Хмельницький національний університет
Факультет інформаційних технологій
Кафедра інженерії програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА

Вебсервіс для обміну мультимедійним контентом між користувачами
Назва теми

Рівень вищої освіти перший (бакалаврський)
Галузь знань 12 «Інформаційні технології»
Спеціальність 121 «Інженерія програмного забезпечення»
Освітня програма Освітньо-професійна програма «Інженерія програмного забезпечення»

Шифр КвРІПЗ.2401218.01.21.00.ПЗ

Виконав студент IV курсу групи ІПЗ-22-1 _____ Максим ЦИМБАЛЮК
Підпис Ім'я, прізвище

Керівник канд. техн. наук, доцент _____ Юрій ФОРКУН
Науковий ступінь, звання Підпис Ім'я, прізвище

Нормоконтролер техн. техн. наук, доцент _____ Оксана ЯШИНА
Підпис Ім'я, прізвище

До захисту допускаю:
Завідувач кафедри інженерії програмного забезпечення _____ Леонід БЕДРАТЮК
Підпис Ім'я, прізвище

9 червня 2026 р.

Хмельницький 2026

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет Інформаційних технологій
Кафедра Інженерії програмного забезпечення
Рівень вищої освіти Перший (бакалаврський)
Галузь знань 12 «Інформаційні технології»
Спеціальність 121 «Інженерія програмного забезпечення»
Освітня програма Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри ІПЗ
Л. П. Бедратюк
02 01 2026 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Цимблюку Максиму Валентиновичу
Прізвище, ім'я, по батькові студента

1. Тема роботи Вебсервіс для обміну мультимедійним контентом між користувачами

Керівник роботи Форкун Юрій Вікторович, канд. техн. наук, доцент
Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

Затверджена наказом ректора університету від 20.01.2026 р. №7
2. Строк подання студентом роботи на кафедру 01.06.26 р.
3. Вихідні дані до роботи Методичні матеріали до кваліфікаційної роботи
4. Зміст пояснювальної записки (перелік питань, які потрібно розробити) Аналіз предметної області, визначення її функціональних та нефункціональних вимог, постановка задачі, проектування програмного забезпечення, програмна реалізація, проведення тестування вебресурсу.
5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень) Три креслення:
1.Діаграма варіантів використання для зареєстрованого користувача
2.Схема бази даних
3.Архітектурне рішення.

6. Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Яшина О.М., доцент кафедри ІПЗ	25.05.26	26.05.26
Антиплагіат	Форкун Ю. В., доцент кафедри ІПЗ	01.05.26	21.05.26

7. Дата видачі завдання « 02 » січня 2026 р.

КАЛЕНДАРНИЙ ПЛАН

Назва етапів (розділів) кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1 Ознайомлення з тематикою кваліфікаційної роботи (КвР), визначення та узгодження індивідуальних тем КвР	01.12 – 31.12.2025	
2 Збір матеріалу за темою КвР; дослідження предметної області, в якій планується використання програмного забезпечення (ПЗ), визначення задач та вимог, розробка технічного завдання	02.01 – 31.01.2026	
3 Проектування програмного забезпечення	01.02 – 28.02.2026	
4 Програмна реалізація з використанням відповідних засобів розробки	01.03 – 10.04.2026	
5 Тестування програмного забезпечення	11.04 – 30.04.2026	
6 Написання вступу, загальних висновків, оформлення переліку джерел посилання та додатків. Оформлення пояснювальної записки КвР згідно вимог	01.05 – 25.05.2025	
7 Попередній захист КвР	травень 2026 (згідно графіка)	
8 Перевірка КвР на плагіат, нормоконтроль, отримання відгуків, рецензій та інших супровідних документів. Брошурування (зшиття) пояснювальної записки.	26.05 – 30.05.2026	
9 Здача КвР на кафедру; підготовка КвР для розміщення у репозиторії ХНУ; підготовка до захисту та захист КвР	з 01.06.2026	

Студент _____ Максим ЦИМБАЛЮК
Підпис Ініціали, прізвище

Керівник роботи _____ Юрій ФОРКУН
Підпис Ініціали, прізвище

АНОТАЦІЯ

Тема кваліфікаційної роботи: Вебплатформа для обміну та зберігання медіа-файлів

Автор роботи: Цимбалюк Максим Валентинович

Керівник роботи: Форкун Юрій Вікторович

Пояснювальна записка: 83 с., 19 рис., 5 табл., 2 дод., 30 джерел.

ВЕБСЕРВІС, MERN, REST API, REACT.JS, MONGODB, JWT, CLOUDINARY.

Метою роботи є розроблення вебсервісу MediaFlow для обміну мультимедійним контентом між користувачами, що забезпечує завантаження, зберігання та перегляд зображень, аудіо та відеоматеріалів у єдиному вебінтерфейсі.

Під час роботи над програмним забезпеченням здійснено аналітичний огляд предметної області, існуючих програмних рішень та засобів, окреслено мету та завдання для кваліфікаційної роботи, в кінці кожного розділу зроблено висновки, а також подано загальні висновки.

Для реалізації серверної частини використано Node.js/Express.js з MongoDB як системою керування базами даних та Cloudinary для хмарного зберігання медіафайлів. Клієнтська частина реалізована на React.js з використанням Redux Toolkit для управління станом, Material UI для компонентів, react-hook-form для валідації форм та axios для HTTP-комунікації.

25.05.2026
Дата

Підпис

ЗМІСТ

Змн.	Арк.	№ докум.	Підпис	Дата	КвРІПЗ. 2401218.01.21.ПЗ		
Виконав		Цимбалюк М.В.		25.05	Вебсервіс для обміну мультимедійним контентом між користувачами	Літ.	Арк. 1
Керівник		Форкун Ю.В.		25.05			
Н. контр.		Яшина О.М.		25.05		ХНУ, ІПЗ-21	
Зав. каф.		Бедратюк Л.П.		25.05			

ЗМІСТ

Змн.	Арк.	№ докум.	Підпис	Дата	КвРІПЗ. 2401218.01.21.ПЗ		
Виконав		Цимбалюк М.В.		25.05	Вебсервіс для обміну мультимедійним контентом між користувачами	Літ.	Арк. 1
Керівник		Форкун Ю.В.		25.05			
Н. контр.		Яшина О.М.		25.05		ХНУ, ІПЗ-21	
Зав. каф.		Бедратюк Л.П.		25.05			

ВСТУП

Стрімкий розвиток Інтернету та повсюдне поширення мобільних пристроїв зумовили кардинальні зміни у способах створення та споживання цифрового контенту. Завантаження, публікація та перегляд мультимедійних матеріалів – зображень, аудіозаписів та відео – стали невід'ємною частиною цифрового життя мільярдів людей. Щохвилини у мережі публікуються десятки тисяч одиниць медіаконтенту, а попит на зручні та функціональні платформи для його обміну постійно зростає.

Попри велику кількість існуючих сервісів, більшість із них орієнтована на один тип контенту: YouTube – виключно відео, SoundCloud – аудіо, Instagram та Pinterest – зображення. Відсутність універсальних платформ, що підтримують одночасно кілька типів мультимедіа в єдиному соціальному середовищі з повноцінними інструментами для авторів, формує незадоволений попит на ринку. Це підтверджується успіхом нішевих платформ на кшталт Newgrounds, що попри застарілий інтерфейс зберігає активну аудиторію саме завдяки мультитипній підтримці контенту.

Розроблення вебсервісу MediaFlow для обміну мультимедійним контентом між користувачами є актуальною задачею, що поєднує сучасні вебтехнології, хмарні сервіси зберігання та принципи UX-проєктування.

Метою кваліфікаційної роботи є розроблення вебсервісу MediaFlow для обміну мультимедійним контентом між користувачами, що забезпечує завантаження, зберігання та перегляд зображень, аудіо та відео в єдиному інтерфейсі.

Для успішного досягнення мети роботи необхідно розв'язати наступні задачі:

– провести аналіз предметної галузі та визначити ключові тенденції ринку вебсервісів для обміну мультимедіа;

– дослідити існуючі програмні рішення (Newgrounds, SoundCloud, Pixabay) та визначити їх переваги і недоліки;

– сформулювати функціональні та нефункціональні вимоги до розроблюваної системи;

– спроєктувати архітектуру вебсервісу на основі стеку MERN з REST API та JWT-автентифікацією;

– спроєктувати схему бази даних MongoDB та визначити API-інтерфейси між модулями;

– реалізувати серверну частину застосунку з підтримкою хмарного зберігання медіафайлів на Cloudinary;

– реалізувати клієнтську частину на React.js з адаптивним інтерфейсом та підтримкою темної і світлої теми;

– провести функціональне тестування вебсервісу.

Об’єкт дослідження — процес обміну мультимедійним контентом між користувачами в мережі Інтернет.

Предмет дослідження — методи та засоби розроблення клієнт-серверних вебзастосунків для публікації, зберігання та перегляду мультимедійного контенту.

Практична цінність роботи полягає у створенні повнофункціональної платформи MediaFlow, яка може бути використана як самостійний продукт або стати основою для комерційного медіахостингу.

1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Аналіз предметної області

Мультимедійний контент — це сукупність цифрових об’єктів, що поєднують різні форми представлення інформації: статичні зображення, аудіозаписи та відеоматеріали. В умовах масового розповсюдження смартфонів, широкосмугового доступу до Інтернету та розвитку соціальних мереж виробництво та споживання мультимедіа набуло безпрецедентних масштабів.

За даними аналітичного звіту Cisco Annual Internet Report (2018–2023), відеотрафік до 2023 року становив понад 82% усього інтернет-трафіку. Щохвилини на YouTube завантажується понад 500 годин відео, а аудіосервіс Spotify налічує понад 100 мільйонів треків у своєму каталозі. Ринок платформ для обміну контентом залишається одним з найдинамічніших сегментів веб-індустрії.

Особливого значення набув сегмент платформ для творчих авторів — художників, музикантів, відеографів, аніматорів. Ці платформи надають не лише інструменти для публікації, але й формують спільноти навколо контенту та створюють економічні можливості для незалежних авторів (creator economy).

1.1.1 Класифікація систем обміну мультимедійним контентом

Системи обміну мультимедійним контентом можна класифікувати за кількома основними ознаками.

За типом підтримуваного контенту:

– монотипні платформи — підтримують лише один тип медіа (наприклад, YouTube — виключно відео, SoundCloud — виключно аудіо);

– мультитипні платформи — підтримують кілька форматів одночасно (наприклад, Newgrounds — зображення, аудіо, відео, ігри).

За моделлю доступу до контенту:

– відкриті платформи — будь-який відвідувач може переглядати опублікований контент без реєстрації;

– закриті платформи — перегляд можливий лише після авторизації;

– гібридні платформи — частина контенту загальнодоступна, частина — лише для зареєстрованих.

За моделлю монетизації:

– безкоштовні з рекламою (ad-supported);

– freemium — базовий функціонал безкоштовний, розширений — платний;

– платні за підпискою (subscription-based);

– платформи з монетизацією авторів (revenue sharing).

За фокусом аудиторії:

– орієнтовані на широку аудиторію (масові платформи);

– орієнтовані на творців (creator platforms);

– стокові платформи (stock media) — зосереджені на пошуку та завантаженні медіаматеріалів, а не на соціальній взаємодії.

Розроблюваний вебсервіс MediaFlow відноситься до категорії мультитипних відкритих платформ, орієнтованих на творчих авторів. Реєстрація потрібна лише для завантаження контенту та соціальних дій (лайки, коментарі), тоді як перегляд доступний усім.

1.1.2 Тенденції розвитку галузі

Аналіз сучасного стану галузі виявляє кілька ключових тенденцій, що формують вимоги до нових платформ.

Зростання обсягів відеоконтенту. За прогнозами Cisco, до 2025 року відеотрафік становитиме понад 85% глобального IP-трафіку. Серед найбільш популярних форматів — короткі вертикальні відео (reels, shorts) тривалістю 15–60 секунд та подкасти у форматі аудіо.

Хмаризація зберігання медіаданих. Більшість сучасних платформ відмовилися від локального зберігання файлів на власних серверах на користь хмарних сервісів (AWS S3, Google Cloud Storage, Cloudinary). Це знижує операційні витрати та забезпечує глобальну доставку контенту через CDN.

Персоналізація стрічки. Алгоритми рекомендацій на основі машинного навчання стали стандартом для великих платформ. Однак нішеві та незалежні платформи часто обирають хронологічну або рейтингову стрічку як прозорішу альтернативу.

Підтримка multiple content types. Аудиторія авторів шукає платформи, що дозволяють публікувати різні типи контенту в єдиному профілі без необхідності вести кілька акаунтів на різних сервісах.

Мобільна доступність. Адаптивний вебдизайн та Progressive Web App (PWA) стали обов'язковими вимогами, оскільки понад 60% інтернет-трафіку генерується з мобільних пристроїв.

З урахуванням виявлених тенденцій, розроблюваний вебсервіс MediaFlow повинен підтримувати кілька типів медіаконтенту, використовувати хмарне зберігання, мати адаптивний інтерфейс та забезпечувати відкритий доступ для перегляду.

1.1.3 Ринок платформ для обміну мультимедіа

Ринок платформ для обміну мультимедійним контентом надзвичайно широкий та охоплює різні ніші. Розглянемо основні сегменти.

Відеохостинг: YouTube (2,5 млрд MAU), Vimeo (260 млн MAU), TikTok (1 млрд MAU). Ринок відеохостингу є найбільшим за обсягом трафіку, однак вхідний поріг для нових гравців надзвичайно високий через необхідність значних інфраструктурних витрат.

Аудіохостинг: SoundCloud (175 млн MAU), Spotify (602 млн MAU). Особливістю сегменту є важливість якості аудіо та наявності стримінгового плеєра.

Фотохостинг: Instagram (2 млрд MAU), Flickr (90 млн MAU), 500px. Найбільш конкурентний сегмент, що охоплює як любительський, так і професійний контент.

Стокові платформи: Pixabay, Unsplash, Pexels, Shutterstock. Орієнтовані на пошук та завантаження безкоштовних або платних медіаматеріалів.

Платформи для творчих авторів: Newgrounds, DeviantArt, ArtStation. Найближча ніша для MediaFlow — мультитипні платформи з акцентом на творчу спільноту.

Аналіз ринку підтверджує, що сегмент мультитипних платформ з підтримкою зображень, аудіо та відео в єдиному просторі залишається незаповненим сучасними технологічними рішеннями з якісним інтерфейсом. Саме цю нішу займає MediaFlow.

1.2 Аналіз наявного програмного забезпечення предметноїобласті

Для формування вимог до розроблюваного вебсервісу проведено детальний аналіз трьох найближчих за функціональністю аналогів: Newgrounds, SoundCloud та Pixabay. Вибір обумовлений тим, що кожна платформа репрезентує окрему субкатегорію мультимедіахостингу та розрахована на творчу аудиторію.

1.2.1 Newgrounds

Newgrounds (newgrounds.com) — одна з найстаріших веб-платформ для творчих авторів, заснована Томом Фулпом у 1995 році. Початково орієнтована на Flash-анімації та браузерні ігри, сьогодні платформа підтримує публікацію арт-робіт, аудіо, відео, ігор та мультиплікації. Станом на 2024 рік платформа налічує понад 500 тисяч зареєстрованих авторів.

Архітектура та технологічний стек. Newgrounds побудований на PHP із власноруч розробленим фреймворком, що використовується з кінця 1990-х. Медіафайли зберігаються на власній інфраструктурі. Платформа не має відкритого API. Перехід на сучасні технології ускладнений масивною legacy-кодовою базою.

Функціональність. Платформа підтримує завантаження анімацій, арт-робіт, аудіо та відео. Наявна система категорій, тегів, рейтингу (0–5 зірок), коментарів та форумів. Профілі авторів включають портфоліо та статистику. Підтримка різних типів медіа є ключовою перевагою.

Інтерфейс. Інтерфейс платформи суттєво застарів та не відповідає сучасним стандартам UX/UI. Відсутній адаптивний дизайн для мобільних пристроїв. Навігація ускладнена великою кількістю застарілих елементів.

Переваги:

– підтримка кількох типів медіаконтенту в єдиному просторі;

– активна спільнота авторів та аудиторія;

– система рейтингів та коментарів;

– категорії та теги для навігації.

Недоліки:

– застарілий та незручний інтерфейс;

– відсутність адаптивності для мобільних пристроїв;

– застаріла технологічна база;

– відсутність відкритого REST API;

– відсутність хмарного зберігання (власна інфраструктура).

1.2.2 SoundCloud

SoundCloud (soundcloud.com) — провідна платформа для публікації та прослуховування аудіоконтенту, заснована в Берліні у 2007 році. Орієнтована на незалежних музикантів, діджеїв, подкастерів та музичних продюсерів. Станом на 2024 рік налічує понад 175 мільйонів активних користувачів та понад 300 мільйонів доріжок.

Технологічний стек. SoundCloud використовує мікросервісну архітектуру з Ruby on Rails на бекенді, React.js на фронтенді. Зберігання аудіофайлів на AWS S3. Власний CDN для стримінгу. Публічний API доступний для партнерів.

Функціональність. Ключовою особливістю є вбудований стримінговий аудіоплеєр з відображенням хвильової форми (waveform), що дозволяє залишати коментарі до конкретних таймкодів треку. Наявна система підписок та фоловерів, статистика відтворень та завантажень, приватні треки та плейлисти.

Переваги:

– вбудований стримінговий плеєр з waveform-візуалізацією;

– система коментарів до конкретних таймкодів;

– розширена статистика для авторів;

– публічний API;

– хмарне зберігання (AWS S3).

Недоліки:

– підтримка виключно аудіоконтенту (без зображень та відео);

– обмежена безкоштовна квота (180 хвилин завантаження);

– складна модель монетизації для авторів;

– відсутність підтримки зображень як самостійного типу контенту.

1.2.3 Pixabay

Pixabay (pixabay.com) — платформа для публікації та пошуку безкоштовних стокових зображень, відео та музики, ліцензованих під Pixabay License. Заснована в 2010 році, станом на 2024 рік налічує понад 4 мільйони медіаматеріалів від більш ніж 1 мільйона авторів.

Технологічний стек. Pixabay використовує PHP (Symfony) на бекенді, Vue.js на фронтенді. Зберігання файлів на власній інфраструктурі з CDN. Публічний REST API для розробників.

