 0.75rem;
}

.search-input {
  flex: 1;
  background: none;
  border: none;
  color: var(--text-primary);
  font-size: 1rem;
  outline: none;
}

.search-input::placeholder {
  color: var(--text-secondary);
}

.filters {
  display: flex;
  gap: 1rem;
```

```
  flex-wrap: wrap;
}

.filter-select {
  padding: 0.75rem 1rem;
  background: var(--bg-secondary);
  border: 1px solid var(--border-color);
  border-radius: 12px;
  color: var(--text-primary);
  font-size: 0.95rem;
  cursor: pointer;
  outline: none;
  min-width: 150px;
}

.filter-select:hover {
  border-color: var(--primary-color);
}

/* Все видео */
.videos-grid {
  display: grid;
  grid-template-columns: repeat(auto-fill, minmax(300px, 1fr));
  gap: 1.5rem;
  margin-top: 1rem;
}

.video-card {
  background: var(--bg-primary);
  border-radius: 12px;
  overflow: hidden;
  cursor: pointer;
  transition: transform 0.2s, box-shadow 0.2s;
  border: 1px solid var(--border-color);
}

.video-card:hover {
  transform: translateY(-4px);
  box-shadow: 0 8px 24px rgba(0, 0, 0, 0.15);
}

.video-thumbnail {
  position: relative;
  height: 180px;
  overflow: hidden;
}

.thumbnail-container {
  width: 100%;
  height: 100%;
  position: relative;
}

.thumbnail-img {
  width: 100%;
  height: 100%;
  object-fit: cover;
  transition: transform 0.3s;
}

.video-card:hover .thumbnail-img {
  transform: scale(1.05);
}

.thumbnail-placeholder {
  width: 100%;
  height: 100%;
  display: flex;
  align-items: center;
  justify-content: center;
  background: var(--bg-tertiary);
  color: var(--text-secondary);
}
```

```
.thumbnail-skeleton {
  position: absolute;
  top: 0;
  left: 0;
  width: 100%;
  height: 100%;
  background: linear-gradient(90deg, var(--bg-tertiary) 25%, var(--bg-secondary) 50%, var(--bg-tertiary) 75%);
  background-size: 200% 100%;
  animation: loading 1.5s infinite;
}

@keyframes loading {
  0% { background-position: 200% 0; }
  100% { background-position: -200% 0; }
}

.video-duration {
  position: absolute;
  bottom: 10px;
  right: 10px;
  background: rgba(0, 0, 0, 0.8);
  color: white;
  padding: 0.25rem 0.5rem;
  border-radius: 4px;
  font-size: 0.75rem;
  font-weight: 600;
}

.video-info {
  padding: 1rem;
}

.video-info h3 {
  margin: 0 0 0.5rem;
  font-size: 1rem;
  line-height: 1.4;
  display: -webkit-box;
  -webkit-line-clamp: 2;
  -webkit-box-orient: vertical;
  overflow: hidden;
  color: var(--text-primary);
}

.video-meta {
  margin: 0 0 0.75rem;
  font-size: 0.875rem;
  color: var(--text-secondary);
  line-height: 1.4;
}

.author-name.clickable {
  color: var(--primary-color);
  cursor: pointer;
  text-decoration: none;
  transition: opacity 0.2s;
}

.author-name.clickable:hover {
  opacity: 0.8;
  text-decoration: underline;
}

.video-tags {
  display: flex;
  flex-wrap: wrap;
  gap: 0.375rem;
}

.tag {
  background: var(--bg-tertiary);
  color: var(--text-secondary);
  padding: 0.25rem 0.5rem;
  border-radius: 6px;
  font-size: 0.75rem;
```

```
  font-weight: 500;
}

.tag-more {
  background: var(--primary-color-light);
  color: var(--primary-color);
  padding: 0.25rem 0.5rem;
  border-radius: 6px;
  font-size: 0.75rem;
  font-weight: 500;
}

/* Скелетоны */
.skeleton-card {
  border: none;
}

.skeleton {
  background: var(--bg-tertiary);
  border-radius: 6px;
  animation: pulse 1.5s infinite;
}

.skeleton-thumbnail {
  height: 180px;
  width: 100%;
}

.skeleton-title {
  height: 1.25rem;
  width: 80%;
  margin-bottom: 0.5rem;
}

.skeleton-meta {
  height: 0.875rem;
  width: 60%;
  margin-bottom: 0.75rem;
}

.skeleton-tags {
  height: 1.5rem;
  width: 100%;
}

@keyframes pulse {
  0%, 100% { opacity: 1; }
  50% { opacity: 0.5; }
}

.empty-videos {
  padding: 4rem 2rem;
  text-align: center;
  color: var(--text-secondary);
  background: var(--bg-secondary);
  border-radius: 16px;
  margin-top: 2rem;
}

.empty-videos p {
  font-size: 1.1rem;
  margin: 0;
}

/* Адаптивность */
@media (max-width: 768px) {
  .recommended-videos {
    grid-template-columns: 1fr;
  }

  .formula {
    justify-content: center;
  }
```

```
  .recommendations-section {
    padding: 1rem;
  }

  .videos-grid {
    grid-template-columns: 1fr;
  }

  .home-page h1 {
    font-size: 1.75rem;
  }
}

.recommendation-container {
  position: relative;
  margin: 20px 0;
}

.help-button {
  position: absolute;
  top: -10px;
  left: 0;
  background: linear-gradient(135deg, #8e66ea 0%, #a24b85 100%);
  border: none;
  border-radius: 50%;
  width: 40px;
  height: 40px;
  color: white;
  cursor: pointer;
  display: flex;
  align-items: center;
  justify-content: center;
  transition: all 0.3s ease;
  box-shadow: 0 4px 15px rgba(255, 255, 255, 0.5);
  z-index: 10;
}

.help-button{
  font-weight: 700;

  font-size: 20px;
}

.help-button:hover {
  transform: scale(1.1) rotate(15deg);
  box-shadow: 0 6px 20px rgba(0, 0, 0, 0.3);
}

.help-button:focus {
  outline: 2px solid #8e66ea;
  outline-offset: 2px;
}

.recommendation-explanation {
  background: linear-gradient(135deg, #f5f7fa 0%, #e4edf5 100%);
  border-radius: 12px;
  padding: 20px;
  margin-top: 10px;
  border: 1px solid #e0e6ed;
  box-shadow: 0 8px 25px rgba(0, 0, 0, 0.1);
  position: relative;
  z-index: 5;

  padding-top: 32px;

}

.explanation-header {
  display: flex;
  justify-content: space-between;
  align-items: center;
  margin-bottom: 15px;
  padding-bottom: 10px;
  border-bottom: 2px solid #8e66ea;
```

```
}

.explanation-header h4 {
  margin: 0;
  color: #2d3748;
  font-size: 1.2rem;
  display: flex;
  align-items: center;
  gap: 8px;
}

.close-button {
  background: none;
  border: none;
  cursor: pointer;
  color: #718096;
  padding: 5px;
  border-radius: 4px;
  transition: all 0.2s ease;
}

.close-button:hover {
  background: #edf2f7;
  color: #2d3748;
  transform: rotate(90deg);
}

.formula {
  display: flex;
  flex-direction: column;
  gap: 12px;
}

.formula-item {
  display: flex;
  flex-direction: column;
  padding: 12px;
  background: white;
  border-radius: 8px;
  border-left: 4px solid;
  transition: transform 0.2s ease;
}

.formula-item:hover {
  transform: translateX(5px);
}

.formula-item:nth-child(1) { border-color: #8e66ea; }
.formula-item:nth-child(2) { border-color: #a24b85; }
.formula-item:nth-child(3) { border-color: #f093fb; }
.formula-item:nth-child(4) { border-color: #4fd1c7; }
.formula-item:nth-child(5) { border-color: #68d391; }

.formula-content {
  display: flex;
  justify-content: space-between;
  align-items: center;
  margin-bottom: 5px;
}

.formula-part {
  font-weight: 600;
  color: #2d3748;
  font-size: 1.1rem;
}

.formula-weight {
  background: linear-gradient(135deg, #8e66ea 0%, #a24b85 100%);
  color: white;
  padding: 4px 12px;
  border-radius: 20px;
  font-weight: 600;
  font-size: 0.9rem;
}
```

```
.formula-description {
  color: #718096;
  font-size: 0.9rem;
  line-height: 1.4;
}

.formula-summary {
  margin-top: 20px;
  padding: 15px;
  background: rgba(102, 126, 234, 0.1);
  border-radius: 8px;
  border-left: 4px solid #8e66ea;
}

.formula-summary p {
  margin: 0;
  color: #2d3748;
}

.formula-math {
  display: block;
  margin-top: 8px;
  padding: 10px;
  background: white;
  border-radius: 6px;
  font-family: 'Courier New', monospace;
  color: #a24b85;
  font-weight: 600;
}

/* Анимации */
.slide-fade-enter-active {
  transition: all 0.3s ease-out;
}

.slide-fade-leave-active {
  transition: all 0.2s cubic-bezier(1, 0.5, 0.8, 1);
}

.slide-fade-enter-from,
.slide-fade-leave-to {
  transform: translateY(-10px);
  opacity: 0;
}

/* Адаптивность */
@media (max-width: 768px) {
  .formula-item {
    padding: 10px;
  }

  .formula-part {
    font-size: 1rem;
  }

  .formula-weight {
    padding: 3px 10px;
    font-size: 0.85rem;
  }

  .help-button {
    top: -15px;
    right: 10px;
    width: 36px;
    height: 36px;
  }
}

</style>
```

InterestsSurvey.vue

```
<template>
  <Transition name="survey-slide">
    <div v-if="isVisible" :class="['interests-survey', { 'dark': isDarkTheme, 'minimized': isMinimized }]">
      <div class="survey-header">
        <div class="survey-title">
          <span class="survey-icon">✨</span>
          <span v-if="!isMinimized">Personalize your feed</span>
        </div>
        <div class="survey-controls">
          <button @click="toggleMinimize" class="survey-control-btn" :title="isMinimized ? 'Expand' : 'Minimize'">
            <svg v-if="isMinimized" width="16" height="16" viewBox="0 0 24 24" fill="none" stroke="currentColor" stroke-
width="2">
              <polyline points="18 15 12 9 6 15"></polyline>
            </svg>
            <svg v-else width="16" height="16" viewBox="0 0 24 24" fill="none" stroke="currentColor" stroke-width="2">
              <polyline points="6 9 12 15 18 9"></polyline>
            </svg>
          </button>
          <button @click="closeSurvey" class="survey-control-btn" title="Close">
            <svg width="16" height="16" viewBox="0 0 24 24" fill="none" stroke="currentColor" stroke-width="2">
              <line x1="18" y1="6" x2="6" y2="18"></line>
              <line x1="6" y1="6" x2="18" y2="18"></line>
            </svg>
          </button>
        </div>
      </div>

      <div v-if="!isMinimized" class="survey-content">
        <div class="survey-progress">
          <div class="progress-bar">
            <div class="progress-fill" :style="{ width: progress + '%' }"></div>
          </div>
          <div class="progress-text">Step {{ currentStep + 1 }} of {{ totalSteps }}</div>
        </div>

        <div class="survey-step">
          <h3 class="step-title">{{ currentCategory.title }}</h3>
          <p class="step-description">{{ currentCategory.description }}</p>

          <div class="interests-grid">
            <button
              v-for="interest in currentCategory.interests"
              :key="interest"
              @click="toggleInterest(interest)"
              :class="['interest-btn', { 'selected': selectedInterests.includes(interest) }]"
            >
              {{ interest }}
              <span v-if="selectedInterests.includes(interest)" class="check-icon">✓</span>
            </button>
          </div>

          <div class="selected-count">
            Selected interests: <strong>{{ selectedInterests.length }}</strong>
          </div>
        </div>

        <div class="survey-actions">
          <button
            v-if="currentStep > 0"
            @click="prevStep"
            class="survey-btn secondary"
          >
            Back
          </button>
          <button
            @click="skipStep"
            class="survey-btn secondary"
          >
            Skip
          </button>
          <button
            @click="nextStep"
            class="survey-btn primary"
            :disabled="!canProceed"
```

```
          >
            {{ currentStep < totalSteps - 1 ? 'Next' : 'Finish' }}
          </button>
        </div>
      </div>
    </div>
  </Transition>
</template>

<script setup>
import { ref, computed, watch, onMounted } from 'vue';

const props = defineProps({
  user: Object,
  userInterests: {
    type: Array,
    default: () => []
  },
  isDarkTheme: Boolean,
  forceShow: {
    type: Boolean,
    default: false
  }
});

const emit = defineEmits(['save-interests', 'close']);

const isVisible = ref(false);
const isMinimized = ref(false);
const currentStep = ref(0);
const selectedInterests = ref([...props.userInterests || []]);
const hasCompletedSurvey = ref(false);

const interestCategories = [
  {
    title: '🎬 Entertainment',
    description: 'What do you like to watch?',
    interests: ['Gaming', 'Comedy', 'Entertainment', 'Funny', 'Memes', 'Anime', 'Film', 'Series']
  },
  {
    title: '🎨 Creativity',
    description: 'What kind of creativity inspires you?',
    interests: ['Music', 'Art', 'Drawing', 'Dance', 'Singing', 'Design', 'Animation', 'Photography']
  },
  {
    title: '📚 Education',
    description: 'What do you want to learn?',
    interests: ['Education', 'Tutorial', 'Science', 'Tech', 'Programming', 'AI', 'Review', 'Documentary']
  },
  {
    title: '💪 Lifestyle',
    description: 'How do you take care of yourself?',
    interests: ['Fitness', 'Health', 'Yoga', 'Cooking', 'Fashion', 'Beauty', 'Travel', 'Lifestyle']
  },
  {
    title: '⚽Activities',
    description: 'What do you like to do?',
    interests: ['Sports', 'Football', 'Basketball', 'Cars', 'DIY', 'Gardening', 'Adventure', 'Nature']
  }
];

const totalSteps = interestCategories.length;

const currentCategory = computed(() => interestCategories[currentStep.value]);
const progress = computed(() => ((currentStep.value + 1) / totalSteps) * 100);
const canProceed = computed(() => selectedInterests.value.length > 0);

