МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра інженерії програмного забезпечення

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри інженерії
програмного забезпечення

_____ Євген ДАВИДЕНКО

«___» _____ 2026 р.

КВАЛІФІКАЦІЙНА РОБОТА
НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВРА

ВЕБЗАСТОСУНОК ВІДЕОХОСТИНГУ

Спеціальність 121 Інженерія програмного забезпечення
Освітня програма «Інженерія програмного забезпечення»

Здобувач _____ **Олександр ШКУРКІН**

«__» _____ 2026 р.

Керівник роботи
PhD, ст. викладач

_____ **Ігор КАНДИБА**

«__» _____ 2026 р.

Миколаїв – 2026

Завдання на виконання кваліфікаційної роботи

Чорноморський національний університет імені Петра Могили

Факультет	Комп’ютерних наук
Кафедра	Інженерії програмного забезпечення
Рівень вищої освіти	Перший (бакалаврський)
Освітній ступінь	Бакалавр
Спеціальність	121 Інженерія програмного забезпечення
Освітня програма	Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри інженерії
програмного забезпечення
_____ Євген ДАВИДЕНКО
«___» _____ 2026 р.

ЗАВДАННЯ

на кваліфікаційну бакалаврську роботу здобувача

Шкуркіна Олександра

1. Тема кваліфікаційної роботи «Вебзастосунок відеохостингу» затверджена наказом ректора ЧНУ ім. Петра Могили № 349 від «26» грудня 2025 р.

2. Строк представлення кваліфікаційної роботи «___» _____ 2026 р.

3. Результатом роботи є розроблений вебзастосунок відеохостингу.

4. Перелік питань, що підлягають розробці:

– проаналізувати існуючі програмні рішення, визначити їх переваги та недоліки;

– сформулювати вимоги та функціонал проєктованої системи;

– виконати проєктування архітектури ПЗ;

– реалізувати інтерфейс вебзастосунку;

– розробити функціонал: завантаження, збереження, перегляду та вивантаження відеоконтенту;

– протестувати розроблений вебзастосунок.

5. Перелік графічних матеріалів: презентація.

6. Консультанти:

Консультант	**Кафедра (організація)**	**Частина роботи**

Дата видачі завдання «01» березня 2026 р.

КАЛЕНДАРНИЙ ПЛАН
виконання кваліфікаційної роботи

Тема: **Вебзастосунок відеохостингу**

№	Найменування роботи	Початок	Закінчення	Примітки
1.	Аналіз предметної області та аналогів	19.01.2026	02.02.2026	Виконано
2.	Формулювання науково-технічної задачі КБР	03.02.2026	10.02.2026	Виконано
3.	Складання календарного плану КБР	11.02.2026	18.02.2026	Виконано
4.	Ознайомлення з літературою за темою роботи	19.02.2026	25.02.2026	Виконано
5.	Проєктування та моделювання інтерфейсу та архітектури ПЗ	19.02.2026	10.03.2026	Виконано
6.	Активна розробка (кодування) ПЗ	12.03.2026	22.05.2026	Виконано
7.	Тестування ПЗ, виправлення помилок, виконання остаточних оптимізацій роботи системи	22.05.2026	25.05.2026	Виконано
8.	Відгук керівника КБР	26.05.2026	26.05.2026	Виконано
9.	Оформлення КБР та презентації	26.05.2026	26.05.2026	Виконано
10.	Попередній захист	27.05.2026	27.05.2026	Виконано
11.	Завершення оформлення КБР та презентації	28.06.2026	09.06.2026	Виконано
12.	Рецензування	15.06.2026	16.06.2026	Виконано
13.	Захист кваліфікаційної роботи	Х.06.2026	Х.06.2026	

Здобувач _____ **Олександр ШКУРКІН**

«__» _____ 2026 р.

Керівник роботи

PhD, ст. викладач

_____ **Ігор КАНДИБА**

«__» _____ 2026 р.

АНОТАЦІЯ

до кваліфікаційної бакалаврської роботи

«Вебзастосунок відеохостингу»

Здобувач 409 гр.: Шкуркін Олександр

Керівник: PhD, ст. викладач Кандиба Ігор

Актуальність теми «Вебзастосунок відеохостингу» зумовлена зростаючою потребою у інструментах збереження та поширення інформації для наукової, освітньої, політичної та розважальної сфер. Інтернет став основним джерелом інформації, а з популяризацією таких платформ як TikTok та Instagram все більше людей віддають перевагу відеоконтенту. Поширення знань у відео форматі призведе до кращого їх розуміння та збільшить загальну освіченість суспільства.