Функціональність. Потужна система пошуку з розширеними фільтрами (колір, орієнтація, розмір, тип файлу). Підтримка зображень, відео та аудіо (безкоштовне завантаження). Ліцензування контенту під власною ліцензією. Профілі авторів зі статистикою завантажень. Вбудований API з широкими можливостями для пошуку.

Переваги:

– розширена система пошуку та фільтрації;

– підтримка зображень, відео та аудіо;

– безкоштовне завантаження всіх матеріалів;

– публічний REST API;

– чіткий та зручний інтерфейс.

Недоліки:

– мінімальна соціальна складова (відсутні коментарі та лайки);

– орієнтована на пошук і завантаження, а не на взаємодію між авторами;

– ліцензія Pixabay License не є стандартизованою (не Creative Commons);

– відсутній аудіоплеєр для прослуховування треків онлайн.

1.2.4 Порівняльний аналіз платформ

На основі проведеного аналізу складено порівняльну таблицю (таблиця 1.1) за ключовими функціональними та технічними характеристиками.

Таблиця 1.1 – Порівняльний аналіз існуючих систем обміну мультимедійним контентом

Характеристика	Newgrounds	SoundCloud	Pixabay	MediaFlow (розроб.)
Зображення	Так	Ні	Так	Так
Аудіо	Так	Так	Так	Так
Відео	Так	Ні	Так	Так
Система лайків	Так	Так	Ні	Так
Коментарі	Так	Так (таймкоди)	Ні	Так
Профіль автора	Так	Так	Так	Так
Теги та категорії	Так	Так	Так	Так
Пошук	Базовий	Розширений	Розширений	Базовий+
REST API	Ні	Так (партнери)	Так (публічний)	Так (відкритий)

Характеристика	Newgrounds	SoundCloud	Pixabay	MediaFlow (розроб.)
Хмарне сховище	Ні	Так (AWS)	Ні (власне)	Так (Cloudinary)
Адаптивний дизайн	Ні	Так	Так	Так
Темна тема	Ні	Так	Ні	Так
Відкрита ліцензія контенту	Ні	Ні	Так	Ні

Проведений порівняльний аналіз підтверджує, що жодна з розглянутих платформ не задовольняє повністю потреби незалежних авторів у публікації різнотипного мультимедійного контенту з повноцінною соціальною складовою та сучасним адаптивним інтерфейсом. MediaFlow покликана об’єднати переваги кожного з аналогів, усуваючи при цьому їх ключові недоліки.

1.3 Визначення вимог до вебсервісу

1.3.1 Актори системи

У системі виділено три категорії акторів, що взаємодіють з вебсервісом з різними правами доступу (Рисунок 1.1).

Гість (незареєстрований відвідувач) — будь-який користувач, що відвідує платформу без авторизації. Має право переглядати стрічку контенту, переглядати окремі медіаматеріали, профілі авторів та здійснювати пошук. Не може завантажувати контент, ставити лайки або залишати коментарі.

Зареєстрований користувач (автор) — користувач, що пройшов реєстрацію та автентифікацію. Отримує повний доступ до всіх функцій платформи: завантаження

медіафайлів, управління власним контентом, соціальна взаємодія, редагування профілю.

Адміністратор — привілейований користувач з роллю «admin» в системі. Має право видаляти будь-який контент та коментарі незалежно від авторства, управляти обліковими записами користувачів.

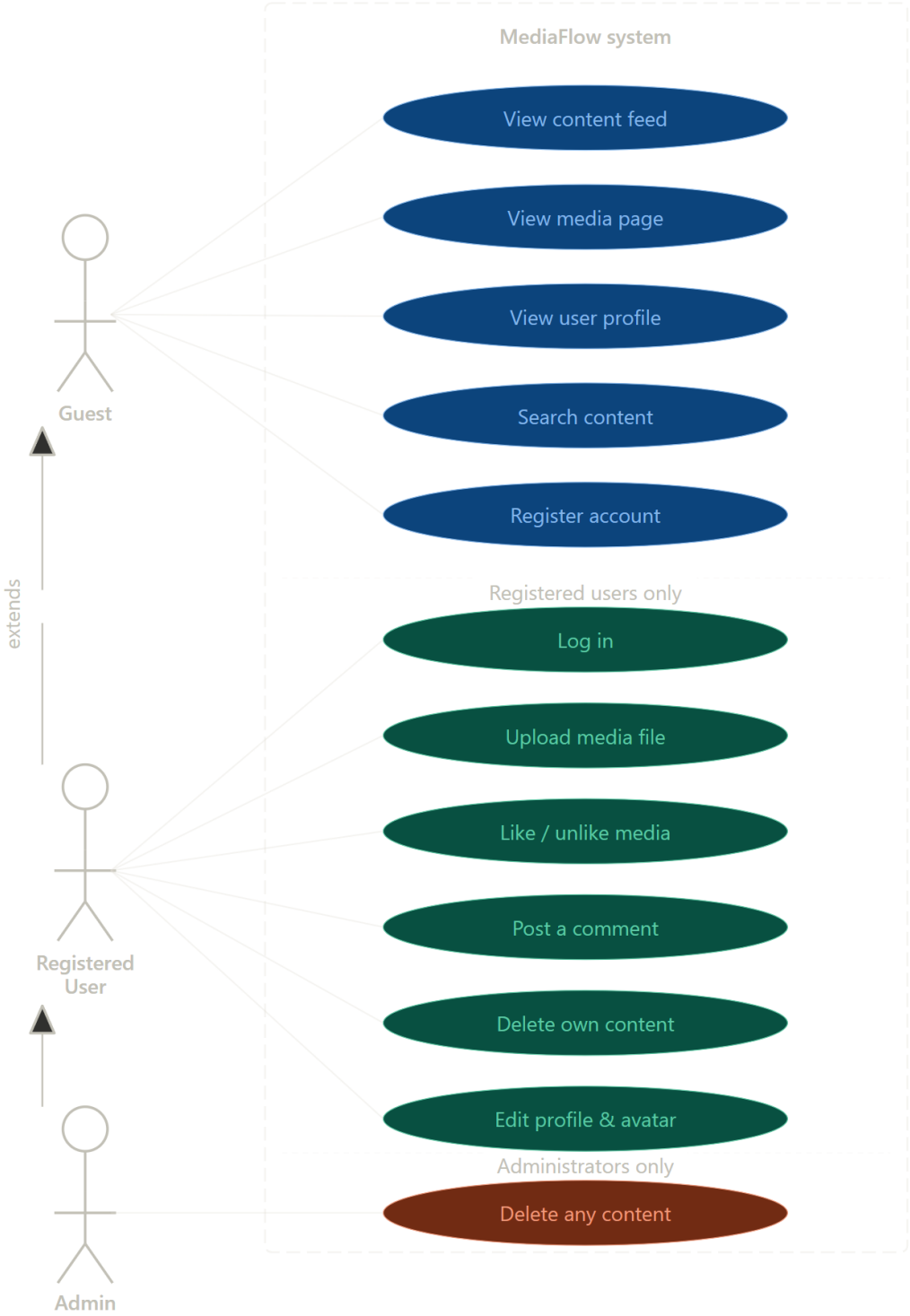


Рисунок 1.1 – Актори системи

1.3.2 Функціональні вимоги

На основі аналізу предметної галузі та аналогів сформульовано функціональні вимоги до системи.

Вимоги до модуля автентифікації:

– ФВ-1. Система повинна надавати можливість реєстрації нового облікового запису із зазначенням унікального нікнейму, email-адреси та пароля.

– ФВ-2. Система повинна виконувати валідацію даних при реєстрації: перевірку унікальності нікнейму та email, мінімальної довжини пароля (6 символів), формату email.

– ФВ-3. Система повинна забезпечувати автентифікацію за email та паролем з видачею JWT-токена.

– ФВ-4. Система повинна зберігати хеш пароля (алгоритм bcrypt, factor=10), а не пароль у відкритому вигляді.

– ФВ-5. Система повинна автоматично завершувати сесію при спробі виконати захищену дію з недійсним або простроченим токеном.

Вимоги до модуля медіаконтенту:

– ФВ-6. Система повинна підтримувати завантаження файлів трьох типів: зображення (JPEG, PNG, GIF, WEBP), аудіо (MP3, WAV, OGG), відео (MP4, WEBM).

– ФВ-7. Система повинна обмежувати розмір завантажуваного файлу до 100 МБ.

– ФВ-8. Система повинна зберігати завантажені файли у хмарному сховищі Cloudinary.

– ФВ-9. Для зображень система повинна автоматично генерувати стиснуте превью розміром 400×300 пікселів.

– ФВ-10. Система повинна відображати стрічку контенту всіх авторів з підтримкою фільтрації за типом медіа, категорією та тегом.

– ФВ-11. Система повинна підтримувати текстовий пошук за назвою, описом та тегами.

– ФВ-12. Система повинна надавати можливість сортування стрічки за датою публікації, кількістю лайків та переглядів.

– ФВ-13. Система повинна реалізувати пагінацію стрічки (12 елементів на сторінку).

– ФВ-14. Система повинна інкрементувати лічильник переглядів при завантаженні сторінки медіаматеріалу.

– ФВ-15. Система повинна надавати можливість видалення власного медіаматеріалу (включно з файлом на Cloudinary).

Вимоги до модуля соціальної взаємодії:

– ФВ-16. Система повинна надавати можливість зареєстрованому користувачу поставити або зняти лайк з медіаматеріалу (один лайк від одного користувача).

– ФВ-17. Система повинна надавати можливість залишати текстові коментарі до медіаматеріалів (до 500 символів).

– ФВ-18. Система повинна надавати можливість видалення власного коментаря.

Вимоги до модуля профілю користувача:

– ФВ-19. Система повинна відображати публічний профіль користувача з портфоліо його робіт та статистикою (кількість завантажень, сумарні лайки, перегляди).

– ФВ-20. Система повинна надавати можливість редагування біографії профілю (до 300 символів).

– ФВ-21. Система повинна надавати можливість завантаження аватара (зображення до 5 МБ) із автоматичним кадруванням до 200×200 пікселів.

1.3.3 Нефункціональні вимоги

Нефункціональні вимоги визначають якісні характеристики системи:

– НФВ-1. Продуктивність: час відповіді сервера на API-запити (без урахування завантаження файлів) не повинен перевищувати 500 мс для стандартних операцій при навантаженні до 100 одночасних користувачів.

– НФВ-2. Надійність: система повинна коректно обробляти всі виняткові ситуації та повертати клієнту зрозумілі повідомлення про помилки з відповідними HTTP-кодами статусу.

– НФВ-3. Безпека: паролі зберігаються у хешованому вигляді (bcrypt, salt rounds=10). Автентифікація на основі JWT з терміном дії 7 діб. CORS налаштований лише для дозволених доменів.

– НФВ-4. Масштабованість: архітектура повинна допускати горизонтальне масштабування серверних інстанцій без модифікації коду (stateless-сервер з JWT).

– НФВ-5. Зручність використання: інтерфейс повинен бути адаптивним та коректно відображатися на екранах шириною від 320px. Підтримка темної та світлої теми.

– НФВ-6. Підтримуваність: серверна частина організована за MVC-патерном, клієнтська — за Flux-архітектурою (Redux). Код структурований за принципом єдиної відповідальності.

– НФВ-7. Сумісність: коректна робота в браузерах Google Chrome 90+, Mozilla Firefox 88+, Microsoft Edge 90+, Safari 14+.

1.3 Висновки. Постановка задачі

У першому розділі проведено комплексний аналіз предметної галузі вебсервісів для обміну мультимедійним контентом. Визначено, що ринок мультитипних платформ для творчих авторів демонструє стале зростання та залишається незадоволеним існуючими сервісами. Жодна з розглянутих платформ (Newgrounds, SoundCloud, Pixabay) не поєднує повноцінну підтримку зображень, аудіо та відео з сучасним адаптивним інтерфейсом та розвиненою соціальною складовою.

Результатом аналізу є сформульовані 21 функціональна та 7 нефункціональних вимог, що утворюють повну специфікацію розроблюваного вебсервісу MediaFlow. Визначено трьох акторів системи: гість, зареєстрований користувач та адміністратор. Ці вимоги є основою для проєктування, що виконується в наступному розділі.

Постановка задачі: розробити вебсервіс MediaFlow для обміну мультимедійним контентом між користувачами на основі стеку MERN (MongoDB, Express.js, React.js, Node.js), що забезпечує завантаження зображень, аудіо та відео на хмарне сховище

Cloudinary, відображення адаптивної стрічки контенту з фільтрацією та пошуком, систему JWT-автентифікації, коментарі, лайки та профілі авторів.

2. ПРОЄКТУВАННЯ ВЕБСЕРВІСУ

2.1 Вибір типу архітектури та шаблонів проєктування

На основі аналізу вимог до вебсервісу MediaFlow обрано клієнт-серверну архітектуру з чітким розподілом відповідальностей між серверною (backend) та клієнтською (frontend) частинами. Взаємодія здійснюється виключно через REST API по протоколу HTTPS.

2.1.1 Клієнт-серверна архітектура

Клієнт-серверна архітектура є де-факто стандартом для сучасних вебзастосунків. Вона забезпечує:

– незалежність розробки — команди фронтенду та бекенду можуть працювати паралельно;

– замінність клієнтів — один і той самий API може обслуговувати вебзастосунок, мобільний додаток або сторонній клієнт;

– масштабованість — серверна та клієнтська частини масштабуються незалежно.

Клієнтська частина реалізована як Single Page Application (SPA) на React.js. SPA-підхід забезпечує плавну навігацію без перезавантаження сторінки, що суттєво покращує UX. Маршрутизація обробляється на стороні клієнта за допомогою React Router v6.

2.1.2 REST API

Взаємодія між клієнтом та сервером реалізована через REST API (Representational State Transfer) (Рисунок 2.1).

Вибір REST обумовлений такими факторами:

– використання стандартних HTTP-методів (GET, POST, PUT, DELETE) для семантично чіткого вираження операцій;

– використання стандартних HTTP-кодів статусу (200, 201, 400, 401, 403, 404, 500);

– незалежність від мови програмування — API може використовуватись будь-яким клієнтом;

– широка підтримка інструментів тестування (Postman, Insomnia, curl);

– зрілість специфікації та велика кількість документації.

Кожен ресурс системи (медіаматеріали, користувачі, коментарі, лайки) представлений окремим URI-ендпоінтом. Тіло запитів та відповідей передається у форматі JSON.

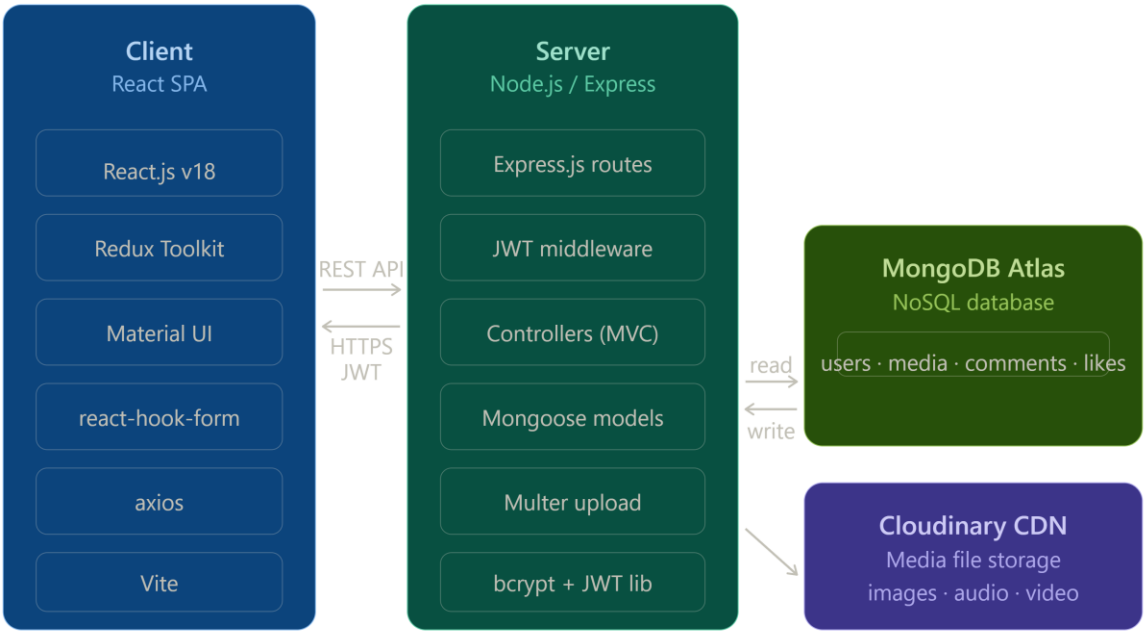


Рисунок 2.1 – Клієнт-серверна архітектура

2.1.3 JWT-автентифікація

Автентифікація реалізована на основі JSON Web Token (JWT) — відкритого стандарту RFC 7519. JWT є рядком, що складається з трьох частин, розділених

крапкою: заголовок (header), корисне навантаження (payload) та підпис (signature).

Заголовок містить алгоритм підпису (HS256 — HMAC SHA-256). Корисне навантаження містить унікальний ідентифікатор користувача (id) та час закінчення терміну дії (exp). Підпис генерується шляхом хешування заголовка та навантаження разом із секретним ключем сервера.

При автентифікації клієнт отримує JWT і зберігає його в localStorage. Кожен наступний запит до захищених ендпоінтів містить JWT в заголовку: Authorization: Bearer <token>. Сервер верифікує підпис токена без звернення до бази даних, що забезпечує горизонтальну масштабованість.

Перевагою JWT перед сесіями на стороні сервера є stateless-характер — сервер не зберігає жодної інформації про авторизованих користувачів, що усуває необхідність у shared session storage при масштабуванні.

2.2 Опис декомпозиції

2.2.1 Загальна декомпозиція системи

Система MediaFlow декомпозована на два рівні: серверний (backend) та клієнтський (frontend). Між ними визначено чіткий контракт взаємодії у вигляді REST API (Рисунок 2.2).

