Київський столичний університет імені Бориса Грінченка
Факультет інформаційних технологій та математики
Кафедра комп’ютерних наук і математики

Допущено до захисту
Завідувач кафедри комп’ютерних наук
доктор технічних наук, професор
Андрій БОНДАРЧУК
« 01 » червня 2026 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня бакалавр
спеціальність 122 Комп’ютерні науки
освітня програма 122.00.01 Інформатика

Тема: ІНТЕРАКТИВНИЙ ДОДАТОК АНАЛІТИЧНОЇ ОБРОБКИ КОНТЕНТУ СОЦІАЛЬНИХ МЕРЕЖ

Виконала
Студентка групи ІНб-1-22-4.0д
Неліпа Вікторія Андріївна

_____
(підпис)

Науковий керівник
к.т.н., доцент,
Мельник І. Ю.

_____
(підпис)

Київ – 2026

Київський столичний університет імені Бориса Грінченка
Факультет інформаційних технологій та математики
Кафедра комп'ютерних наук і математики

«Затверджую»
Завідувач кафедри комп'ютерних наук
доктор технічних наук, професор
Андрій БОНДАРЧУК
«31» жовтня 2025 р.

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА
«Інтерактивний додаток аналітичної обробки контенту соціальних мереж»

Виконавець – студентка групи ІНб-1-22-4.0д,
спеціальності 122 Комп'ютерні наук,
освітньої програми 122.00.01 Інформатика
Неліпа Вікторія Андріївна

1. Вихідні дані: наукові публікації та технічна документація з питань отримання, обробки та аналізу цифрового контенту платформ YouTube і TikTok; офіційна документація до програмних засобів для розробки веб-додатків та роботи з базами даних.
2. Основні завдання: провести аналіз предметної галузі та дослідити особливості платформ YouTube і TikTok як джерел цифрового контенту; виконати огляд програмних засобів-аналогів та визначити вимоги до системи; спроектувати архітектуру веб-додатка, структуру бази даних та інтерфейс; розробити веб-додаток для автоматизованого отримання, завантаження, збереження, конвертації та аналітичної обробки контенту; здійснити тестування розробленої системи та оформити пояснювальну записку.
3. Пояснювальна записка: Обсяг близько 60 сторінок формату А4 комп'ютерного набору з дотриманням вимог стандартів і методичних рекомендацій кафедри.
4. Графічні матеріали: структурна схема системи, схема бази даних, знімки екрана веб-додатка.
5. Додатки: лістинг програмного коду.
6. Строк подання роботи на кафедру: «01» червня 2026 р.

Науковий керівник
к.т.н., доцент
_____ Мельник І.Ю.
«31» жовтня 2025 р.

Виконавець:
_____ Неліпа В.А.
«31» жовтня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

1	Затвердження теми кваліфікаційної роботи бакалавра	31.10.25	Виконано
2	Аналіз проблеми, вивчення аналогів, формування цілей і завдань	01.11.25 – 11.01.26	Виконано
3	Аналіз предметної галузі та існуючих засобів обробки контенту соціальних мереж	26.01.26 – 15.03.26	Виконано
4	Дослідження особливостей отримання даних із платформ YouTube та TikTok	16.03.26 – 31.03.26	Виконано
5	Розробка функціональної структури, архітектури веб-додатка та структури бази даних	01.04.26 – 15.04.26	Виконано
6	Тестування функцій отримання, відображення та завантаження контенту	16.04.26 – 30.04.26	Виконано
7	Впровадження розробленої системи та оцінка її працездатності на реальних даних	01.05.26 – 15.05.26	Виконано
8	Підготовка пояснювальної записки, графічних матеріалів, додатків	25.05.26 – 15.06.26	Виконано
9	Захист кваліфікаційної роботи	15.06.26	

«Затверджую»
Науковий керівник
к.п.н., доцент, Мельник І.Ю.

Індивідуальний план склала
Неліпа В.А.

РЕФЕРАТ

Кваліфікаційна робота: 56 с., 18 рис., 10 табл., 40 посилань.

Актуальність. Стрімке зростання обсягів мультимедійного контенту на платформах YouTube і TikTok зумовлює потребу в програмних засобах, здатних забезпечити комплексне отримання, збереження, обробку та аналіз цифрових матеріалів у межах єдиної системи. Існуючі веб-сервіси орієнтовані на виконання окремих операцій і не забезпечують повного циклу роботи з контентом обох платформ, що підтверджує актуальність теми дослідження.

Об'єкт дослідження: інтерактивний веб-додаток для отримання, завантаження, збереження, конвертації та аналітичної обробки мультимедійного контенту соціальних платформ.

Предмет дослідження: методи, програмні засоби та технології розробки інтерактивного веб-додатка для завантаження, конвертації та аналітичної обробки контенту платформ YouTube і TikTok.

Мета роботи: розробка інтерактивного веб-додатка для отримання, завантаження, збереження, конвертації та аналітичної обробки контенту платформ YouTube і TikTok.

Завдання:

1. Провести аналіз предметної галузі та дослідити особливості платформ YouTube і TikTok як джерел цифрового контенту.
2. Виконати огляд програмних засобів-аналогів та визначити вимоги до системи; спроектувати архітектуру веб-додатка, структуру бази даних та інтерфейс.
3. Розробити веб-додаток для автоматизованого отримання, завантаження, збереження, конвертації та аналітичної обробки контенту.
4. Здійснити тестування розробленої системи.

Основні результати. Одержані результати можуть бути використані в навчальній діяльності, при підготовці інформаційних матеріалів, у роботі з

мультимедійними ресурсами, а також як основа для подальшого розвитку систем аналітичної обробки цифрового контенту.

Практичне значення: одержані результати можуть бути використані в навчальній діяльності, при підготовці інформаційних матеріалів, у роботі з мультимедійними ресурсами, а також як основа для подальшого розвитку систем аналітичної обробки цифрового контенту.

Ключові слова: веб-додаток, соціальні мережі, мультимедійний контент, YouTube, TikTok, метадані, конвертація, REST API, Node.js, React.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

AAC (Advanced Audio Coding) – формат стиснення аудіо з втратами, що забезпечує високу якість звуку

API (Application Programming Interface) – програмний інтерфейс застосунку для взаємодії між компонентами програмного забезпечення

AVI (Audio Video Interleave) – формат мультимедійного контейнера для зберігання відео та аудіо даних

БД (База даних) – організована сукупність структурованих даних, що зберігається в електронному вигляді

CRF (Constant Rate Factor) – параметр якості відеокодування, що визначає співвідношення між якістю та розміром файлу

CSS (Cascading Style Sheets) – каскадні таблиці стилів для оформлення вигляду веб-сторінок

HTTP (HyperText Transfer Protocol) – протокол передачі гіпертексту, що використовується для обміну даними в мережі Інтернет

I/O (Input/Output) – введення/виведення даних між програмою та зовнішніми пристроями або файловою системою

JSON (JavaScript Object Notation) – текстовий формат обміну даними, зручний для читання людиною та машинної обробки

MP3 (MPEG Audio Layer III) – широко розповсюджений формат стиснення аудіо з втратами

MP4 (MPEG-4 Part 14) – універсальний формат мультимедійного контейнера для зберігання відео, аудіо та субтитрів

REST (Representational State Transfer) – архітектурний стиль проєктування API для розподілених систем

SCSS (Sassy CSS) – розширений препроцесор каскадних таблиць стилів із підтримкою змінних та вкладеності

SPA (Single Page Application) – односторінковий веб-застосунок, що динамічно оновлює вміст без перезавантаження сторінки

SQLite (Structured Query Language lite) – легковагова вбудована реляційна система управління базами даних, що не потребує окремого сервера

SSE (Server-Sent Events) – технологія однонаправленої передачі серверних подій клієнту у реальному часі

UI (User Interface) – інтерфейс користувача, що забезпечує взаємодію людини з програмним застосунком

URL (Uniform Resource Locator) – уніфікований локатор ресурсів для визначення адреси об'єкта в мережі Інтернет

WAV (Waveform Audio File Format) – формат аудіофайлу без стиснення, що забезпечує високу якість звучання

WebM (Web Media) – відкритий мультимедійний формат для веб, оптимізований для потокової передачі відео в браузері

ЗМІСТ

ВСТУП

Умови сучасного цифрового середовища характеризуються стрімким зростанням обсягів мультимедійного контенту, що поширюється через соціальні мережі та відеоплатформи. Особливе місце в цьому процесі посідають платформи YouTube і TikTok, на яких щоденно публікується значна кількість відео- та аудіоматеріалів, доступних для перегляду, поширення, аналізу й подальшого використання. Такий контент активно застосовується в інформаційно-комунікаційній, освітній та прикладній діяльності, що зумовлює потребу у програмних засобах, здатних забезпечити його оперативне отримання, збереження, обробку та впорядковане використання.

Зі збільшенням обсягів цифрового контенту зростають вимоги до швидкості доступу до нього, можливості вибору формату та якості, подальшої конвертації, а також до зручності роботи з супровідними метаданими. Водночас сучасні платформи ускладнюють механізми доступу до контенту й пов’язаних із ним відомостей, що обмежує можливості централізованої роботи з мультимедійними матеріалами в межах єдиного програмного середовища. За таких умов актуальним є створення веб-додатка, який поєднуватиме функції отримання інформації про контент, його завантаження, збереження, конвертації та аналітичного опрацювання.

Існуючі сервіси та програмні засоби, призначені для роботи з мультимедійним контентом, здебільшого орієнтовані на виконання окремих операцій, зокрема завантаження файлів, отримання метаданих або перетворення форматів. Така фрагментарність функціоналу ускладнює організацію цілісного процесу роботи з контентом і знижує практичну ефективність використання подібних рішень. У зв’язку з цим виникає потреба у створенні інтерактивного веб-додатка, який забезпечуватиме комплексну взаємодію з контентом платформ YouTube і TikTok у межах єдиної системи.

Метою кваліфікаційної роботи є розробка інтерактивного веб-додатка для отримання, завантаження, збереження, конвертації та аналітичної обробки контенту платформ YouTube і TikTok.

Для досягнення поставленої мети необхідно вирішити такі завдання:

Завдання:

1. Провести аналіз предметної галузі та дослідити особливості платформ YouTube і TikTok як джерел цифрового контенту.
2. Виконати огляд програмних засобів-аналогів та визначити вимоги до системи; спроектувати архітектуру веб-додатка, структуру бази даних та інтерфейс.
3. Розробити веб-додаток для автоматизованого отримання, завантаження, збереження, конвертації та аналітичної обробки контенту.
4. Здійснити тестування розробленої системи.

Об’єктом дослідження є інтерактивний веб-додаток для отримання, завантаження, збереження, конвертації та аналітичної обробки мультимедійного контенту соціальних платформ.

Предметом дослідження є методи, програмні засоби та технології розробки інтерактивного веб-додатка для завантаження, конвертації та аналітичної обробки контенту платформ YouTube і TikTok.

Методологічну основу дослідження становлять загальнонаукові та спеціальні методи аналізу, синтезу, узагальнення, порівняння, проектування та моделювання, застосування яких дало змогу дослідити особливості предметної галузі, проаналізувати наявні програмні рішення, визначити вимоги до системи, обґрунтувати вибір технологій і забезпечити реалізацію програмного засобу відповідно до поставлених завдань. Для перевірки працездатності веб-додатка використано методи функціонального тестування програмного забезпечення.

Практичне значення одержаних результатів полягає у створенні веб-додатка, придатного до автоматизованої роботи з контентом платформ YouTube і TikTok, зокрема до отримання інформації про мультимедійні матеріали, їх завантаження, вибору доступних форматів і параметрів якості, подальшої конвертації та відображення в зручному користувацькому інтерфейсі. Одержані результати можуть бути використані в навчальній діяльності, під час підготовки інформаційних матеріалів, у роботі з мультимедійними ресурсами, а також як основа для подальшого розвитку систем подібного призначення.

РОЗДІЛ 1

АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ІСНУЮЧИХ ЗАСОБІВ ОБРОБКИ КОНТЕНТУ СОЦІАЛЬНИХ МЕРЕЖ

1.1 Сутність контенту соціальних мереж як об’єкта аналізу

У сучасному цифровому середовищі контент соціальних мереж належить до основних форм інформаційної взаємодії, комунікації та поширення мультимедійних даних. Соціальні платформи надають користувачам можливість створювати, публікувати, поширювати й коментувати різні типи матеріалів, зокрема текстові повідомлення, зображення, аудіозаписи, відео та комбіновані мультимедійні публікації. У науковому та прикладному аспектах такий контент доцільно розглядати як різновид користувацького контенту, що формується в мережевому середовищі, відображає творчий внесок користувачів і виникає поза межами традиційних професійних практик медіавиробництва. Саме такий підхід до тлумачення користувацького контенту (user-created content) запропоновано OECD, яка визначає його як контент, відкрито доступний через Інтернет, такий, що містить елемент творчої діяльності й створюється поза професійними процедурами виробництва медіа [1].