// Проверяем localStorage на факт прохождения опроса
const checkSurveyStatus = () => {
  if (!props.user) return;

  // Если forceShow = true, показываем опрос принудительно
  if (props.forceShow) {
    isVisible.value = true;
```

```
    return;
  }

  const surveyKey = `survey_completed_${props.user.uid}`;
  const completed = localStorage.getItem(surveyKey);
  hasCompletedSurvey.value = completed === 'true';

  // Показываем опрос через 3 секунды, если не был пройден и нет интересов
  if (!completed && props.userInterests.length === 0) {
    setTimeout(() => {
      isVisible.value = true;
    }, 3000);
  }
};

const toggleInterest = (interest) => {
  const index = selectedInterests.value.indexOf(interest);
  if (index > -1) {
    selectedInterests.value.splice(index, 1);
  } else {
    selectedInterests.value.push(interest);
  }
};

const nextStep = () => {
  if (currentStep.value < totalSteps - 1) {
    currentStep.value++;
  } else {
    completeSurvey();
  }
};

const prevStep = () => {
  if (currentStep.value > 0) {
    currentStep.value--;
  }
};

const skipStep = () => {
  if (currentStep.value < totalSteps - 1) {
    currentStep.value++;
  } else {
    completeSurvey();
  }
};

const completeSurvey = () => {
  if (props.user) {
    const surveyKey = `survey_completed_${props.user.uid}`;
    localStorage.setItem(surveyKey, 'true');
    hasCompletedSurvey.value = true;
  }
  emit('save-interests', selectedInterests.value);
  closeSurvey();
};

const closeSurvey = () => {
  isVisible.value = false;
  emit('close');
};

const toggleMinimize = () => {
  isMinimized.value = !isMinimized.value;
};

watch(() => props.user, () => {
  if (props.user) {
    checkSurveyStatus();
  }
}, { immediate: true });

watch(() => props.forceShow, (newVal) => {
  if (newVal && props.user) {
    isVisible.value = true;
```

```
  }
});

onMounted(() => {
  if (props.user) {
    checkSurveyStatus();
  }
});
</script>

<style scoped>
/* Interests Survey */
.interests-survey {
  position: fixed;
  bottom: 20px;
  right: 20px;
  width: 420px;
  max-height: 600px;
  background: white;
  border-radius: 16px;
  box-shadow: 0 8px 32px rgba(0, 0, 0, 0.15);
  overflow: hidden;
  z-index: 1000;
  transition: all 0.3s ease;
  border: 1px solid #e0e0e0;
}

.interests-survey.dark {
  background: #212121;
  border-color: #333;
}

.interests-survey.minimized {
  width: 280px;
  max-height: 60px;
}

.survey-header {
  display: flex;
  justify-content: space-between;
  align-items: center;
  padding: 1rem 1.25rem;
  background: linear-gradient(135deg, #8e66ea 0%, #a24b85 100%);
  color: white;
}

.survey-title {
  display: flex;
  align-items: center;
  gap: 0.5rem;
  font-weight: 600;
  font-size: 1rem;
}

.survey-icon {
  font-size: 1.25rem;
}

.survey-controls {
  display: flex;
  gap: 0.5rem;
}

.survey-control-btn {
  background: rgba(255, 255, 255, 0.2);
  border: none;
  width: 28px;
  height: 28px;
  border-radius: 6px;
  cursor: pointer;
  display: flex;
  align-items: center;
  justify-content: center;
  transition: all 0.2s;
```

```
  color: white;
}

.survey-control-btn:hover {
  background: rgba(255, 255, 255, 0.3);
}

.survey-content {
  padding: 1.5rem 1.25rem;
  max-height: 520px;
  overflow-y: auto;
}

.survey-progress {
  margin-bottom: 1.5rem;
}

.progress-bar {
  height: 6px;
  background: #e0e0e0;
  border-radius: 3px;
  overflow: hidden;
  margin-bottom: 0.5rem;
}

.interests-survey.dark .progress-bar {
  background: #333;
}

.progress-fill {
  height: 100%;
  background: linear-gradient(90deg, #8e66ea 0%, #a24b85 100%);
  transition: width 0.3s ease;
}

.progress-text {
  font-size: 0.8rem;
  color: #666;
  text-align: center;
}

.interests-survey.dark .progress-text {
  color: #aaa;
}

.survey-step {
  margin-bottom: 1.5rem;
}

.step-title {
  font-size: 1.25rem;
  margin-bottom: 0.5rem;
  color: #000;
}

.interests-survey.dark .step-title {
  color: #f1f1f1;
}

.step-description {
  font-size: 0.9rem;
  color: #666;
  margin-bottom: 1.25rem;
}

.interests-survey.dark .step-description {
  color: #aaa;
}

.interests-grid {
  display: grid;
  grid-template-columns: repeat(2, 1fr);
  gap: 0.5rem;
  margin-bottom: 1rem;
```

```
}

.interest-btn {
  padding: 0.75rem;
  border: 2px solid #e0e0e0;
  background: white;
  border-radius: 10px;
  cursor: pointer;
  font-size: 0.85rem;
  font-weight: 500;
  font-family: inherit;
  transition: all 0.2s;
  position: relative;
  color: #333;
  text-align: left;
}

.interests-survey.dark .interest-btn {
  background: #1a1a1a;
  border-color: #444;
  color: #f1f1f1;
}

.interest-btn:hover {
  border-color: #8e66ea;
  transform: translateY(-2px);
}

.interest-btn.selected {
  background: linear-gradient(135deg, #8e66ea 0%, #a24b85 100%);
  color: white;
  border-color: transparent;
}

.check-icon {
  position: absolute;
  top: 6px;
  right: 6px;
  width: 18px;
  height: 18px;
  background: rgba(255, 255, 255, 0.3);
  border-radius: 50%;
  display: flex;
  align-items: center;
  justify-content: center;
  font-size: 0.7rem;
}

.selected-count {
  font-size: 0.85rem;
  color: #666;
  text-align: center;
  padding: 0.75rem;
  background: #f5f5f5;
  border-radius: 8px;
}

.interests-survey.dark .selected-count {
  background: #1a1a1a;
  color: #aaa;
}

.selected-count strong {
  color: #8e66ea;
  font-weight: 700;
}

.survey-actions {
  display: flex;
  gap: 0.5rem;
  margin-top: 1rem;
}

.survey-btn {
```

```
  flex: 1;
  padding: 0.75rem;
  border: none;
  border-radius: 8px;
  font-weight: 600;
  font-size: 0.9rem;
  cursor: pointer;
  transition: all 0.2s;
  font-family: inherit;
}

.survey-btn.primary {
  background: linear-gradient(135deg, #8e66ea 0%, #a24b85 100%);
  color: white;
}

.survey-btn.primary:hover:not(:disabled) {
  transform: translateY(-2px);
  box-shadow: 0 4px 12px rgba(102, 126, 234, 0.4);
}

.survey-btn.primary:disabled {
  opacity: 0.5;
  cursor: not-allowed;
}

.survey-btn.secondary {
  background: #f0f0f0;
  color: #333;
}

.interests-survey.dark .survey-btn.secondary {
  background: #2a2a2a;
  color: #f1f1f1;
}

.survey-btn.secondary:hover {
  background: #e0e0e0;
}

.interests-survey.dark .survey-btn.secondary:hover {
  background: #333;
}

/* Анимации */
.survey-slide-enter-active,
.survey-slide-leave-active {
  transition: all 0.3s ease;
}

.survey-slide-enter-from {
  transform: translateX(100%);
  opacity: 0;
}

.survey-slide-leave-to {
  transform: translateX(100%);
  opacity: 0;
}

/* Адаптивность */
@media (max-width: 768px) {
  .interests-survey {
    bottom: 10px;
    right: 10px;
    left: 10px;
    width: auto;
    max-width: none;
  }

  .interests-survey.minimized {
    left: auto;
    width: 220px;
  }
```

```
  .interests-grid {
    grid-template-columns: 1fr;
  }
}

/* Скроллбар для контента опроса */
.survey-content::-webkit-scrollbar {
  width: 6px;
}

.survey-content::-webkit-scrollbar-track {
  background: #f1f1f1;
  border-radius: 3px;
}

.interests-survey.dark .survey-content::-webkit-scrollbar-track {
  background: #1a1a1a;
}

.survey-content::-webkit-scrollbar-thumb {
  background: #888;
  border-radius: 3px;
}

.survey-content::-webkit-scrollbar-thumb:hover {
  background: #555;
}
</style>
```

ProfilePage.vue

```
<script setup>
import { computed, ref } from 'vue';

const props = defineProps({
  profileData: Object,
  user: Object,
  videos: Array,
  subscriptions: Array,
  watchLater: Array,
  history: Array,
  formatDate: Function
});

const emit = defineEmits([
  'toggle-subscription',
  'upload-avatar',
  'watch-video',
  'delete-video',
  'update-interests',
  'open-survey'
]);

const isEditingInterests = ref(false);

const userVideos = computed(() => {
  return props.videos.filter(v => v.uploadedBy === props.profileData.userId);
});

const userLikedVideos = computed(() => {
  if (!props.profileData.isOwn) return [];
  return props.videos.filter(v => v.likedBy?.includes(props.user?.uid));
});

const handleVideoClick = (video) => {
  emit('watch-video', video);
};

const getVideoAuthorAvatar = (video) => {
  return props.profileData.avatar || '';
};
```

```
const toggleEditMode = () => {
  isEditingInterests.value = !isEditingInterests.value;
};

const removeInterest = (interest) => {
  const updatedInterests = props.profileData.interests.filter(i => i !== interest);
  emit('update-interests', updatedInterests);
};

const clearAllInterests = () => {
  if (confirm('Ви впевнені що хочете видалити всі інтереси?')) {
    emit('update-interests', []);
    isEditingInterests.value = false;
  }
};
</script>

<template>
  <div class="profile-page">
    <div class="profile-header">
      <div class="profile-avatar" @click="profileData.isOwn ? emit('upload-avatar') : null" :class="{ clickable:
profileData.isOwn }">
        <img v-if="profileData.avatar" :src="profileData.avatar" alt="Avatar" class="avatar-img" />
        <svg v-else width="80" height="80" viewBox="0 0 24 24" fill="none" stroke="currentColor" stroke-width="2">
          <path d="M20 21v-2a4 4 0 0 0-4-4H8a4 4 0 0 0-4 4v2"></path>
          <circle cx="12" cy="7" r="4"></circle>
        </svg>
        <div v-if="profileData.isOwn" class="avatar-overlay">
          <span> Змінити</span>
        </div>
      </div>
      <h1>{{ profileData.isOwn ? `Привет, ${profileData.name || 'User'}!` : profileData.name }}</h1>
      <p class="profile-email">{{ profileData.email }}</p>

      <button
        v-if="!profileData.isOwn && user"
        class="btn-subscribe profile-subscribe"
        :class="{ subscribed: subscriptions.includes(profileData.userId) }"
        @click="emit('toggle-subscription', profileData.userId)"
      >
        {{ subscriptions.includes(profileData.userId) ? '✓ Підписані' : '+ Підписатись' }}
      </button>
    </div>

    <div class="profile-section">
      <div class="section-header">
        <h2>Інтереси</h2>
        <div v-if="profileData.isOwn" class="interests-actions">
          <button
            v-if="profileData.interests.length > 0"
            @click="toggleEditMode"
            class="btn-edit-interests"
            :class="{ active: isEditingInterests }"
          >
            {{ isEditingInterests ? '✓ Готово' : '☒ Редагувати' }}
          </button>
          <button
            @click="emit('open-survey')"
            class="btn-add-interests"
          >
            ✨{{ profileData.interests.length === 0 ? 'Додати інтереси' : 'Пройти опитування' }}
          </button>
          <button
            v-if="isEditingInterests && profileData.interests.length > 0"
            @click="clearAllInterests"
            class="btn-clear-interests"
          >
             Очистити все
          </button>
        </div>
      </div>

      <div v-if="profileData.interests.length === 0" class="empty-state">
        <p>{{ profileData.isOwn ? 'Вы еще не указали свои интересы' : 'Користувач не вказав інтереси' }}</p>
```

```
    <p v-if="profileData.isOwn" class="empty-hint">Лайкайте видео чтобы добавить интересы</p>
  </div>
  <div v-else class="interests-list">
    <div
      v-for="interest in profileData.interests"
      :key="interest"
      :class="['interest-tag', { editable: isEditingInterests && profileData.isOwn }]"
    >
      {{ interest }}
      <button
        v-if="isEditingInterests && profileData.isOwn"
        @click.stop="removeInterest(interest)"
        class="remove-interest-btn"
        :title="`Видалити ${interest}`"
      >
        ×
      </button>
    </div>
  </div>
</div>