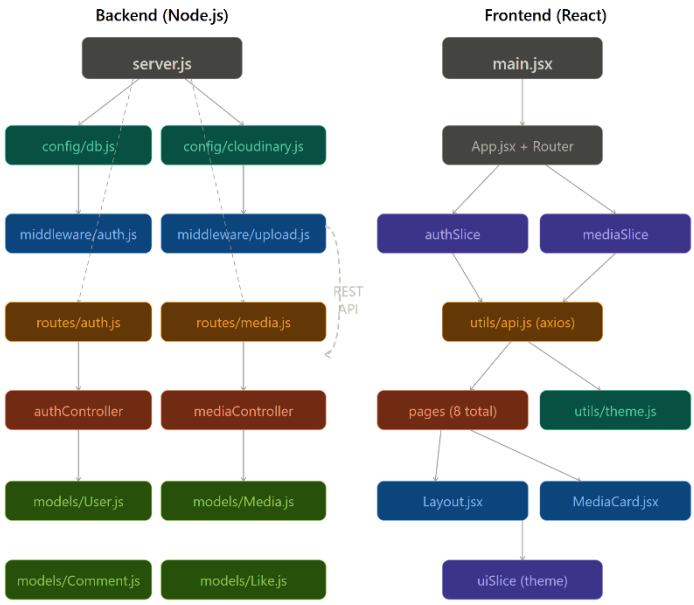


Рисунок 2.2 – Система MediaFlow

2.2.2 Декомпозиція серверних модулів

Модуль автентифікації (authController.js) реалізує три операції: register (реєстрація нового користувача з хешуванням пароля та генерацією JWT), login (перевірка облікових даних та видача JWT) та getMe (повернення даних поточного авторизованого користувача).

Модуль медіаконтенту (mediaController.js) є центральним та реалізує: uploadMedia (стримінгове завантаження на Cloudinary, збереження метаданих у MongoDB), getMedia (стрічка з фільтрацією та пагінацією), getMediaById (деталі одного матеріалу з інкрементом переглядів), deleteMedia (видалення з Cloudinary та MongoDB), toggleLike (атомарне перемикання лайку).

Модуль профілю (userController.js): getUserProfile (публічний профіль з портфоліо та статистикою через MongoDB aggregation), updateProfile (редагування біографії), updateAvatar (завантаження аватара з обрізанням 200×200px через Cloudinary transformation).

Модуль коментарів (commentController.js): getComments (список коментарів до матеріалу), addComment (додавання коментаря з оновленням commentsCount у Media), deleteComment (видалення власного коментаря).

2.2.3 Декомпозиція клієнтських модулів

authSlice керує станом автентифікації: user (об’єкт поточного користувача), token (JWT з localStorage), loading, error, initialized. Реалізує асинхронні операції через createAsyncThunk: register, login, fetchMe. Синхронні дії: logout (очищення стану та localStorage), clearError, setInitialized.

mediaSlice керує станом медіаконтенту: feed (масив медіаматеріалів стрічки), currentMedia (поточний переглянутий матеріал), total та pages (для пагінації), loading, uploading. Асинхронні операції: fetchMedia, fetchMediaById, uploadMedia, toggleLike, deleteMedia.

uiSlice зберігає стан інтерфейсу: themeMode (‘dark’ або ‘light’). Значення зберігається в localStorage при зміні.

2.3 Опис залежностей

Залежності між модулями системи MediaFlow визначають потік даних та відповідальності кожного компонента.

Серверні залежності:

– server.js → config/db.js (ініціалізація підключення до MongoDB при старті);

– routes/*.js → middleware/auth.js (захист ендпоінтів через protect/optionalAuth);

– routes/media.js → middleware/upload.js (обробка файлів через Multer+Cloudinary);

– controllers/*Controller.js → models/*.js (CRUD-операції через Mongoose);

– controllers/mediaController.js → config/cloudinary.js (видалення файлів через Cloudinary API);

– controllers/userController.js → config/cloudinary.js (завантаження та видалення аватарів).

Клієнтські залежності:

– App.jsx → features/auth/authSlice (ініціалізація стану автентифікації);

– pages/*.jsx → features/media/mediaSlice (отримання та мутація стану медіаконтенту);

– features/*.slice.js → utils/api.js (HTTP-запити через axios instance);

– utils/api.js → localStorage (зчитування та очищення JWT);

– main.jsx → utils/theme.js + features/ui/uiSlice (динамічна генерація MUI-теми);

– components/layout/Layout.jsx → features/auth/authSlice + features/ui/uiSlice (навбар).

Зовнішні залежності:

– backend → MongoDB Atlas (зберігання метаданих через mongoose);

– backend → Cloudinary API (зберігання медіафайлів та генерація превью);

– frontend → backend REST API (всі HTTP-запити через /api/*).

2.4 Опис інтерфейсу модулів

Інтерфейс між клієнтською та серверною частинами визначається REST API. У таблиці 2.2 наведено повний перелік ендпоінтів системи.

Таблиця 2.2 – REST API ендпоінти вебсервісу MediaFlow

Метод	URI	Доступ	Опис	Тіло запиту / Параметри
POST	/api/auth/register	Публічний	Реєстрація	username, email, password
POST	/api/auth/login	Публічний	Вхід, отримання JWT	email, password
GET	/api/auth/me	JWT	Дані поточного user	—
GET	/api/media	Публічний	Стрічка + фільтри	query: category, mediaType, tag, search, sort, page
POST	/api/media	JWT	Завантаження медіа	form-data: file, title, description, category, tags
GET	/api/media/:id	Публічний	Деталі медіа + +1 view	—
DELETE	/api/media/:id	JWT (автор)	Видалення медіа	—
POST	/api/media/:id/like	JWT	Лайк / Анлайк	—
GET	/api/users/:username	Публічний	Профіль + портфоліо	—

Метод	URI	Доступ	Опис	Тіло запиту / Параметри
PUT	/api/users/profile	JWT	Оновлення біографії	bio
PUT	/api/users/avatar	JWT	Оновлення аватара	form-data: file
GET	/api/comments/:mediaId	Публічний	Список коментарів	—
POST	/api/comments/:mediaId	JWT	Новий коментар	text
DELETE	/api/comments/:id	JWT (автор)	Видалення коментаря	—

Всі відповіді API повертаються у форматі JSON. У разі помилки відповідь містить поле message з текстом помилки та відповідний HTTP-код статусу. Успішне завантаження медіаматеріалу повертає HTTP 201 Created, решта успішних операцій — HTTP 200 OK. Видалення повертає JSON-об’єкт з полем message, а не HTTP 204 No Content, що спрощує обробку на клієнті.

2.5 Проєктування інтерфейсу користувача

При проєктуванні інтерфейсу вебсервісу MediaFlow дотримано принципів Material Design та специфіки мультимедіа-платформ. Ключовим рішенням є підтримка темної та світлої теми з акцентним кольором #5865F2 (Discord-blurple), що асоціюється з сучасними креативними платформами.

Головна сторінка / Стрічка контенту. Основний вигляд — адаптивна сітка карток медіаматеріалів (4 колонки на широких екранах, 3 — на планшеті, 2 — на смартфоні, 1 — на вузькому екрані). Кожна картка містить: превью або іконку типу

медіа, назву, аватар та нікнейм автора, лічильники лайків та переглядів, категорію у вигляді чіпа.

Панель фільтрів. Розміщена вище сітки: поле текстового пошуку, кнопки перемикання типу медіа (ToggleButtonGroup), випадаючий список категорій, випадаючий список сортування. При активних фільтрах відображаються чіпи для індивідуального скидання.

Сторінка медіаматеріалу. Верхня частина — медіаплеєр (нативний img/audio/video HTML-елемент). Нижче — метаінформація (назва, автор, категорія, теги, дата), кнопка лайку з лічильником. Секція коментарів з полем вводу для авторизованих та пропозицією авторизуватись для гостей.

Сторінка завантаження. Двоколонний layout: дропзона для завантаження файлу (drag & drop або вибір через діалог) та форма з полями назви, опису, категорії та тегів. Після вибору файлу дропзона відображає його назву, тип та розмір.

Сторінка профілю. Верхня частина — шапка з аватаром, нікнеймом, біографією та статистикою (завантаження, лайки, перегляди). Для власного профілю — кнопка «Редагувати профіль». Нижче — сітка робіт автора.

Навігаційна панель. Зафіксована зверху (sticky). Зліва — логотип MediaFlow з градієнтним ефектом. Праворуч — перемикач теми, кнопка «Upload» та аватар користувача з dropdown-меню (або кнопки Login/Register для гостей).

2.6 Детальне проєктування модулів

2.6.1 Схема бази даних

База даних MongoDB включає чотири колекції. Для оптимізації запитів визначено індекси.

Колекція users: поля username (String, unique, 3-30 символів), email (String, unique, lowercase), password (String, bcrypt-хеш), avatar (String, URL), avatarPublicId (String, Cloudinary public_id), bio (String, до 300 символів), role (String, enum: 'user'|'admin', default: 'user'), timestamps (createdAt, updatedAt). Визначено хуки pre('save') для хешування пароля та методи matchPassword та toPublic.

Колекція media: поля title (String, обов'язкове, до 100 символів), description (String, до 2000 символів), fileUrl (String, URL на Cloudinary), publicId (String, Cloudinary public_id), thumbnailUrl (String, URL превью), mediaType (String, enum: 'image'|'audio'|'video'),

category (String, enum: art|music|video|animation|photography|other), tags (Array<String>), author (ObjectId → users), views (Number, default: 0), likesCount (Number, default: 0), commentsCount (Number, default: 0). Індекси: { category, createdAt }, { author }, { tags }, повнотекстовий на title+description+tags.

Колекція comments: поля text (String, до 500 символів), author (ObjectId → users), media (ObjectId → media). Індекс: { media, createdAt: -1 }.

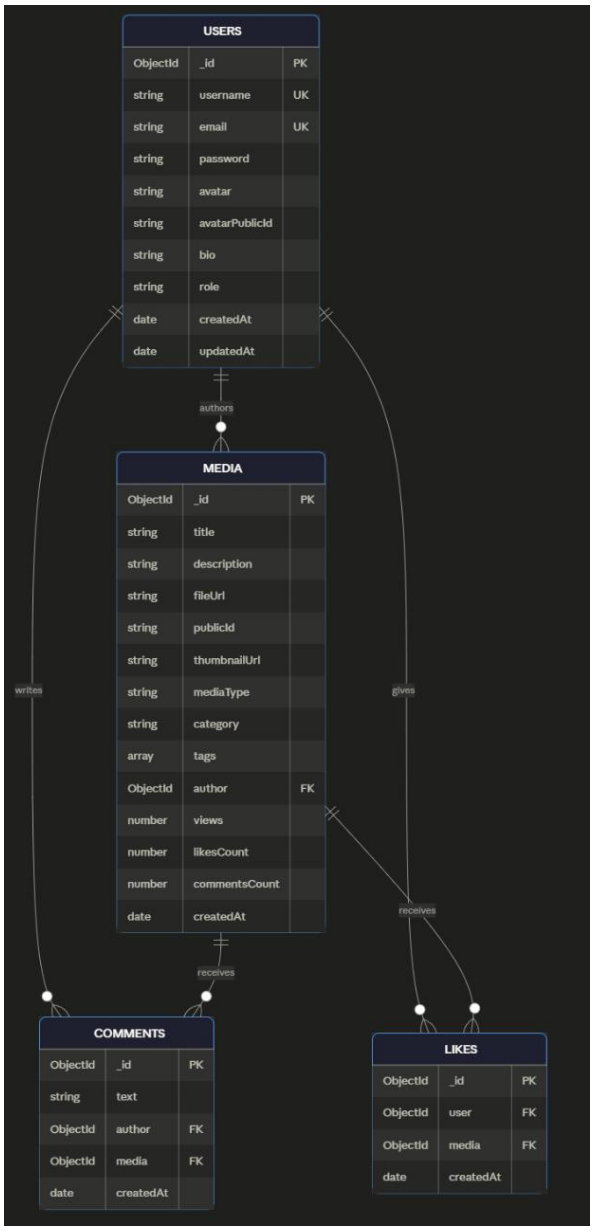


Рисунок 2.2 – Схема бази даних

2.6.2 Алгоритм завантаження медіафайлу

Процес завантаження медіафайлу на платформу включає такі кроки:

– Клієнт відправляє POST /api/media з multipart/form-data, що містить файл та метадані.

– Middleware protect верифікує JWT та завантажує об’єкт поточного користувача в req.user.

– Middleware upload (Multer з CloudinaryStorage) перевіряє MIME-тип файлу та його розмір (≤100 МБ). Файл стримінгово завантажується на Cloudinary без збереження на диску сервера.

– Якщо файл є зображенням, автоматично застосовується Cloudinary-трансформація для генерації превью: ширина 400px, висота 300px, режим обрізання fill, якість auto.

– Контролер uploadMedia визначає mediaType на основі MIME-типу, формує об’єкт медіаматеріалу та зберігає його в колекцію media через Mongoose.

– Об’єкт медіаматеріалу заповнюється (populate) даними автора та повертається клієнту з HTTP 201 Created.

2.6.3 Алгоритм фільтрації та пагінації стрічки

Функція getMedia реалізує динамічну фільтрацію через формування об’єкта filter залежно від query-параметрів: category та mediaType встановлюються безпосередньо, tag призначається в поле tags ($in-запит в MongoDB), search активує $text-оператор повнотекстового пошуку. Сортування обирається з трьох опцій (newest → {createdAt: -1}, popular → {likesCount: -1}, views → {views: -1}). Пагінація реалізована через skip=(page-1)×limit та limit=12.

Паралельно виконуються два запити: отримання даних та підрахунок загальної кількості (Promise.all), що мінімізує час очікування.

2.7 Аналіз та вибір технологій і методів реалізації

2.7.1 Серверна частина

Node.js. Середовище виконання JavaScript поза браузером на основі рушія V8 від Google. Обрано через можливість використання єдиної мови (JavaScript/TypeScript) для клієнтської та серверної частин, що скорочує cognitive overhead розробника. Неблокуюча I/O-модель (event loop) є особливо ефективною при обробці файлових операцій та великої кількості одночасних HTTP-підключень — ключових операцій для медіахостингу. Версія 18 LTS.

Express.js. Мінімалістичний та гнучкий фреймворк для Node.js, що надає маршрутизацію, middleware-конвеєр та набір утиліт для роботи з HTTP. Обрано через відсутність зайвих абстракцій (на відміну від NestJS чи Fastify з їх IoC-контейнерами), широку екосистему middleware-пакетів, документованість та зрілість (запущений у 2010 р.).

MongoDB Atlas. Повністю керована хмарна база даних MongoDB як сервіс. Документо-орієнтована NoSQL СУБД обрана через гнучку схему документів (не потрібно ALTER TABLE при додаванні нових полів медіаматеріалів), нативну підтримку JSON-подібних документів, природну інтеграцію з JavaScript. MongoDB Atlas надає безкоштовний M0-тір (512 МБ), автоматичне резервне копіювання та глобальні кластери.

Mongoose. ODM (Object Document Mapper) для MongoDB. Надає схеми з валідацією, middleware-хуки (pre/post), методи документів та статичні методи моделей. Mongoose abstractує низькорівневий MongoDB Node.js Driver та суттєво зменшує кількість шаблонного коду.

Cloudinary. Хмарний сервіс зберігання та трансформації медіафайлів. Обрано через: безкоштовний план (25 ГБ сховища, 25 кредитів/міс.), вбудовані on-the-fly трансформації (зміна розміру, обрізання, стиснення) через URL-параметри, підтримку зображень, відео та аудіо, CDN-доставку контенту. Інтеграція через multer-storage-cloudinary дозволяє стримінгово завантажувати файли на Cloudinary без проміжного збереження на диску сервера.

jsonwebtoken, bcryptjs. jsonwebtoken — найпопулярніша Node.js-реалізація стандарту JWT (RFC 7519). bcryptjs — pure-JavaScript реалізація алгоритму bcrypt без нативних залежностей, що спрощує розгортання.

2.7.2 Клієнтська частина

Бібліотека React.js (v18) для побудови користувацьких інтерфейсів, розроблена Facebook. Обрано через декларативний компонентний підхід, Concurrent Mode та автоматичне батчування в React 18, що покращує продуктивність при частих оновленнях UI, велику екосистему та спільноту.

Redux Toolkit (RTK). Офіційний набір інструментів для Redux. Суттєво зменшує обсяг шаблонного коду завдяки createSlice (автогенерація action creators та reducer), createAsyncThunk (обробка асинхронних операцій з автоматичними pending/fulfilled/rejected-станами) та immer (іммутабельні оновлення через мутабельний синтаксис).

Material UI (MUI v5). Бібліотека React-компонентів, що реалізує Google Material Design 3. Надає 50+ готових компонентів (TextField, Button, Card, Grid, Dialog та ін.), систему темізації через ThemeProvider, підтримку CSS-in-JS через @emotion. Обрано через швидкість розробки та якість компонентів.

react-hook-form. Бібліотека управління формами, що використовує нативний DOM та ref-based підхід замість controlled components. Ключова перевага — мінімальна кількість ре-рендерів (validation on submit/blur, не на кожне натискання клавіші), що покращує продуктивність форм з великою кількістю полів.

HTTP-клієнт з підтримкою браузерних та Node.js середовищ. Обрано через зручний API для конфігурації базового URL та заголовків, підтримку request/response interceptors для централізованої обробки авторизації та помилок, автоматичне перетворення JSON.

Vite. Сучасний інструмент збірки, що використовує ES-модулі в режимі розробки (без попередньої збірки) та Rollup для production-бандла. Забезпечує

практично миттєвий холодний старт та блискавичний Hot Module Replacement, що суттєво пришвидшує розробку порівняно з webpack.

2.8 Висновки

У другому розділі спроєктовано повну архітектуру вебсервісу MediaFlow. Обрано клієнт-серверну архітектуру з REST API та JWT-автентифікацією як основу системи. Серверна частина організована за MVC-патерном, клієнтська — за Flux-архітектурою (Redux Toolkit).

Спроєктовано схему бази даних MongoDB з чотирма колекціями та оптимізованими індексами. Визначено повний перелік 14 REST API ендпоінтів з описом методів, шляхів, прав доступу та форматів даних.

Обрано технологічний стек на основі MERN (MongoDB, Express.js, React.js, Node.js), доповнений Redux Toolkit, MUI, react-hook-form, axios, Cloudinary та Vite. Кожна технологія обґрунтована з урахуванням вимог до системи, переваг над альтернативами та практичних факторів (ціна, підтримка, екосистема).

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ

3.1 Особливості програмної реалізації

3.1.1 Структура серверного застосунку

Серверний застосунок ініціалізується у файлі server.js. Підключаються middleware глобального рівня: cors (дозволяє запити з клієнтського домену), express.json (парсинг JSON-тіла запиту), morgan (логування запитів у форматі dev). Підключення до MongoDB виконується асинхронно через connectDB при старті застосунку. Всі маршрути підключаються через app.use з базовим шляхом /api/*.