Контент соціальних мереж слід розглядати не просто як інформаційне повідомлення, а як структурований цифровий об’єкт (рисунок 1.1)[1, 4, 5], що поєднує мультимедійний матеріал і сукупність супровідних характеристик. У практиці програмного аналізу значення має як сам медіафайл, так і пов’язані з ним відомості: назва, опис, дата публікації, автор, тривалість, тематичні ознаки, кількісні показники взаємодії аудиторії, а також технічні параметри відображення чи вбудовування. З огляду на це контент соціальних мереж доцільно визначати як комплексний інформаційний об’єкт, придатний до дослідження, систематизації, збереження та подальшого використання в системах автоматизованої обробки даних [2, 3].

Узагальнену структуру контенту соціальних мереж як об’єкта аналізу наведено на рисунку 1.1.

Рисунок 1.1. Структура контенту соціальних мереж як об’єкта аналізу

Як видно з рисунка 1.1, контент соціальних мереж доцільно розглядати як поєднання мультимедійного об’єкта, описових метаданих, технічних параметрів і показників взаємодії аудиторії, сукупність яких створює підґрунтя для автоматизованого отримання, збереження, опрацювання та подальшого аналізу цифрового контенту в межах веб-додатка.

Особливий інтерес у цьому дослідженні становить контент платформ YouTube і TikTok, оскільки саме вони належать до найбільш поширених відеоорієнтованих платформ і формують значний масив мультимедійних даних, доступних для перегляду, аналізу та технічної обробки. За даними Pew Research Center, YouTube залишається однією з найуживаніших платформ: її використовують 83% дорослих у США, тоді як TikTok використовують 33%. Серед підлітків YouTube також посідає провідні позиції, а 73% опитаних зазначили щоденне використання цієї платформи, тоді як TikTok щоденно використовують близько шести з десяти респондентів [4, 5]. Хоча наведені показники не претендують на універсальний глобальний вимір, вони

переконливо відображають домінування відеоорієнтованого соціального контенту в сучасному цифровому середовищі та обґрунтовують вибір саме цих платформ як об’єкта дослідження.

Важливою особливістю контенту платформ YouTube і TikTok є наявність структурованих метаданих, що супроводжують мультимедійний матеріал і забезпечують можливість його автоматизованого опрацювання. Поряд із самим відеоконтентом ці платформи містять відомості про час публікації, назву, опис, тривалість, засоби візуального представлення та показники користувацької взаємодії, що дає підстави визначати контент YouTube і TikTok як комплексний інформаційний об’єкт, придатний до систематизації, фільтрації, сортування, порівняльного аналізу, збереження та подальшого відображення у веб-додатку.

Контент соціальних мереж як об’єкт аналізу характеризується сукупністю властивостей, що визначають особливості його дослідження та подальшого опрацювання в програмних системах, серед яких провідне місце посідають динамічність, мультимодальність, платформозалежність і наявність кількісних показників користувацької взаємодії. Динамічний характер такого контенту виявляється в його постійному оновленні, доповненні новими публікаціями та зміні під впливом активності користувачів, тоді як мультимодальність зумовлена поєднанням у межах однієї публікації відеоряду, аудіосупроводу, текстового опису, прев’ю-зображень та елементів інтерактивної взаємодії. Платформозалежність проявляється у відмінностях способу подання матеріалу, складу доступних метаданих і механізмів отримання інформації, що визначаються функціональними можливостями конкретного сервісу, а кількісні показники взаємодії аудиторії, зокрема перегляди, вподобання, коментарі та поширення, дозволяють розглядати такий контент як інформаційний об’єкт із вираженими поведінковими характеристиками. Сукупність зазначених ознак зумовлює придатність контенту соціальних мереж до аналітичної обробки в межах інтерактивних програмних систем.

Під час проектування веб-додатка контент соціальних мереж доцільно розглядати як сукупність двох взаємопов’язаних складових: мультимедійного об’єкта та набору його метаданих. Мультимедійний об’єкт виступає основою для завантаження, збереження та конвертації, тоді як метадані забезпечують ідентифікацію, класифікацію, відображення та подальший аналіз контенту. Такий підхід формує методологічну основу побудови системи, у якій процеси отримання інформації, збереження результатів, роботи з базою даних і реалізації користувацького інтерфейсу підпорядковуються єдиній логіці обробки цифрового контенту соціальних платформ. Отже, контент соціальних мереж у цьому дослідженні розглядається як складний цифровий об’єкт, що поєднує медіадані, описові характеристики та показники взаємодії, а відтак може бути повноцінним об’єктом програмного аналізу, збереження та подальшого використання.

Визначені особливості контенту соціальних мереж як об’єкта аналізу зумовлюють потребу в окремому розгляді способів отримання даних із платформ YouTube і TikTok, що й становить предмет наступного підрозділу.

1.2 Особливості отримання даних із платформ YouTube та TikTok

Отримання даних із платформ YouTube та TikTok становить одну з ключових складових побудови веб-додатка, орієнтованого на збір, збереження та обробку мультимедійного контенту, оскільки в цьому випадку предметом доступу виступає не сам відеоматеріал у відриві від супровідної інформації, а пов’язаний із ним комплекс відомостей, до складу якого входять метадані, технічні характеристики та показники взаємодії аудиторії. Попри належність YouTube і TikTok до відеоорієнтованих соціальних платформ, механізми доступу до контенту на них істотно відрізняються, що позначається на принципах побудови програмної системи, призначеної для роботи з такими джерелами інформації.

Для платформи YouTube характерний більш упорядкований і передбачуваний механізм отримання даних, зумовлений наявністю офіційного

програмного інтерфейсу, призначеного для роботи з відомостями про відео, результатами пошуку, каналами та іншими пов’язаними об’єктами; за таких обставин доступ до частини публічних даних є відносно простим і не потребує складної взаємодії з користувачем. Для TikTok, натомість, типовим є більш обмежений режим доступу, регламентований правилами платформи, системою дозволів і порядком авторизації, у зв’язку з чим отримання інформації потребує ретельнішого врахування платформених умов і технічних обмежень.

Спільною рисою YouTube і TikTok є супровід мультимедійного матеріалу сукупністю відомостей, до яких належать назва або опис відео, дата публікації, тривалість, зображення-прев’ю, а також показники взаємодії користувачів, зокрема перегляди, вподобання, коментарі та поширення; наявність такого інформаційного набору визначає придатність контенту цих платформ до збереження, фільтрації, сортування, систематизації та подальшого відображення у веб-додатку.

Найбільш виразні відмінності між YouTube і TikTok виявляються в аспекті обмежень доступу до даних, адже для YouTube суттєвим чинником є квотування запитів, що впливає на інтенсивність використання програмного інтерфейсу та організацію роботи із сервісом, тоді як для TikTok визначального значення набувають вимоги, пов’язані з авторизацією, правилами доступу та погодженням використання інструментів розробника. У зв’язку з цим під час проектування веб-додатка необхідно враховувати як склад доступних метаданих, так і загальні умови взаємодії з кожною платформою, від яких залежить побудова підсистеми збору інформації.

Отже, YouTube і TikTok, зберігаючи спільну відеоорієнтовану природу, відрізняються за рівнем відкритості доступу до даних, складністю авторизації та умовами використання інструментів розробника, а врахування цих відмінностей належить до необхідних передумов побудови підсистеми збору інформації та подальшого проектування веб-додатка.

Порівняльну характеристику платформ YouTube та TikTok наведено в таблиці 1.1.

Таблиця 1.1

Порівняльна характеристика платформ YouTube та TikTok

Характеристика	**YouTube**	**TikTok**
Тип платформи	Відеохостинг із широким спектром контенту	Короткі та вертикальні відео, орієнтовані на алгоритм
Офіційний API	YouTube Data API v3 (квотування запитів)	Обмежений доступ, авторизація через акаунт
Метадані відео	Заголовок, опис, дата, тривалість, канал, теги, категорія, перегляди, вподобання	Заголовок, автор, перегляди, лайки, коментарі (обмежено)
Доступ до підписників	Кількість підписників каналу доступна через API	Кількість підписників доступна через неофіційні засоби
Завантаження без водяного знака	Водяний знак відсутній	Офіційний контент містить водяний знак платформи
Обмеження доступу	Квоти API, блокування автоматизованих запитів	Сувора авторизація, правила доступу, ліміти
Підтримка yt-dlp	Повна підтримка, активне оновлення	Підтримка через неофіційний API-хост

1.3 Аналіз існуючих веб-сервісів для отримання та обробки контенту соціальних мереж

Отримання та обробка контенту соціальних мереж у сучасних умовах здійснюються не лише за допомогою окремих програмних інструментів, а й через веб-сервіси, призначені для завантаження мультимедійних файлів без встановлення додаткового програмного забезпечення. Переважно такі сервіси реалізують спрощений сценарій взаємодії, за якого користувач вводить посилання на матеріал, обирає доступний формат або якість і отримує готовий файл для подальшого збереження. Подібний принцип організації роботи покладено і в основу веб-додатків цього типу, у зв’язку з чим аналіз

відповідних рішень дає можливість визначити вже реалізовані функціональні можливості, їх переваги та обмеження.

Серед веб-сервісів, орієнтованих на роботу з контентом платформ YouTube і TikTok, простежуються два основні підходи: використання більш універсальних рішень, що поєднують функції завантаження та конвертації, і застосування вузькоспеціалізованих сервісів, зосереджених переважно на одній платформі. У цьому контексті доцільним є розгляд веб-сервісу Y2Down, який поєднує функції завантаження та конвертації контенту, і веб-сервісу SnapTik, орієнтованого на швидке завантаження відео з TikTok без водяного знака. Зіставлення цих веб-сервісів дозволяє простежити відмінності у підходах до організації рішень такого типу та сформувати підґрунтя для подальшого визначення вимог до веб-додатка.

1.3.1 Аналіз веб-сервісу Y2Down

Y2Down є веб-сервісом, призначеним для завантаження та конвертації мультимедійного контенту, насамперед із платформи YouTube. Принцип його роботи ґрунтується на взаємодії через веб-інтерфейс: користувач вводить URL-адресу матеріалу, обирає формат збереження та ініціює завантаження без потреби в реєстрації чи встановленні додаткового програмного забезпечення. Заявлений функціонал охоплює перетворення відео у формати MP3, MP4, WEBM та інші, а також вибір роздільної здатності в діапазоні від 144p до 4K і 8K. Окремо декларується підтримка плейлистів, каналів і роботи з великою кількістю сайтів, хоча функціональна спрямованість ресурсу переважно пов’язана саме з YouTube.

До основних функціональних можливостей Y2Down належать:

1. Завантаження мультимедійного контенту за посиланням через веб-інтерфейс;
2. Конвертація відео в аудіо- та відеоформати;
3. Вибір доступної якості завантаження;
4. підтримка роботи з плейлистами та каналами;

5. використання без встановлення окремого програмного забезпечення.

Суттєвими перевагами Y2Down є простота використання, широкий вибір форматів і наявність функції конвертації в межах одного веб-середовища. Такий підхід спрощує роботу користувача, оскільки завантаження та перетворення контенту відбуваються без переходу до сторонніх інструментів. Практичну цінність становить і підтримка плейлистів, а також заявлена багатоплатформність, що розширює сферу можливого застосування сервісу. Разом із тим окремі характеристики Y2Down подаються в виразно маркетинговій формі, тому його доцільно розглядати насамперед як приклад веб-рішення з акцентом на зручність використання та конвертацію контенту.

До основних обмежень Y2Down належать:

1. Виражена орієнтація переважно на платформу YouTube;
2. Відсутність чітко окресленого механізму роботи з метаданими на рівні користувацького інтерфейсу;
3. Подання частини характеристик у рекламному стилі, що ускладнює об’єктивну оцінку стабільності та повноти функціоналу.

Отже, Y2Down (рисунок 1.2)[6] є показовим прикладом веб-сервісу, що поєднує функції завантаження та конвертації мультимедійного контенту в межах одного інтерфейсу. До його сильних сторін належать простота взаємодії, підтримка різних форматів і можливість вибору якості, тоді як основним обмеженням залишається виражена орієнтація на YouTube, що звужує придатність цього рішення для комплексної роботи з контентом різних соціальних платформ.

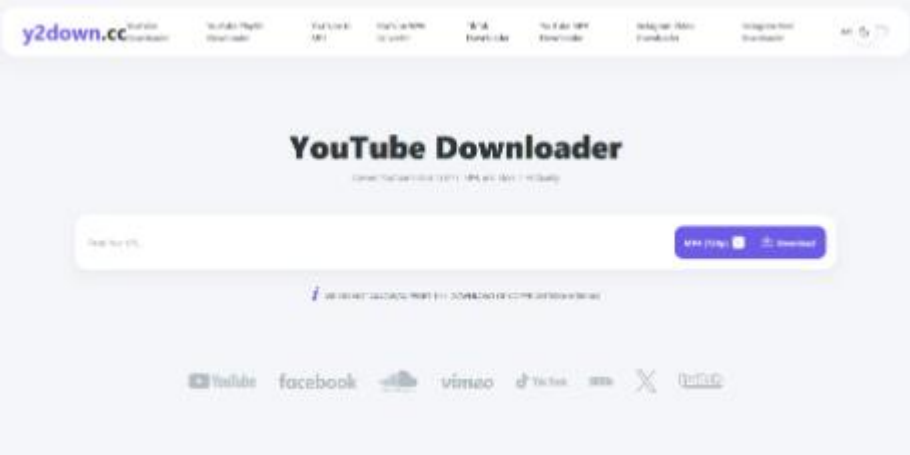

Рисунок 1.2. Інтерфейс веб-сервісу Y2Down

1.3.2 Аналіз веб-сервісу SnapTik

SnapTik є спеціалізованим веб-сервісом, призначеним для завантаження контенту з платформи TikTok. Його функціональне призначення полягає в отриманні відеоматеріалів за посиланням без потреби у встановленні додаткового програмного забезпечення. Сервіс працює через браузер і підтримує використання на різних типах пристроїв, зокрема на персональних комп’ютерах, планшетах і мобільних телефонах. Основний сценарій взаємодії з користувачем є максимально спрощеним: користувач копіює посилання на відео, вставляє його у відповідну веб-форму та отримує файл для подальшого завантаження.