Для України ця тема є особливо актуальна – із збільшенням кількості аварійних відключень електроенергії велика кількість здобувачів освіти часто не має можливості відвідувати заняття за графіком, що може призвести до прогалин у знаннях. За наявності інструментів хостингу відеоконтенту педагоги мають можливість зберігати записи своїх попередніх занять на вебсайті, що дозволить учням наздогнати пропущене у будь-який доступний або зручний час.

Метою кваліфікаційної роботи є розробка вебзастосунку відеохостингу для полегшення поширення наукової та освітньої інформації.

Об’єктом кваліфікаційної роботи є процес зберігання та поширення відеоконтенту.

Предметом кваліфікаційної роботи є методи та засоби реалізації компонентів вебзастосунку відеохостингу.

У вступній частині підкреслено актуальність розробки вебзастосунку відеохостингу, сформульовано мету та завдання роботи, визначено об’єкт та предмет розробки.

У першому розділі досліджується предметна область, аналізуються наявні аналоги програмному рішенню, що розробляється.

У другому розділі визначаються вимоги до програмного забезпечення, описується функціонал, аналізується наявний інструментарій та методології розробки.

У третьому розділі моделюється архітектура програмного забезпечення, проєктуються функції, створюються діаграми та описи структури вебзастосунку, визначаються технології та мови програмування, що будуть використані під час програмної реалізації.

У четвертому розділі представляється процес кодування та тестування програмного забезпечення, надається аналіз результатів тестування, описується фінальний продукт.

У розділі висновків оцінюються результати виконаних досліджень та розробки, підкреслюється практичне значення результатів, порівнюються характеристики програмного забезпечення із поставленими завданнями.

КБР викладена на ___сторінки, вона містить ___ розділи, ___ілюстрацій, ___ таблиці, ___ джерел в переліку посилань.

Ключові слова: *збереження інформації, поширення інформації, вебзастосунок, відеоконтент, освітня сфера, наукова сфера*.

ABSTRACT

of the Bachelor's Thesis

"Video hosting web application"

Student of group 409: Shkurkin Oleksandr

Supervisor: PhD, Senior Lect. Kandyba Ihor

The relevance of the topic "Video hosting web application" is due to the growing need for tools for storing and distributing information for scientific, educational, political and entertainment purposes. The Internet has become the main source of information, and with the popularization of platforms such as TikTok and Instagram, more and more people prefer video content. The dissemination of knowledge in video format will lead to a better understanding of it and increase the general education of society.

This topic is especially relevant for Ukraine – with the increase in the number of emergency power outages, a large number of students often do not have the opportunity to attend classes according to the schedule, which can lead to gaps in knowledge. With the availability of video content hosting tools, teachers have the opportunity to store recordings of their previous classes on the website, which will allow students to catch up on what they missed at any available or convenient time.

The purpose of the bachelor's thesis is to develop a video hosting web application to facilitate the dissemination of scientific and educational information.

The object of the bachelor's thesis is the process of storing and distributing video content.

The subject of the bachelor's thesis is the methods and means of implementing the components of the video hosting web application.

The introduction highlights the relevance of developing a video hosting web application, sets out the aim and objectives of the work, and defines the scope and subject of the development.

The first chapter examines the subject area and analyses existing equivalents to the software solution under development.

The second chapter defines the software requirements, describes the functionality, and analyses the available tools and development methodologies.

The third chapter models the software architecture, designs the functions, creates diagrams and descriptions of the web application's structure, and identifies the technologies and programming languages to be used during software implementation.

The fourth chapter presents the process of coding and testing the software, provides an analysis of the test results, and describes the final product.

The conclusions chapter evaluates the results of the research and development carried out, highlights the practical significance of the results, and compares the software's characteristics with the tasks set.

The bachelor's thesis is laid out on _ pages, it contains _ chapters, _ images, _ tables, _ references.