Глобальний middleware обробки помилок визначений останнім із чотирма параметрами (err, req, res, next) — Express розпізнає його саме за кількістю параметрів. Перехоплює всі помилки, передані через next(err) або кинуті в async-контролерах.

Для зручності розробки використовується nodemon — утиліта автоматичного перезапуску Node.js-процесу при зміні файлів. Конфігурація зберігається в .env та завантажується через dotenv на першому рядку server.js.

3.1.2 Налаштування Cloudinary та Multer

Конфігурація Cloudinary у файлі config/cloudinary.js ініціалізує cloudinary.v2 з ключами з .env та визначає CloudinaryStorage — динамічне Multer-сховище, що для кожного файлу обчислює папку, resource_type та трансформації залежно від MIME-типу.

Cloudinary використовує resource_type "video" як для відео, так і для аудіо (оскільки аудіофайли є підмножиною медіаресурсів у термінології API).

Зображення трансформуються: {width: 1920, height: 1080, crop: "limit", quality: "auto"} — не збільшуються понад оригінал, якість визначається автоматично.

Multer-middleware у файлі middleware/upload.js визначає fileFilter, що дозволяє лише дозволені MIME-типи, та limits.fileSize = 100МБ. Формат завантаження — single("file"), що відповідає полю форми.

3.1.3 JWT та bcrypt

Генерація JWT в authController: jwt.sign({ id: user._id }, process.env.JWT_SECRET, { expiresIn: "7d" }). Верифікація в middleware protect: jwt.verify(token, process.env.JWT_SECRET) повертає decoded-об’єкт з id, за яким завантажується User з БД через findById.select("-password").

Хешування паролів реалізовано через pre("save")-хук моделі User. Хук активується лише якщо поле password було змінено (isModified("password")). bcrypt.hash виконується з salt rounds = 10. Метод matchPassword використовує bcrypt.compare для безпечного порівняння без можливості відновлення пароля.

3.1.4 Redux Toolkit та axios interceptors

Axios instance налаштований з baseURL: "/api" — що дозволяє Vite proxy перенаправляти запити на backend сервер під час розробки. Request interceptor зчитує token з localStorage та додає його в заголовок Authorization: Bearer. Response interceptor перехоплює 401-статус та виконує dispatch(logout()) через window.location.href = "/login" (оскільки interceptor не має доступу до Redux store).

createAsyncThunk автоматично генерує три action creators: uploadMedia.pending, uploadMedia.fulfilled, uploadMedia.rejected.

У extraReducers кожного slice ці стани обробляються через builder.addCase. Immer (вбудований у RTK) дозволяє "мутувати" стан у reducers — насправді створюючи нову іммутабельну копію.

3.1.5 Система тем

Темна/світла тема реалізована через Redux uiSlice.themeMode та компонент ThemeWrapper у main.jsx. ThemeWrapper зчитує themeMode з Redux store та через useMemo генерує MUI theme об’єкт за допомогою функції getTheme(mode). ThemeWrapper знаходиться всередині Redux Provider, що дозволяє йому

використовувати useSelector. MUI ThemeProvider автоматично поширює тему через React Context на всі дочірні компоненти.

3.2 Програмна реалізація модулів

3.2.1 Реалізація автентифікації

Контролер register виконує послідовно: перевірку унікальності email та username через findOne-запити, створення нового User-документа (хешування відбувається у pre("save")-хуку), генерацію JWT та повернення { token, user: user.toPublic() }. Якщо email або username вже існують — повертається HTTP 400 з відповідним повідомленням.

Контролер login виконує: пошук User за email через findOne, перевірку пароля через user.matchPassword(password) (bcrypt.compare — асинхронна операція), генерацію JWT та повернення { token, user }. Якщо користувача не знайдено або пароль невірний — повертається HTTP 401, що не дозволяє визначити чи існує email (захист від user enumeration).

3.2.2 Реалізація модуля медіаконтенту

uploadMedia отримує завантажений файл з req.file (заповнений Multer після успішного завантаження на Cloudinary). Із req.file.mimetype визначається mediaType. Для зображень URL превью формується через рядкову підстановку в fileUrl (заміна /upload/ на /upload/w_400,h_300,c_fill/). Media-документ зберігається в MongoDB із усіма метаданими.

getMedia використовує Promise.all для паралельного виконання двох запитів: отримання сторінки даних та підрахунку загальної кількості документів. Сортування реалізовано через словник sortOptions — об’єкт із MongoDB sort-специфікаціями. populate("author", "username avatar") підставляє лише необхідні поля автора замість повного об’єкта.

toggleLike реалізує ідемпотентне перемикання: якщо Like-документ із {user, media} вже існує — видаляє його та декрементує likesCount; інакше — створює та інкрементує. $inc-оператор MongoDB виконується атомарно, що виключає race conditions (Рисунок 3.1).

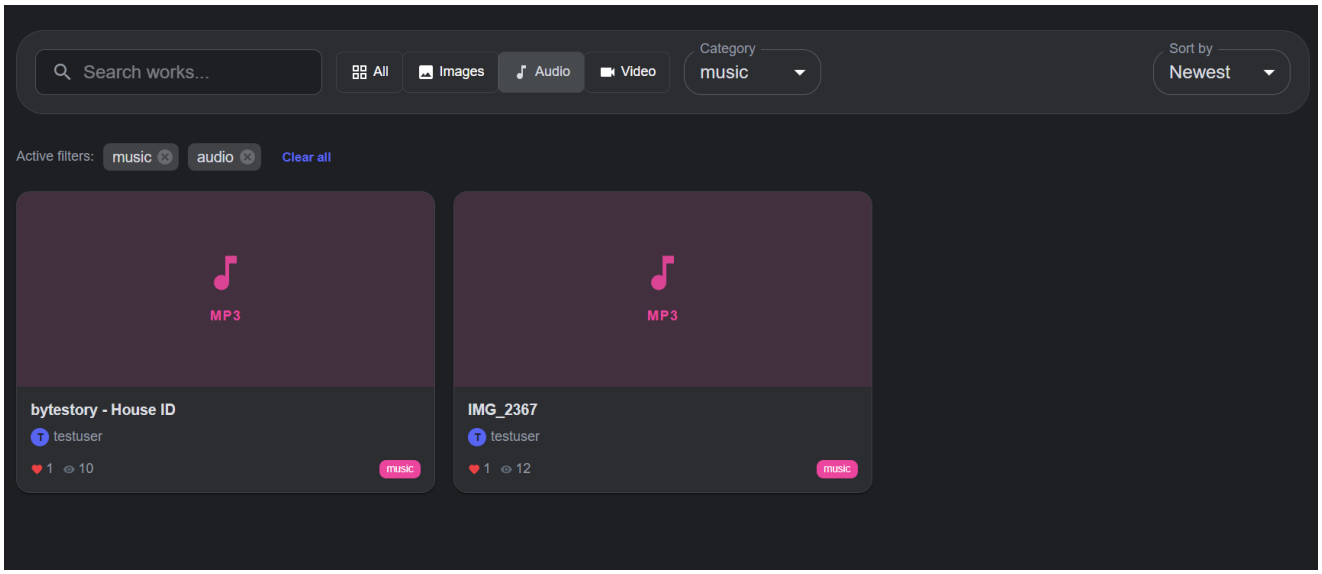


Рисунок 3.1 – Схема модуля медіаконтенту

3.2.3 Реалізація профілю користувача

getUserProfile використовує MongoDB aggregation pipeline для підрахунку totalLikes та totalViews всіх медіаматеріалів автора. Паралельно через Promise.all виконується запит останніх 20 робіт автора. Aggregation виконується в один database roundtrip замість кількох окремих запитів.

updateAvatar реалізований як окрема функція замість middleware, оскільки потребує доступу до res для відправки відповіді після завантаження. Використовує окремий avatarUpload multer-instance (відмінний від основного upload) з власним CloudinaryStorage, що вказує папку mediaflow/avatars та трансформацію {width: 200 , height: 200, crop: "fill", gravity: "face"} —

Cloudinary автоматично визначає та центрує обличчя на аватарі. Перед збереженням нового аватара старий видаляється з Cloudinary через cloudinary.uploader.destroy (Рисунок 3.2)

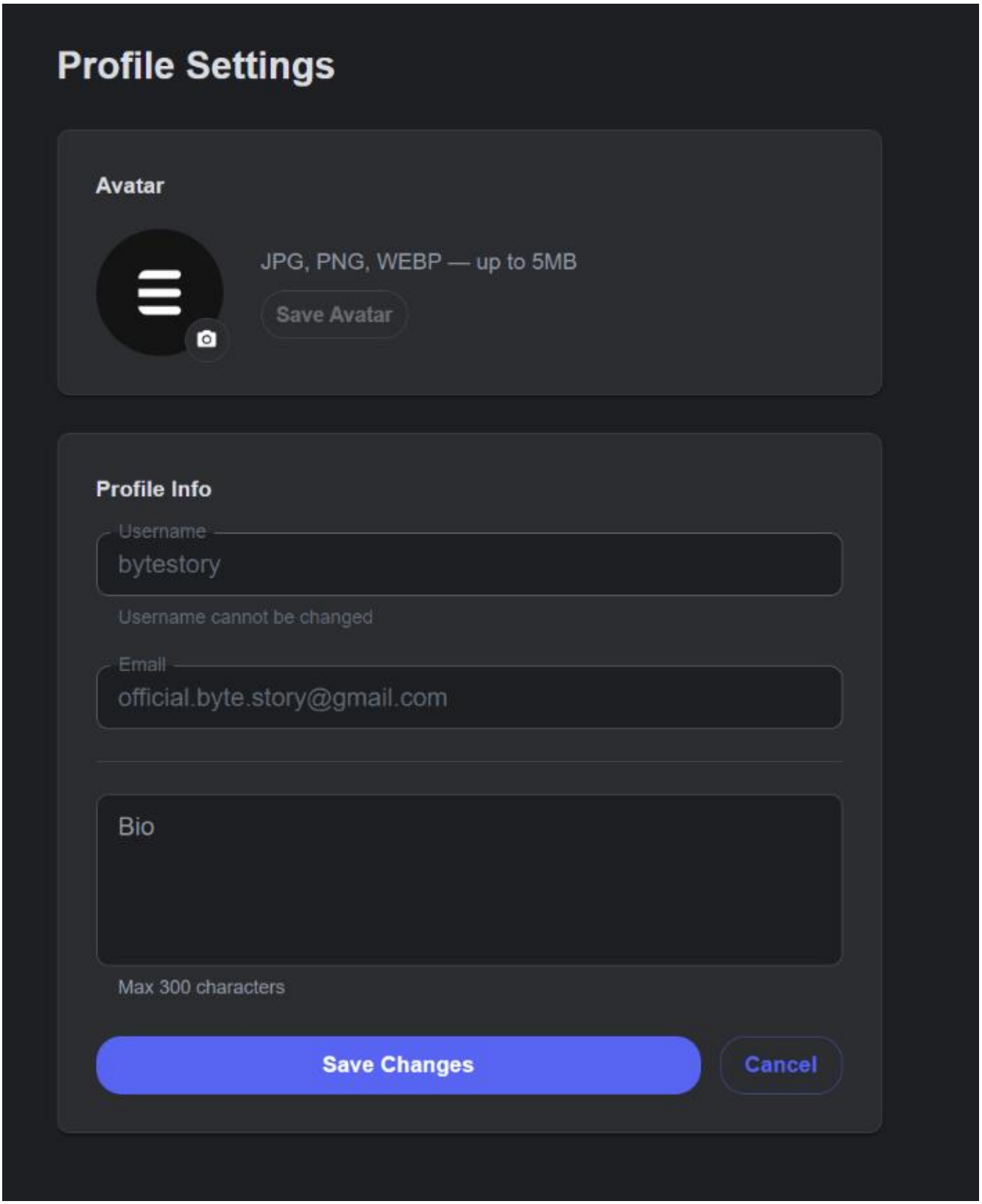


Рисунок 3.2 – Налаштуваня «Profile Settings»

3.3 Реалізація інтерфейсу користувача

3.3.1 Ініціалізація застосунку

Точка входу клієнтського застосунку main.jsx обгортає App у Provider (Redux store), BrowserRouter (маршрутизація) та ThemeWrapper (динамічна тема). ThemeWrapper рендерить MUI ThemeProvider та CssBaseline (нормалізація CSS). SnackbarProvider (notistack) надає систему toast-повідомлень.

App.jsx при монтуванні виконує useEffect: якщо token є в localStorage – dispatch(fetchMe()) для завантаження даних користувача; якщо ні – dispatch(setInitialized()) для позначення, що ініціалізація завершена. Поки initialized = false, App рендерить CircularProgress по центру сторінки.

3.3.2 Маршрутизація та захищені маршрути

Маршрутизація реалізована через React Router v6 з вкладеними маршрутами. Layout компонент рендерить <Outlet />, що замінюється відповідним компонентом сторінки. PrivateRoute перевіряє token та initialized: якщо токена немає – перенаправляє на /login через <Navigate replace>. Якщо initialized = false – повертає null (сторінка ще завантажується).

3.3.3 Реалізація форм з react-hook-form

Форми реєстрації та входу використовують useForm з валідацією при сабміті (mode: "onSubmit" за замовчуванням). Поля підключені через register з правилами валідації: required, minLength, maxLength, pattern, validate (для підтвердження пароля). Помилки відображаються через helperText пропс TextField. Форма зображена на Рисунку 3.3

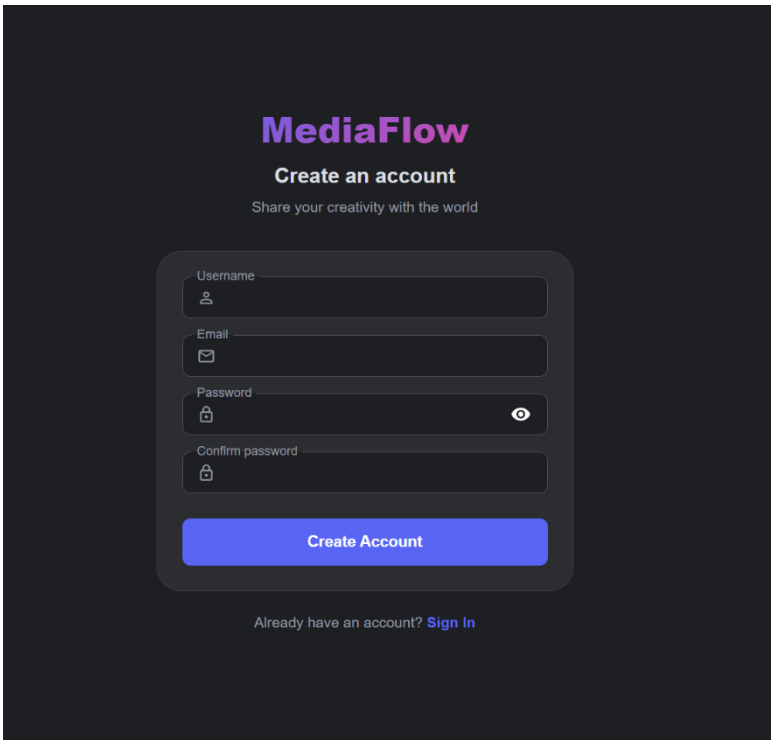


Рисунок 3.3 – Форми реєстрації та входу

Форма завантаження медіа (UploadPage) використовує Controller для поля category (Select-компонент не підтримує нативний register). Дропзона реалізована без сторонніх бібліотек: через onDrop, onDragOver, onDragLeave та onClick обробники на звичайному Box.

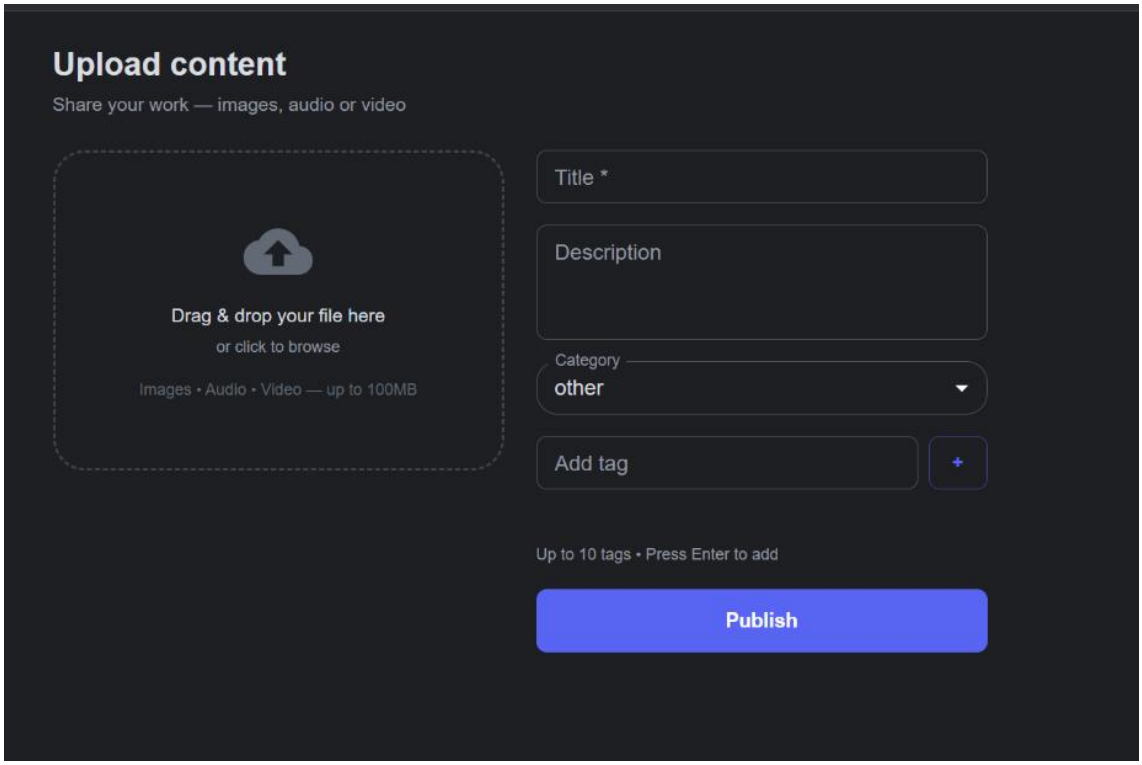


Рисунок 3.3 – Форма завантаження медіа

3.3.4 MediaCard та динамічний рендеринг

Компонент MediaCard рендерить картку медіаматеріалу в сітці стрічки. Для зображень відображається превью (thumbnailUrl з Cloudinary), для аудіо та відео — кольоровий плейсхолдер із відповідною іконкою. Колір плейсхолдера та ефект при hover (box-shadow) визначаються типом медіа через об’єкт TYPE_CONFIG. Компонент обгорнутий у CardActionArea component={Link}, що робить всю картку клікабельним посиланням на сторінку медіаматеріалу (Рисунок 3.4)