<div class="profile-section">
  <h2>{{ profileData.isOwn ? 'Мої відео' : 'Відео автора' }}</h2>
  <div v-if="userVideos.length === 0" class="empty-state">
    <p>{{ profileData.isOwn ? 'Ви ще не завантажили жодного відео' : 'Користувач не завантажив жодного відео' }}</p>
  </div>
  <div v-else class="videos-grid">
    <div v-for="video in userVideos" :key="video.id" class="video-card" @click="handleVideoClick(video)">
      <div class="video-thumbnail">
        <video :src="video.videoUrl"></video>
      </div>
      <div class="video-info">
        <h3>{{ video.title }}</h3>
        <div class="video-author-with-avatar">
          <div class="author-avatar-small">
            <img
              v-if="getVideoAuthorAvatar(video)"
              :src="getVideoAuthorAvatar(video)"
              :alt="profileData.name"
              class="avatar-img-tiny"
            />
            <svg
              v-else
              width="20"
              height="20"
              viewBox="0 0 24 24"
              fill="none"
              stroke="currentColor"
              stroke-width="2"
            >
              <path d="M20 21v-2a4 4 0 0 0-4-4H8a4 4 0 0 0-4 4v2"></path>
              <circle cx="12" cy="7" r="4"></circle>
            </svg>
          </div>
          <p class="video-meta">{{ video.views }} переглядів • {{ formatDate(video.createdAt) }}</p>
        </div>
      </div>
    </div>
  </div>
</div>

<template v-if="profileData.isOwn">
  <div class="profile-section">
    <h2>Liked Videos</h2>
    <div v-if="userLikedVideos.length === 0" class="empty-state">
      <p>Вы еще не лайкали видео</p>
    </div>
    <div v-else class="videos-grid">
      <div v-for="video in userLikedVideos" :key="video.id" class="video-card" @click="handleVideoClick(video)">
        <div class="video-thumbnail">
          <video :src="video.videoUrl"></video>
        </div>
        <div class="video-info">
          <h3>{{ video.title }}</h3>
```

```
              <p class="video-meta">{{ video.views }} переглядів • {{ formatDate(video.createdAt) }}</p>
            </div>
          </div>
        </div>
      </div>

      <div class="profile-section">
        <h2>Watch Later</h2>
        <div v-if="watchLater.length === 0" class="empty-state">
          <p>Список порожній</p>
        </div>
        <div v-else class="videos-grid">
          <div v-for="videoId in watchLater" :key="videoId" class="video-card" @click="handleVideoClick(videos.find(v => v.id
=== videoId))">
            <template v-if="videos.find(v => v.id === videoId)">
              <div class="video-thumbnail">
                <video :src="videos.find(v => v.id === videoId).videoUrl"></video>
              </div>
              <div class="video-info">
                <h3>{{ videos.find(v => v.id === videoId).title }}</h3>
                <p class="video-meta">{{ videos.find(v => v.id === videoId).views }} переглядів</p>
              </div>
            </template>
          </div>
        </div>
      </div>

      <div class="profile-section">
        <h2>History</h2>
        <div v-if="history.length === 0" class="empty-state">
          <p>Історія порожня</p>
        </div>
        <div v-else class="videos-grid">
          <div v-for="videoId in history.slice(0, 20)" :key="videoId" class="video-card"
@click="handleVideoClick(videos.find(v => v.id === videoId))">
            <template v-if="videos.find(v => v.id === videoId)">
              <div class="video-thumbnail">
                <video :src="videos.find(v => v.id === videoId).videoUrl"></video>
              </div>
              <div class="video-info">
                <h3>{{ videos.find(v => v.id === videoId).title }}</h3>
                <p class="video-meta">{{ videos.find(v => v.id === videoId).views }} переглядів</p>
              </div>
            </template>
          </div>
        </div>
      </div>
    </template>
  </div>
</template>

<style scoped>
.section-header {
  display: flex;
  justify-content: space-between;
  align-items: center;
  margin-bottom: 1.5rem;
}

.section-header h2 {
  margin: 0;
}

.interests-actions {
  display: flex;
  gap: 0.5rem;
}

.btn-edit-interests,
.btn-clear-interests,
.btn-add-interests {
  padding: 0.5rem 1rem;
  border: 1px solid #ddd;
  background: white;
```

```
  border-radius: 8px;
  cursor: pointer;
  font-size: 0.875rem;
  font-weight: 500;
  font-family: inherit;
  transition: all 0.2s;
}

body.dark-theme .btn-edit-interests,
body.dark-theme .btn-clear-interests,
body.dark-theme .btn-add-interests {
  background: #2a2a2a;
  border-color: #444;
  color: #f1f1f1;
}

.btn-edit-interests:hover {
  background: #f5f5f5;
  border-color: #8e66ea;
}

body.dark-theme .btn-edit-interests:hover {
  background: #333;
  border-color: #8e66ea;
}

.btn-edit-interests.active {
  background: #8e66ea;
  color: white;
  border-color: #8e66ea;
}

.btn-add-interests {
  background: linear-gradient(135deg, #8e66ea 0%, #a24b85 100%);
  color: white;
  border-color: transparent;
}

.btn-add-interests:hover {
  transform: translateY(-2px);
  box-shadow: 0 4px 12px rgba(102, 126, 234, 0.4);
}

.btn-clear-interests {
  border-color: #e53e3e;
  color: #e53e3e;
}

.btn-clear-interests:hover {
  background: #e53e3e;
  color: white;
}

body.dark-theme .btn-clear-interests {
  border-color: #fc8181;
  color: #fc8181;
}

body.dark-theme .btn-clear-interests:hover {
  background: #fc8181;
  color: #000;
}

.interests-list {
  display: flex;
  flex-wrap: wrap;
  gap: 0.75rem;
}

.interest-tag {
  padding: 0.5rem 1rem;
  background: linear-gradient(135deg, #8e66ea 0%, #a24b85 100%);
  color: white;
  border-radius: 20px;
```

```
  font-size: 0.9rem;
  font-weight: 500;
  position: relative;
  transition: all 0.2s;
}

.interest-tag.editable {
  padding-right: 2.5rem;
  cursor: default;
}

.interest-tag.editable:hover {
  transform: scale(1.05);
  box-shadow: 0 4px 12px rgba(102, 126, 234, 0.3);
}

.remove-interest-btn {
  position: absolute;
  top: 50%;
  right: 6px;
  transform: translateY(-50%);
  width: 24px;
  height: 24px;
  border: none;
  background: rgba(255, 255, 255, 0.3);
  color: white;
  border-radius: 50%;
  cursor: pointer;
  font-size: 1.2rem;
  font-weight: 700;
  line-height: 1;
  display: flex;
  align-items: center;
  justify-content: center;
  transition: all 0.2s;
  padding: 0;
}

.remove-interest-btn:hover {
  background: rgba(229, 62, 62, 0.9);
  transform: translateY(-50%) scale(1.1);
}

@media (max-width: 768px) {
  .section-header {
    flex-direction: column;
    align-items: flex-start;
    gap: 1rem;
  }

  .interests-actions {
    width: 100%;
  }

  .btn-edit-interests,
  .btn-clear-interests,
  .btn-add-interests {
    flex: 1;
  }
}
</style>
```

UploadModal.vue

```
<script setup>
const props = defineProps({
  user: Object,
  uploadTitle: String,
  uploadDescription: String,
  selectedTags: Array,
  tagGroups: Array
});
```

```
const emit = defineEmits([
  'update:uploadTitle',
  'update:uploadDescription',
  'update:selectedTags',
  'close',
  'open-cloudinary-widget',
  'toggle-tag'
]);

const handleTagClick = (tag) => {
  emit('toggle-tag', tag);
};
</script>

<template>
  <div class="modal-overlay" @click="emit('close')">
    <div class="modal upload-modal" @click.stop>
      <h2>Завантажити відео</h2>
      <div class="form-group">
        <label>Назва *</label>
        <input
          :value="uploadTitle"
          @input="emit('update:uploadTitle', $event.target.value)"
          type="text"
          placeholder="Назва відео"
          required
        />
      </div>
      <div class="form-group">
        <label>Опис</label>
        <textarea
          :value="uploadDescription"
          @input="emit('update:uploadDescription', $event.target.value)"
          placeholder="Опис відео"
          rows="3"
        ></textarea>
      </div>
      <div class="form-group">
        <label>Теги (оберіть до 10)</label>
        <div class="tags-selector">
          <div v-for="group in tagGroups" :key="group.name" class="tag-group">
            <div class="tag-group-name">{{ group.name }}</div>
            <div class="tag-group-items">
              <button
                v-for="tag in group.tags"
                :key="tag"
                type="button"
                class="tag-option"
                :class="{ selected: selectedTags.includes(tag) }"
                @click="handleTagClick(tag)"
              >
                {{ tag }}
              </button>
            </div>
          </div>
        </div>
        <small style="color: #999;">Обрано: {{ selectedTags.length }}/10</small>
      </div>

      <div class="modal-buttons">
        <button class="btn btn-secondary" @click="emit('close')">Скасувати</button>
        <button class="btn btn-primary" @click="emit('open-cloudinary-widget')" :disabled="!uploadTitle ||
selectedTags.length === 0">
          Вибрати відео
        </button>
      </div>
    </div>
  </div>
</template>
```

VideoCard.vue

```
<script setup>
```

```
const props = defineProps({
  video: Object,
  formatDate: Function,
  showAuthor: {
    type: Boolean,
    default: true
  },
  showTags: {
    type: Boolean,
    default: true
  },
  showStats: {
    type: Boolean,
    default: true
  },
  clickable: {
    type: Boolean,
    default: true
  }
});

const emit = defineEmits(['click', 'author-click']);

const handleClick = () => {
  if (props.clickable) {
    emit('click', props.video);
  }
};

const handleAuthorClick = (e) => {
  e.stopPropagation();
  emit('author-click', props.video.uploadedBy);
};
</script>

<template>
  <div
    class="video-card"
    :class="{ 'no-click': !clickable }"
    @click="handleClick"
  >
    <div class="video-thumbnail">
      <video :src="video.videoUrl"></video>
      <div class="video-duration">{{ video.tags?.length || 0 }} tags</div>
    </div>
    <div class="video-info">
      <h3>{{ video.title }}</h3>
      <p v-if="showStats" class="video-meta">
        <span
          v-if="showAuthor"
          class="author-name clickable"
          @click="handleAuthorClick"
        >
          {{ video.uploaderName }}
        </span>
        <span v-if="showAuthor && showStats"> • </span>
        <span v-if="showStats">
          {{ video.views }} переглядів • {{ formatDate(video.createdAt) }}
        </span>
      </p>
      <div v-if="showTags && video.tags" class="video-tags">
        <span v-for="tag in video.tags.slice(0, 3)" :key="tag" class="tag">{{ tag }}</span>
        <span v-if="video.tags.length > 3" class="tag-more">+{{ video.tags.length - 3 }}</span>
      </div>
    </div>
  </div>
</template>
```

VideoStatsModal.vue

```
<script setup>
const props = defineProps({
  video: Object,
```

```
  formatDate: Function,
  getVideoChartData: Function,
  getDetailedVideoStats: Function
});

const emit = defineEmits(['close']);

const detailedStats = props.getDetailedVideoStats(props.video);
const chartData = props.getVideoChartData(props.video);
</script>

<template>
  <div class="modal-overlay" @click="emit('close')">
    <div class="modal stats-modal" @click.stop>
      <h2>📊 Детальна статистика</h2>

      <div class="stats-video-info">
        <video :src="video.videoUrl" controls style="width: 100%; border-radius: 8px; max-height: 300px;"></video>
        <h3>{{ video.title }}</h3>
      </div>

      <div class="stats-grid">
        <div class="stat-item">
          <div class="stat-label">Перегляди</div>
          <div class="stat-big">{{ video.views || 0 }}</div>
        </div>
        <div class="stat-item">
          <div class="stat-label">Лайки</div>
          <div class="stat-big">{{ video.likes || 0 }}</div>
        </div>
        <div class="stat-item">
          <div class="stat-label">Дизлайки</div>
          <div class="stat-big">{{ video.dislikes || 0 }}</div>
        </div>
        <div class="stat-item">
          <div class="stat-label">Like Ratio</div>
          <div class="stat-big">
            {{ (video.likes || 0) + (video.dislikes || 0) > 0
              ? ((video.likes / ((video.likes || 0) + (video.dislikes || 0))) * 100).toFixed(1)
              : 0 }}%
          </div>
        </div>
      </div>

      <div class="stats-grid" style="margin-top: 1rem;">
        <div class="stat-item">
          <div class="stat-label">Engagement Rate</div>
          <div class="stat-big">{{ detailedStats.engagementRate }}%</div>
        </div>
        <div class="stat-item">
          <div class="stat-label">CTR</div>
          <div class="stat-big">{{ detailedStats.ctr }}%</div>
        </div>
        <div class="stat-item">
          <div class="stat-label">Час перегляду</div>
          <div class="stat-big">{{ detailedStats.avgWatchTime }}%</div>
        </div>
        <div class="stat-item">
          <div class="stat-label">Пік перегляду</div>
          <div class="stat-big">{{ detailedStats.peakViewingTime }}</div>
        </div>
      </div>

      <div class="chart-section">
        <h4>Зростання за 30 днів</h4>
        <div class="simple-chart">
          <div
            v-for="(point, index) in chartData"
            :key="index"
            class="chart-bar"
            :style="{ height: (point.views / video.views * 100) + '%' }"
            :title="`${point.day}: ${point.views} переглядів, ${point.likes} лайків`"
          >
            <span class="chart-label">{{ index % 5 === 0 ? point.day : '' }}</span>
```