До ключових функціональних можливостей SnapTik належать:

1. завантаження відео з TikTok без водяного знака;
2. робота через браузер без встановлення окремих програм;
3. підтримка різних типів пристроїв;
4. завантаження TikTok-слайдшоу у форматі MP4;
5. необмежена кількість завантажень;
6. надання доступу до відео у вищій якості, якщо така доступна.

Функціональна організація SnapTik засвідчує орієнтацію сервісу на один конкретний сценарій використання, а саме швидке отримання відео з TikTok без додаткових налаштувань і складних проміжних дій. Такий підхід визначає

основні переваги сервісу, серед яких варто виокремити простоту використання, зрозумілий інтерфейс, відсутність потреби у встановленні програмного забезпечення та можливість роботи на різних пристроях. Практичну цінність становить і завантаження відео без водяного знака, що є однією з найбільш затребуваних функцій у межах сервісів цього типу. Окремо підкреслюється, що сервіс не зберігає копії завантажених відео та не веде історію завантажень користувача, що позиціонується як перевага з погляду конфіденційності.

Поряд із цим SnapTik має низку обмежень, зумовлених його вузькою функціональною спеціалізацією. Сервіс орієнтований виключно на платформу TikTok і не підтримує ширших сценаріїв роботи з контентом, пов’язаних із масовим завантаженням кількох відео, отриманням усіх роликів з акаунта або завантаженням контенту за хештегами. Крім того, його функціонал не охоплює редагування відео чи перетворення відеоматеріалів в аудіоформати. Унаслідок цього SnapTik доцільно розглядати як інструмент для виконання чітко визначеної прикладної завдання, а не як універсальне рішення для комплексної роботи з мультимедійним контентом соціальних платформ.

Отже, SnapTik є показовим прикладом вузькоспеціалізованого веб-сервісу, орієнтованого на швидке та зручне завантаження відео з TikTok (рисунок 1.3)[7]. До його сильних сторін належать простота інтерфейсу, відсутність потреби у встановленні додаткового програмного забезпечення та підтримка завантаження без водяного знака. Водночас обмежений набір функцій не дає підстав розглядати цей сервіс як універсальне рішення для повноцінної роботи з контентом соціальних мереж.

Рисунок 1.3. Інтерфейс веб-сервісу SnapTik

1.3.3 Порівняльний аналіз веб-сервісів Y2Down та SnapTik

Розглянуті веб-сервіси Y2Down і SnapTik репрезентують два різні підходи до організації онлайн-засобів для роботи з мультимедійним контентом соціальних платформ. Y2Down орієнтований на поєднання функцій завантаження та конвертації, підтримує вибір формату і якості та в більшій мірі пов'язаний із роботою з контентом YouTube. SnapTik, натомість, характеризується вузькою платформеною спеціалізацією й зосереджується на реалізації максимально спрощеного сценарію завантаження відео з TikTok без водяного знака. Унаслідок цього зазначені сервіси відображають різні моделі побудови веб-рішень: багатофункціональну та спеціалізовану.

З погляду функціональних можливостей Y2Down є більш гнучким сервісом, оскільки підтримує різні формати, вибір роздільної здатності, завантаження плейлистів і окремі елементи конвертації в межах одного середовища. SnapTik не забезпечує такого рівня керування параметрами контенту, проте вирізняється простотою використання та чіткою орієнтацією на один конкретний сценарій взаємодії з користувачем. Якщо Y2Down доцільно розглядати як приклад веб-сервісу з акцентом на функціональну гнучкість, то SnapTik є прикладом вузькоспеціалізованого рішення, у якому пріоритет надано швидкості, доступності та зручності використання.

Порівняння зазначених веб-сервісів має практичне значення для подальшого проектування системи, оскільки дозволяє виокремити

функціональні риси, доцільні для врахування під час розробки веб-додатка. Y2Down демонструє доцільність поєднання завантаження, вибору якості та конвертації в межах одного інтерфейсу, тоді як SnapTik акцентує увагу на важливості простого користувацького сценарію та чіткої функціональної спрямованості. З огляду на це під час проектування веб-додатка доцільно враховувати переваги обох підходів, поєднуючи гнучкість роботи з форматами й параметрами контенту зі зрозумілою та логічно організованою взаємодією користувача із системою.

1.4 Визначення вимог до системи

Аналіз веб-сервісів Y2Down і SnapTik показує, що найбільш затребуваними для користувача є функції отримання контенту за посиланням, вибору формату, доступної якості та швидкого завантаження готового файлу. Разом із тим наявні сервіси не забезпечують однакового рівня гнучкості, універсальності та повноти роботи з метаданими. Це свідчить про потребу у створенні веб-додатка, який поєднуватиме простий сценарій використання з ширшими можливостями обробки контенту.

Під час формування вимог до системи необхідно враховувати як специфіку платформ YouTube і TikTok, так і обмеження, виявлені під час аналізу існуючих веб-сервісів. Саме вимоги визначають набір функцій майбутнього веб-додатка, його основні якісні характеристики та загальну логіку побудови. На їх основі надалі формуються архітектура системи, структура даних і принципи організації користувацького інтерфейсу.

1.4.1 Визначення функціональних вимог до системи

Функціональні вимоги є формалізованим описом основних можливостей веб-додатка та визначають перелік операцій, які система повинна виконувати під час роботи з контентом платформ YouTube і TikTok. Вони фіксують очікувану поведінку системи в межах ключових сценаріїв взаємодії з користувачем, охоплюючи етапи приймання посилання, отримання

метаданих, вибору параметрів завантаження, конвертації мультимедійних файлів та подальшого відображення результатів. У такому розумінні функціональні вимоги задають межі прикладного призначення веб-додатка та визначають зміст його основних функціональних компонентів.

Встановлення функціональних вимог становить необхідний етап проектування системи, оскільки забезпечує перехід від загального опису призначення веб-додатка до чітко окресленої специфікації його можливостей. Їх формалізація дозволяє узгодити логіку роботи програмних модулів, структуру даних і принципи організації користувацького інтерфейсу, а також створює основу для подальшої перевірки відповідності реалізованого програмного продукту поставленим завданням.

Результати аналізу платформ YouTube і TikTok, а також розглянутих веб-сервісів-аналогів свідчать, що визначальними для систем цього типу є функції отримання контенту за посиланням, доступу до супровідних метаданих, вибору формату та якості.

Таблиця 1.2

Функціональні вимоги до системи

№	**Функціональна вимога**	**Опис**
1	Отримання метаданих відео за URL	Система приймає посилання на відео YouTube або TikTok і повертає метадані: назву, автора, дату, тривалість, перегляди, вподобання, коментарі, теги, мініатюру
2	Автоматичне визначення платформи	Система автоматично визначає платформу (YouTube або TikTok) на основі доменного імені в URL без участі користувача
3	Відображення доступних форматів	Система відображає перелік доступних форматів і роздільних здатностей відео для вибору користувачем
4	Завантаження відеофайлу	Система завантажує відеофайл у обраному форматі та якості і автоматично зберігає його на пристрої користувача
5	Завантаження лише аудіо	Система підтримує завантаження виключно аудіодоріжки відео у форматі MP3

Продовження Таблиці 1.2

6	Передача прогресу завантаження	Система відображає поточний прогрес завантаження файлу у реальному часі через прогрес-бар
7	Конвертація медіафайлів	Система приймає локальний медіафайл від користувача і конвертує його у один з підтримуваних форматів (MP4, MP3, WAV, WebM, AAC, AVI)
8	Збереження метаданих у базі даних	Система зберігає метадані кожного запитаного відео у локальній базі даних для подальшого перегляду та аналізу
9	Відображення аналітики завантажень	Система надає статистику завантажень: загальна кількість, розподіл за платформами, форматами та якістю відео
10	Перегляд та фільтрація історії	Система відображає хронологічний список завантажень із можливістю фільтрації за платформою та сортування за датою і розміром файлу

Функціональні вимоги, вище наведені в таблиці 1.2, визначають основні сценарії роботи веб-додатка та встановлюють послідовність операцій, які система повинна підтримувати під час взаємодії з контентом соціальних платформ. Їх урахування на етапі проектування забезпечує узгодженість між логікою обробки даних, структурою програмних модулів і користувацьким інтерфейсом, що є необхідною умовою побудови цілісної та функціонально завершеної системи.

1.4.2 Визначення нефункціональних вимог до системи

Нефункціональні вимоги визначають сукупність якісних характеристик веб-додатка та встановлюють умови, за яких система повинна функціонувати стабільно, коректно та зручно для користувача. На відміну від функціональних вимог, що окреслюють перелік операцій і сценаріїв роботи, нефункціональні вимоги характеризують властивості системи з погляду швидкодії, надійності, сумісності, підтримуваності, масштабованості та загальної придатності до практичного використання. Саме вони формують вимоги до якості реалізації програмного рішення.

Для веб-додатка, орієнтованого на отримання, завантаження, конвертацію та відображення мультимедійного контенту, нефункціональні вимоги мають принципове значення, оскільки наявність необхідного функціоналу сама по собі не забезпечує ефективності системи без належного рівня стабільності, зрозумілості інтерфейсу та прийнятного часу обробки запитів. Урахування таких вимог дозволяє розглядати веб-додаток не лише як сукупність окремих функцій, а як цілісне програмне рішення, здатне до коректної роботи в реальних умовах використання.

Аналіз існуючих веб-сервісів та особливостей платформ YouTube і TikTok показує, що для систем цього типу визначальними є стабільність роботи, зручність користування, сумісність із сучасними браузерами, можливість подальшого розширення функціоналу та придатність до супроводу.

Таблиця 1.3

Нефункціональні вимоги до системи

№	**Нефункціональна вимога**	**Показник**	**Пріоритет**
1	Час отримання метаданих	Не більше 5 секунд для обох платформ	Високий
2	Час завантаження сторінки аналітики	Не більше 2 секунд	Середній
3	Час запису у базу даних	Не більше 1 секунди	Середній
4	Стабільність роботи	Відсутність критичних помилок при 20 послідовних операціях	Високий
5	Сумісність браузерів	Коректна робота у Chrome, Firefox та Edge актуальних версій	Високий
6	Адаптивність інтерфейсу	Коректне відображення на пристроях від 375 пікселів ширини	Середній
7	Крос-платформне розгортання	Запуск без нативної компіляції на Windows, macOS та Linux	Високий
8	Розширюваність системи	Модульна архітектура, що допускає додавання нових платформ	Середній
9	Безпека запитів	Перевірка вхідних URL та захист від path traversal	Середній

Нефункціональні вимоги, наведені в таблиці 1.3, визначають основні якісні параметри веб-додатка та окреслюють умови, за яких система повинна залишатися стабільною, зручною та придатною до практичного використання. Їх формалізація створює підґрунтя для побудови програмного рішення, яке відповідатиме функціональним очікуванням користувача та вимогам до якості його реалізації.

Висновки до розділу 1

У процесі дослідження особливостей платформ YouTube і TikTok встановлено, що, попри їхню спільну відеоорієнтовану природу, вони відрізняються за механізмами доступу до даних, рівнем відкритості інформації, умовами авторизації та правилами використання інструментів розробника. Визначені особливості мають принципове значення для побудови підсистеми збору інформації, оскільки безпосередньо впливають на способи отримання контенту, склад доступних метаданих і загальні умови взаємодії з кожною платформою.

У межах аналізу існуючих веб-сервісів для отримання та обробки контенту соціальних мереж розглянуто сервіси Y2Down і SnapTik, що представляють різні підходи до організації рішень цього типу. Проведене порівняння дозволило встановити, що більш універсальні сервіси орієнтовані на поєднання функцій завантаження, вибору параметрів і конвертації, тоді як вузькоспеціалізовані рішення забезпечують простіший і швидший сценарій взаємодії з контентом окремої платформи. Аналіз їхніх переваг і обмежень дав змогу визначити функціональні риси, які доцільно врахувати під час створення власного веб-додатка.

На основі проведеного аналізу сформульовано функціональні та нефункціональні вимоги до системи, що визначають її основні можливості, якісні характеристики та загальні умови практичного використання. Одержані результати створюють підґрунтя для переходу до наступного розділу, у якому

розглядатимуться питання проектування інтерактивного веб-додатка аналітичної обробки контенту соціальних мереж, його архітектури, функціональної структури та основних модулів.