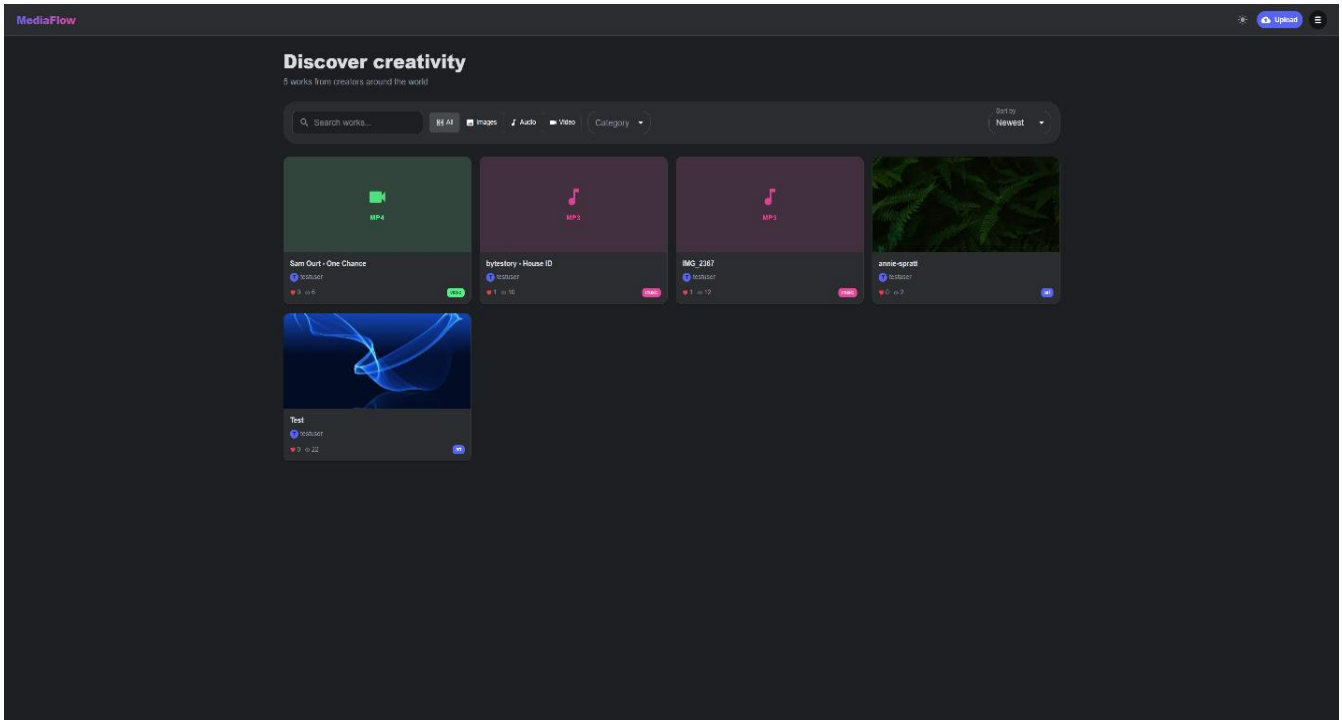


Рисунок 3.4 – Компонент MediaCard

3.3.5 Сторінка медіаматеріалу

MediaPage завантажує медіаматеріал та коментарі паралельно при монтуванні. Медіаплеєр реалізований через умовний рендеринг: для зображень — <Box component="img">, для аудіо — нативний <audio controls>, для відео — нативний <video controls>. Нативні елементи обрані замість кастомних плеєрів через кросплатформну підтримку та відповідність браузерним стандартам (Рисунок 3.5)

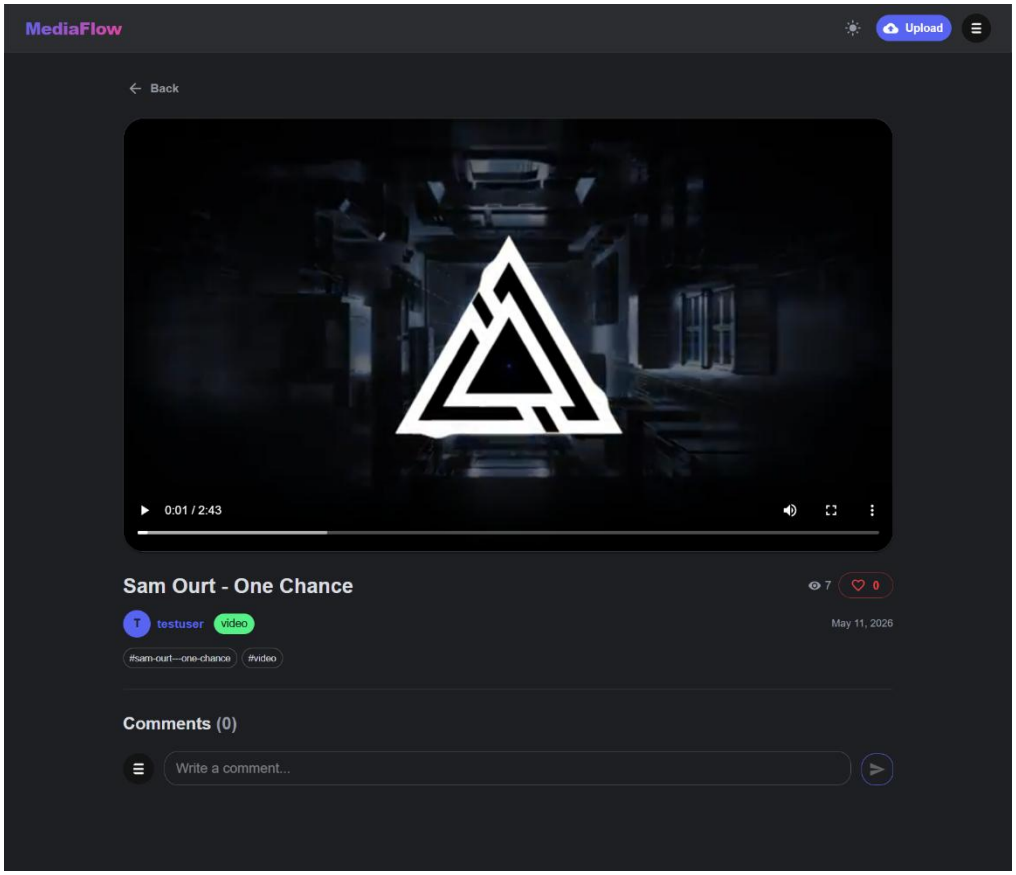


Рисунок 3.5 – Сторінка медіаматеріалу

Коментарі завантажуються через прямий виклик api.get (не через Redux) у локальний стан компонента, оскільки коментарі не потрібні за межами цієї сторінки. Додавання та видалення коментаря оновлює локальний масив comments без перезавантаження всіх коментарів.

3.4 Керівництво користувача

Системні вимоги

Для роботи з платформою MediaFlow необхідний сучасний веббраузер з підтримкою JavaScript: Google Chrome 90+, Mozilla Firefox 88+, Microsoft Edge 90+ або Safari 14+. Стабільне підключення до Інтернету обов’язкове для завантаження та перегляду медіафайлів.

Реєстрація та авторизація

Реєстрація нового акаунту:

– Перейдіть на головну сторінку MediaFlow.

– Натисніть кнопку «Sign Up» в правому верхньому куті.

– Введіть унікальний нікнейм (3-30 символів, лише латиниця, цифри та _).

– Введіть email-адресу.

– Введіть пароль (мінімум 6 символів) та повторіть його для підтвердження.

– Натисніть «Create Account». При успішній реєстрації ви автоматично увійдете в систему.

Авторизація:

– Натисніть кнопку «Log In» в правому верхньому куті.

– Введіть email та пароль зареєстрованого акаунту.

– Натисніть «Sign In». Токен авторизації зберігається в браузері на 7 діб.

В.3 Перегляд контенту

Головна сторінка відображає стрічку всіх опублікованих медіаматеріалів. Для фільтрації стрічки використовуйте панель фільтрів:

– кнопки перемикання «All / Images / Audio / Video» — для фільтрації за типом медіа;

– випадаючий список «Category» — для вибору категорії (art, music, video, animation, photography, other);

– випадаючий список «Sort by» — для сортування (Newest, Most Liked, Most Viewed);

– поле пошуку — введіть запит та натисніть Enter для повнотекстового пошуку за назвою та тегами.

Для перегляду медіаматеріалу натисніть на його картку. Відкриється сторінка з плеєром (зображення, аудіоплеєр або відеоплеєр) та метаінформацією. Зображення можна розглянути у повному розмірі.

В.4 Завантаження контенту

Для завантаження медіаматеріалу необхідна авторизація. Натисніть кнопку «Upload» в навігаційній панелі.

– Перетягніть файл у дропзону або натисніть на неї для відкриття діалогу вибору файлу. Підтримуються формати: зображення (JPEG, PNG, GIF, WEBP), аудіо (MP3, WAV, OGG), відео (MP4, WEBM). Максимальний розмір файлу — 100 МБ.

– Заповніть поля форми: назва (обов’язкове), опис (необов’язкове), категорія.

– Додайте теги: введіть тег у відповідне поле та натисніть «+» або Enter. Можна додати до 10 тегів.

– Натисніть «Publish». Після успішного завантаження відбудеться перехід на сторінку опублікованого матеріалу.

В.5 Соціальна взаємодія

– Лайк: натисніть кнопку із серцем на сторінці медіаматеріалу. Повторне натискання знімає лайк.

– Коментар: введіть текст у поле «Write a comment...» та натисніть кнопку відправки або Enter (без Shift). Максимальна довжина коментаря — 500 символів.

– Видалення коментаря: наведіть курсор на власний коментар та натисніть іконку кошика.

В.6 Профіль користувача

Для перегляду власного профілю натисніть на аватар у правому верхньому куті та оберіть «My Profile». Для перегляду профілю іншого автора натисніть на його нікнейм у картці або на сторінці медіаматеріалу.

Для редагування профілю перейдіть в «Settings» через меню аватара:

– завантажте новий аватар, натиснувши на іконку камери під аватаром (до 5 МБ, зображення);

– відредагуйте біографію (до 300 символів);

– натисніть «Save Changes» для збереження.

3.5 Вимоги до технічних та програмних засобів

Для запуску вебсервісу MediaFlow у режимі розробки необхідне таке програмне та апаратне забезпечення.

Серверне середовище:

– Node.js версії 18 LTS або вище (рекомендовано 20 LTS);

– npm версії 9.0 або вище;

– Активний акаунт MongoDB Atlas (безкоштовний M0 tier достатній);

– Активний акаунт Cloudinary (безкоштовний план: 25 ГБ сховища).

Клієнтське середовище (браузер):

– Google Chrome 90+ / Mozilla Firefox 88+ / Microsoft Edge 90+ / Safari 14+;

– JavaScript та localStorage не повинні бути заблоковані.

Мінімальні апаратні вимоги до серверного середовища:

– RAM: 512 МБ (рекомендовано 1 ГБ);

– CPU: 1 ядро (рекомендовано 2 ядра);

– Дисковий простір: 500 МБ (для node_modules та коду).

Для розгортання в production обрано безкоштовні хмарні сервіси: Render.com для backend (Node.js Web Service, 512 МБ RAM) та Vercel для frontend (статичний хостинг з CDN). Render.com автоматично встановлює залежності та запускає npm start при деплої.

3.6 Тестування вебсервісу

Аналіз методів тестування вебсервісу для обміну мультимедійним контентом між користувачами.

Тестування вебсервісів може проводитись на кількох рівнях. Модульне тестування (unit testing) перевіряє окремі функції та методи у ізоляції. Інтеграційне тестування перевіряє взаємодію між компонентами системи. End-to-end тестування перевіряє повний ланцюжок від клієнта до сервера та бази даних. Функціональне (manual) тестування перевіряє відповідність реалізованої функціональності вимогам.

Для даного проєкту обрано функціональне тестування за сценаріями використання як найбільш доцільний метод для перевірки відповідності системи функціональним вимогам в умовах обмеженого часу. Кожен тестовий сценарій перевіряє конкретну функціональну вимогу з визначеними передумовами, вхідними даними, кроками виконання та очікуваним результатом.

3.5.2 Аналіз результатів тестування

Функціональне тестування проведено за 12 тестовими сценаріями. Результати наведено в таблиці 3.1.

Таблиця 3.1 – Результати функціонального тестування вебсервісу MediaFlow

№	Тестовий сценарій	Вхідні дані	Очікуваний результат	Фактичний результат	Статус
1	Реєстрація з коректними даними	username: testuser, email: test@test.com, password: 123456	HTTP 201, JWT в відповіді, user в MongoDB	Відповідає очікуваному	Пройдено
2	Реєстрація з дублікатом email	email, що вже існує в БД	HTTP 400, повідомлення про помилку	Відповідає очікуваному	Пройдено
3	Реєстрація з дублікатом username	username, що вже зайнятий	HTTP 400, повідомлення про помилку	Відповідає очікуваному	Пройдено
4	Вхід з коректними даними	email та password зареєстрованого user	HTTP 200, JWT в відповіді	Відповідає очікуваному	Пройдено
5	Вхід з невірним паролем	Вірний email, невірний password	HTTP 401, повідомлення про помилку	Відповідає очікуваному	Пройдено
6	Завантаження зображення (JPEG, 2 МБ)	JPEG-файл, title, category	HTTP 201, fileUrl та thumbnailUrl на Cloudinary	Відповідає очікуваному	Пройдено
7	Завантаження аудіо (MP3, 8 МБ)	MP3-файл, title, category	HTTP 201, fileUrl на Cloudinary, відтворення	Відповідає очікуваному	Пройдено
8	Перегляд стрічки (без фільтрів)	Стрічка без параметрів	12 елементів, JSON з items/total/pages	Відповідає очікуваному	Пройдено
9	Фільтрація за типом "audio"	mediaType=audio	Лише аудіоматеріали	Відповідає очікуваному	Пройдено
10	Лайк / Анлайк	POST /api/media/:id/like двічі	1-й: liked=true; 2-й: liked=false	Відповідає очікуваному	Пройдено

11	Видалення контенту	DELETE /api/media/:id автором	HTTP 200, файл видалено з Cloudinary та MongoDB	Відповідає очікуваному	Пройдено
12	Доступ до /upload без авторизації	Запит без JWT-токена	Редірект на /login	Відповідає очікуваному	Пройдено

Всі 12 тестових сценаріїв успішно пройдені. Фактичні результати відповідають очікуваним. Помилок або відхилень у функціональності не виявлено. Перевірено коректну роботу автентифікації, завантаження файлів, фільтрації стрічки, системи лайків, видалення контенту та захисту маршрутів.

3.7 Висновки

У третьому розділі описано особливості програмної реалізації вебсервісу MediaFlow. Детально розглянуто механізм JWT-автентифікації (генерація, верифікація, middleware protect), стримінгове завантаження медіафайлів на Cloudinary через Multer без збереження на диску сервера, алгоритм перемикання лайків ($inc + unique index), систему тем (Redux uiSlice + ThemeWrapper + getTheme), а також ініціалізацію застосунку та захищені маршрути.

Описано реалізацію всіх основних серверних та клієнтських модулів: автентифікацію, медіаконтент, профілі, коментарі, форми з react-hook-form, MediaCard, MediaPage, UploadPage з дропзоною. Визначено технічні вимоги до середовища виконання.

Проведено функціональне тестування за 12 тестовими сценаріями, що охоплюють усі ключові функціональні вимоги. Всі сценарії успішно пройдені, що підтверджує відповідність реалізованого вебсервісу MediaFlow поставленим вимогам.

ВИСНОВКИ

У кваліфікаційній роботі розроблено вебсервіс для обміну мультимедійним контентом між користувачами — платформу MediaFlow. Виконання роботи дозволило досягти поставленої мети та вирішити всі сформульовані задачі.

У результаті виконання кваліфікаційної роботи проведено аналіз предметної галузі вебсервісів для обміну мультимедіа. Виявлено, що більшість існуючих платформ орієнтовані на один тип контенту, а сегмент мультитипних платформ для творчих авторів з сучасним інтерфейсом залишається незаповненим.

Досліджено три ключові аналоги — Newgrounds, SoundCloud та Pixabay. Проведено порівняльний аналіз за 13 критеріями у вигляді таблиці. Визначено переваги та недоліки кожної платформи, що стали основою для формування вимог до MediaFlow.

Сформульовано 21 функціональну та 7 нефункціональних вимог до системи. Визначено трьох акторів: гість, зареєстрований користувач та адміністратор.

Спроєктовано клієнт-серверну архітектуру на базі стеку MERN з REST API та JWT-автентифікацією. Визначено схему бази даних з чотирма колекціями (users, media, comments, likes) та 14 API-ендпоінтів. Вибір технологій обґрунтовано з урахуванням вимог та переваг над альтернативами.

Реалізовано серверну частину: REST API, JWT-автентифікація (bcrypt+jsonwebtoken), стримінгове завантаження медіафайлів на Cloudinary (multer-storage-cloudinary), атомарна система лайків ($inc + unique index), MongoDB aggregation для статистики профілю.

Реалізовано клієнтську частину: SPA на React.js з Redux Toolkit, адаптивний інтерфейс з підтримкою темної/світлої теми, стрічка контенту з фільтрацією та пошуком, нативний медіаплеєр, форми з react-hook-form, захищені маршрути.

Проведено функціональне тестування за 12 тестовими сценаріями. Всі сценарії успішно пройдені.

Практична цінність роботи полягає у створенні повнофункціонального вебсервісу MediaFlow, що може використовуватись як самостійний продукт або як

основа для комерційного медіахостингу. Платформа розгорнута з використанням безкоштовних хмарних сервісів (MongoDB Atlas M0, Cloudinary free tier).