```
          </div>
        </div>
      </div>



      <button class="btn btn-primary" @click="emit('close')" style="margin-top: 2rem;">Закрити</button>
    </div>
  </div>
</template>
```

WatchPage.vue

```
<script setup>
import { computed, onMounted, ref } from 'vue';

const props = defineProps({
  currentVideo: Object,
  user: Object,
  userName: String,
  videos: Array,
  subscriptions: Array,
  watchLater: Array,
  comments: Array,
  commentText: String,
  usersData: Object,
  formatDate: Function,
  getOptimizedVideoUrl: Function
});

const emit = defineEmits([
 'update:commentText',
  'toggle-subscription',
  'toggle-watch-later',
  'delete-video',
  'like',
  'dislike',
  'share',
  'go-to-user-profile',
  'go-home',
  'add-comment',
  'toggle-comment-like',
  'reply-to-comment',
  'watch-video'
]);

// ДОБАВЛЕНО: Ref для видео элемента
const videoPlayer = ref(null);

onMounted(() => {
  // Оптимизация: предзагружаем видео при монтировании
  if (props.currentVideo && videoPlayer.value) {
    videoPlayer.value.preload = 'auto';
  }
});

const recommendedVideos = computed(() => {
  if (!props.currentVideo || props.videos.length <= 1) return [];

  const current = props.currentVideo;
  const otherVideos = props.videos.filter(v => v.id !== current.id);

  const scored = otherVideos.map(video => {
    const commonTags = video.tags.filter(tag => current.tags.includes(tag)).length;
    const viewsScore = video.views / 100;
    const recentScore = video.createdAt ? (Date.now() - video.createdAt.toDate().getTime()) / (1000 * 60 * 60 * 24) : 0;
    const trendScore = recentScore < 7 ? (video.views / (recentScore + 1)) : 0;

    return {
      ...video,
      score: (commonTags * 10) + viewsScore + (trendScore * 5)
    };
```

```
  });

  return scored.sort((a, b) => b.score - a.score).slice(0, 5);
});

const handleAuthorClick = (userId) => {
  emit('go-to-user-profile', userId);
};

const handleRecommendedClick = (video) => {
  emit('watch-video', video);
};

const getUserAvatar = (userId) => {
  return props.usersData[userId]?.avatar || '';
};

const getUserName = (userId) => {
  return props.usersData[userId]?.name || 'Користувач';
};

// ДОБАВЛЕНО: Оптимизированный URL для текущего видео
const optimizedVideoUrl = computed(() => {
  return props.getOptimizedVideoUrl(props.currentVideo.videoUrl);
});
</script>

<template>
  <div class="watch-page">
    <div class="watch-content">
      <div class="watch-main">
        <div class="video-player">
          <!-- ИЗМЕНЕНО: Используем оптимизированный URL и настройки для быстрой загрузки -->
          <video
            ref="videoPlayer"
            :src="optimizedVideoUrl"
            controls
            autoplay
            preload="metadata"
            playsinline
            style="width: 100%; border-radius: 12px;"
            @loadeddata="videoPlayer?.play()"
          >
            <source :src="optimizedVideoUrl" type="video/mp4">
            Ваш браузер не підтримує відео тег.
          </video>
        </div>

        <div class="video-details">
          <h1>{{ currentVideo.title }}</h1>
          <div class="video-stats">
            <div>
              <span>{{ currentVideo.views }} переглядів</span>
              <span class="video-date"> • {{ formatDate(currentVideo.createdAt) }}</span>
            </div>
            <div class="video-actions">
              <button
                class="action-btn"
                :class="{ active: currentVideo.likedBy?.includes(user?.uid) }"
                @click="emit('like', currentVideo)"
              >
                <svg width="20" height="20" viewBox="0 0 24 24" fill="none" stroke="currentColor" stroke-width="2">
                  <path d="M14 9V5a3 3 0 0 0-3-3l-4 9v11h11.28a2 2 0 0 0 2-1.7l1.38-9a2 2 0 0 0-2-2.3zM7 22H4a2 2 0 0 1-2-2v-
7a2 2 0 0 1 2-2h3"></path>
                </svg>
                {{ currentVideo.likes }}
              </button>
              <button
                class="action-btn"
                :class="{ active: currentVideo.dislikedBy?.includes(user?.uid) }"
                @click="emit('dislike', currentVideo)"
              >
                <svg width="20" height="20" viewBox="0 0 24 24" fill="none" stroke="currentColor" stroke-width="2"
style="transform: rotate(180deg)">
```

```
                <path d="M14 9V5a3 3 0 0 0-3-3l-4 9v11h11.28a2 2 0 0 0 2-1.7l1.38-9a2 2 0 0 0-2-2.3zM7 22H4a2 2 0 0 1-2-2v-
7a2 2 0 0 1 2-2h3"></path>
              </svg>
              {{ currentVideo.dislikes }}
            </button>
            <button class="action-btn" @click="emit('share', currentVideo)">
              <svg width="20" height="20" viewBox="0 0 24 24" fill="none" stroke="currentColor" stroke-width="2">
                <circle cx="18" cy="5" r="3"></circle>
                <circle cx="6" cy="12" r="3"></circle>
                <circle cx="18" cy="19" r="3"></circle>
                <line x1="8.59" y1="13.51" x2="15.42" y2="17.49"></line>
                <line x1="15.41" y1="6.51" x2="8.59" y2="10.49"></line>
              </svg>
              Share
            </button>
          </div>
        </div>

        <div class="video-description">
          <div class="author-section">
            <p>
              <strong
                class="author-name clickable"
                @click="handleAuthorClick(currentVideo.uploadedBy)"
              >
                {{ currentVideo.uploaderName }}
              </strong>
            </p>
            <button
              v-if="user && currentVideo.uploadedBy !== user.uid"
              class="btn-subscribe"
              :class="{ subscribed: subscriptions.includes(currentVideo.uploadedBy) }"
              @click="emit('toggle-subscription', currentVideo.uploadedBy)"
            >
              {{ subscriptions.includes(currentVideo.uploadedBy) ? 'Відписатись' : 'Підписатись' }}
            </button>
            <button
              v-if="user && currentVideo.uploadedBy === user.uid"
              class="btn-delete"
              @click="emit('delete-video', currentVideo.id)"
            >
               Видалити
            </button>
          </div>
          <p>{{ currentVideo.description }}</p>
          <div class="video-tags">
            <span v-for="tag in currentVideo.tags" :key="tag" class="tag">{{ tag }}</span>
          </div>
          <button
            class="btn-watch-later"
            :class="{ active: watchLater.includes(currentVideo.id) }"
            @click="emit('toggle-watch-later', currentVideo)"
          >
            {{ watchLater.includes(currentVideo.id) ? '✓ В Watch Later' : '+ Watch Later' }}
          </button>
        </div>
      </div>

      <div class="comments-section">
        <h3>Коментарі ({{ comments.length }})</h3>

        <div v-if="user" class="add-comment">
          <input
            :value="commentText"
            @input="emit('update:commentText', $event.target.value)"
            type="text"
            placeholder="Додати коментар..."
            @keyup.enter="emit('add-comment')"
          />
          <button class="btn btn-primary" @click="emit('add-comment')">Відправити</button>
        </div>
        <p v-else style="color: #999; margin-bottom: 1rem;">Увійдіть щоб коментувати</p>

        <div class="comments-list">
```

```
      <div v-for="comment in comments" :key="comment.id" class="comment">
        <div class="comment-header">
          <div class="comment-avatar">
            <img
              v-if="getUserAvatar(comment.userId)"
              :src="getUserAvatar(comment.userId)"
              :alt="getUserName(comment.userId)"
              class="avatar-img-small"
              loading="lazy"
            />
            <svg
              v-else
              width="32"
              height="32"
              viewBox="0 0 24 24"
              fill="none"
              stroke="currentColor"
              stroke-width="2"
            >
              <path d="M20 21v-2a4 4 0 0 0-4-4H8a4 4 0 0 0-4 4v2"></path>
              <circle cx="12" cy="7" r="4"></circle>
            </svg>
          </div>
          <div class="comment-author-info">
            <strong
              class="author-name clickable"
              @click="handleAuthorClick(comment.userId)"
            >
              {{ comment.userName }}
            </strong>
            <span class="comment-date">{{ formatDate(comment.createdAt) }}</span>
          </div>
        </div>
        <p>{{ comment.text }}</p>
        <div class="comment-actions">
          <button
            class="comment-action-btn"
            :class="{ liked: comment.likedBy?.includes(user?.uid) }"
            @click="emit('toggle-comment-like', comment)"
          >
             {{ comment.likes || 0 }}
          </button>
          <button class="comment-action-btn" @click="emit('reply-to-comment', comment.id)">
             Відповісти
          </button>
        </div>
      </div>
    </div>
  </div>
</div>

<div class="watch-sidebar">
  <h3>Рекомендації</h3>
  <div class="recommended-videos">
    <div
      v-for="video in recommendedVideos"
      :key="video.id"
      class="recommended-card"
      @click="handleRecommendedClick(video)"
    >
      <div class="recommended-thumbnail">
        <!-- ИЗМЕНЕНО: Оптимизированные превью для рекомендаций -->
        <img
          v-if="getOptimizedVideoUrl(video.videoUrl)"
          :src="getOptimizedVideoUrl(video.videoUrl).replace(/\.(mp4|mov|avi|webm)$/, '.jpg')"
          :alt="video.title"
          loading="lazy"
          class="recommended-thumbnail-img"
        />
        <div v-else class="recommended-thumbnail-placeholder">
          <svg width="24" height="24" viewBox="0 0 24 24" fill="none" stroke="currentColor" stroke-width="1">
            <rect x="3" y="3" width="18" height="18" rx="2" ry="2"></rect>
          </svg>
        </div>
```

```
            </div>
            <div class="recommended-info">
              <h4>{{ video.title }}</h4>
              <p class="recommended-meta">{{ video.uploaderName }}</p>
              <p class="recommended-meta">{{ video.views }} переглядів • {{ formatDate(video.createdAt) }}</p>
              <div class="recommended-tags">
                <span v-for="tag in video.tags.slice(0, 3)" :key="tag" class="mini-tag">{{ tag }}</span>
              </div>
            </div>
          </div>

          <div v-if="recommendedVideos.length === 0" class="no-recommendations">
            <p>Немає рекомендацій</p>
          </div>
        </div>
      </div>
    </div>
  </div>
</template>
```

main.css.min