РОЗДІЛ 2

ПРОЕКТУВАННЯ ІНТЕРАКТИВНОГО ДОДАТКА АНАЛІТИЧНОЇ ОБРОБКИ КОНТЕНТУ СОЦІАЛЬНИХ МЕРЕЖ

2.1 Формування функціональної структури системи

Формування функціональної структури системи є необхідним кроком проектування, що дозволяє розподілити відповідальність між компонентами та забезпечити незалежність їхньої реалізації. Для веб-додатка Loadly виокремлено чотири самостійні функціональні модулі. Кожен модуль відповідає за конкретну ділянку роботи і не залежить від внутрішньої реалізації інших.

Перший модуль – збору інформації – визначає платформу за посиланням, вилучає метадані відео через yt-dlp та нормалізує їх у єдиний формат незалежно від того, YouTube це чи TikTok. Другий модуль відповідає за збереження та аналіз отриманих даних. Він записує відомості у базу даних SQLite і надає агреговану статистику для сторінки аналітики. Третій модуль виконує завантаження відеофайлів. Він запускає yt-dlp з потрібними параметрами і передає прогрес завантаження клієнту у реальному часі. Четвертий модуль обробляє конвертацію вже наявних файлів між форматами через ffmpeg.

Така архітектурна розбивка відповідає принципу єдиної відповідальності: зміна в логіці однієї платформи або формату потребує оновлення лише відповідного модуля, не торкаючись ре торкнеться. Те саме стосується конвертації: якщо з'явиться новий формат або зміняться параметри якості, достатньо підправити один файл. Це класичний принцип єдиної відповідальності, який реально спрощує роботу.

Взаємодія між модулями реалізована таким чином (рисунок 2.1): модуль збору передає результат своєї роботи до модуля збереження; модуль завантаження після отримання файлу повідомляє модуль збереження, щоб той

зафіксував технічні характеристики файлу. Модуль конвертації працює незалежно від інших.

Рисунок 2.1. функціональна структура системи

2.2 Проектування архітектури та основних модулів веб-додатка

Вибір архітектурного рішення для веб-додатка, що поєднує різнорідні завдання з обробки мультимедійного контенту, потребує обґрунтованого балансу між організованістю системи та простотою її реалізації.

Для Loadly вибрано трирівневу клієнт-серверну архітектуру (рисунок 2.2). Перший рівень, клієнтський, реалізований як SPA на React 18 і відповідає за все, що бачить користувач. Другий рівень, серверний, написаний на Node.js з Express.js і містить всю бізнес-логіку. Третій рівень, рівень даних, це SQLite-база, де зберігається інформація про завантажені відео. Між першим і другим рівнями дані передаються через REST API у форматі JSON. Для передачі прогресу завантаження у реальному часі використовується SSE, тобто Server-Sent Events, що дозволяє серверу надсилати повідомлення клієнту через одне постійне HTTP-з'єднання.

Вибір Node.js для серверного рівня обґрунтований його асинхронною моделлю виконання, яка забезпечує ефективний паралельний запуск дочірніх процесів yt-dlp і ffmpeg, не блокуючи при цьому обробку решти запитів. Якби для серверного рівня було обрано платформу з блокуючою моделлю I/O,

кожне завантаження відео фактично паралізувало б сервер до свого завершення. З Node.js цієї проблеми немає.

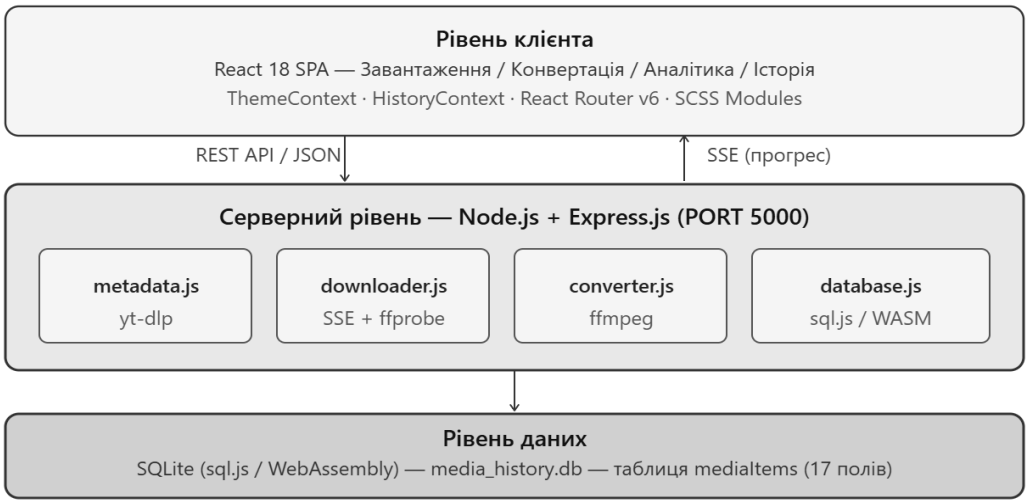

Рисунок 2.2. Трирівнева клієнт-серверна архітектура веб-додатка Loadly

Серверна частина організована за модульним принципом. Файл server.js є точкою входу: він ініціалізує Express, реєструє middleware і підключає чотири маршрутизатори. Кожен маршрутизатор відповідає за свою ділянку API і лише делегує виклики відповідному модулю. Самі модулі не знають нічого про HTTP. Вони отримують дані, обробляють їх і повертають результат. Це дозволяє легко тестувати кожен модуль окремо.

2.2.1 Проектування модуля збору інформації з платформ YouTube та TikTok

Основним інструментом для збору метаданих стала утиліта yt-dlp. Це активно підтримуваний форк youtube-dl з набагато швидшим циклом оновлень і підтримкою понад 1700 платформ. На момент розробки жоден інший інструмент не міг запропонувати порівнянного покриття у поєднанні з простотою інтеграції через командний рядок.

Модуль реалізований у файлі modules/metadata.js. Його структуру складають три функції. Функція detectPlatform() перевіряє URL і повертає рядок "youtube" або "tiktok" залежно від домену. Функція validateUrl()

перевіряє коректність протоколу і відповідність URL підтримуваним платформам. Якщо щось не так, вона кидає виключення з описом проблеми, яке потім перетворюється на HTTP-відповідь 400 з поясненням для клієнта. Основна функція fetchMetadata() запускає yt-dlp через child_process.execFile з прапорами --dump-json --no-playlist --no-download, парсить JSON-відповідь і нормалізує поля.

Нормалізація важлива тому, що yt-dlp повертає трохи різні структури для різних платформ. Наприклад, для YouTube кількість підписників каналу може бути в полі channel_follower_count, а для деяких платформ цього поля взагалі немає. Тому в модулі є логіка послідовної перевірки кількох можливих полів з fallback-значенням.

Для TikTok запуск yt-dlp відрізняється наявністю додаткового аргументу --extractor-args з адресою API-хоста. Без цього TikTok може повертати відео з накладеним водяним знаком.

Результатом роботи модуля є нормалізований об'єкт, який містить вичерпну інформацію про медіаресурс. До основних полів належать: ідентифікатор відео, платформа, заголовок, опис, назва каналу, URL каналу, кількість підписників, дата публікації, тривалість у секундах, кількість переглядів, вподобань і коментарів, країна виробника, категорія, теги (перші вісім), ознака вікового обмеження, URL мініатюри, оригінальний URL, а також масив доступних форматів завантаження. Така структура забезпечує єдине уніфіковане представлення даних незалежно від платформи, з якої було отримано контент, що спрощує подальшу обробку інформації в інших модулях системи. Масив форматів є ключовим елементом об'єкта, оскільки надає користувачу повну інформацію про можливі варіанти отримання медіафайлу. Кожен елемент цього масиву містить ідентифікатор формату, розширення файлу, роздільну здатність, приблизний розмір файлу, а також прапори наявності відео- та аудіопотоків, що дозволяє точно визначити вміст кожного з доступних варіантів. Завдяки такому підходу клієнтська частина застосунку

отримує всі необхідні дані в єдиному запиті, що мінімізує кількість звернень до сервера та підвищує загальну продуктивність системи.

Таблиця 2.1

Склад метаданих, що вилучаються модулем збору інформації

Поле об'єкта	**Тип**	**Відповідне поле yt-dlp**	**Опис**
id	string	id	Унікальний ідентифікатор відео на платформі
platform	string	webpage_url	Назва платформи: youtube або tiktok
title	string	title	Заголовок відео
description	string	description	Текстовий опис відео
uploader	string	uploader / channel	Назва каналу або автора
uploaderUrl	string	uploader_url	Посилання на сторінку каналу
subscriberCount	number	channel_follower_count	Кількість підписників каналу
uploadDate	string	upload_date	Дата публікації у форматі РРРР-ММ-ДД
duration	number	duration	Тривалість у секундах
viewCount	number	view_count	Кількість переглядів
likeCount	number	like_count	Кількість вподобань
commentCount	number	comment_count	Кількість коментарів
country	string	release_country	Країна виробника (якщо доступна)
tags	string[]	tags	Перші 8 тегів відео
ageLimit	number	age_limit	Вікове обмеження (0 означає відсутнє)
thumbnail	string	thumbnail	URL зображення-мініатюри
formats	object[]	formats	Масив доступних форматів відео/аудіо

2.2.2 Проектування модуля збереження, обробки та аналізу метаданих

Для зберігання даних вибрано SQLite в реалізації через бібліотеку sql.js. sql.js – це компіляція SQLite у WebAssembly, що дає одну принципову перевагу: її можна встановити звичайним npm install без жодної нативної компіляції. На Windows це важливо, бо альтернативні бібліотеки, зокрема better-sqlite3 або node-sqlite3, вимагають Visual Studio Build Tools. Для навчального проекту такі вимоги до розгортання виглядають надмірними.

Особливість sql.js полягає в тому, що база даних живе у пам'яті як масив байтів. Зміни не записуються на диск автоматично. Тому в модулі після кожної операції INSERT або UPDATE викликається функція persist(), яка серіалізує поточний стан і зберігає його у файл media_history.db. Це трохи повільніше порівняно з синхронними SQLite-прив'язками, але для масштабу проекту різниця непомітна.

Схема бази даних свідомо максимально проста: одна таблиця mediaItems з шістнадцятьма полями. Складений унікальний ключ по парі videoId + platform вирішує питання дублікатів. Якщо одне й те саме відео запитати двічі, другий запит просто оновить динамічні показники: перегляди, вподобання, коментарі. Запис при цьому не подвоюється.

Для аналітики модуль надає три функції. Функція getHistory() повертає список записів з фільтрацією за платформою і пагінацією. Функція getStats() рахує загальну кількість завантажень, розподіл за платформами, середню тривалість і суму розмірів файлів одним SQL-запитом з умовними агрегатами. Функція getQualityStats() групує завантаження за полем fileQuality, що дозволяє побачити, скільки відео було завантажено в 1080p, 720p, аудіо тощо.

2.2.3 Проектування модуля завантаження та конвертації мультимедійних файлів

Модуль завантаження вирішує кілька нетривіальних технічних завдань, що визначають надійність та коректність роботи системи.

Перше завдання – надійна ідентифікація завантаженого файлу. Назва файлу формується з метаданих відео і може містити довільні символи. Початковий підхід, коли назва будувалася з UUID і yt-dlp записував файл у загальну папку, часом давав збій, бо файл міг мати не те розширення, яке очікувалося. Вирішення просте: кожне завантаження тепер отримує власну тимчасову підпапку з UUID yt-dlp пише у цю папку, а після завершення скрипт читає вміст папки і знаходить файл без жодних припущень щодо його імені. Потім файл переноситься у загальну папку downloads із фінальним іменем, а тимчасова папка видаляється.

Друге завдання – забезпечення надійності завантаження з YouTube, яка активно блокує автоматизовані запити. Для цього yt-dlp запускається з реалістичним User-Agent браузера і заголовком Accept-Language. Також встановлено три повторні спроби на рівні як фрагментів, так і з'єднань. Якщо YouTube все одно повертає HTML замість відео, скрипт перевіряє розширення отриманого файлу і якщо це .htm або .html, видаляє його і повертає клієнту зрозуміле повідомлення про блокування.

Третє завдання – визначення якості відео після завантаження. Якість не можна отримати з назви файлу або форматного рядка yt-dlp, оскільки вони не завжди збігаються з реальними параметрами відеопотоку, особливо після злиття форматів. Замість цього після завантаження запускається ffprobe, який читає висоту відеопотоку безпосередньо з файлу. З цього числа виводиться мітка якості: більше або рівне 2160 це 4K, більше або рівне 1080 це 1080p і так далі.

Конвертація реалізована в окремому модулі converter.js. Він виконує виклик ffmpeg, підбирає параметри кодека залежно від цільового формату і

відслідковує прогрес через парсинг stderr ffmpeg, звідки вилучається поточна позиція time= і загальна тривалість Duration:. Один нетривіальний момент тут: на Windows ffmpeg може бути встановлений у дуже різних місцях, залежно від того, чи використовувався winget, chocolatey, scoop або просте копіювання архіву. Тому в модулі реалізована функція findBinary(), яка послідовно перевіряє змінну оточення, результат команди where, і кілька типових шляхів встановлення включно з папкою WinGet Packages.