Перспективами подальшого розвитку системи є: реалізація системи підписок між користувачами, алгоритмічні рекомендації контенту на основі вподобань, адміністративна панель для модерації, підтримка плейлистів для аудіоконтенту, PWA-обгортка для мобільних пристроїв, реалізація публічного API для сторонніх розробників.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Flanagan D. JavaScript: The Definitive Guide. 7th edition. Sebastopol: O'Reilly Media, 2020. 706 p.

2. Brown E. Web Development with Node and Express: Leveraging the JavaScript Stack. 2nd edition. Sebastopol: O'Reilly Media, 2019. 332 p.

3. Banks A., Porcello E. Learning React: Modern Patterns for Developing React Apps. 2nd edition. Sebastopol: O'Reilly Media, 2020. 310 p.

4. Chodorow K. MongoDB: The Definitive Guide. Powerful and Scalable Data Storage. 3rd edition. Sebastopol: O'Reilly Media, 2019. 514 p.

5. Richardson L., Ruby S. RESTful Web Services: Web Services for the Real World. Sebastopol: O'Reilly Media, 2007. 448 p.

6. React Documentation. URL: https://react.dev/learn (дата звернення: 01.04.2025).

7. Redux Toolkit Documentation. URL: https://redux-toolkit.js.org/introduction/getting-started (дата звернення: 01.04.2025).

8. Express.js Application Generator. URL: https://expressjs.com/en/starter/generator.html (дата звернення: 02.04.2025).

9. Mongoose Documentation. Schemas. URL: https://mongoosejs.com/docs/guide.html (дата звернення: 02.04.2025).

10. Cloudinary Documentation. Node.js SDK. URL: https://cloudinary.com/documentation/node_integration (дата звернення: 03.04.2025).

11. JSON Web Tokens. Introduction. URL: https://jwt.io/introduction (дата звернення: 03.04.2025).

12. Material UI. Getting Started. URL: https://mui.com/material-ui/getting-started (дата звернення: 04.04.2025).

13. React Hook Form. Documentation. URL: https://react-hook-form.com/get-started (дата звернення: 04.04.2025).

14. Multer. GitHub. URL: https://github.com/expressjs/multer (дата звернення: 05.04.2025).

15. Vite. Guide. URL: https://vitejs.dev/guide (дата звернення: 05.04.2025).

16. Newgrounds. URL: https://www.newgrounds.com (дата звернення: 06.04.2025).

17. SoundCloud. URL: https://soundcloud.com (дата звернення: 06.04.2025).

18. Pixabay. URL: https://pixabay.com (дата звернення: 06.04.2025).

19. Cisco Annual Internet Report (2018–2023). URL: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html (дата звернення: 07.04.2025).

20. MongoDB Atlas. Documentation. URL: https://www.mongodb.com/docs/atlas (дата звернення: 07.04.2025).

21. bcryptjs. npm package. URL: https://www.npmjs.com/package/bcryptjs (дата звернення: 08.04.2025).

22. Axios. Documentation. URL: https://axios-http.com/docs/intro (дата звернення: 08.04.2025).

23. React Router. Documentation. URL: https://reactrouter.com/en/main (дата звернення: 09.04.2025).

24. Render. Cloud Application Hosting. URL: https://render.com/docs (дата звернення: 09.04.2025).

25. ДСТУ ISO/IEC 25010:2016. Вимоги та оцінювання якості систем та програмного забезпечення. Київ: ДП «УкрНДНЦ», 2016. 40 с.

ДОДАТОК А

(обов’язковий)

А.1 Діаграма варіантів використання (Use Case Diagram)

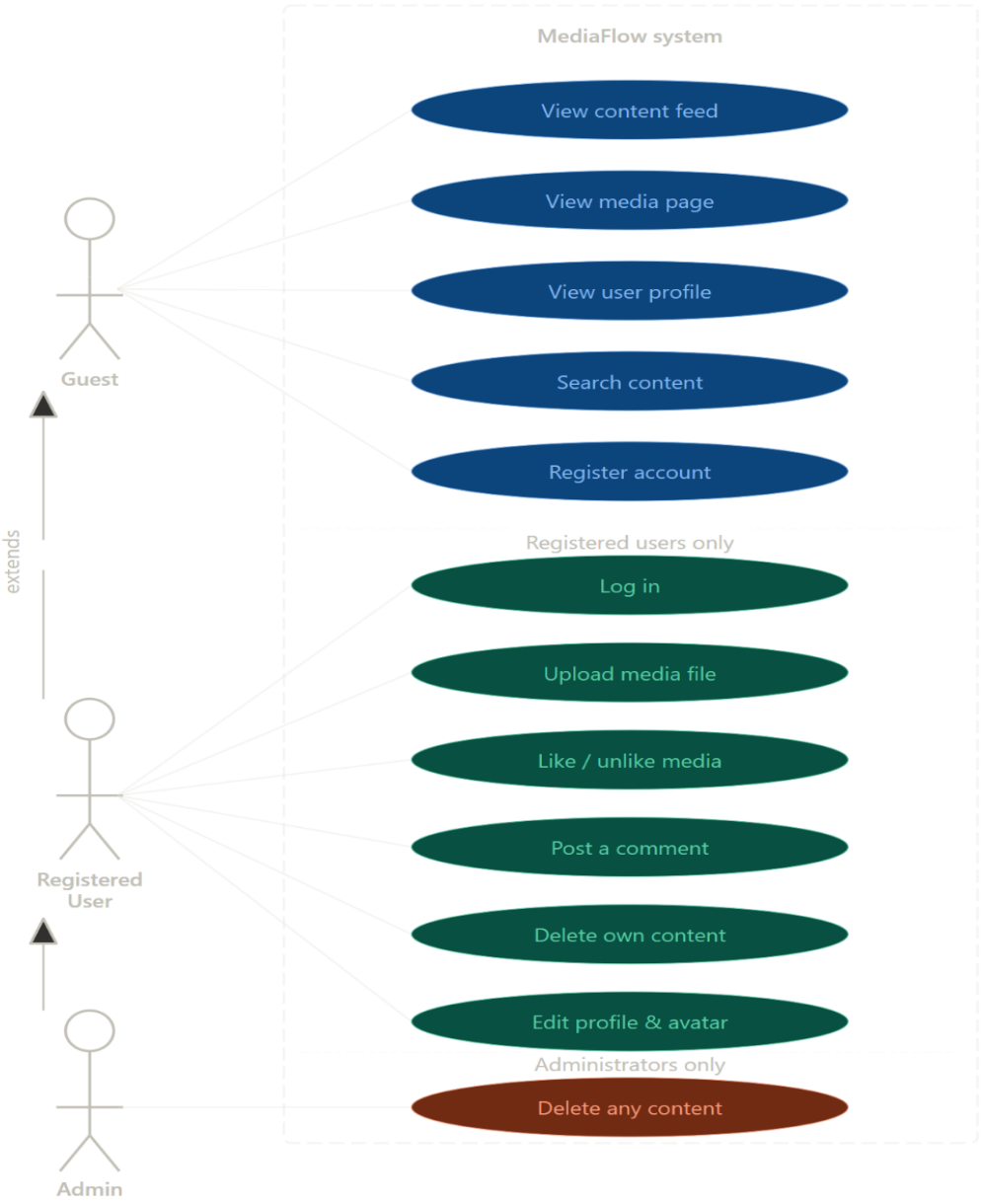


Рисунок А.1 – Діаграма варіантів використання

А.2 Діаграма зв’язків модулів

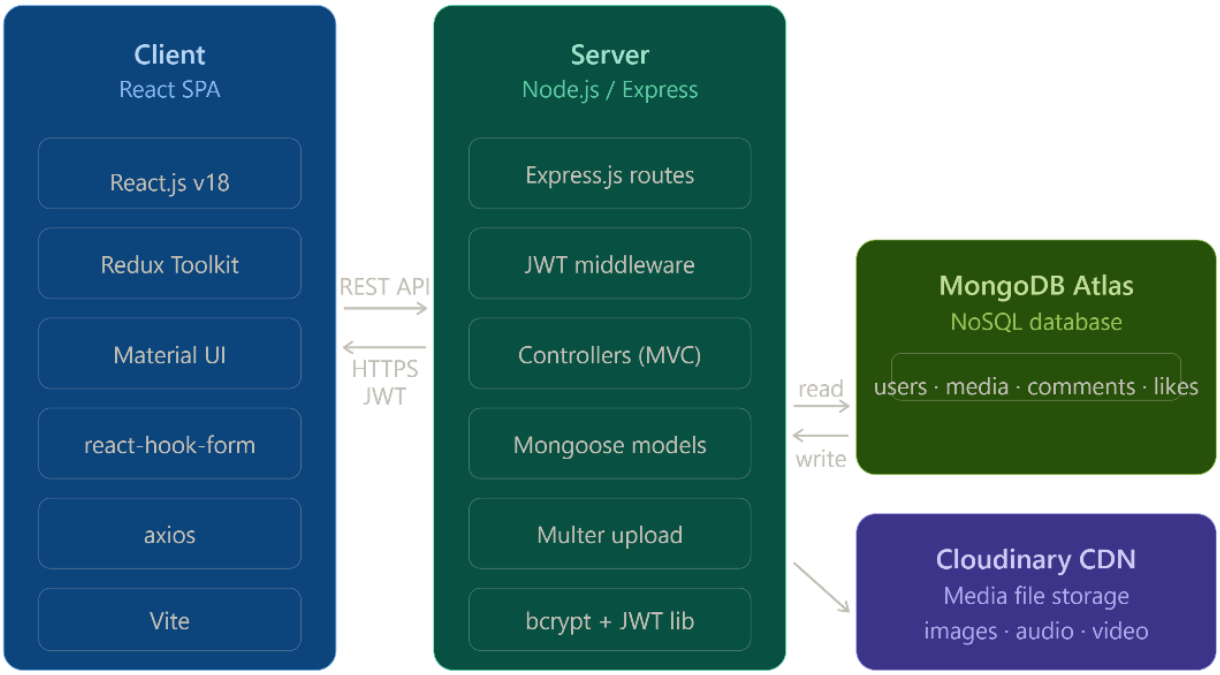


Рисунок А.2 – Діаграма зв’язків модулів]

А.3 Діаграма класів / Схема бази даних

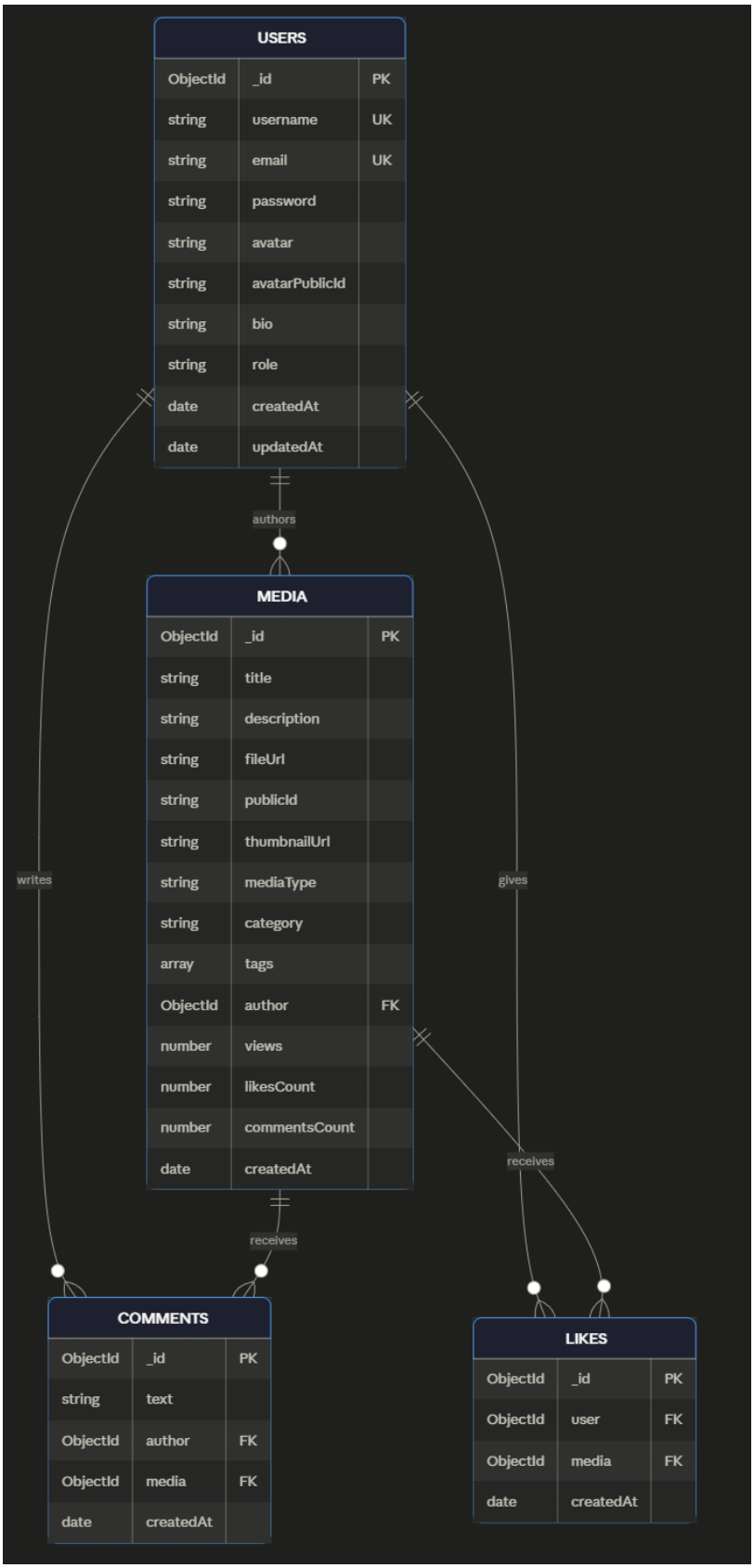


Рисунок А.3 – Діаграма класів / Схема бази даних

ДОДАТОК Б

(обов’язковий)

ПРОГРАМНИЙ КОД ОСНОВНИХ МОДУЛІВ

Б.1 Точка входу серверного застосунку (server.js)

```
const express = require('express');
const cors = require('cors');
const morgan = require('morgan');
const dotenv = require('dotenv');
const connectDB = require('./config/db');

dotenv.config();
connectDB();

const app = express();

app.use(cors({
  origin: process.env.CLIENT_URL || 'http://localhost:3000',
  credentials: true,
}));
app.use(express.json());
app.use(morgan('dev'));

app.use('/api/auth',     require('./routes/auth'));
app.use('/api/media',    require('./routes/media'));
app.use('/api/users',    require('./routes/users'));
app.use('/api/comments', require('./routes/comments'));
```

```
app.use((err, req, res, next) => {
  console.error(err.stack);
  res.status(err.status || 500).json({
    message: err.message || 'Server Error',
  });
});

const PORT = process.env.PORT || 5000;
app.listen(PORT, () => console.log(`Server running on port
${PORT}`));
```

Б.2 Модель користувача (models/User.js)

```
const mongoose = require('mongoose');
const bcrypt   = require('bcryptjs');

const userSchema = new mongoose.Schema({
  username: { type: String, required: true, unique: true,
              minlength: 3, maxlength: 30 },
  email:    { type: String, required: true, unique: true,
              lowercase: true },
  password: { type: String, required: true, minlength: 6 },
  avatar:   { type: String, default: '' },
  avatarPublicId: { type: String, default: '' },
  bio:      { type: String, maxlength: 300, default: '' },
  role:     { type: String, enum: ['user','admin'], default:
'user' },
}, { timestamps: true });
```

```
userSchema.pre('save', async function(next) {
  if (!this.isModified('password')) return next();
  this.password = await bcrypt.hash(this.password, 10);
  next();
});

userSchema.methods.matchPassword = function(pwd) {
  return bcrypt.compare(pwd, this.password);
};

userSchema.methods.toPublic = function() {
  const obj = this.toObject();
  delete obj.password;
  return obj;
};

module.exports = mongoose.model('User', userSchema);
```

Б.3 Контролер автентифікації (controllers/authController.js)

```
const jwt = require('jsonwebtoken');
const User = require('../models/User');

const generateToken = (id) =>
  jwt.sign({ id }, process.env.JWT_SECRET,
           { expiresIn: process.env.JWT_EXPIRES_IN || '7d' });

// POST /api/auth/register
const register = async (req, res) => {
```

```
  try {
    const { username, email, password } = req.body;
    if (await User.findOne({ email }))
      return res.status(400).json({ message: 'Email вже зареєстрований' });
    if (await User.findOne({ username }))
      return res.status(400).json({ message: 'Нікнейм вже зайнятий' });
    const user = await User.create({ username, email, password });
    res.status(201).json({ token: generateToken(user._id), user: user.toPublic() });
  } catch (err) { res.status(500).json({ message: err.message }); }
};

// POST /api/auth/login
const login = async (req, res) => {
  try {
    const { email, password } = req.body;
    const user = await User.findOne({ email });
    if (!user || !(await user.matchPassword(password)))
      return res.status(401).json({ message: 'Невірний email або пароль' });
    res.json({ token: generateToken(user._id), user: user.toPublic() });
  } catch (err) { res.status(500).json({ message: err.message }); }
};

module.exports = { register, login, getMe: (req, res) => res.json(req.user) };
```