Keywords: *information storing, information distribution, web application, video content, educational sphere, scientific sphere*.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ ТА ТЕРМІНІВ

БД	– база даних
ПЗ	– програмне забезпечення
РСКБД	– реляційна система керування базами даних
Ajax	– asynchronous JavaScript and XML
API	– application programming interface
CSRF	– cross-site request forgery
DOM	– document object model
DTO	– data transfer object
E2e	– end-to-end
EF Core	– entity framework core
HTTP	– hypertext transfer protocol
HTTPS	– hypertext transfer protocol secure
JSON	– JavaScript object notation
JWT	– JSON web token
LINQ	– language integrated query
MVC	– model-view-controller
MVVM	– model-view-viewmodel
ORM	– object-relational mapping
REST	– архітектурний стиль API
SQL	– structured query language
SSL/TLS	– secure socket layer/transport layer security
UML	– unified modelling language
URL	– uniform resource locator
XSS	– cross-site scripting

ВСТУП

Актуальність теми «Вебзастосунок відеохостингу» зумовлена зростаючою потребою у інструментах збереження та поширення інформації для наукової, освітньої, політичної та розважальної сфер. Інтернет став основним джерелом інформації, а з популяризацією таких платформ як TikTok та Instagram все більше людей віддають перевагу відеоконтенту. Поширення знань у відео форматі призведе до кращого їх розуміння та збільшить загальну освіченість суспільства.

Для України ця тема є особливо актуальна – із збільшенням кількості аварійних відключень електроенергії велика кількість здобувачів освіти часто не має можливості відвідувати заняття за графіком, що може призвести до прогалин у знаннях. За наявності інструментів хостингу відеоконтенту педагоги мають можливість зберігати записи своїх попередніх занять на вебсайті, що дозволить учням наздогнати пропущене у будь-який доступний або зручний час.

Метою кваліфікаційної роботи є розробка вебзастосунку відеохостингу для полегшення поширення наукової та освітньої інформації.

Відповідно до мети визначено такі завдання:

1. проаналізувати існуючі програмні рішення, визначити їх переваги та недоліки;
2. сформулювати вимоги та функціонал проєктованої системи;
3. виконати проєктування архітектури ПЗ;
4. реалізувати інтерфейс вебзастосунку;
5. розробити функціонал: завантаження, збереження, перегляду та вивантаження відеоконтенту;
6. протестувати розроблений вебзастосунок.

Об’єктом роботи є процес зберігання та поширення відеоконтенту.

Предметом роботи є методи та засоби реалізації компонентів вебзастосунку відеохостингу.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Опис предметної області, об’єкту та предмету дослідження

Відеохостинг це один із типів вебсервісу, який призначений для обробки, зберігання та поширення відеоконтенту. Вже довгий час такі сервіси є невід’ємною частиною віртуального середовища, адже велика кількість сфер діяльності активно використовують функціонал та інструментарій таких систем. Розвиток та масове впровадження високошвидкісного інтернету разом із використанням нових вебтехнологій сприяли значному зростанню популярності відеохостингів.

Сьогодні відеохостинг активно використовується в різних галузях, зокрема у розважальній, освітній, науковій, маркетинговій та політичній сферах. Розважальний сектор займає провідні позиції за обсягом споживаного контенту, оскільки користувачі активно переглядають відео з метою дозвілля. Водночас освітні платформи використовують відеохостинг для розміщення лекцій, курсів та навчальних матеріалів, що допомагає підтримувати можливість дистанційного навчання. У науковій сфері відеоконтент застосовується для демонстрації експериментів, презентації результатів досліджень та поширення наукових знань. Політичні організації використовують відеохостинг як інструмент комунікації з аудиторією та формування громадської думки.

Незважаючи на значну кількість існуючих рішень у сфері відеохостингу, більшість із них орієнтовані переважно на розважальний контент. Це обмежує можливості їх використання у більш спеціалізованих сферах, де потрібні додаткові функціональні можливості, що включають підтримку категоризації, механізми розширеного пошуку, фільтрації та рекомендацій. Крім того, актуальними є питання безпеки, захисту авторських прав, контролю доступу та ефективного управління великими обсягами даних.

Актуальність створення вебзастосунку відеохостингу полягає у необхідності розробки універсального програмного рішення, яке

забезпечувало б гнучкість функціоналу відповідно до потреб різних категорій користувачів. Такий застосунок повинен підтримувати масштабованість, високу продуктивність та ефективну обробку мультимедійних даних.

Об’єктом дослідження виступають процеси зберігання, обробки та розповсюдження відеоконтенту. Ці процеси охоплюють повний життєвий цикл сервісу: від завантаження відеофайлу користувачем, його обробки та зберігання у хмарному сховищі, до стримінгу кінцевому користувачеві. Важливою складовою об’єкта дослідження також є взаємодія між клієнтами та системою, що включає механізми авторизації, управління контентом та реакції (такі як підписки та коментарі).