Таблиця 2.2

Підтримувані формати модуля конвертації

Формат	**Тип**	**Кодек відео**	**Кодек аудіо**	**Рівні якості**
MP4	Відео	libx264	AAC	CRF 18 / 23 / 28
WebM	Відео	libvpx-vp9	libopus	Стандартний
AVI	Відео	copy (без перекодування)	copy	Оригінальна
MP3	Аудіо	відсутній	libmp3lame -q:a 2	Висока (VBR)
WAV	Аудіо	відсутній	pcm_s16le	Без стиснення
AAC	Аудіо	відсутній	AAC 192 kbps	Фіксована

2.3 Проектування структури бази даних

Для зберігання метаданих мультимедійного контенту та відомостей про завантажені файли в системі використано базу даних під керуванням системи управління базами даних SQLite. За моделлю організації даних база є реляційною: інформація зберігається у вигляді таблиць, рядки яких відповідають окремим записам, а стовпці – атрибутам цих записів із чітко визначеними типами даних та обмеженнями цілісності.

Таблиця 2.3

Загальна характеристика бази даних

Характеристика	**Значення**
Модель даних	Реляційна
Система управління (СУБД)	SQLite
Тип СУБД	Вбудована (serverless), однофайлова

Продовження Таблиці 2.3

Характеристика	Значення
Мова запитів	SQL
Кількість таблиць	1 (mediaItems)
Кількість полів	17
Первинний ключ	id (INTEGER, AUTOINCREMENT)
Складений унікальний ключ	videoId + platform
Індекси	2 (за полями platform та createdAt)
Рівень нормалізації	Третя нормальна форма (3НФ)

Вибір реляційної моделі замість нереляційної (NoSQL) зумовлений характером предметної області. Метадані відеоконтенту мають однорідну структуру з фіксованим набором атрибутів, що природно відображається у табличній формі. Реляційна модель забезпечує контроль цілісності даних через обмеження типів, унікальності та значень за замовчуванням, а також підтримує декларативну мову запитів SQL, необхідну для реалізації аналітичних функцій системи. Використання нереляційного сховища документоорієнтованого або ключ-значення типу для такої структурованої та однотипної інформації не дало б суттєвих переваг, натомість ускладнило б реалізацію аналітичних запитів.

СУБД SQLite обрано з огляду на її вбудований (serverless) характер: база даних зберігається в єдиному файлі та не потребує розгортання окремого сервера, що спрощує встановлення та супровід застосунку. Структуру бази даних спроектовано відповідно до вимог нормалізації — кожен атрибут є атомарним і функціонально залежить від первинного ключа, що усуває надмірність даних. Логічну структуру бази даних наведено у вигляді таблиці її полів (табл. 2.4).

База даних системи має одну таблицю mediaItems. Це свідоме рішення. Коли спочатку розглядалася схема з двома таблицями, де основна таблиця зберігає метадані, а інша, download_history, зберігає деталі завантаження,

виникала проблема JOIN: щоб показати список завантажень з розміром файлу, потрібен LEFT JOIN, а якщо на момент читання запис у другій таблиці ще не зафіксований, у полі буде NULL. Оскільки операція saveMediaItem() і saveDownload() викликаються послідовно в одному запиті, але між ними є пауза на саме завантаження, виникали ситуації, де розмір файлу показувався як нуль. Після того як схему спростили до однієї таблиці і поля fileSize з fileQuality стали полями тієї ж таблиці, що оновлюються окремим UPDATE після завершення завантаження, проблема зникла.

Кожен рядок у таблиці mediaItems описує одне завантажене відео. Пара полів videoId + platform утворює складений унікальний ключ. Два індекси прискорюють найпоширеніші запити: перший по полю platform, другий по createdAt. Поля fileFormat, fileSize і fileQuality залишаються порожніми до моменту фактичного завантаження файлу.

Таблиця 2.4

Структура таблиці mediaItems

Поле	Тип даних	Обмеження	Опис
id	INTEGER	PRIMARY KEY AUTOINCREMENT	Числовий ідентифікатор запису
videoId	TEXT	NOT NULL	Ідентифікатор відео на платформі
platform	TEXT	NOT NULL	youtube або tiktok
url	TEXT	NOT NULL	Повна URL-адреса відео
title	TEXT	DEFAULT ''	Заголовок відео
description	TEXT	DEFAULT ''	Опис відео
uploader	TEXT	DEFAULT ''	Назва каналу або автора
uploadDate	TEXT	DEFAULT ''	Дата публікації відео
duration	INTEGER	DEFAULT 0	Тривалість у секундах
viewCount	INTEGER	DEFAULT 0	Кількість переглядів
likeCount	INTEGER	DEFAULT 0	Кількість вподобань
commentCount	INTEGER	DEFAULT 0	Кількість коментарів
thumbnail	TEXT	DEFAULT ''	URL мініатюри
fileFormat	TEXT	DEFAULT ''	Розширення завантаженого файлу
fileQuality	TEXT	DEFAULT ''	Роздільна здатність (1080p, 720p тощо)
fileSize	INTEGER	DEFAULT 0	Розмір файлу у байтах

Поле	Тип даних	Обмеження	Опис
createdAt	TEXT	DEFAULT (datetime('now'))	Час збереження запису

2.4 Проектування веб-інтерфейсу користувача

Інтерфейс Loadly складається з чотирьох основних сторінок. Переходи між ними відбуваються через навігаційну панель у верхній частині екрана без перезавантаження сторінки, що є стандартною поведінкою для SPA-застосунків. Кожен перехід супроводжується легкою CSS-анімацією: елементи з'являються з невеликим зсувом знизу і плавно матеріалізуються, що створює відчуття динамічності без надмірної помітності.

Сторінка завантаження, яка є головною, відкривається першою. Вгорі розміщено пошуковий рядок, куди користувач вставляє посилання. Нижче, після введення URL, з'являється картка метаданих відео, а ще нижче, панелі вибору формату і завантаження. Розміщення саме в такому порядку відповідає природному сценарію: спочатку знаходиш відео, потім дивишся деталі, потім вибираєш параметри і завантажуєш.

Сторінка конвертації влаштована простіше. Є зона для перетягування файлу, набір кнопок-форматів і вибір якості. Все компактно і без зайвих елементів.

Сторінка аналітики відображає статистику накопичених завантажень. Чотири картки зверху показують загальні числа, нижче ідуть розподіли за платформами, форматами і якістю. Можна перемикати між "Всі", "YouTube" і "TikTok" і всі секції оновлюються синхронно.

Сторінка історії являє собою список усіх завантажених відео у вигляді карток з мініатюрами, що дозволяє користувачу зручно переглядати раніше збережені матеріали. Кожна картка відображає ключову інформацію про відео, зокрема назву, платформу, дату завантаження та розмір файлу. Передбачено можливість фільтрування записів за платформою, а також сортування за датою

або розміром файлу в обох напрямках, що суттєво спрощує навігацію при великій кількості збережених матеріалів.

Весь інтерфейс підтримує темну і світлу теми оформлення, що забезпечує комфортну роботу з застосунком у різних умовах освітлення. Перемикач теми розміщено у правій частині навігаційної панелі для зручного та швидкого доступу. Обрана тема зберігається у localStorage і автоматично відновлюється при наступному відкритті застосунку, завдяки чому користувачу не потрібно щоразу налаштовувати інтерфейс повторно. Зміна теми відбувається миттєво без будь-яких затримок: замінюється атрибут data-theme на кореневому елементі html, а всі кольори компонентів визначені через CSS custom properties. Такий підхід повністю виключає необхідність перерендеру або перезавантаження сторінки, що забезпечує плавний і відгукуючий користувацький досвід.

Таблиця 2.5

Сторінки інтерфейсу та їхнє функціональне призначення

Сторінка	**Маршрут**	**Призначення**	**Ключові компоненти**
Завантаження	/	Введення URL, перегляд метаданих, вибір формату, завантаження	SearchBar, MetadataCard, FormatSelector, DownloadPanel
Конвертація	/convert	Перетворення локальних медіафайлів між форматами	DropZone, FormatGrid, QualitySelector, ProgressBar
Аналітика	/analytics	Статистика і розподіл завантажень	StatCard, PlatformBar, FormatChart, QualityChart
Історія	/history	Хронологічний список завантажень з фільтрацією	HistoryItem, FilterTabs, SortButtons

При проектуванні інтерфейсу орієнтувалися на мінімальну кількість дій для типового сценарію. Щоб завантажити відео, потрібно три кроки: вставити посилання, вибрати формат і натиснути кнопку. Все інше система робить сама. Параметри за замовчуванням підібрані так, щоб підійти більшості випадків без додаткового налаштування.

Висновки до розділу 2

У цьому розділі описано ключові проектні рішення, що лягли в основу системи Loadly. Сформовано функціональну структуру з чотирьох незалежних модулів, де кожен відповідає за свою ділянку роботи. Обрано трирівневу клієнт-серверну архітектуру з React 18 на клієнтському рівні, Node.js з Express.js на серверному і SQLite через sql.js на рівні даних. Вибір зумовлений тим, що Node.js оптимально підходить для асинхронного запуску дочірніх процесів, а sql.js не потребує нативної компіляції, що спрощує розгортання на Windows.

Схема бази даних зведено до однієї таблиці mediaItems з сімнадцятьма полями, що усунуло проблему порожніх значень розміру файлу при JOIN-запитах. Розроблено концепцію інтерфейсу з чотирьох сторінок і підтримкою темної та світлої тем через CSS custom properties.

РОЗДІЛ 3

ПРАКТИЧНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ СИСТЕМИ

3.1 Програмна реалізація модуля збору даних із YouTube та TikTok

Файл modules/metadata.js відповідає за весь цикл отримання метаданих: від перевірки вхідного URL до повернення нормалізованого об'єкта. Виклик yt-dlp здійснюється через promisify-версію execFile з модуля child_process. Встановлено ліміт виконання 30 секунд, після якого процес примусово зупиняється, щоб зависання на стороні yt-dlp не блокувало сервер нескінченно.

Перевірка URL відбувається у два етапи. Спочатку конструктор URL перевіряє формат і протокол. Потім функція detectPlatform() шукає в URL підрядки youtube.com, youtu.be і tiktok.com. Якщо жоден не знайдено, клієнт отримує HTTP 400 з повідомленням, що підтримуються лише YouTube і TikTok. Це запобігає марним викликам yt-dlp для посилань на сторонні ресурси.

Після запуску yt-dlp JSON-рядок зі stdout парситься і обходиться по полях. Частина полів має прямі відповідності: title є title, view_count є viewCount. Для інших потрібна додаткова логіка. Дата публікації upload_date у форматі YYYYMMDD перетворюється на YYYY-MM-DD через просту функцію parseDate(). Кількість підписників шукається послідовно у полях channel_follower_count і uploader_subscriber_count, бо yt-dlp використовує різні поля залежно від типу сторінки. Масив форматів фільтрується від технічних форматів з розширенням mhtml, що є службовими форматами і непотрібні користувачу.

Отримані метадані зберігаються у базі одразу після успішної відповіді yt-dlp, ще до того як результат повертається клієнту. Якщо відео вже є у базі, INSERT ... ON CONFLICT DO UPDATE SET оновлює тільки показники взаємодії аудиторії, не зачіпаючи інші поля.

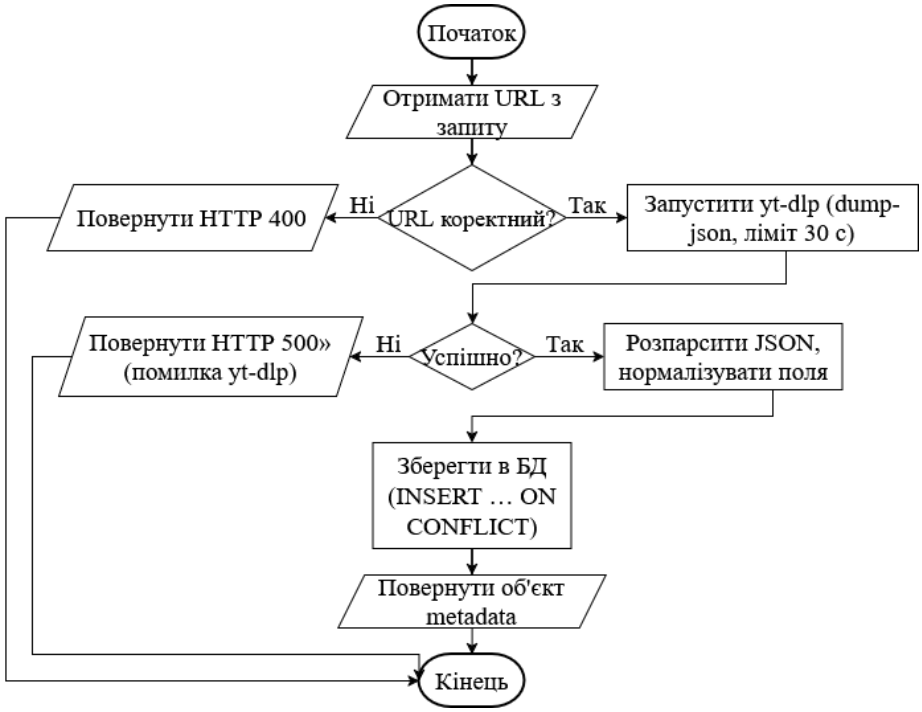

Рисунок 3.1. Схема реалізації функції fetchMetadata()

Варто зазначити один практичний нюанс. У перших версіях модуля yt-dlp запускався через spawn і читав stdout порядково, що теоретично мало б бути ефективніше. Але на практиці для операції dump-json, де весь JSON видається одним блоком після завершення роботи, подієва модель з spawn додавала зайву складність без жодної переваги. Тому перейшли на execFile з очікуванням повного завершення процесу.