```
@import url(https://fonts.googleapis.com/css2?family=Inter+Tight:ital,wght@0,100..900;1,100..900&display=swap);.action-btn,.video-actions{gap:.5rem;display:flex}.comments-section,.dashboard-section,.header,.profile-header,.profile-section,.video-details{box-shadow:0 2px 8px rgba(0,0,0,.1)}.action-btn,.add-comment input,.btn,.filter-select,.search-input,.tag-option{font-family:inherit}.action-btn,.btn,.filter-select,.profile-avatar.clickable,.tag-option,.video-card,.video-row-title{cursor:pointer}.stat-card:hover,.summary-card:hover,.video-card:hover{transform:translateY(-4px)}.empty-videos,.profile-header,.stat-item,.video-row-stat{text-align:center}.action-btn.active svg,.comment-avatar svg,.mini-avatar svg,.profile-avatar svg{stroke:white}img,video{max-width:100%}.avatar-img,.avatar-img-small,.avatar-img-table,.avatar-img-tiny,.extreme-video video,.recommended-thumbnail-img,.thumbnail-img,.video-row-thumbnail video{object-fit:cover}.recommended-info h4,.video-info h3{-webkit-line-clamp:2;-webkit-box-orient:vertical}*{margin:0;padding:0;box-sizing:border-box}body{font-family:"Inter Tight",sans-serif;background-color:#f5f5f5}#app{min-height:100vh}.header{background-color:#000;padding:1.25rem 0}.video-card:hover{box-shadow:0 4px 16px rgba(0,0,0,.15)}.video-thumbnail{width:100%;height:180px;background:#000;position:relative;overflow:hidden}.comment-avatar img,.mini-avatar img,.recommended-thumbnail video,.table-avatar img,.top-user-avatar img,.user-avatar img,.video-thumbnail video{width:100%;height:100%;object-fit:cover}.video-duration{position:absolute;bottom:8px;right:8px;background:rgba(0,0,0,.8);color:#fff;padding:2px 6px;border-radius:4px;font-size:.75rem}.recommended-info,.video-info{padding:1rem}.video-info h3{font-size:1rem;margin-bottom:.5rem;color:#000;display:-webkit-box;overflow:hidden}.video-meta{font-size:.875rem;color:#666;margin-bottom:.5rem}.interests-badges,.video-tags{display:flex;flex-wrap:wrap;gap:.5rem;margin-top:.5rem}.tag{padding:.25rem .75rem;background:#e0e0e0;border-radius:16px;font-size:.75rem;color:#333}.profile-page,.watch-page{max-width:1200px;margin:0 auto;padding:2rem}.comments-section h3,.tag-group,.video-details,.video-player{margin-bottom:1.5rem}.video-details{background:#fff;padding:1.5rem;border-radius:12px}.video-details h1{font-size:1.75rem;margin-bottom:1rem}.video-stats{display:flex;justify-content:space-between;align-items:center;padding-bottom:1rem;border-bottom:1px solid #eee;margin-bottom:1rem}.action-btn{align-items:center;padding:.5rem 1rem;border:1px solid #ddd;border-radius:20px;background:#fff;transition:.2s;font-size:.9rem}.action-btn:hover,.btn-watch-later:hover,.recommended-card:hover{background:#f5f5f5}.action-btn.active,.btn-watch-later.active{background:#000;color:#fff;border-color:#000}.video-description{padding:1rem;background:#f9f9f9;border-radius:8px}.comments-section{background:#fff;padding:1.5rem;border-radius:12px}.add-comment{display:flex;gap:1rem;margin-bottom:2rem}.add-comment input{flex:1;padding:.75rem;border:1px solid #ddd;border-radius:8px;font-size:1rem}.comments-list,.recommended-videos,.top-users-list{display:flex;flex-direction:column;gap:1rem}.comment-avatar,.profile-avatar{align-items:center;display:flex}.comment{display:flex;gap:1rem;padding:1rem;background:#f9f9f9;border-radius:8px}body.dark-theme .comment,body.dark-theme .users-table tbody tr:hover,body.dark-theme .video-row:not(.video-row-header):hover{background:#2a2a2a}.comment-avatar{background:linear-gradient(135deg,#8e66ea 0,#a24b85 100%);justify-content:center;flex-shrink:0}.comment-content,.recommended-info,.watch-main{flex:1;min-width:0}.comment-header{display:flex;justify-content:space-between;align-items:start}.comment-date{font-size:.8rem;color:#999;font-weight:400;min-width:100px}.comment strong{display:block;margin-bottom:.5rem;color:#000}.comment p{color:#333;width:100%}.profile-header{padding:3rem 2rem;background:#fff;border-radius:12px;margin-bottom:2rem}.profile-avatar{width:120px;height:120px;margin:0 auto 1.5rem;background:linear-gradient(135deg,#8e66ea 0,#a24b85 100%);border-radius:50%;justify-content:center;position:relative;overflow:hidden}.avatar-img{width:100%;height:100%;border-radius:50%}.btn-theme,.logo img{height:40px}.avatar-overlay{position:absolute;top:0;left:0;right:0;bottom:0;background:rgba(0,0,0,.6);display:flex;align-items:center;justify-content:center;opacity:0;transition:opacity .3s;border-radius:50%;color:#fff;font-weight:600;font-size:.9rem}.lazy-load.loaded,.profile-avatar.clickable:hover .avatar-overlay{opacity:1}.profile-header h1{margin:.5rem 0;font-size:2.5rem;color:#000}.profile-email{color:#666;font-size:1.1rem}.profile-joined{color:#666;font-size:.95rem;margin-top:.5rem}body.dark-theme .bar-name,body.dark-theme .chart-label,body.dark-theme .chart-labels,body.dark-theme .comment p,body.dark-theme .comment-action-btn,body.dark-theme .demo-label,body.dark-theme .explanation-text p,body.dark-theme .extreme-info p,body.dark-theme .profile-email,body.dark-theme .profile-joined,body.dark-theme .search-bar svg,body.dark-theme .stat-label,body.dark-theme .summary-label,body.dark-theme .top-tag-stats,body.dark-theme .traffic-percent,body.dark-theme .video-meta,body.dark-theme .video-row-stat{color:#aaa}.dashboard-section,.profile-section{background:#fff;padding:2rem;border-radius:12px;margin-bottom:2rem}.profile-section h2{font-size:1.5rem;margin-bottom:1.5rem;color:#333}.empty-state{color:#999}.empty-hint{font-size:.9rem;color:#bbb;margin-top:.5rem}.countries-list,.interests-list{display:flex;flex-wrap:wrap;gap:.75rem}.interest-tag{padding:.5rem 1rem;background:linear-gradient(135deg,#8e66ea 0,#a24b85 100%);color:#fff;border-radius:20px;font-size:.9rem;font-weight:500}.modal-
```

```
overlay{position:fixed;top:0;left:0;right:0;bottom:0;background-color:rgba(0,0,0,.7);display:flex;justify-
content:center;align-items:center;z-index:1000;overflow-y:auto;padding:2rem}.modal{background-color:#fff;padding:2rem;border-
radius:12px;width:90%;max-width:400px;box-shadow:0 10px 40px rgba(0,0,0,.3)}.upload-modal{max-width:600px;max-
height:90vh;overflow-y:auto}.tags-selector{display:block;max-height:400px;overflow-y:auto;padding:1.5rem;border:1px solid
#ddd;border-radius:8px;background:#f9f9f9;margin-bottom:.5rem}.tag-group-name{font-weight:600;font-
size:.95rem;color:#333;margin-bottom:.75rem;padding-bottom:.5rem;border-bottom:2px solid #eee}.tag-group-
items{display:flex;flex-wrap:wrap;gap:.5rem}.tag-option{padding:.5rem 1rem;border:1px solid #ddd;border-
radius:20px;background:#fff;transition:.2s;font-size:.85rem}.tag-option:hover{background:#f0f0f0;border-color:#999}.tag-
option.selected{background:linear-gradient(135deg,#8e66ea 0,#a24b85 100%);color:#fff;border-color:#8e66ea;font-
weight:500}.modal h2{margin:0 0 1.5rem;color:#000;font-size:1.5rem}.form-group{margin-bottom:1.25rem}.form-group
label{display:block;margin-bottom:.5rem;color:#333;font-weight:500;font-size:.9rem}.form-group input,.form-group
textarea{width:100%;padding:.75rem;border:1px solid #ddd;border-radius:6px;font-size:1rem;font-
family:inherit;transition:border-color .3s}.filter-select:focus,.form-group input:focus,.form-group
textarea:focus{outline:0;border-color:#000}.form-group textarea{resize:vertical;min-height:80px}.form-group
small{display:block;margin-top:.25rem;font-size:.8rem}.error{color:#e53e3e;font-size:.875rem;margin:.5rem 0}.modal-
buttons{display:flex;gap:.75rem;margin-top:1.5rem}.explanation-text,.modal-buttons .btn,.top-user-info{flex:1}@media (max-
width:768px){.container{padding:0 1rem}.nav-buttons{gap:.5rem}.btn{padding:.5rem 1rem;font-size:.875rem}.videos-grid{grid-
template-columns:repeat(auto-fill,minmax(280px,1fr))}.video-stats{flex-direction:column;align-items:flex-
start;gap:1rem}.video-actions{width:100%;justify-content:space-between}.action-btn{flex:1;justify-
content:center}}.container{max-width:1400px;margin:0 auto;padding:0 2rem;display:flex;justify-content:space-between;align-
items:center}img,video{height:auto}.analytics-controls,.nav-buttons{display:flex;gap:1rem;align-
items:center}.btn{padding:.625rem 1.5rem;font-size:.95rem;font-weight:500;border:none;border-
radius:6px;transition:.3s;display:flex;align-items:center;gap:.5rem}.btn:disabled{opacity:.6;cursor:not-allowed}.btn-
profile,.btn-secondary,.btn-upload{background-color:transparent;color:#fff;border:1px solid rgba(255,255,255,.3)}.btn-
profile:hover:not(:disabled),.btn-secondary:hover:not(:disabled),.btn-theme:hover,.btn-
upload:hover:not(:disabled){background-color:rgba(255,255,255,.1);border-color:rgba(255,255,255,.5)}.btn-primary{background-
color:#fff;color:#000;font-weight:600}.btn-theme,body.dark-theme .recommended-info h4{color:#fff}.btn-
primary:hover:not(:disabled){background-color:#f0f0f0;transform:translateY(-1px)}.btn-theme{background-
color:transparent;border:1px solid rgba(255,255,255,.3);padding:.625rem;display:flex;align-items:center;justify-
content:center;width:40px}.btn-theme svg{filter:invert(100%);width:100%;height:auto}.dashboard-page,.home-page{max-
width:1400px;margin:0 auto;padding:2rem}.home-page h1{font-size:2rem;margin-bottom:2rem;color:#000}.empty-
videos{padding:4rem;color:#999;font-size:1.2rem}.videos-grid{display:grid;grid-template-columns:repeat(auto-
fill,minmax(320px,1fr));gap:1.5rem}.video-card{background:#fff;border-radius:12px;overflow:hidden;transition:transform
.2s,box-shadow .2s}.watch-content{display:flex;gap:1.5rem;max-width:1600px;margin:0 auto}.watch-sidebar{width:400px;flex-
shrink:0}.watch-sidebar h3{font-size:1.25rem;margin-bottom:1rem;color:#000}.recommended-meta,.video-
date{color:#666}.recommended-card{gap:.75rem;padding:.5rem;border-radius:8px;cursor:pointer;transition:background
.2s}.recommended-thumbnail{background:#000;flex-shrink:0}.recommended-info h4{color:#000;display:-webkit-box}body.dark-theme
.action-btn.active svg{stroke:#000}.recommended-tags{display:flex;flex-wrap:wrap;margin-top:.5rem}.mini-tag{padding:.15rem
.5rem;background:#e0e0e0;border-radius:10px;font-size:.65rem;color:#555}.no-recommendations{text-
align:center;padding:2rem;color:#999}@media (max-width:1200px){.watch-content{flex-direction:column}.watch-
sidebar{width:100%}.recommended-videos{flex-direction:row;flex-wrap:wrap}.recommended-card{flex-
direction:column;width:calc(50% - .5rem)}.recommended-thumbnail{width:100%;height:120px}}@media (max-
width:640px){.container{padding:0 1rem}.nav-buttons{gap:.5rem}.btn{padding:.5rem 1rem;font-
size:.875rem}.modal{padding:1.5rem}.home-page h1{font-size:2rem}.profile-header h1{font-size:1.75rem}.profile-
page{padding:2rem 1rem}.recommended-card{width:100%}}body.dark-theme{background-color:#0f0f0f;color:#f1f1f1}body.dark-theme
.header{background-color:#212121}body.dark-theme .bar-value,body.dark-theme .chart-section h4,body.dark-theme .chart-
title,body.dark-theme .comment strong,body.dark-theme .countries-section h4,body.dark-theme .dashboard-page h1,body.dark-
theme .dashboard-section h2,body.dark-theme .demo-section h4,body.dark-theme .explanation-text strong,body.dark-theme
.extreme-card h3,body.dark-theme .extreme-info h4,body.dark-theme .form-group label,body.dark-theme .home-page h1,body.dark-
theme .modal h2,body.dark-theme .profile-header h1,body.dark-theme .profile-section h2,body.dark-theme .stat-value,body.dark-
theme .stats-video-info h3,body.dark-theme .top-tag-name,body.dark-theme .traffic-section h4,body.dark-theme .users-stats-
page h1,body.dark-theme .users-table th,body.dark-theme .video-details h1,body.dark-theme .video-info h3,body.dark-theme
.video-row-title strong,body.dark-theme .watch-sidebar h3{color:#f1f1f1}body.dark-theme .comments-section,body.dark-theme
.modal,body.dark-theme .profile-header,body.dark-theme .profile-section,body.dark-theme .video-card,body.dark-theme .video-
details{background-color:#212121;color:#f1f1f1}body.dark-theme .recommended-card:hover,body.dark-theme .video-
card:hover{background-color:#2a2a2a}body.dark-theme .action-btn,body.dark-theme .add-comment input,body.dark-theme .filter-
select,body.dark-theme .form-group input,body.dark-theme .form-group textarea,body.dark-theme .search-input,body.dark-theme
.tag-option{background-color:#2a2a2a;border-color:#444;color:#f1f1f1}body.dark-theme .add-comment input:focus,body.dark-theme
.filter-select:focus,body.dark-theme .form-group input:focus,body.dark-theme .form-group textarea:focus{border-
color:#fff}body.dark-theme .tags-selector{background-color:#1a1a1a;border-color:#444}body.dark-theme .action-
btn:hover,body.dark-theme .tag-option:hover{background-color:#333}body.dark-theme .video-description{background-
color:#1a1a1a}body.dark-theme .action-btn.active{background-color:#f1f1f1;color:#000}.skeleton-card{pointer-
events:none}.skeleton{background:linear-gradient(90deg,#f0f0f0 25%,#e0e0e0 50%,#f0f0f0 75%);background-size:200%
100%;animation:1.5s infinite skeleton-loading;border-radius:8px}body.dark-theme .skeleton{background:linear-
gradient(90deg,#2a2a2a 25%,#333 50%,#2a2a2a 75%);background-size:200% 100%}@keyframes skeleton-loading{0%{background-
position:200% 0}100%{background-position:-200% 0}}.skeleton-thumbnail{width:100%;height:180px}.skeleton-
title{width:80%;height:20px;margin-bottom:.5rem}.skeleton-meta{width:60%;height:14px;margin-bottom:.5rem}.skeleton-
tags{width:90%;height:24px}.search-filters{display:flex;gap:1rem;margin-bottom:2rem;flex-wrap:wrap}.search-bar{flex:1;min-
width:300px;position:relative;display:flex;align-items:center}.search-bar svg{position:absolute;left:1rem;color:#666}.search-
input{width:100%;padding:.75rem 1rem .75rem 3rem;border:1px solid #ddd;border-radius:24px;font-
size:1rem;transition:.3s}.search-input:focus{outline:0;border-color:#000;box-shadow:0 0 0 3px rgba(0,0,0,.1)}body.dark-theme
.search-input:focus{border-color:#fff;box-shadow:0 0 0 3px rgba(255,255,255,.1)}.filters{display:flex;gap:.75rem}.filter-
select{padding:.75rem 1rem;border:1px solid #ddd;border-radius:8px;font-size:.95rem;background:#fff;transition:.3s}.author-
section{display:flex;align-items:center;gap:1rem;margin-bottom:1rem}.btn-subscribe{padding:.5rem
```