Б.4 Контролер медіаконтенту (controllers/mediaController.js — фрагмент)

```
const Media = require('../models/Media');
const Like  = require('../models/Like');

// POST /api/media
const uploadMedia = async (req, res) => {
  if (!req.file) return res.status(400).json({ message: 'Файл не завантажено' });
  const { title, description, category, tags } = req.body;
  const parsedTags = tags ? tags.split(',').map(t => t.trim()).filter(Boolean) : [];
  const mediaType = req.file.mimetype.startsWith('image/') ? 'image'
    : req.file.mimetype.startsWith('audio/') ? 'audio' : 'video';
  const thumbnailUrl = mediaType === 'image'
    ? req.file.path.replace('/upload/', '/upload/w_400,h_300,c_fill/') : '';
  const media = await Media.create({
    title, description, fileUrl: req.file.path,
    publicId: req.file.filename, thumbnailUrl,
    mediaType, category: category || 'other', tags: parsedTags,
    author: req.user._id,
  });
  await media.populate('author', 'username avatar');
  res.status(201).json(media);
};

// GET /api/media
const getMedia = async (req, res) => {
```

```
  const { category, mediaType, tag, search,
          page = 1, limit = 12, sort = 'newest' } = req.query;
  const filter = {};
  if (category)  filter.category  = category;
  if (mediaType) filter.mediaType = mediaType;
  if (tag)       filter.tags      = tag;
  if (search)    filter.$text     = { $search: search };
  const sortOptions = {
    newest:  { createdAt: -1 },
    popular: { likesCount: -1 },
    views:   { views: -1 },
  };
  const skip = (Number(page) - 1) * Number(limit);
  const [items, total] = await Promise.all([
    Media.find(filter)
      .populate('author', 'username avatar')
      .sort(sortOptions[sort] || sortOptions.newest)
      .skip(skip).limit(Number(limit)),
    Media.countDocuments(filter),
  ]);
  res.json({ items, total, page: Number(page),
             pages: Math.ceil(total / limit) });
};

// POST /api/media/:id/like
const toggleLike = async (req, res) => {
  const media = await Media.findById(req.params.id);
  if (!media) return res.status(404).json({ message: 'Не
знайдено' });
```

```
  const existing = await Like.findOne({
    user: req.user._id, media: media._id });
  if (existing) {
    await existing.deleteOne();
    await Media.findByIdAndUpdate(media._id, { $inc: {
likesCount: -1 } });
    return res.json({ liked: false, likesCount: media.likesCount
- 1 });
  }
  await Like.create({ user: req.user._id, media: media._id });
  await Media.findByIdAndUpdate(media._id, { $inc: { likesCount:
1 } });
  res.json({ liked: true, likesCount: media.likesCount + 1 });
};
```

Б.5 Redux Toolkit authSlice (фрагмент)

```
import { createSlice, createAsyncThunk } from '@reduxjs/toolkit';
import api from '../../utils/api';

export const login = createAsyncThunk('auth/login',
  async (data, { rejectWithValue }) => {
    try {
      const res = await api.post('/auth/login', data);
      localStorage.setItem('token', res.data.token);
      return res.data;
    } catch (err) {
      return rejectWithValue(err.response?.data?.message);
    }
  });
```

```
const authSlice = createSlice({
  name: 'auth',
  initialState: {
    user: null,
    token: localStorage.getItem('token'),
    loading: false, error: null, initialized: false,
  },
  reducers: {
    logout(state) {
      state.user = null; state.token = null;
      localStorage.removeItem('token');
    },
    setInitialized(state) { state.initialized = true; },
  },
  extraReducers: (builder) => {
    builder
      .addCase(login.pending,    s => { s.loading = true; })
      .addCase(login.fulfilled,  (s, a) => {
        s.loading = false; s.user = a.payload.user;
        s.token = a.payload.token;
      })
      .addCase(login.rejected,   (s, a) => {
        s.loading = false; s.error = a.payload;
      });
  },
});
```

```
export const { logout, setInitialized } = authSlice.actions;
export default authSlice.reducer;
```

Б.6 Axios instance з interceptors (utils/api.js)

```
import axios from 'axios';

const api = axios.create({ baseURL: '/api' });

// Додаємо JWT до кожного запиту
api.interceptors.request.use((config) => {
  const token = localStorage.getItem('token');
  if (token) config.headers.Authorization = `Bearer ${token}`;
  return config;
});

// Перехоплюємо 401 — виконуємо logout
api.interceptors.response.use(
  (res) => res,
  (err) => {
    if (err.response?.status === 401) {
      localStorage.removeItem('token');
      window.location.href = '/login';
    }
    return Promise.reject(err);
  }
);

export default api;
```

Б.7 Функція генерації теми (utils/theme.js — фрагмент)

```
import { createTheme } from '@mui/material/styles';

export const getTheme = (mode) => createTheme({
  palette: {
    mode,
    primary: { main: '#5865F2' },
    secondary: { main: '#eb459e' },
    background: mode === 'dark'
      ? { default: '#1e1f22', paper: '#2b2d31' }
      : { default: '#f2f3f5', paper: '#ffffff' },
  },
  components: {
    MuiButton: {
      styleOverrides: {
        root: { borderRadius: 8, textTransform: 'none',
                fontWeight: 600 },
      },
    },
  },
});
```

ДОДАТОК В

(Обов’язковий)

КЕРІВНИЦТВО КОРИСТУВАЧА

В.1 Системні вимоги

Для роботи з платформою MediaFlow необхідний сучасний веббраузер з підтримкою JavaScript: Google Chrome 90+, Mozilla Firefox 88+, Microsoft Edge 90+ або Safari 14+. Стабільне підключення до Інтернету обов’язкове для завантаження та перегляду медіафайлів.

В.2 Реєстрація та авторизація

Реєстрація нового акаунту:

– Перейдіть на головну сторінку MediaFlow.

– Натисніть кнопку «Sign Up» в правому верхньому куті.

– Введіть унікальний нікнейм (3-30 символів, лише латиниця, цифри та _).

– Введіть email-адресу.

– Введіть пароль (мінімум 6 символів) та повторіть його для підтвердження.

– Натисніть «Create Account». При успішній реєстрації ви автоматично увійдете в систему.

Авторизація:

– Натисніть кнопку «Log In» в правому верхньому куті.

– Введіть email та пароль зареєстрованого акаунту.

– Натисніть «Sign In». Токен авторизації зберігається в браузері на 7 діб.

В.3 Перегляд контенту

Головна сторінка відображає стрічку всіх опублікованих медіаматеріалів. Для фільтрації стрічки використовуйте панель фільтрів:

– кнопки перемикання «All / Images / Audio / Video» — для фільтрації за типом медіа;

– випадаючий список «Category» — для вибору категорії (art, music, video, animation, photography, other);

– випадаючий список «Sort by» — для сортування (Newest, Most Liked, Most Viewed);

– поле пошуку — введіть запит та натисніть Enter для повнотекстового пошуку за назвою та тегами.

Для перегляду медіаматеріалу натисніть на його картку. Відкриється сторінка з плеєром (зображення, аудіоплеєр або відеоплеєр) та метаінформацією. Зображення можна розглянути у повному розмірі.

В.4 Завантаження контенту

Для завантаження медіаматеріалу необхідна авторизація. Натисніть кнопку «Upload» в навігаційній панелі.

– Перетягніть файл у дропзону або натисніть на неї для відкриття діалогу вибору файлу. Підтримуються формати: зображення (JPEG, PNG, GIF, WEBP), аудіо (MP3, WAV, OGG), відео (MP4, WEBM). Максимальний розмір файлу — 100 МБ.

– Заповніть поля форми: назва (обов’язкове), опис (необов’язкове), категорія.

– Додайте теги: введіть тег у відповідне поле та натисніть «+» або Enter. Можна додати до 10 тегів.

– Натисніть «Publish». Після успішного завантаження відбудеться перехід на сторінку опублікованого матеріалу.

В.5 Соціальна взаємодія

– Лайк: натисніть кнопку із серцем на сторінці медіаматеріалу. Повторне натискання знімає лайк.

– Коментар: введіть текст у поле «Write a comment...» та натисніть кнопку відправки або Enter (без Shift). Максимальна довжина коментаря — 500 символів.

– Видалення коментаря: наведіть курсор на власний коментар та натисніть іконку кошика.

В.6 Профіль користувача

Для перегляду власного профілю натисніть на аватар у правому верхньому куті та оберіть «My Profile». Для перегляду профілю іншого автора натисніть на його нікнейм у картці або на сторінці медіаматеріалу.

Для редагування профілю перейдіть в «Settings» через меню аватара:

– завантажте новий аватар, натиснувши на іконку камери під аватаром (до 5 МБ, зображення);

– відредагуйте біографію (до 300 символів);

– натисніть «Save Changes» для збереження.

ДОДАТОК Г

(Обов’язковий)