3.2 Програмна реалізація веб-інтерфейсу додатка

Клієнтська частина організована у п'ять каталогів. У components/ зберігаються перевикористовувані компоненти, у pages/ компоненти окремих сторінок, у hooks/ кастомні хуки, у api/ шар взаємодії з сервером, у context/ контексти глобального стану. Стилі зосереджені в styles/ зі змінними і глобальними правилами, а кожен компонент має власний .module.scss файл.

Глобальний стан керується через два React-контексти. ThemeContext відповідає за тему і синхронізацію з localStorage. HistoryContext це єдине місце, де живе список завантажених відео. Коли користувач видаляє запис на сторінці Історія, масив оновлюється в контексті, і сторінка Аналітика, яка теж

читає цей самий контекст, автоматично перераховує всі показники. Це вирішує проблему синхронізації між сторінками без додаткових запитів до сервера.

3.2.1 Головна сторінка та отримання метаданих

На головній сторінці (рисунок 3.2) SearchBar першим реагує на введення URL. Компонент стежить за кожним натисканням клавіші в полі введення і перевіряє, чи є в URL підрядки youtube або tiktok. Якщо є, поряд з полем з'являється кольоровий значок платформи з SVG-іконкою. Це відбувається миттєво, без будь-яких запитів до сервера.

Після відправлення запиту хук useMetadata надсилає GET /api/metadata і очікує відповіді. Поки відповідь не прийшла, на місці картки метаданих відображається skeleton-анімація: ті самі прямокутники розміру, що й реальні елементи, але заповнені кольором, що переливається. Це набагато приємніший користувацький досвід, ніж порожній простір або спінер посеред сторінки.

Картка метаданих MetadataCard показує мініатюру, мітку тривалості у правому нижньому куті мініатюри, назву відео, назву каналу, кількість підписників, дату публікації, країну, категорію, чотири показники в ряд (перегляди, вподобання, коментарі, тривалість), теги і посилання на оригінальне відео.

Нижче картки метаданих розміщено компонент FormatSelector. Він розбиває формати на дві вкладки: Відео і Аудіо. У вкладці Відео перелік спочатку проходить через функцію дедублікації deduplicateFormats(): вона групує записи за парою висота + розширення і для кожної пари залишає тільки один варіант, з перевагою MP4 над WebM і більшого розміру файлу над меншим. В результаті замість двадцяти і більше технічних рядків від yt-dlp користувач бачить чистий список: 144p .mp4, 360p .mp4, 720p .mp4, 1080p .mp4, 1080p .webm і так далі. Розширення підсвічені різними кольорами для швидшого розпізнавання.

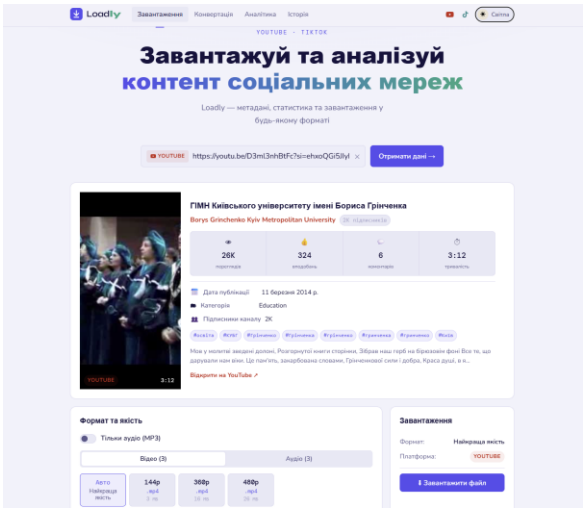

Рисунок 3.2. Головна сторінка: метадані відео, компоненти FormatSelector та DownloadPanel

3.2.2 Сторінки аналітики та історії

Сторінка аналітики (рисунок 3.3) завантажує список відео з HistoryContext і рахує все на клієнті через useMemo. Жодних додаткових HTTP-запитів при перемиканні фільтра за платформою немає. Дані якості завантажень отримуються окремим запитом GET /api/history/quality-stats при першому відкритті сторінки.

Секція розподілу за платформами показує два кола з числами і горизонтальний бар, де ліва частина це YouTube (червоний), права це TikTok (бірюзовий). Пропорції оновлюються при зміні даних.

Секція форматів файлів і секція якості відображають горизонтальні бари з підписами. Довжина кожного бара пропорційна до найбільшого значення у групі.

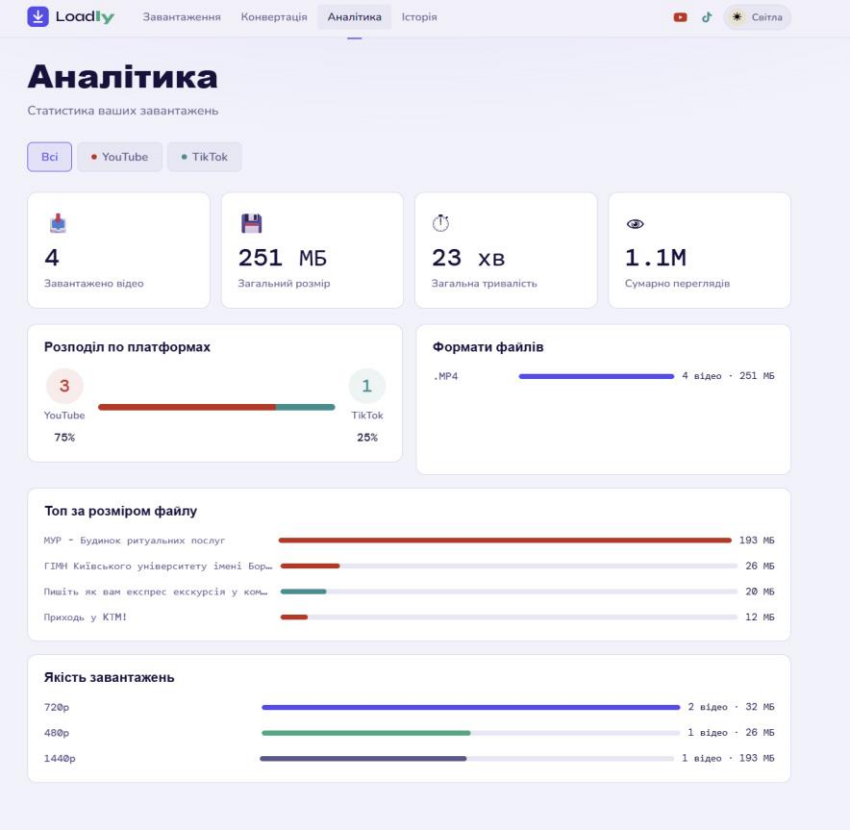

Рисунок 3.3. Сторінка «Аналітика»: статистичні картки та розподіл за платформами, форматами, якістю

Сторінка «Історія» (рисунок 3.4) відображає картки завантажень. Кожна картка містить мініатюру розміром 210 на 118 пікселів, назву відео, назву каналу, дату збереження, показники взаємодії та технічні дані файлу: формат, якість і розмір. Фільтр за платформою і сортування виконуються через useMemo на клієнті. Видалення запису відразу оновлює масив allItems у HistoryContext, що миттєво відображається і на сторінці Аналітика.

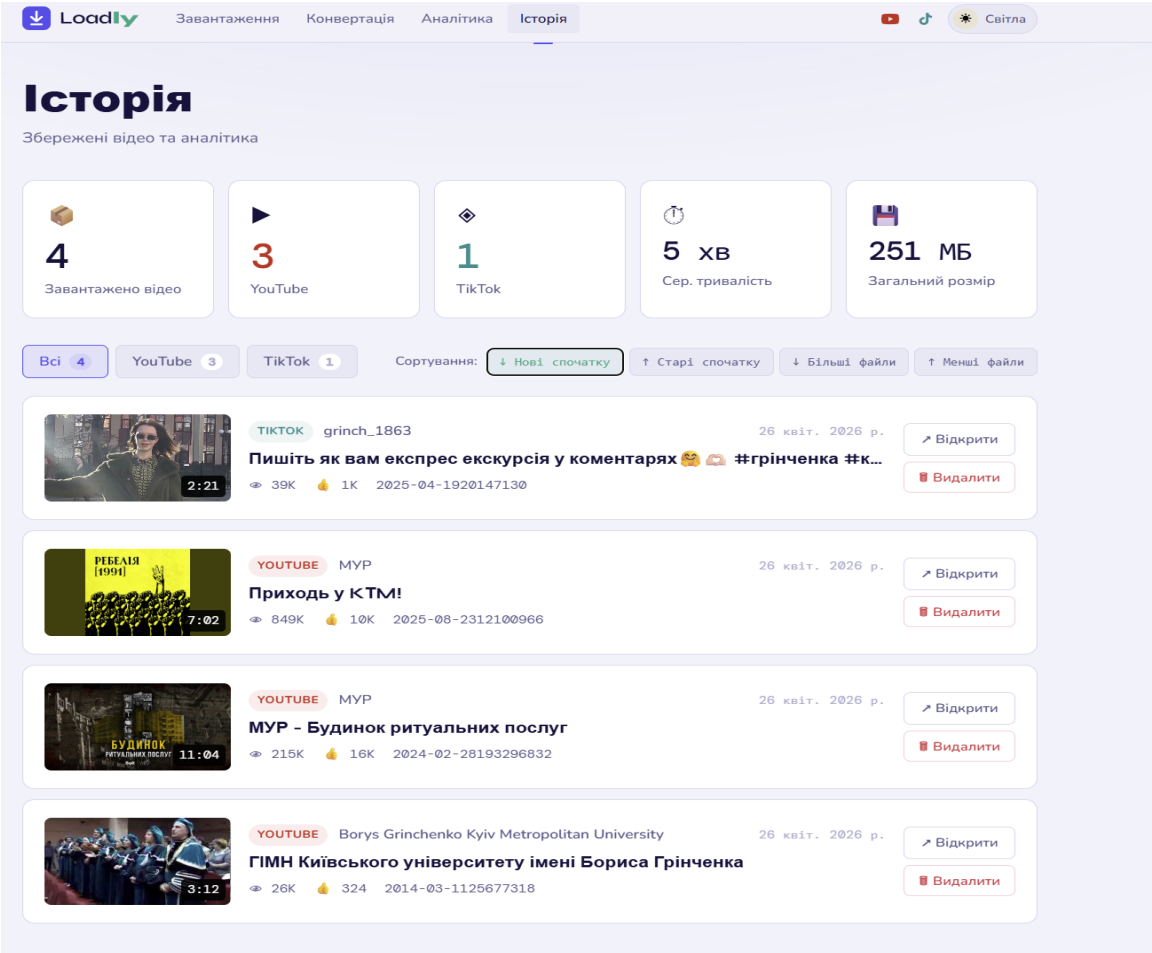

Рисунок 3.4. Сторінка Історія: список завантажених відео з фільтром та сортуванням

3.3 Реалізація функцій завантаження та конвертації мультимедійних файлів

3.3.1 Завантаження через SSE

Завантаження відеофайлів реалізоване через SSE-ендпоінт GET /api/download/stream (рисунок 3.5). Клієнт відкриває з'єднання через стандартний браузерний API EventSource і слухає події. Сервер одразу після встановлення з'єднання надсилає подію status з повідомленням "Отримання метаданих..." і запускає fetchMetadata(). Після отримання метаданих надсилається подія metadata з заголовком відео і назвою платформи. Потім стартує завантаження.

Функція download() у модулі downloader.js запускає yt-dlp у тимчасовій підпапці з унікальним ідентифікатором. Прогрес читається зі stdout: yt-dlp при прапорі --newline видає окремий рядок на кожне оновлення прогресу у форматі [download] X.X% of Y.YYMiB. Регулярний вираз вилучає відсоток, і той передається у callback onProgress, а звідти через SSE надходить до клієнта як подія progress з числом від 0 до 100.

Після завершення завантаження запускається ffprobe, визначається якість відео, файл переноситься з тимчасової папки у загальну директорію downloads/ і надсилається подія done з ім'ям файлу, його розміром і URL для отримання. Тоді ж оновлюється запис у базі даних.

На клієнтській стороні хук useDownload при отриманні події done відразу запускає завантаження файлу в браузері. Це робиться через fetch(), що отримує файл як Blob, після чого для цього Blob створюється тимчасовий об'єктний URL, програмно клікається невидимий елемент з атрибутом download, URL звільняється. Все відбувається без участі користувача одразу після завершення завантаження на сервері.