```
1.5rem;background:#000;color:#fff;border:none;border-radius:20px;font-weight:600;cursor:pointer;transition:.3s;font-family:inherit;font-size:.9rem}.btn-delete,.btn-watch-later{padding:.5rem 1rem;font-size:.9rem;cursor:pointer;font-family:inherit}.btn-subscribe:hover,body.dark-theme .btn-watch-later:hover,body.dark-theme .comment-action-btn:hover,body.dark-theme .demo-bar,body.dark-theme .traffic-bar{background:#333}.btn-subscribe.subscribed{background:#999}body.dark-theme .btn-subscribe,body.dark-theme .btn-watch-later.active{background:#f1f1f1;color:#000}body.dark-theme .btn-subscribe:hover{background:#ddd}.btn-delete{background:#e53e3e;color:#fff;border:none;border-radius:8px}.btn-delete:hover{background:#c53030}.btn-watch-later{margin-top:1rem;background:#fff;border:1px solid #ddd;border-radius:8px;transition:.3s}body.dark-theme .btn-watch-later{background:#2a2a2a;border-color:#444;color:#f1f1f1}.comment-actions{display:flex;gap:1rem;margin-top:.5rem}.comment-action-btn{padding:.25rem .75rem;background:0 0;border:none;color:#666;cursor:pointer;font-size:.85rem;border-radius:16px;transition:.2s;font-family:inherit}.comment-action-btn:hover,.video-row:not(.video-row-header):hover{background:#f0f0f0}.comment-action-btn.liked{color:#2563eb;font-weight:600}body.dark-theme .author-name.clickable:hover,body.dark-theme .comment-action-btn.liked{color:#60a5fa}.author-with-avatar,.mini-avatar{display:inline-flex;align-items:center}.author-name.clickable{cursor:pointer;transition:color .2s}.author-name.clickable:hover{color:#2563eb;text-decoration:underline}.profile-subscribe{margin-top:1.5rem;padding:.75rem 2rem;font-size:1rem}.author-with-avatar{gap:.5rem}.mini-avatar{width:20px;height:20px;border-radius:50%;background:linear-gradient(135deg,#8e66ea 0,#a24b85 100%);justify-content:center;overflow:hidden;flex-shrink:0}.mini-avatar svg{width:12px;height:12px}.dashboard-page h1,.users-stats-page h1{font-size:2.5rem;margin-bottom:2rem;color:#000}.stats-overview{display:grid;grid-template-columns:repeat(auto-fit,minmax(200px,1fr));gap:1.5rem;margin-bottom:3rem}.stats-grid,.stats-video-info{margin-bottom:2rem}.stat-card{background:#fff;box-shadow:0 2px 8px rgba(0,0,0,.1);display:flex;align-items:center;transition:transform .2s}body.dark-theme .dashboard-section,body.dark-theme .stat-card,body.dark-theme .summary-card,body.dark-theme .users-table-container{background:#212121}.stat-value{color:#000}.stat-label{color:#666;margin-top:.25rem}.dashboard-section h2{font-size:1.5rem;margin-bottom:1.5rem;color:#000}.bar-chart,.traffic-list,.videos-table{display:flex;flex-direction:column;gap:.75rem}.video-row{display:grid;grid-template-columns:120px 2fr 100px 80px 80px 100px 100px;gap:1rem;align-items:center;padding:1rem;background:#f9f9f9;border-radius:8px;transition:background .2s}body.dark-theme .chart-container,body.dark-theme .demo-section,body.dark-theme .extreme-card,body.dark-theme .simple-chart,body.dark-theme .stat-item,body.dark-theme .top-tag-item,body.dark-theme .video-row{background:#1a1a1a}.video-row-header{background:#e0e0e0;font-weight:600;color:#333}body.dark-theme .video-row-header{background:#333;color:#f1f1f1}.video-row-thumbnail video{width:100%;height:70px;border-radius:6px;cursor:pointer}.video-row-title strong{display:block;margin-bottom:.25rem;color:#000}.video-tags-small{display:flex;gap:.25rem;margin-top:.5rem}.tag-small{padding:.15rem .5rem;background:#e0e0e0;border-radius:10px;font-size:.7rem;color:#555}body.dark-theme .interest-more,body.dark-theme .tag-small{background:#333;color:#aaa}.video-row-stat{font-weight:500;color:#333}.video-row-actions{display:flex;gap:.5rem;justify-content:center}.btn-delete-small,.btn-stats{padding:.5rem;background:0 0;border:1px solid #ddd;border-radius:6px;cursor:pointer;font-size:1.2rem;transition:.2s}.simple-chart,.stat-item{background:#f9f9f9;padding:1rem;border-radius:8px}.chart-bar,.line-chart-bar{transition:.3s;cursor:pointer}.btn-stats:hover{background:#2563eb;border-color:#2563eb}.btn-delete-small:hover{background:#e53e3e;border-color:#e53e3e}body.dark-theme .btn-delete-small,body.dark-theme .btn-stats{border-color:#444}.stats-modal{max-width:1000px;max-height:90vh;overflow-y:auto}.stats-video-info h3{margin-top:1rem;color:#000}.stats-grid{display:grid;grid-template-columns:repeat(auto-fit,minmax(150px,1fr))}.stat-big{font-size:2rem;font-weight:700;color:#2563eb;margin-top:.5rem}.chart-section,.countries-section,.traffic-section{margin:2rem 0}.chart-section h4,.countries-section h4,.extreme-card h3,.traffic-section h4{margin-bottom:1rem;color:#000}.simple-chart{display:flex;align-items:flex-end;justify-content:space-around;height:200px;gap:.5rem}.chart-bar{flex:1;background:linear-gradient(180deg,#2563eb 0,#3b82f6 100%);border-radius:4px 4px 0 0;min-height:10px;position:relative;display:flex;align-items:flex-end;justify-content:center}.chart-bar:hover,.line-chart-bar:hover{filter:brightness(1.2)}.chart-label{position:absolute;bottom:-25px;font-size:.7rem;color:#666;white-space:nowrap}.chart-container{background:#f9f9f9;padding:1.5rem;border-radius:8px;margin-top:1rem}.chart-title{font-weight:600;margin-bottom:1rem;color:#000}.line-chart{display:flex;align-items:flex-end;justify-content:space-between;height:150px;gap:2px}.line-chart-bar{flex:1;background:linear-gradient(180deg,#10b981 0,#059669 100%);border-radius:2px 2px 0 0;min-height:2px}.chart-labels{display:flex;justify-content:space-between;margin-top:.5rem;font-size:.75rem;color:#666}.top-tags-list{display:flex;flex-direction:column;gap:1rem;margin-top:1rem}.top-tag-item{display:flex;align-items:center;gap:1rem;padding:1rem;background:#f9f9f9;border-radius:8px;position:relative;overflow:hidden}.top-tag-rank{font-size:1.5rem;font-weight:700;color:#2563eb;min-width:40px}.top-tag-info{flex:1;z-index:1}.top-tag-name{font-weight:600;font-size:1.1rem;margin-bottom:.25rem;color:#000}.top-tag-stats{font-size:.85rem;color:#666}.top-tag-bar{position:absolute;left:0;top:0;bottom:0;background:linear-gradient(90deg,rgba(37,99,235,.1) 0,transparent 100%);transition:width .3s}.demo-section,.extreme-card{padding:1.5rem;background:#f9f9f9;border-radius:8px}.extremes-grid{display:grid;grid-template-columns:repeat(auto-fit,minmax(300px,1fr));gap:1.5rem;margin-top:1rem}.extreme-video{cursor:pointer;transition:transform .2s}.extreme-video:hover{transform:scale(1.02)}.extreme-video video{width:100%;height:150px;border-radius:8px;margin-bottom:1rem}.extreme-info h4{margin-bottom:.5rem;color:#000}.extreme-info p{margin:.25rem 0;color:#666;font-size:.9rem}.demo-section{margin:2rem 0}.demo-section h4{margin-bottom:1.5rem;color:#000}.demo-grid{display:grid;grid-template-columns:repeat(auto-fit,minmax(250px,1fr));gap:2rem}.demo-category h5{margin-bottom:1rem;color:#333;font-size:1rem}body.dark-theme .demo-category h5,body.dark-theme .traffic-name{color:#ddd}.demo-bar-container{margin-bottom:.75rem}.demo-label{display:block;font-size:.85rem;margin-bottom:.25rem;color:#666}.demo-bar{background:#e0e0e0;height:24px;border-radius:12px;overflow:hidden}.demo-bar-fill{background:linear-gradient(90deg,#3b82f6 0,#2563eb 100%);height:100%;display:flex;align-items:center;justify-content:flex-end;padding-right:.5rem;color:#fff;font-size:.75rem;font-weight:600;transition:width .5s}.traffic-item{display:grid;grid-template-columns:150px 1fr 60px;align-items:center;gap:1rem}.traffic-name{font-weight:500;color:#333}.traffic-bar{background:#e0e0e0;height:20px;border-radius:10px;overflow:hidden}.traffic-bar-fill{background:linear-gradient(90deg,#10b981 0,#059669 100%);height:100%;transition:width .5s}.traffic-percent{text-align:right;font-weight:600;color:#666}.centered,.empty-state{text-align:center}.country-badge{padding:.5rem 1rem;background:linear-gradient(135deg,#8e66ea 0,#a24b85 100%);color:#fff;border-radius:20px;font-weight:500;font-size:.9rem}.summary-card,.users-table-container{background:#fff;box-shadow:0 2px 8px rgba(0,0,0,.1)}.users-stats-page{max-width:1600px;margin:0 auto;padding:2rem}.stats-summary{display:grid;grid-template-columns:repeat(auto-fit,minmax(250px,1fr));gap:1.5rem;margin-bottom:3rem}.summary-card{padding:2rem;border-radius:12px;display:flex;align-items:center;gap:1.5rem;transition:transform .2s}.stat-icon-large,.summary-icon{font-size:3rem}.summary-value{font-
```