ПРЕЗЕНТАЦІЙНІ МАТЕРІАЛИ

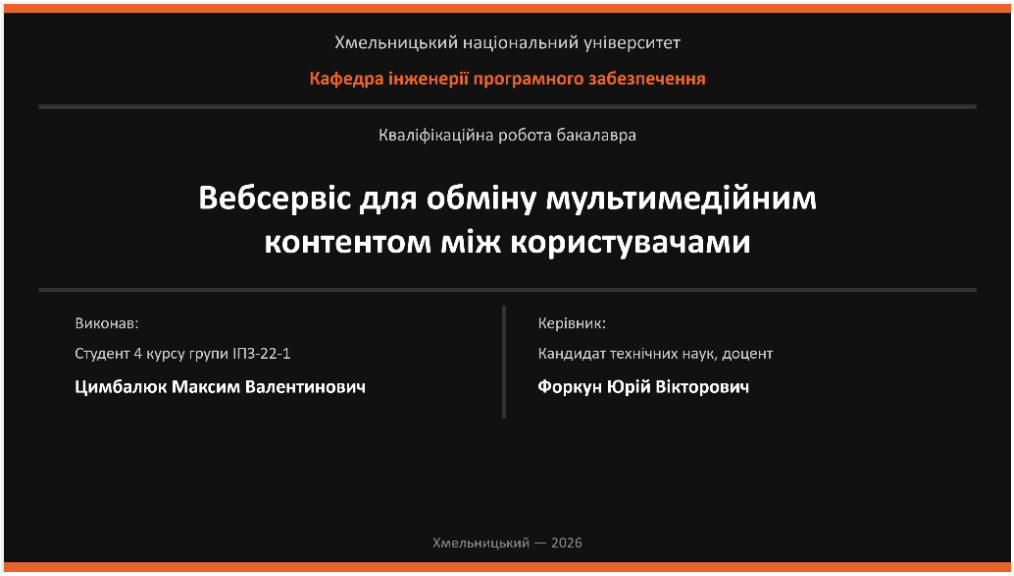


Рисунок В.1 – Слайд 1: Титульний слайд

Рисунок В.2 – Слайд 2: Предметна область

Рисунок В.3 – Слайд 3: Мета та завдання роботи

Рисунок В.4 – Слайд 4: Аналіз предметної області

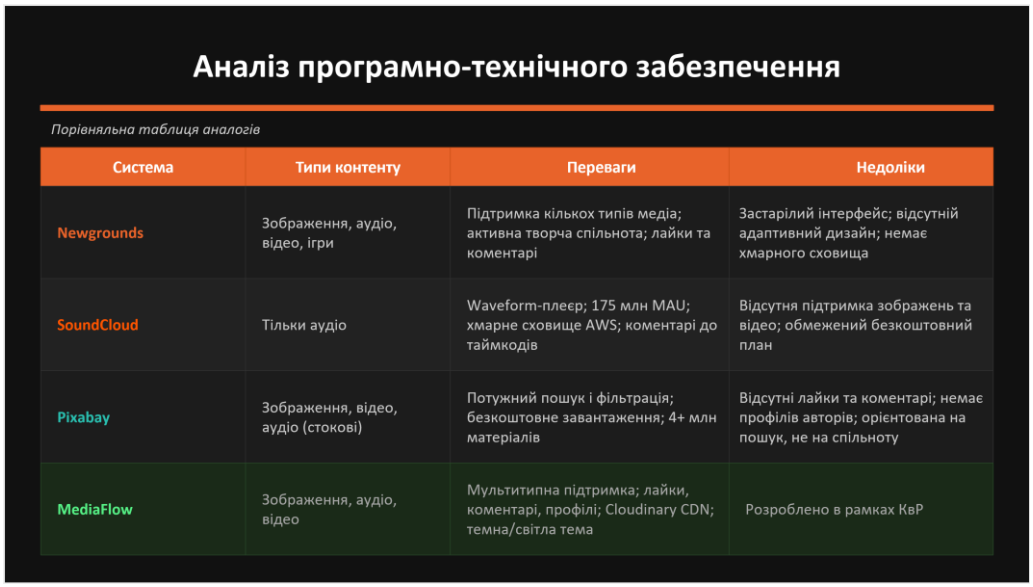


Рисунок В.5 – Слайд 5: Аналіз програмно-технічого забезпечення

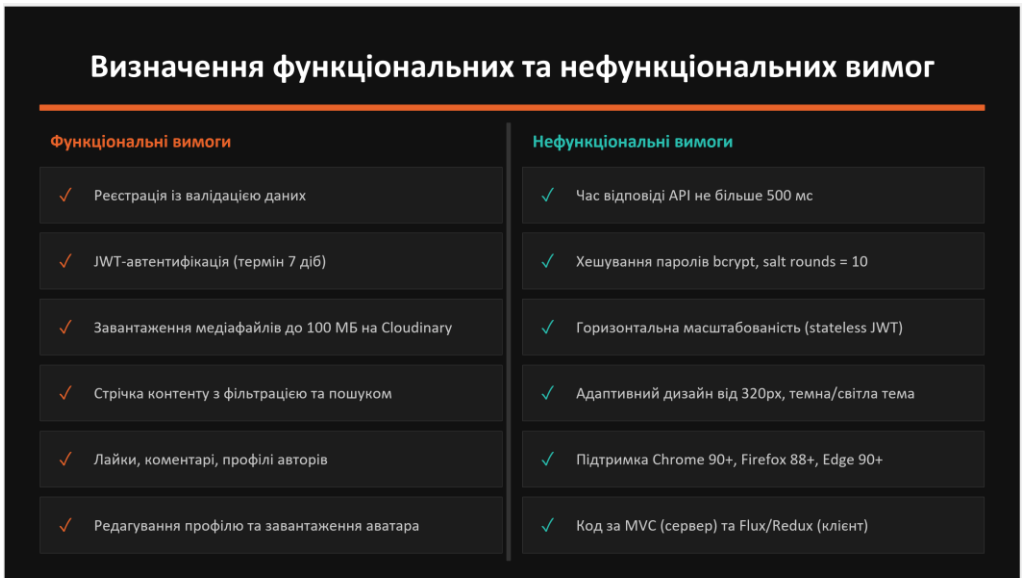


Рисунок В.6 – Слайд 6: Визначення вимог

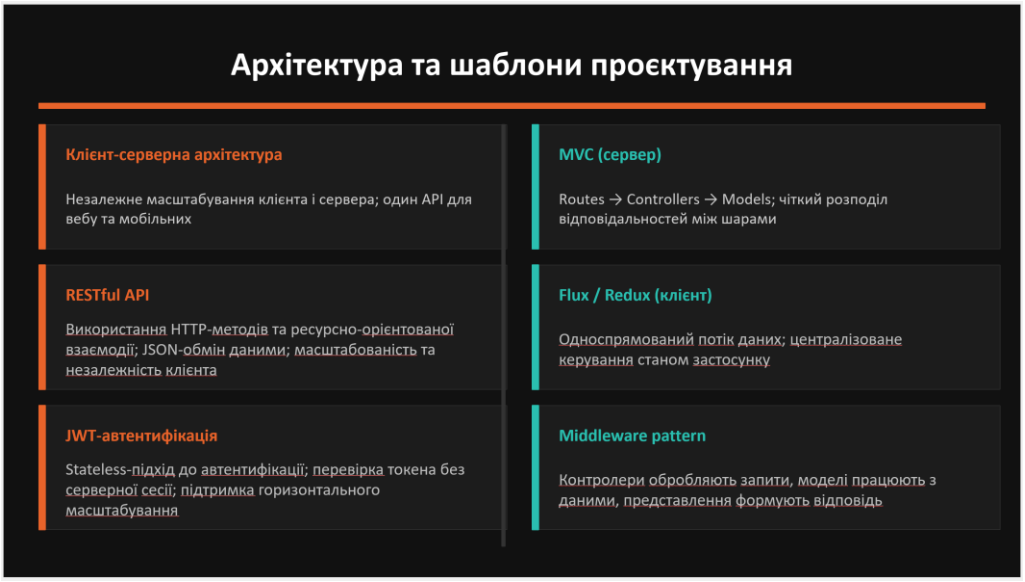


Рисунок В.7 – Слайд 7: Архітектура та шаблони проєктування

Рисунок В.8 – Слайд 8: Декомпозиція та інтерфейси

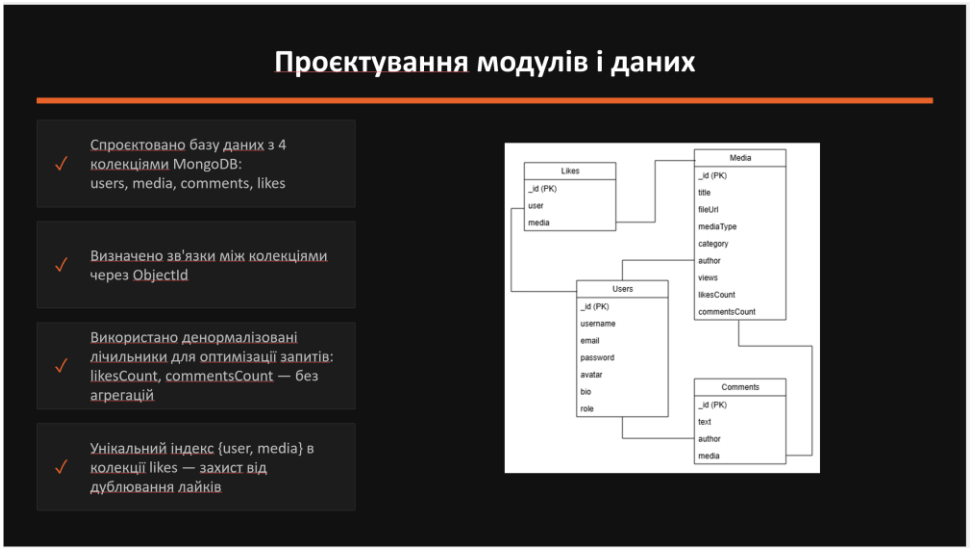


Рисунок В.9 – Слайд 9: Проєктування модулів і даних

Рисунок В.10 – Слайд 10: Аналіз та вибір технологій

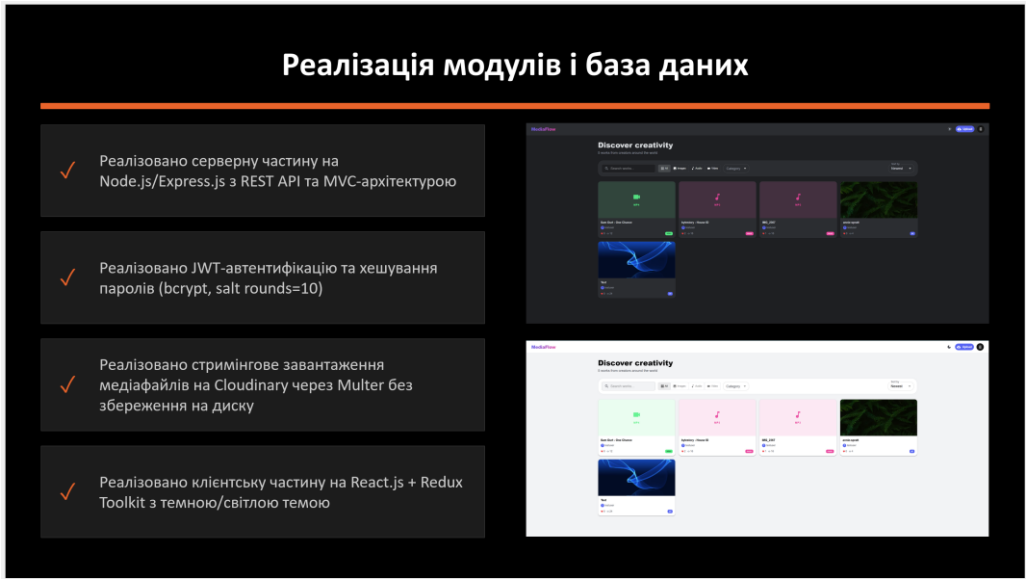


Рисунок В.11 – Слайд 11: Реалізація модулів і БД

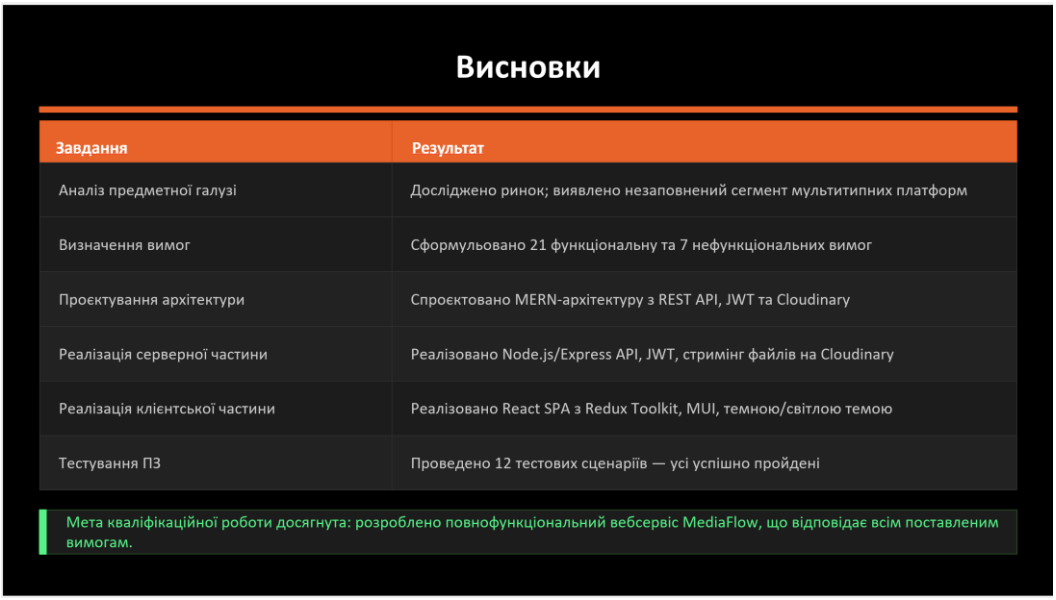
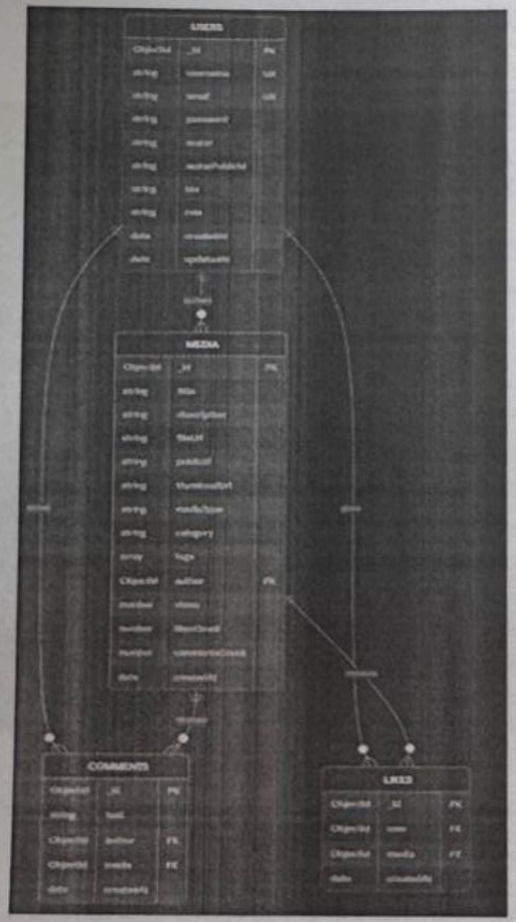
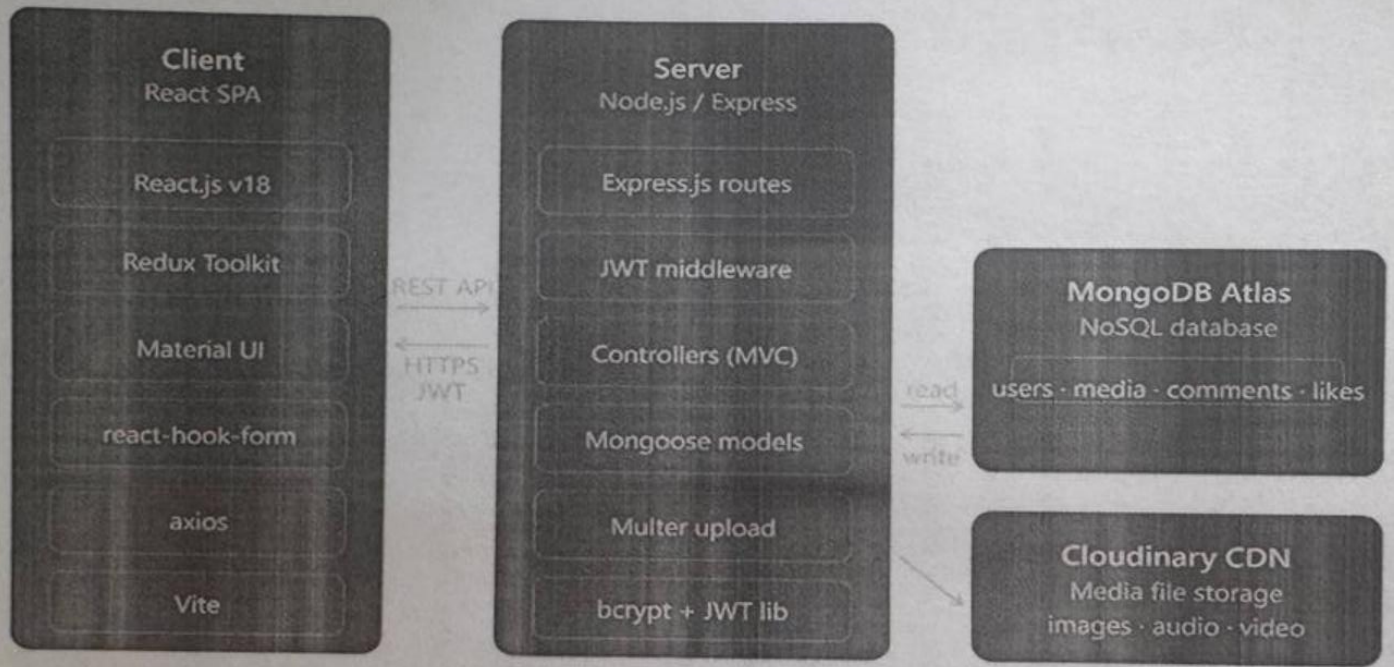


Рисунок В.12 – Слайд 12: Висновки

ГРАФІЧНА ЧАСТИНА



					КвРІПЗ.220218.01.21.ГБ		
Змн.	Лист	№ докум.	Підпис	Дата			
Розробив		Цимбалюк М.			Схема бази даних	Літ.	Маса	Масштаб
Керівник		Федун І.О.						
						Арк. 1	Аркушів 3	
Н. контр.		Ільчук О. М.				КНУ, ІПЗ–22–1		
Затвердив		Сабанчюк Л.						



КаРІПЗ.220218.01.21.Г8

Цимбалюк М.

Діаграма зв'язків модулів

Арк. 1 Аркушів 3

КНУ, ІПЗ-22-1

СУПРОВІДНІ ДОКУМЕНТИ

Завідувачу кафедри інженерії програмного забезпечення проф.
Леоніду БЕДРАТЮКУ здобувача
вищої освіти
Цимбалюка Максима Валентиновича
факультет ІТ, IV курс, група ІПЗ-22-1

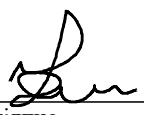
ЗАЯВА

З правилами чинного Положення про систему забезпечення академічної доброчесності в Хмельницькому національному університеті, згідно з яким виявлення академічного плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту і застосування заходів дисциплінарної та академічної відповідальності, ознайомлений. Про використання програмно-технічних засобів для перевірки кваліфікаційних робіт здобувачів вищої освіти на наявність академічного плагіату оповіщений та надаю свою згоду на обробку й збереження університетом моєї роботи в інституційному репозитарії Хмельницького національного університету.

Також надаю університету право на передачу моєї роботи для обробки та збереження в базах даних програмно-обчислювального комплексу StrikePlagiarism та/або програмно-технічного засобу AntiPlagiarism і використання роботи для виявлення академічного плагіату в інших роботах, які перевіряються програмно-технічними засобами та користувачами, що мають доступ до цих програмно-технічних засобів, виключно в обмежених цілях для виявлення текстових збігів у роботах.

Робота надається для перевірки в електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

09.05.2026
дата


підпис

Протокол аналізу звіту подібності науковим керівником

Заявляю, що я ознайомився (-лась) з Повним звітом подібності, який був згенерований Системою виявлення і запобігання плагіату щодо роботи:

Автор: Максим ЦИМБАЛЮК

Співавтор:

Назва: Вебсервіс для обміну мультимедійним контентом між користувачами

Науковий керівник: канд. техн. наук, доцент Юрій ФОРКУН

Підрозділ: Кафедра інженерії програмного забезпечення

Коефіцісит подібності 1:5.68%

Коефіцісит подібності 2:1.99%

Мікропробіли: 48

Заміна букв: 1

Інтервали: 0

Білі знаки: 0

Дата створення звіту: 2026-06-08 13:14:20.0

Після аналізу Звіту подібності констатую наступне:

☑ **Запозичення, виявлені в роботі є законними і не є плагіатом. Рівень подібності не перевищує допустимої межі. Таким чином робота незалежна і приймається.**

☐ **Запозичення не є плагіатом, але перевищено граничне значення рівня подібностей. Таким чином робота повертається на доопрацювання.**

☐ **Виявлено запозичення і плагіат або навмисні текстові спотворення (маніпуляції), як передбачувані спроби укриття плагіату, які роблять роботу невідповідною вимогам законодавства (Ст. 32. ЗУ Про вищу освіту, пункт 3.1, Ст. 42. ЗУ Про освіту) та вимог НАЗЯВО (Критерій 5), а також кодексу етики і процедурам. Таким чином робота не приймається.**

Обгрунтування:

Дата 08.06.2026 експерт Форкун Ю.В.

Anti-Plagiarism (http://ap.km.ua) v-16.718

Максимальне співпадіння з одним документом 3.0%

Словники перевірки: UA, US, RU. **Помилок в документах: 24%**

	Документ		Сумарний збіг по Базі Даних	
ID: 274247 Назва: БКРВебсервіс для обміну мультимедійним контентом між користувачами Додано в БД: 2026-06-09 Автора: Максим ЦИМБАЛЮК Керівники: канд. техн. наук, доцент Юрій ФОРКУН Консультанти: Опоненти:	Символи	Лексеми	Символи	Лексеми
	54724	526	4824 (9%)	60 (11%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ
освітнього ступеня «Бакалавр»

Дипломник Цимбалюк Максим Валентинович

Тема Веб-платформа для обміну та зберігання медіа між користувачами

Спеціальність 121 – Інженерія програмного забезпечення

Обсяг кваліфікаційної роботи:

Кількість листів креслень 3; кількість сторінок записки 75

1.Короткий зміст пояснювальної записки та прийнятих рішень У кваліфікаційній роботі досліджено і проаналізовано предметну область вебсервісів для обміну мультимедійним контентом між користувачами та визначено функціональні й нефункціональні вимоги. Проведено аналіз трьох провідних платформ на ринку (Newgrounds, SoundCloud, Pixabay), розглянуто їхні архітектурні переваги й недоліки та доведено актуальність вебсервісу «MediaFlow». Розглянуто клієнт-серверний стек технологій на базі React.js, Node.js та MongoDB, в результаті чого створено програмне забезпечення. Проведено функціональне тестування вебсервісу, за результатами якого доведено, що розроблене програмне забезпечення працює коректно та готове до експлуатації.

2. Висновок про відповідність роботи поставленому завданню Кваліфікаційна робота виконана відповідно до поставленого завдання та з дотриманням всіх вимог.

3. Характеристика виконання кожного розділу роботи, ступінь використання останніх досягнень науки і техніки та передових методів роботи У вступі обґрунтовано актуальність теми, визначено мету та завдання бакалаврської роботи. У першому розділі проведено змістовний аналіз предметної галузі вебсервісів для обміну мультимедійним контентом, виконано технічний огляд трьох наявних рішень на ринку (Newgrounds, SoundCloud, Pixabay) та чітко визначено 21 функціональну й 7 нефункціональних вимог до вебсервісу. У другому розділі проаналізовано сучасні архітектурні підходи й визначено, що система базуватиметься на клієнт-серверній моделі з REST API та JWT-автентифікацією з використанням технологій React.js, Node.js та MongoDB. У третьому розділі реалізовано базу даних MongoDB із чотирма колекціями, реалізовано програмні модулі інтерфейсу з використанням Material UI та описано інструкцію користувача. Також у цьому розділі виконано функціональне тестування вебсервісу, за результатами якого підтверджено коректність та стабільність роботи створеного вебсервісу MediaFlow.

4. Позитивні сторони роботи Тематика кваліфікаційної роботи є актуальною, оскільки більшість існуючих платформ орієнтовані лише на один тип контенту, тоді як MediaFlow забезпечує єдиний простір для публікації зображень, аудіо та відео. Автором застосовано сучасний стек технологій MERN (MongoDB, Express.js, React.js, Node.js), реалізовано хмарне зберігання медіафайлів на Cloudinary з CDN-доставкою, впроваджено stateless JWT-автентифікацію та підтримку темної і світлої теми інтерфейсу.

5. Негативні сторони роботи У роботі відсутня система підписок між авторами та алгоритмічні рекомендації контенту – у майбутньому доцільно реалізувати персоналізовану стрічку на основі вподобань користувача. Також пошук реалізовано засобами MongoDB text без ранжування за релевантністю – для розширення функціональності варто розглянути інтеграцію з Elasticsearch.

6. Оцінка графічного оформлення та пояснювальної записки Графічне оформлення виконано відповідно до теми кваліфікаційної роботи та подано у вигляді діаграми варіантів використання, схеми бази даних та архітектурного рішення. Пояснювальна записка оформлена згідно з вимогами чинних стандартів.

7. Відгук про кваліфікаційну роботу в цілому Кваліфікаційна робота заслуговує позитивної оцінки. Матеріал пояснювальної записки структурований, послідовний та чіткий, що дозволяє зрозуміти викладений матеріал у рамках тематики розробки вебсервісу «MediaFlow». Графічний матеріал дає можливість наочно побачити деталі архітектури, схему бази даних та інтерфейс системи.

8. Інші зауваження

9. Оцінка кваліфікаційної роботи Кваліфікаційна робота виконана у повному обсязі, відповідає поставленій задачі та заслуговує на оцінку «добре».

РЕЦЕНЗЕНТ (прізвище, ім'я, по-батькові, посада, місце роботи) Тимбук Є.Г.
д.т.н., проф. каф. КІІС

"05" 05 2026 р. _____
(підпис)

РІШЕННЯ ЕКСПЕРНОЇ КОМІСІЇ
КАФЕДРИ ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ
ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатами звіту/звітів перевірки роботи, продукованими програмно-технічним засобом (ами), на наявність текстових збігів.
Назва кваліфікаційної роботи: Веб-платформа для обміну та зберігання медіа між користувачами
Автор: Цимбалюк Максим Валентинович
Освітня програма: Освітньо-професійна програма «Інженерія програмного забезпечення»
Спеціальність: 121 – Інженерія програмного забезпечення
Науковий керівник: Форкун Юрій Вікторович, кандидат технічних наук, доцент
Після аналізу звіту/звітів зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі є законними і не є плагіатом. Рівень подібності не перевищує допустимої межі. Таким чином робота незалежна і приймається до захисту.	**відповідає**
2	Виявлені запозичення не є академічним плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи. Робота приймається до захисту, але має бути відкоригована.	
3	Виявлені запозичення не є плагіатом, але перевищено граничне значення рівня подібностей. Виявлені запозичення частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнуті. Робота може бути допущена до захисту після того, як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття текстових запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	
5	Інше:	

Підтвердження:
Запозичення, виявлені у роботі, є законними і не є плагіатом, оскільки:
1) у тексті кваліфікаційної роботи системою перевірки на плагіат Anti-Plagiarism виявлено схожість з деякими документами у частині загальновживаних обов'язкових словосполучень у стандартних бланках, у структурі змісту, назвах розділів/підрозділів, рамках форм, у назвах та URL-адресах публікацій переліку джерел посилання;
2) запозичення, виявлені у тексті роботи, є фрагментарними.
Загальна сумарна подібність у базі даних складає 3 % за символами та 24 % за лексемами. Крім того за результатами додаткового аналізу системи StrikePlagiarism коефіцієнт подібності 1 становить 5,68 %, коефіцієнт подібності 2 – 1,99 %. З урахуванням наведених обґрунтувань, відповідає характеру теми і свідчить на користь кваліфікаційної роботи.

Дата 04.06.26

Завідувач кафедри _____ Леонід БЕДРАТЮК

Гарант освітньої програми _____ Леонід БЕДРАТЮК

Керівник кваліфікаційної роботи _____ Юрій ФОРКУН