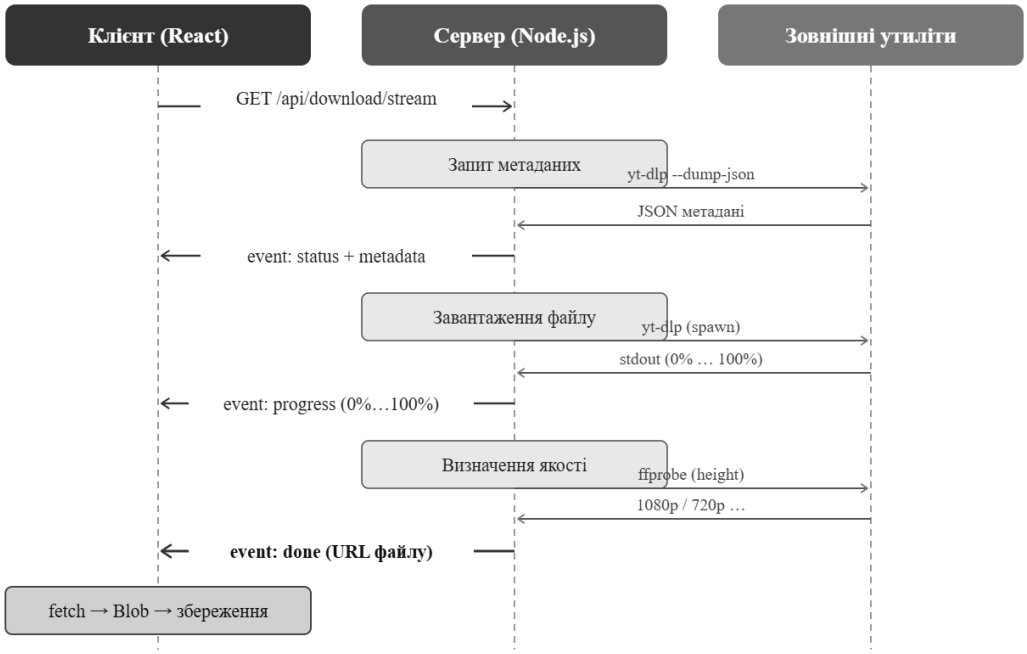

Рисунок 3.5. Схема SSE-потоку завантаження: послідовність подій від сервера до клієнта

3.3.2 Конвертація локальних файлів

Сторінка «Конвертація» (рисунок 3.6) дозволяє перетворити будь-який медіафайл з пристрою користувача у один з шести форматів. Файл передається на сервер через FormData з полями file, targetFormat і quality.

На серверній стороні middleware multer зберігає файл у папці downloads/ під тимчасовим іменем. Оригінальне ім'я файлу, яке multer зберігає в полі originalname, декодується з latin1 у UTF-8 через Buffer.from(name, 'latin1').toString('utf8'). Це вирішує проблему кириличних символів в іменах файлів: Windows передає їх у формі latin1, і без перекодування назва перетворюється на нечитабельний набір символів.

Після перекодування запускається ffmpeg з параметрами, що відповідають обраному формату. Для MP4 з рівнем якості high параметр CRF встановлюється на 18, для medium на 23, для low на 28. CRF 23 це рекомендоване за замовчуванням значення бібліотеки libx264 і забезпечує розумний баланс між якістю і розміром файлу. Для аудіоформатів відеопотік вимикається прапором -vn. Результуючий файл отримує ім'я OriginalName_converted.ext.

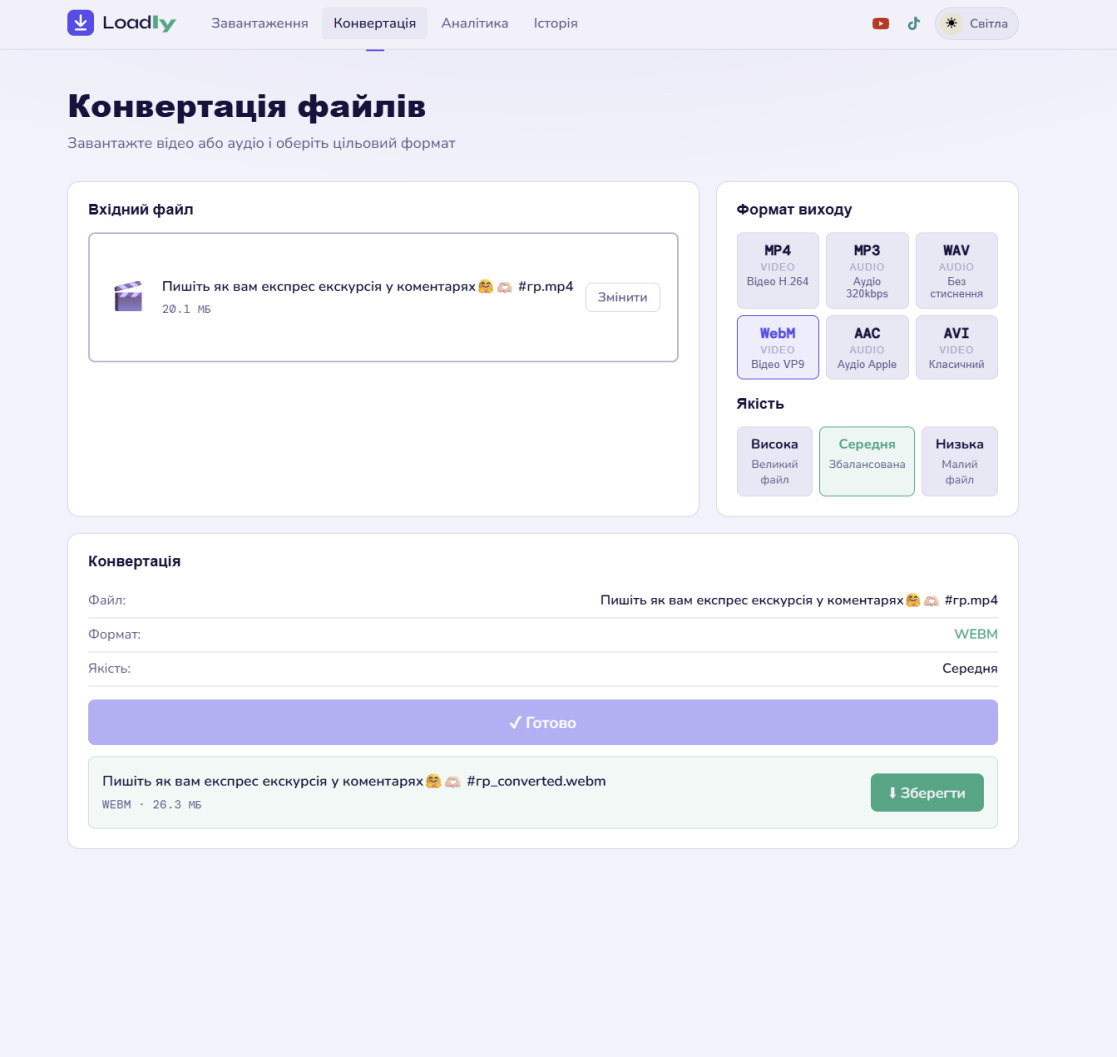

Рисунок 3.6. Сторінка «Конвертація»: зона drag and drop, вибір формату і якості, прогрес конвертації

3.4 Тестування функцій отримання, відображення та завантаження контенту

Тестування проводилось у два основних напрямки: перевірка функціональних сценаріїв через браузерний інтерфейс і пряма перевірка REST API через адресний рядок та консоль браузера. Для кожного сценарію фіксувалося вхідне посилання або параметри запиту, очікуваний результат і те, що система фактично повернула.

Таблиця 3.1

Результати функціонального тестування основних сценарії

N	Сценарій	Очікуваний автором результат	Фактичний результат	Статус
1	Запит метаданих коректного YouTube URL	HTTP 200, JSON з 14+ полями	HTTP 200, всі поля заповнені	Пройдено
2	Запит метаданих TikTok URL	HTTP 200, platform: tiktok	HTTP 200, платформа визначена коректно	Пройдено
3	Некоректний URL (без протоколу)	HTTP 400, опис помилки	HTTP 400, "Некоректне посилання"	Пройдено
4	Посилання на непідтримувану платформу	HTTP 400	HTTP 400, "Підтримуються лише YouTube та TikTok"	Пройдено
5	SSE-завантаження відео у форматі MP4 720p	Послідовні події: status, progress 0-100, done	Всі події отримано, файл збережено	Пройдено
6	Завантаження лише аудіо (audioOnly=true)	Файл .mp3, fileQuality = Аудіо у БД	MP3 завантажено, якість збережена	Пройдено
7	Автоматичне збереження файлу у браузері	Браузер зберігає файл без дій користувача	Файл збережений автоматично	Пройдено
8	Запис fileSize та fileQuality у БД	Значення > 0 після завантаження	Обидва поля заповнені коректно	Пройдено
9	Конвертація MP4 у MP3	HTTP 200, outputFile з розширенням .mp3	Файл конвертовано та збережено	Пройдено
10	Конвертація файлу з кириличною назвою	Кирилиця у назві _converted файлу	Назва відображається коректно	Пройдено
11	Видалення запису з Історії	Запис видалено, Аналітика оновлена	Обидві сторінки оновлено без перезавантаження	Пройдено
12	Фільтр за YouTube на сторінці Аналітика	Відображаються лише YouTube-відео	Фільтр застосовано коректно	Пройдено
13	Сортування за розміром (більші спочатку)	Записи впорядковані за спаданням fileSize	Сортування коректне	Пройдено
14	Перемикання теми (темна / світла)	Зміна теми без перезавантаження, збереження	Тема змінена і збережена у localStorage	Пройдено
15	Перевірка /api/health	HTTP 200, {status: "ok"}	HTTP 200, сервер відповідає	Пройдено

Усі 15 сценаріїв завершилися успішно. Найбільш трудомісткими при налагодженні виявилися сценарії 8, 10 і 11. Проблема з нульовим fileSize

виникла через відсутність await перед асинхронними викликами saveMediaItem() і saveDownload(): функції запускалися, але результат не очікувався, і запис у базу відбувався вже після того, як HTTP-відповідь була надіслана клієнту. Проблема з кириличними символами у назвах конвертованих файлів вирішилася через перекодування latin1 в UTF-8. Проблема з синхронізацією Аналітики та Історії вирішилася переносом allItems у спільний HistoryContext.

Окремо перевірялася коректна робота SSE у трьох браузерах: Chrome 124, Firefox 125 і Edge 124. У всіх трьох браузерах EventSource відпрацював коректно, події progress надходили в реальному часі, автоматичне завантаження через Blob URL спрацьовувало без блокування з боку браузера.

Перевірку адаптивності виконано через інструменти розробника у трьох режимах: 375 px (iPhone SE), 390 px (iPhone 14) і 768 px (iPad). На всіх трьох роздільних здатностях інтерфейс коректно адаптувався. Навігаційна панель переходила у двохрядний режим, картка метаданих ставала вертикальною з мініатюрою зверху, сітки форматів переходили у три колонки.

3.5 Оцінка працездатності та ефективності розробленої системи на реальних даних

Після того як основне тестування підтвердило коректність роботи системи, її перевірили на реальних відео для оцінки продуктивності в умовах, наближених до реального використання. Тестування проводилося на відео різної тривалості і якості з YouTube (7 відео) і TikTok (5 відео), що дозволило охопити широкий спектр можливих сценаріїв взаємодії з системою. Вибір відео здійснювався таким чином, щоб забезпечити рівномірне покриття різних форматів, роздільних здатностей та тривалостей, від коротких кліпів до повнометражних матеріалів. Для кожної операції вимірювався час виконання від натискання кнопки до повного завершення дії, що забезпечило об'єктивну оцінку швидкодії кожного модуля системи. Отримані результати фіксувалися

та порівнювалися між собою з метою виявлення можливих вузьких місць у продуктивності та визначення напрямків подальшої оптимізації застосунку.

Таблиця 3.2

Результати вимірювань часу виконання операцій на реальних даних

Операція	**Платформа**	**Мін., с**	**Макс., с**	**Середнє, с**	**Відповідність вимогам**
Отримання метаданих	YouTube	1,1	3,4	2,0	Так (вимога < 5 с)
Отримання метаданих	TikTok	1,4	4,1	2,5	Так (вимога < 5 с)
Завантаження 360p (~10 МБ)	YouTube	6	14	9	Так
Завантаження 1080p (~120 МБ)	YouTube	35	95	58	Залежить від мережі
Завантаження без водяного знака	TikTok	4	11	7	Так
Конвертація MP4 у MP3, ~50 МБ	-	4	9	6	Так (вимога < 60 с)
Запис у базу даних	-	< 0,1	< 0,1	< 0,1	Так (вимога < 1 с)
Завантаження сторінки «Аналітика»	-	< 0,3	< 0,3	< 0,3	Так (вимога < 2 с)

З таблиці видно, що час отримання метаданих не перевищує чотирьох секунд для обох платформ, що відповідає вимозі менше п'яти секунд. Операції з базою даних практично миттєві. Час завантаження відео залежить від

швидкості інтернет-з'єднання і розміру файлу, це об'єктивний фактор, не пов'язаний з архітектурою системи. Завантаження сторінки «Аналітика» обходиться без додаткових HTTP-запитів для більшості операцій, що і забезпечує таку малу затримку.