```
size:2.5rem;font-weight:700;color:#2563eb}.centered,.user-details strong{font-weight:600}.summary-label{font-size:.95rem;color:#666;margin-top:.5rem}.users-table-container{overflow-x:auto}.interest-badge,.table-avatar{background:linear-gradient(135deg,#8e66ea 0,#a24b85 100%);display:flex}.users-table thead{background:#f5f5f5;border-bottom:2px solid #e0e0e0}body.dark-theme .users-table thead{background:#1a1a1a;border-bottom-color:#333}.users-table th{color:#333;font-size:.9rem}.users-table td{border-bottom:1px solid #e0e0e0;color:#333}body.dark-theme .users-table td{border-bottom-color:#333;color:#f1f1f1}.users-table tbody tr:hover{background:#f9f9f9}.table-avatar{width:48px;height:48px;border-radius:50%;align-items:center;justify-content:center;overflow:hidden}.table-avatar svg{width:28px;height:28px;stroke:white}.stat-cell,.tab-btn{font-weight:500}.interests-cell{display:flex;flex-wrap:wrap;gap:.5rem;align-items:center}.interest-badge{white-space:nowrap;justify-content:center;align-items:center}.interest-more{padding:.25rem .5rem;background:#e0e0e0;border-radius:12px;font-size:.75rem;color:#666;cursor:help}.no-interests{color:#999;font-size:.85rem;font-style:italic}.btn-view-profile{padding:.5rem 1rem;background:#2563eb;color:#fff;border:none;border-radius:8px;cursor:pointer;font-family:inherit;font-size:.85rem;transition:.2s;display:inline-flex;align-items:center;gap:.5rem}.btn-view-profile:hover{background:#1d4ed8;transform:translateY(-2px)}@media (max-width:1200px){.users-table,.video-row{font-size:.9rem}.video-row{grid-template-columns:100px 1.5fr 80px 70px 70px 90px 80px;gap:.5rem}.users-table td,.users-table th{padding:.75rem}}.tags-table td,.tags-table th,.users-table td,.users-table th{padding:1rem;border-bottom:1px solid var(--border-color)}@media (max-width:768px){.search-filters{flex-direction:column}.search-bar{min-width:100%}.filters,.video-row-thumbnail{width:100%}.filter-select{flex:1}.author-section{flex-direction:column;align-items:flex-start}.stats-overview,.stats-summary,.video-row{grid-template-columns:1fr}.video-row{gap:.5rem}.video-row-header{display:none}.video-row-thumbnail video{height:150px}.video-row-stat::before{content:attr(data-label);font-weight:600;margin-right:.5rem}.users-table-container{overflow-x:scroll}.users-table{min-width:800px}}.author-avatar-small svg,.comment-avatar svg{height:100%;width:100%;color:var(--text-secondary)}.comment-avatar{width:32px;height:32px;border-radius:50%;overflow:hidden;margin-right:10px;flex-shrink:0}.author-avatar-small,.avatar-img-tiny{width:20px;height:20px;border-radius:50%;margin-right:6px}.comment-header{display:flex;align-items:center;margin-bottom:8px}.comment-author-info,.stat-info,.stat-info-large,.user-details{display:flex;flex-direction:column}.video-author-with-avatar{display:flex;align-items:center;margin-top:8px}.author-avatar-small{overflow:hidden;flex-shrink:0}.thumbnail-container{position:relative;width:100%;height:0;padding-bottom:56.25%;overflow:hidden;background:var(--bg-tertiary)}.recommended-thumbnail-img,.recommended-thumbnail-placeholder,.thumbnail-img,.thumbnail-placeholder,.thumbnail-skeleton{top:0;height:100%;position:absolute;left:0;width:100%}.thumbnail-img{transition:opacity .3s}.thumbnail-placeholder{display:flex;align-items:center;justify-content:center;background:var(--bg-tertiary);color:var(--text-secondary)}.thumbnail-skeleton{background:linear-gradient(90deg,var(--bg-tertiary) 25%,var(--bg-secondary) 50%,var(--bg-tertiary) 75%);background-size:200% 100%;animation:1.5s infinite loading}@keyframes loading{0%{background-position:200% 0}100%{background-position:-200% 0}}.recommended-thumbnail{width:100%;padding-bottom:56.25%;border-radius:8px}.recommended-thumbnail-placeholder{display:flex;align-items:center;justify-content:center;background:var(--bg-tertiary)}video{background:#000}.lazy-load{opacity:0;transition:opacity .3s}.loading-progress{position:absolute;bottom:0;left:0;width:100%;height:3px;background:var(--primary-color);transform-origin:left;animation:2s ease-in-out infinite loading-progress}@keyframes loading-progress{0%{transform:scaleX(0)}50%{transform:scaleX(.5)}100%{transform:scaleX(1)}}.analytics-page{padding:2rem;max-width:1400px;margin:0 auto}.analytics-tabs{display:flex;gap:.5rem;margin-bottom:2rem;border-bottom:1px solid var(--border-color);padding-bottom:1rem}.tab-btn{padding:.75rem 1.5rem;border:none;background:#e4e4e4;border-radius:8px;cursor:pointer;transition:.2s;display:flex;align-items:center;gap:.5rem;margin-top:12px}.tab-btn.active{background:#b4b4b4}.loading-analytics{display:flex;flex-direction:column;align-items:center;justify-content:center;padding:4rem;color:var(--text-secondary)}.loading-spinner{width:50px;height:50px;border:3px solid var(--border-color);border-top-color:var(--primary-color);border-radius:50%;animation:1s linear infinite spin;margin-bottom:1rem}@keyframes spin{to{transform:rotate(360deg)}}.analytics-header{display:flex;justify-content:space-between;align-items:center;margin-bottom:1.5rem}.tags-table-container,.users-table-container{background:var(--bg-secondary);border-radius:12px;overflow:hidden;margin-bottom:2rem;border:1px solid var(--border-color)}.tags-table,.users-table{width:100%;border-collapse:collapse}.tags-table th,.users-table th{background:var(--bg-tertiary);text-align:left;font-weight:600;color:var(--text-secondary)}.tags-table tbody tr:hover,.users-table tbody tr:hover{background:var(--bg-tertiary);cursor:pointer}.tags-table tbody tr:last-child td,.users-table tbody tr:last-child td{border-bottom:none}.bar-chart-item,.user-info{display:flex;align-items:center;gap:1rem}.user-avatar{width:40px;height:40px;border-radius:50%;overflow:hidden;flex-shrink:0}.category-info,.stat-label,.user-details small{font-size:.875rem;color:var(--text-secondary)}.stat-badge{display:inline-block;padding:.25rem .75rem;border-radius:20px;font-size:.875rem;font-weight:600}.stat-badge.subscribers{background:rgba(74,222,128,.2);color:#10b981}.stat-badge.subscriptions{background:rgba(96,165,250,.2);color:#3b82f6}.stat-badge.videos{background:rgba(249,168,212,.2);color:#ec4899}.stat-number{font-weight:600;color:var(--text-primary)}.date-cell,.interest-tag-small,.more-tags{color:var(--text-secondary)}.interests-cell{max-width:200px}.interests-list{display:flex;flex-wrap:wrap;gap:.25rem}.interest-tag-small{background:var(--bg-tertiary);padding:.125rem .5rem;border-radius:12px;font-size:.75rem;white-space:nowrap}.more-tags{font-size:.75rem;padding:.125rem .25rem}.date-cell{font-size:.875rem}.tag-name{min-width:150px}.tag-badge{background:#000;color:#fff;padding:.25rem .75rem;border-radius:20px;font-size:.875rem;font-weight:600}.stats-grid,.users-stats-summary{display:grid;grid-template-columns:repeat(auto-fit,minmax(200px,1fr));gap:1rem;margin-top:2rem}.stat-card,.stat-card-large{border:1px solid var(--border-color);display:flex;background:var(--bg-secondary)}.stat-card{border-radius:12px;padding:1.5rem;align-items:center;gap:1rem}.stat-icon{font-size:2rem}.stat-value{font-size:1.5rem;font-weight:700;color:var(--text-primary)}.stat-card-large{border-radius:16px;padding:2rem;align-items:center;gap:1.5rem}.stat-value-large{font-size:2.5rem;font-weight:800;color:var(--text-primary)}.stat-label-large{font-size:1rem;color:var(--text-secondary)}.tags-charts{display:grid;grid-template-columns:repeat(auto-fit,minmax(500px,1fr));gap:2rem;margin-top:2rem}.chart-section{background:var(--bg-secondary);border:1px solid var(--border-color);border-radius:12px;padding:1.5rem}.chart-section h3{margin-bottom:1rem;color:var(--text-primary)}.bar-label{width:120px;font-size:.875rem;color:var(--text-primary);overflow:hidden;text-overflow:ellipsis;white-space:nowrap}.bar-container{flex:1;height:30px;background:var(--bg-tertiary);border-radius:6px;overflow:hidden;position:relative}.bar-fill{height:100%;border-radius:6px;display:flex;align-items:center;justify-content:flex-end;padding-right:.5rem;transition:width .5s}.bar-value{font-size:.75rem;text-shadow:1px 1px 2px rgba(0,0,0,.3)}.top-tags-cloud{display:flex;flex-wrap:wrap;gap:.75rem;padding:1.5rem;background:var(--bg-tertiary);border-radius:12px}.tag-cloud-item,.top-user-card{align-items:center;display:flex;cursor:pointer}.tag-cloud-
```

```
item{justify-content:center;padding:.5rem 1rem;background:#000;color:#fff;border-radius:20px;font-
weight:600;transition:transform .2s}.empty-state,.rec-tab,.top-user-rank,.top-user-stats{color:var(--text-secondary)}.tag-
cloud-item:hover{transform:scale(1.05)}.top-user-card{gap:1rem;padding:1rem;background:var(--bg-tertiary);border-
radius:12px;transition:background .2s}.recommendations-section,.top-user-card:hover{background:var(--bg-secondary)}.top-user-
rank{font-size:1.5rem;font-weight:800;min-width:40px}.top-user-avatar{width:50px;height:50px;border-
radius:50%;overflow:hidden}.top-user-stats{display:flex;gap:1rem;margin-top:.5rem;font-size:.875rem}.top-user-stats
span{display:flex;align-items:center;gap:.25rem}.empty-state{padding:3rem}.avatar-img-small,.avatar-img-
table{width:40px;height:40px;border-radius:50%}@media (max-width:1200px){.tags-charts{grid-template-
columns:1fr}}.recommendations-section{margin-bottom:3rem;padding:1.5rem;border-radius:16px;border:1px solid var(--border-
color)}.rec-tab,.your-interests{background:var(--bg-tertiary)}.recommendation-tabs{display:flex;gap:.5rem;margin-
bottom:1.5rem;flex-wrap:wrap}.rec-tab{padding:.75rem 1.5rem;border:none;border-radius:12px;cursor:pointer;font-
weight:500;transition:.2s;display:flex;align-items:center;gap:.5rem}.rec-tab:hover{background:var(--primary-color-
light);color:var(--primary-color)}.rec-tab.active{background:var(--primary-color);color:#fff}.your-interests{margin-
bottom:1.5rem;padding:1rem;border-radius:12px}.interest-badge{background:linear-gradient(135deg,var(--primary-
color),#8b5cf6);color:#fff;padding:.5rem 1rem;border-radius:20px;font-size:.875rem;font-weight:600}.more-
interests{padding:.5rem;color:var(--text-secondary);font-size:.875rem}.recommendation-category{margin-top:1.5rem}.category-
header{display:flex;justify-content:space-between;align-items:center;margin-bottom:1rem}.recommended-grid{display:grid;grid-
template-columns:repeat(auto-fill,minmax(300px,1fr));gap:1.5rem}.recommended-video-card{background:var(--bg-primary);border-
radius:12px;overflow:hidden;cursor:pointer;transition:transform .2s,box-shadow .2s;border:1px solid var(--border-
color)}.recommended-video-card:hover{transform:translateY(-4px);box-shadow:0 8px 24px rgba(0,0,0,.15)}.recommended-
thumbnail{position:relative;height:180px;overflow:hidden}.recommended-thumbnail .thumbnail-
container{width:100%;height:100%}.recommended-badge,.similar-badge,.trending-
badge{position:absolute;top:10px;left:10px;background:rgba(0,0,0,.8);color:#fff;width:36px;height:36px;border-
radius:50%;display:flex;align-items:center;justify-content:center;font-size:1.2rem}.growth-rate,.relevance-
score{position:absolute;bottom:10px;right:10px;background:rgba(0,0,0,.8);color:#fff;padding:.25rem .5rem;border-
radius:12px;font-size:.75rem;font-weight:600}.recommended-info h4{margin:0 0 .5rem;font-size:1rem;line-height:1.4;display:-
webkit-box;overflow:hidden}.recommendations-section .recommended-info h4{color:#000!important}.recommended-meta{margin:0 0
.5rem;font-size:.875rem;color:var(--text-secondary)}.recommended-tags{display:flex;flex-wrap:wrap;gap:.25rem;margin-
bottom:.5rem}.tag.matching-tag{background:linear-gradient(135deg,#10b981,#059669);color:#fff}.recommendation-reason,.similar-
reason{font-size:.75rem;color:var(--text-secondary);margin-top:.5rem;line-height:1.4}.trending-stats{display:flex;justify-
content:space-between;align-items:center;font-size:.875rem}.trend-score{background:linear-
gradient(135deg,#f59e0b,#d97706);color:#fff;padding:.25rem .5rem;border-radius:12px;font-weight:600}.empty-
recommendations{padding:3rem;text-align:center;color:var(--text-secondary);background:var(--bg-tertiary);border-
radius:12px}@media (max-width:768px){.analytics-page{padding:1rem}.analytics-tabs{flex-wrap:wrap}.analytics-header{flex-
direction:column;gap:1rem;align-items:flex-start}.analytics-controls{width:100%;flex-direction:column}.tags-table-
container,.users-table-container{overflow-x:auto}.recommended-grid,.stats-grid,.users-stats-summary{grid-template-
columns:1fr}.top-user-stats{flex-direction:column;gap:.25rem}.category-header{flex-direction:column;align-items:flex-
start;gap:.5rem}.recommendation-tabs{justify-content:center}.rec-tab{flex:1;min-width:120px;justify-content:center}}.modal-
buttons .btn{background:#000;color:#fff}.score-breakdown{margin-top:1rem;padding:0;background:rgba(102,126,234,.05);border-
radius:8px;border:1px solid rgba(102,126,234,.1);overflow:hidden}.score-breakdown-header{display:flex;justify-content:space-
between;align-items:center;padding:.75rem;cursor:pointer;user-select:none;transition:background .2s}.score-breakdown-
header:hover{background:rgba(102,126,234,.08)}.score-breakdown-header.expanded{border-bottom:1px solid
rgba(102,126,234,.15)}.breakdown-title{font-size:.85rem;font-weight:600;color:#8e66ea;display:flex;align-
items:center;gap:.5rem}.expand-icon{transition:transform .3s}.expand-icon.rotated{transform:rotate(90deg)}.breakdown-
total{font-size:.9rem;font-weight:700;color:#8e66ea}.score-breakdown-content{padding:.75rem}.score-breakdown-
bars{display:flex;flex-direction:column;gap:.5rem;margin-bottom:.75rem}.score-bar-wrapper{display:flex;flex-
direction:column;gap:.25rem}.score-bar-label{display:flex;justify-content:space-between;font-size:.75rem}.bar-
name{color:#666;font-weight:500}.bar-value{color:#333;font-weight:600}.score-
bar{height:6px;background:rgba(0,0,0,.05);border-radius:3px;overflow:hidden}.score-bar-fill{height:100%;border-
radius:3px;transition:width .3s}.score-bar-fill.context{background:linear-gradient(90deg,#8e66ea,#a24b85)}.score-bar-
fill.social{background:linear-gradient(90deg,#f093fb,#f5576c)}.score-bar-fill.temporal{background:linear-
gradient(90deg,#4facfe,#00f2fe)}.score-bar-fill.affinity{background:linear-gradient(90deg,#43e97b,#38f9d7)}.score-bar-
fill.diversity{background:linear-gradient(90deg,#fa709a,#fee140)}.score-
explanation{display:flex;gap:.75rem;padding:.75rem;background:#fff;border-radius:6px;border:1px solid
rgba(102,126,234,.15);margin-top:.75rem}.explanation-icon{font-size:1.5rem;line-height:1}.explanation-text
strong{display:block;font-size:.85rem;color:#333;margin-bottom:.35rem}.explanation-text p{font-
size:.75rem;color:#666;margin:0 0 .5rem;line-height:1.4}.explanation-text p span{display:inline}.explanation-formula{font-
size:.7rem;color:#999;font-family:'Courier New',monospace;padding:.5rem;background:rgba(0,0,0,.02);border-radius:4px;line-
height:1.6;word-wrap:break-word}.explanation-formula strong{color:#8e66ea;font-weight:700}.score-expand-enter-active,.score-
expand-leave-active{transition:.3s;max-height:500px;overflow:hidden}.score-expand-enter-from,.score-expand-leave-to{max-
height:0;opacity:0}.subscription-badge{position:absolute;top:10px;left:10px;background:linear-
gradient(135deg,gold,#ffed4e);color:#000;width:28px;height:28px;border-radius:50%;display:flex;align-items:center;justify-
content:center;font-size:1rem;box-shadow:0 2px 8px rgba(255,215,0,.4);animation:2s infinite pulse-star}.matching-tag-
more,body.dark-theme .score-breakdown-header:hover{background:rgba(102,126,234,.15)}@keyframes pulse-
star{0%,100%{transform:scale(1)}50%{transform:scale(1.1)}}.matching-tag-more{color:#8e66ea;padding:.25rem .5rem;border-
radius:4px;font-size:.7rem;font-weight:600}body.dark-theme .score-breakdown{background:rgba(102,126,234,.1);border-
color:rgba(102,126,234,.2)}body.dark-theme .score-bar{background:rgba(255,255,255,.05)}body.dark-theme .score-
explanation{background:#1a1a1a;border-color:rgba(102,126,234,.2)}body.dark-theme .explanation-
formula{background:rgba(0,0,0,.3)}
```

main.js

```
import './assets/main.css'

import { createApp } from 'vue'
import App from './App.vue'

createApp(App).mount('#app')
```

ДОДАТОК В

Керівництво користувача

ЗАТВЕРДЖУЮ
Перший проректор
Українського державного
університету науки і технологій
Анатолій РАДКЕВИЧ

MYTUBE
Керівництво користувача
ЛИСТ ЗАТВЕРДЖЕННЯ
44165850.1549 – 01 12 01

Завідувач кафедри КІТ
_____Вадим ГОРЯЧКІН
Керівник розробки
_____Тетяна ГРИШЕЧКІНА
Виконавець
_____Максим ПОПОВ
Нормоконтролер
_____Світлана ВОЛКОВА

2026

ЗАТВЕРДЖЕНО
44165850.1549– 01 12 01

MYTUBE
Керівництво користувача
Листів 9

2026

ВСТУП

Система MyTube – це платформа для завантаження, перегляду та обміну відеоконтентом. Додаток надає можливості соціальної взаємодії між користувачами через коментарі, лайки та підписки. Рекомендаційна система на основі штучного інтелекту автоматично підбирає контент відповідно до інтересів користувача. Система розрахована на користувачів з базовими навичками роботи з веб-застосунками та не вимагає спеціальної технічної підготовки. Перед початком роботи рекомендується ознайомитися з документацією щодо встановлення та налаштування системи.

ПРИЗНАЧЕННЯ ТА УМОВИ ЗАСТОСУВАННЯ

MyTube призначена для автоматизації процесів публікації відеоконтенту, взаємодії між авторами та глядачами, аналізу ефективності контенту та персоналізації рекомендацій. Система працює у веб-браузерах Chrome, Firefox, Safari та Edge останніх версій. Для роботи необхідне стабільне інтернет-з'єднання зі швидкістю не менше 5 Мбіт/с. Користувачі повинні мати активну електронну пошту для реєстрації. Рекомендується екран з роздільною здатністю не менше 1280×720 пікселів для комфортного перегляду інтерфейсу.

ПІДГОТОВКА ДО РОБОТИ

Для початку роботи відкрийте веб-браузер та перейдіть за адресою додатка. При першому відвідуванні система автоматично завантажить необхідні компоненти інтерфейсу. Створіть обліковий запис, натиснувши кнопку "Create Account" та заповнивши форму реєстрації з ім'ям, електронною поштою та паролем. Після успішної реєстрації система запропонує пройти опитування інтересів для налаштування персональних рекомендацій. Перевірте працездатність системи, переглянувши кілька відео з головної сторінки та перевіривши відтворення контенту.

ОПИС ОПЕРАЦІЙ

Перегляд відео: На головній сторінці відображається сітка доступних відео. Натисніть на превью бажаного відео для відкриття сторінки перегляду. Відео автоматично почне відтворюватися після завантаження. Використовуйте стандартні елементи керування плеєра для паузи, регулювання гучності та повноекранного режиму.

Завантаження відео: Натисніть кнопку "Upload" у верхньому меню. Заповніть форму з назвою та описом відео. Оберіть до десяти тегів з запропонованих категорій для класифікації контенту. Натисніть "Вибрати відео" та оберіть файл на вашому комп'ютері. Система автоматично завантажить та оптимізує відео. Після завершення завантаження відео з'явиться у вашому профілі. Взаємодія з контентом: Під відеоплеєром розташовані кнопки лайків та дизлайків для оцінки контенту. Натискання кнопки "Share" копіює посилання на відео у буфер обміну. Кнопка "Watch Later" додає відео до списку відкладеного перегляду. Для підписки на автора натисніть кнопку "Підписатись" під відео.

Коментування: У розділі коментарів під відео введіть текст у поле "Додати коментар" та натисніть "Відправити". Ваш коментар з'явиться у загальному списку з позначкою часу публікації. Натисніть для лайка чужого коментаря або "Відповісти" для цитування.

Персоналізація рекомендацій: При першому вході система запропонує опитування про ваші інтереси. Оберіть категорії контенту, що вас цікавлять, натискаючи на відповідні теги. Система проаналізує ваші вподобання та лайки для автоматичного формування персональної стрічки. У профілі можна редагувати список інтересів, видаляючи непотрібні теги у режимі редагування.

Аналітика: Автори можуть переглядати статистику своїх відео через розділ Dashboard. Клацніть на кнопку біля відео для перегляду детальної аналітики з графіками переглядів, демографією аудиторії та джерелами трафіку. Розділ

Analytics надає загальну статистику платформи з рейтингами користувачів та популярних тегів.

АВАРІЙНІ СИТУАЦІЇ

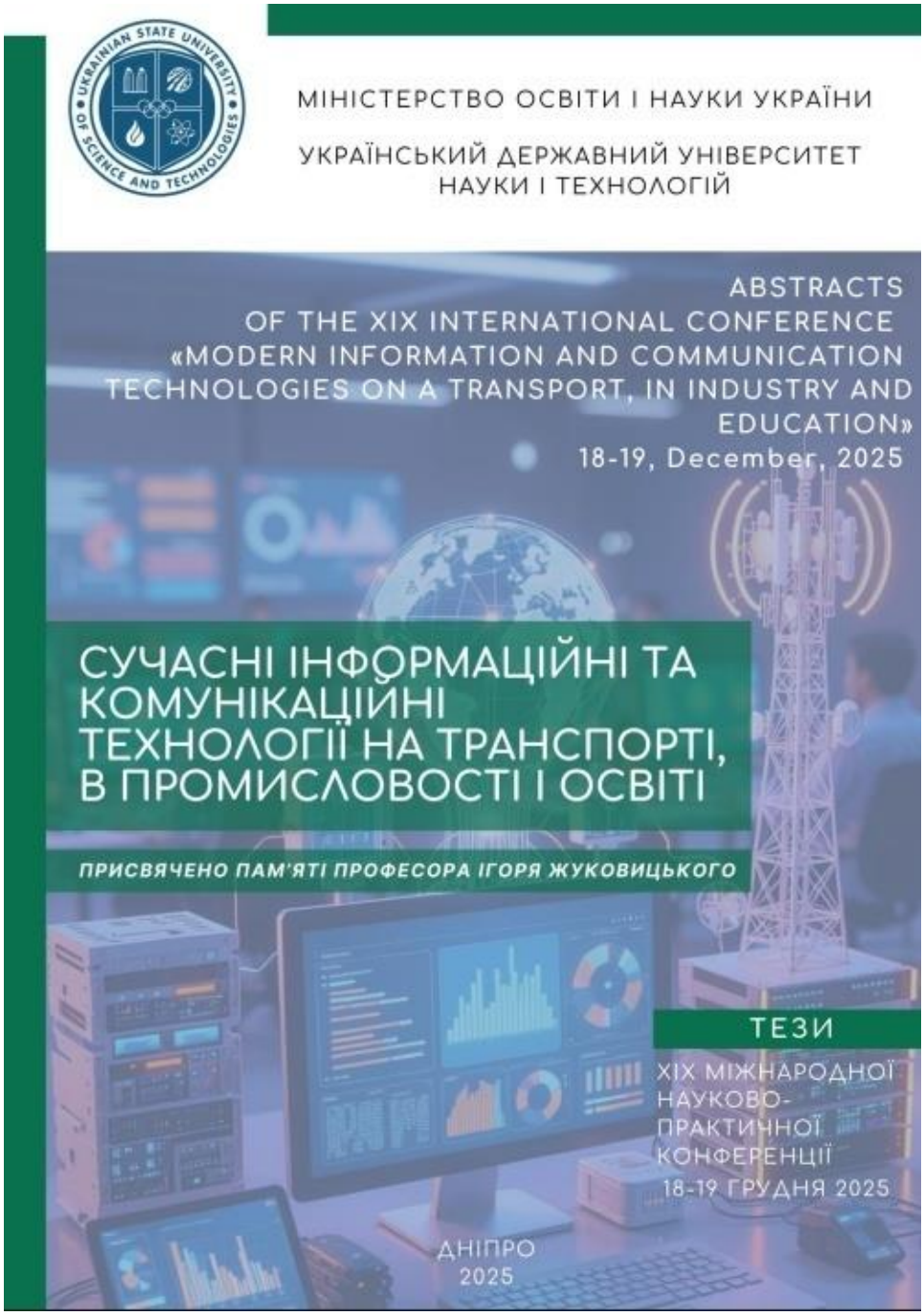
При виникненні помилок завантаження відео перевірте якість інтернет-з'єднання та формат файлу. Система підтримує MP4, MOV, AVI та WebM з максимальним розміром 100 МБ. Якщо відео не відтворюється, спробуйте оновити сторінку або очистити кеш браузера. При тривалих технічних проблемах дані про ваші перегляди та лайки зберігаються у базі даних Firebase та будуть відновлені після усунення несправності. У разі виявлення недопустимого контенту використовуйте кнопку видалення власних відео або зверніться до адміністрації щодо контенту інших користувачів.

РЕКОМЕНДАЦІЇ ЩОДО ЗАСВОЄННЯ

Для ефективного освоєння системи рекомендується почати з перегляду кількох відео різних категорій та оцінки їх лайками. Це допоможе алгоритму рекомендацій краще зрозуміти ваші вподобання. Спробуйте завантажити тестове відео тривалістю до хвилини з чіткою назвою та релевантними тегами. Додайте кілька коментарів під чужими відео для практики взаємодії зі спільнотою. Перегляньте розділ Dashboard після завантаження відео для ознайомлення з інструментами аналітики. Експериментуйте з різними комбінаціями тегів при завантаженні контенту для оптимізації охоплення аудиторії. Регулярне використання системи протягом тижня допоможе повністю освоїти всі функціональні можливості платформи.

ДОДАТОК Г

ТЕЗИ



Сторінка 130

Рекомендаційні системи в динамічному середовищі: підходи до оцінювання

Попов М.С., Український державний університет науки і технологій (УДУНТ), Україна

Уподобання користувачів щодо контенту з часом змінюються, оскільки з'являється новий вибір. Аналогічно, схильності користувачів розвиваються, що призводить до постійного переосмислення їхнього смаку [1].

Оцінювання рекомендаційних систем зазвичай проводиться з використанням кінцевих наборів даних. Це означає, що традиційні методології оцінювання застосовні лише в офлайн-експериментах, де дані та моделі стаціонарні. Однак у реальних системах зворотний зв'язок від користувачів генерується постійно з непередбачуваною швидкістю [2].

Критерії оцінки якості рекомендаційних систем можуть виходити далеко за межі класичної точності прогнозування. Сучасні підходи враховують суб'єктивне сприйняття системи користувачем: новизну рекомендацій, їхню різноманітність та навіть здатність приємно дивувати. Найбільш об'єктивним критерієм є реальна поведінка користувача - кліки, перегляди чи покупки, отримані через A/B-тестування. Таким чином, ефективна рекомендаційна система має задовольняти користувача комплексно, пропонуючи релевантний, різноманітний та корисний контент.

Для ефективного моніторингу систем можна розглянути кілька рішень. По-перше, впроваджувати візуалізацію метрик у вигляді часових графіків, які показують продуктивність по тижнях чи місяцях. Особливо корисними є графіки активності користувачів та розподілу оцінок, які показують зміни задоволеності з часом. Теплові карти візуалізують сезонні паттерни - наприклад, зростання інтересу до певних категорій у конкретні місяці. По-друге, використовувати механізми часової уваги, які враховують повторювані паттерни поведінки користувачів. Дашборди з інтерактивними графіками дозволяють швидко виявляти аномалії та падіння ефективності.

Отже, ефективні рекомендаційні системи потребують комплексного підходу до оцінювання, який виходить за межі традиційних метрик точності. Візуалізація часових трендів, моніторинг поведінкових патернів та безперервна адаптація до змін у вподобаннях користувачів є ключовими факторами успіху. Лише інтегруючи часову динаміку, різноманітні критерії оцінки та інструменти аналітики, можна створити системи, які залишаються релевантними та корисними для користувачів у довгостроковій перспективі.

1. **Koren, Y.** Collaborative filtering with temporal dynamics / Y. Koren // Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '09). – Paris, France, 2009. – P. 447–456. – DOI: 10.1145/1557019.1557072. – URL: https://dl.acm.org/doi/10.1145/1557019.1557072 (дата звернення: 10.12.2024).

2. **Vinagre, J.** Evaluation of recommender systems in streaming environments / J. Vinagre, A. M. Jorge, J. Gama // Workshop on Recommender Systems Evaluation: Dimensions and Design (REDD 2014), held in conjunction with RecSys 2014. – Silicon Valley, USA, 2014. – arXiv:1504.08175. – URL: https://www.researchgate.net/publication/265913731_Evaluation_of_recommender_systems_in_streaming_environments (дата звернення: 10.12.2024).