Щоб перевірити стабільність, виконали двадцять послідовних завантажень протягом однієї сесії. Після кожного завантаження перевіряли запис у базі даних. Всі двадцять операцій завершилися успішно, fileSize і fileQuality заповнені коректно у кожному записі. Сервер не перезапускався і не видавав критичних помилок протягом усього тестування.

Алгоритм дедублікації форматів скоротив середній перелік з 22 технічних варіантів yt-dlp до 8 зручних для вибору опцій. Це підтвердило правильність рішення фільтрувати формати на клієнті, а не показувати користувачу весь технічний список.

Висновки до розділу 3

У третьому розділі описано практичну реалізацію і тестування веб-додатка Loadly. Реалізовано чотири серверних модулі і клієнтський SPA з чотирма сторінками. Ключові технічні рішення, описані у другому розділі, перевірено і підтверджено у роботі.

Модуль збору метаданих через yt-dlp нормалізує дані від YouTube і TikTok у єдиний формат і зберігає їх у базі без дублікатів. Модуль завантаження використовує SSE для передачі прогресу в реальному часі і тимчасові підпапки для надійного пошуку отриманого файлу. Модуль конвертації коректно обробляє кириличні назви файлів і підтримує шість форматів з трьома рівнями якості. Синхронізація сторінок Аналітика та Історія забезпечується через спільний HistoryContext.

Функціональне тестування охопило 15 сценаріїв і всі завершилися успішно в трьох браузерах. Тестування на реальних даних (12 відео, 20 послідовних завантажень) підтвердило відповідність системи встановленим нефункціональним вимогам за часом відповіді і стабільністю роботи.

ВИСНОВКИ

У кваліфікаційній роботі вирішено актуальне науково-практичне завдання – розроблено інтерактивний веб-додаток Loadly для автоматизованого отримання, завантаження, збереження, конвертації та аналітичної обробки контенту платформ YouTube і TikTok. За результатами виконання роботи зроблено такі висновки.

Проведено аналіз предметної галузі та досліджено особливості контенту соціальних мереж як об'єкта аналізу. Встановлено, що контент соціальних мереж є комплексним цифровим об'єктом, що характеризується динамічністю, мультимодальністю та платформозалежністю. Визначено відмінності між платформами YouTube і TikTok за механізмами доступу до даних. Аналіз веб-сервісів Y2Down і SnapTik засвідчив, що жоден із них не забезпечує повного циклу роботи з контентом обох платформ у межах єдиної системи. На основі проведеного аналізу сформульовано функціональні та нефункціональні вимоги до веб-додатка.

Спроектовано трирівневу клієнт-серверну архітектуру системи: React 18 – клієнтський рівень, Node.js з Express.js – серверний рівень, SQLite через sql.js – рівень даних. Виокремлено чотири функціональні модулі відповідно до принципу єдиної відповідальності. Схему бази даних спрощено до однієї таблиці mediaItems, що усунуло проблему порожніх значень поля fileSize при паралельних операціях запису. Спроектовано веб-інтерфейс з чотирьох сторінок із підтримкою темної та світлої тем.

Реалізовано веб-додаток Loadly, що забезпечує: отримання та нормалізацію метаданих із YouTube і TikTok засобами yt-dlp; передачу прогресу завантаження у реальному часі технологією SSE; конвертацію між шістьма медіаформатами з трьома рівнями якості; збереження метаданих у базі даних SQLite; відображення аналітики та хронологічної історії завантажень. Проведено функціональне тестування 15 сценаріїв у трьох браузерах – усі завершилися успішно. Тестування продуктивності підтвердило відповідність системи встановленим нефункціональним вимогам.

Практичне значення одержаних результатів полягає у тому, що розроблений веб-додаток може бути використаний у навчальній діяльності, при підготовці інформаційних матеріалів та у роботі з мультимедійними ресурсами. Перспективами подальшого розвитку системи є розширення переліку підтримуваних платформ, реалізація пакетного завантаження та інтеграція засобів автоматичного розпізнавання мовлення для генерації субтитрів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. OECD. Participative Web and User-Created Content: Web 2.0, Wikis and Social Networking. Paris: OECD Publishing, 2007. 124 p. URL: https://www.oecd.org/en/publications/participative-web-and-user-created-content_9789264037472-en.html

2. Chaffey D. Social media statistics 2024. Smart Insights. 2024. URL: https://www.smartinsights.com/social-media-marketing/social-media-strategy/new-global-social-media-research

3. Auxier B., Anderson M. Social Media Use in 2021. Pew Research Center. 2021. URL: https://www.pewresearch.org/internet/2021/04/07/social-media-use-in-2021/

4. Vogels E., Gelles-Watnick R., Massarat N. Teens, Social Media and Technology 2022. Pew Research Center. 2022. URL: https://www.pewresearch.org/internet/2022/08/10/teens-social-media-and-technology-2022/

5. Y2Down. Офіційний сайт. URL: https://y2down.cc/enZv/

6. SnapTik. Офіційний сайт. URL: https://snaptik.app

7. yt-dlp. Документація проекту на GitHub. URL: https://github.com/yt-dlp/yt-dlp

8. FFmpeg. Офіційна документація. URL: https://ffmpeg.org/documentation.html

9. Node.js Foundation. Node.js Documentation. URL: https://nodejs.org/en/docs

10. Express.js. Офіційна документація. URL: https://expressjs.com

11. SQLite. Офіційна документація. URL: https://www.sqlite.org/docs.html

12. sql.js. Документація на GitHub. URL: https://github.com/sql-js/sql.js

13. React. Офіційна документація. URL: https://react.dev

14. React Router. Офіційна документація. URL: https://reactrouter.com

15. MDN Web Docs. Using server-sent events. URL: https://developer.mozilla.org/en-US/docs/Web/API/Server-sent_events/Using_server-sent_events

16. MDN Web Docs. Fetch API. URL: https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API

17. Sass. Офіційна документація. URL: https://sass-lang.com/documentation

18. State of JS 2023. Результати щорічного опитування розробників. URL: https://stateofjs.com/en-US

19. npm. Офіційна документація реєстру пакетів. URL: https://docs.npmjs.com

20. Multer. Документація на GitHub. URL: https://github.com/expressjs/multer

21. Pérez-Torres V., Pastor-Sánchez J. A., Castillo-Peces C. Analysis of the Use of Short Video on TikTok as a Social Media Platform from a Scientific Perspective. Publications. 2023. Vol. 11, № 1. P. 1–17. DOI: 10.3390/publications11010002.

22. YouTube Data API Overview. Google Developers. 2024. URL: https://developers.google.com/youtube/v3/getting-started (дата звернення: 14.04.2026).

23. TikTok for Developers. TikTok Developer Documentation. 2024. URL: https://developers.tiktok.com (дата звернення: 14.04.2026).

24. Fielding R. T., Taylor R. N. Principled Design of the Modern Web Architecture. ACM Transactions on Internet Technology. 2002. Vol. 2, № 2. P. 115–150. DOI: 10.1145/514183.514185.

25. Richardson L., Ruby S. RESTful Web Services. Sebastopol : O'Reilly Media, 2007. 454 p.

26. Sommerville I. Software Engineering. 10th ed. Harlow : Pearson, 2016. 810 p.

27. Martin R. C. Clean Architecture: A Craftsman's Guide to Software Structure and Design. Boston : Prentice Hall, 2017. 432 p.

28. Fowler M. Patterns of Enterprise Application Architecture. Boston : Addison-Wesley, 2002. 560 p.

29. Bass L., Clements P., Kazman R. Software Architecture in Practice. 3rd ed. Boston : Addison-Wesley, 2012. 640 p.

30. Cooper A., Reimann R., Cronin D. About Face: The Essentials of Interaction Design. 4th ed. Indianapolis : Wiley, 2014. 720 p.

31. Pressman R. S., Maxim B. R. Software Engineering: A Practitioner's Approach. 8th ed. New York : McGraw-Hill, 2015. 976 p.

32. Tilkov S., Vinoski S. Node.js: Using JavaScript to Build High-Performance Network Programs. IEEE Internet Computing. 2010.Vol.14,№ 6.P.80–83. DOI: 10.1109/MIC.2010.145.

33. Галузов, С. Ю., Бондарчук, А. П., Бажан, Т. О., & Корецька, В. О. (2023). Застосування методів data science для прогнозування попиту в ритейлі. Телекомунікаційні та інформаційні технології, (3), 58-64.

34. MDN Web Docs. Child Process. URL: https://nodejs.org/api/child_process.html (дата звернення: 17.04.2026).

35. Shantyr, A., Zinchenko, O., Storchak, K., Bondarchuk, A., & Pepa, Y. (2025). Prediction of quality software quality indicators with applied modifications of integrated gradiates methods. Informatyka, Automatyka, Pomiary w Gospodarce i Ochronie Środowiska, 15(2), 139-146.

36. Cockburn A. Writing Effective Use Cases.Boston:Addison-Wesley, 2001.304 p.

37. Мілованова, М. В., & Бондарчук, А. П. (2020). Оцінка ефективності інформаційного пошуку. Телекомунікаційні та інформаційні технології, (1), 45-52.

38. Pew Research Center. Social Media Use in 2023. URL: https://www.pewresearch.org/internet/2023/12/31/americans-social-media-use/ (дата звернення: 20.04.2026).

39. Abramov, V., Astafieva, M., Boiko, M., Bodnenko, D., Bushma, A., Vember, V., ... & Yaskevych, V. (2021). Theoretical and practical aspects of the use of mathematical methods and information technology in education and science.

40. Машкіна, І. В., Абрамов, В. О., Носенко, Т. І., Іваніченко, Є. В., & Яскевич, В. О. (2024). Навчально-методичний посібник до виконання і захисту кваліфікаційної роботи бакалавра спеціальностей 122 Комп'ютерні науки для здобувачів вищої освіти денної форми навчання.

ДОДАТОК А

Лістинг функції fetchMetadata() (modules/metadata.js)

```
const { execFile } = require('child_process');
const { promisify } = require('util');
const execFileAsync = promisify(execFile);


function detectPlatform(url) {
if (url.includes('youtube.com') || url.includes('youtu.be')) return 'youtube';
if (url.includes('tiktok.com')) return 'tiktok';
throw new Error('Підтримуються лише YouTube та TikTok');}

function parseDate(raw) {
if (!raw || raw.length !== 8) return '';
return raw.slice(0,4) + '-' + raw.slice(4,6) + '-' + raw.slice(6,8);}

async function fetchMetadata(url) {
const platform = detectPlatform(url);
const args = ['--dump-json', '--no-playlist', '--no-download'];

if (platform === 'tiktok') {
args.push('--extractor-args',
'tiktok:api_hostname=api22-normal-c-useast2a.tiktokv.com');}

args.push(url);
const { stdout } = await execFileAsync('yt-dlp', args, { timeout: 30000 });
const raw = JSON.parse(stdout);

return {
id: raw.id, platform,
title: raw.title || '',
uploader: raw.uploader || raw.channel || '',
uploadDate: parseDate(raw.upload_date),
duration: raw.duration || 0,
viewCount: raw.view_count || 0,
likeCount: raw.like_count || 0,
thumbnail: raw.thumbnail || '',

   formats: (raw.formats || []).filter(f => f.ext !== 'mhtml'),
};
}
module.exports = { fetchMetadata, detectPlatform };
```

ДОДАТОК Б

Лістинг модуля завантаження (modules/downloader.js)

```
const path = require('path');
const fs = require('fs');
const { execFile } = require('child_process');
const { v4: uuidv4 } = require('uuid');

async function download(url, formatId, audioOnly, onProgress) {
const tmpDir = path.join(__dirname, '../downloads', uuidv4());
fs.mkdirSync(tmpDir, { recursive: true });
return new Promise((resolve, reject) => {

const args = [
'--newline', '--retries', '3',
'--add-header', 'User-Agent:Mozilla/5.0 (Windows NT 10.0; Win64; x64)',
'--add-header', 'Accept-Language:uk-UA,uk;q=0.9',
'-o', path.join(tmpDir, '%(title)s.%(ext)s'),
];

if (audioOnly) args.push('-x', '--audio-format', 'mp3');
else if (formatId) args.push('-f', formatId);
args.push(url);
const proc = execFile('yt-dlp', args);
proc.stdout.on('data', (data) => {
const match = data.toString().match(/[download]\s+([\d.]+)%/);
if (match) onProgress(parseFloat(match[1]));
});

proc.on('close', (code) => {
const files = fs.readdirSync(tmpDir);
if (!files.length || code !== 0) {
fs.rmdirSync(tmpDir, { recursive: true });
return reject(new Error('Помилка завантаження файлу'));}

const dest = path.join(__dirname, '../downloads', files[0]);
fs.renameSync(path.join(tmpDir, files[0]), dest);
fs.rmdirSync(tmpDir, { recursive: true });

resolve(dest);
});
});
}
module.exports = { download };
```