Предметом дослідження є методи, моделі та засоби реалізації компонентів вебзастосунку відеохостингу. До них належать архітектурні рішення, алгоритми обробки відео, технології зберігання даних та інструменти розробки клієнтської та серверної частин застосунку.

Окремо предмет дослідження охоплює методи оптимізації продуктивності системи, такі як кешування та асинхронну обробку даних. Ще одним важливим аспектом є забезпечення інформаційної безпеки, що включає захист персональних даних користувачів та контроль над доступом до контенту та функцій застосунку.

1.2 Структурні та функціональні особливості вебзастосунку

Враховуючи специфіку вебсервісів відеохостингу, під час проєктування такої системи необхідно врахувати найпоширеніші структурі та функціональні виклики, що можуть створити проблеми під час розробки:

– зберігання та швидке вилучення великих обсягів даних;
– обробка відео – транскодування файлів різних форматів;
– забезпечення швидкого та стабільного потокового відтворення;
– оптимізація систем пошуку та фільтрації;
– забезпечення безпеки персональних даних та контроль доступу до функціоналу.

Для спрощення вирішення цих викликів вебзастосунок використовуватиме трирівнену архітектуру – поділ на рівні клієнта, серверу та даних дозволяє розподілити відповідальності системи між компонентами та гарантувати її масштабованість у майбутньому.

Клієнтський рівень, або frontend, реалізовано із використанням бібліотеки React – це дозволяє створити компонентний односторінковий вебзастосунок. Такий підхід забезпечує швидке завантаження сторінок та покращений користувацький досвід, що частково вирішує проблему стабільного відтворення контенту за рахунок кешування даних.

Серверний рівень, або backend, реалізовано із використанням платформи .NET, що є сучасним середовищем для розробки вебзастосунків клієнт-серверної архітектури мовою C#. Наявність вбудованого функціоналу обробки HTTP-запитів та маршрутизації дозволяє створити швидкий та розширюваний REST API для обробки запитів і даних користувачів. Ця платформа також підтримує велику кількість бібліотек та інструментів, необхідних для реалізації деякого функціоналу вебсервісу:

1. FFmpeg [1] – інструмент, що дозволяє виконувати транскодування відеофайлів у різні формати та забезпечувати підтримку декількох варіантів якості відтворення;
2. bcrypt [2] – бібліотека хешування чутливих даних, таких як паролі користувачів, що забезпечує їх безпеку у разі несанкціонованого доступу до бази даних;
3. JwtBearer [3] – офіційний пакет від Microsoft, що надає можливість реалізувати механізми JWT автентифікації та авторизації для контролю за доступом до частин та ресурсів системи.

Рівень даних складається із локального та хмарного сховищ. Локальне сховище представляє РСКБД Microsoft SQL Server, яка забезпечує зберігання структурованих даних та підтримує індексацію для прискорення пошуку, що дозволяє вирішити проблему швидкої роботи з великими обсягами інформації. Хмарна частина реалізована за допомогою сервісу Amazon S3 і

використовується для зберігання відеофайлів та зображень – це дозволяє легко масштабувати систему та зберігати великі обсяги даних.

1.3 Аналіз сучасних аналогічних програмних рішень

YouTube

YouTube [4] є найбільшою у світі платформою відеохостингу, яка обслуговує мільярди користувачів і забезпечує доступ до величезної кількості відеоконтенту різних категорій. Сервіс активно використовується як для розважальних, так і для освітніх, наукових і комерційних цілей. Платформа постійно розвивається, оновлюючи свій алгоритм рекомендацій та інструменти поширення і зберігання контенту.

Розробники: Стів Чен, Джавед Карім, Чад Герлі.

Власник: Google.

Архітектура: Microservices.

Мови реалізації: JavaScript, C, C++, Go, Java, Python.

Основні функції YouTube:

– завантаження та перегляд відео контенту;
– персоналізована головна сторінка із рекомендаціями;
– алгоритм підбору наступних відео для перегляду;
– трансляція відео на екран телевізору чипроєктора;
– створення плейлістів та списку відео «Переглянути пізніше»;
– можливість створення відео-черги.

Переваги YouTube:

– безкоштовний доступ до більшої частини контенту;
– велика різноманітність контенту;
– можливість заробітку для контент-мейкерів, монетизуючи свої відео за допомогою реклами та спонсорства;
– можливість взаємодії із автором та користувачами за допомогою коментарів та лайків/дизлайків.

Недоліки YouTube:

– надмірна кількість реклами;

– дуже велика конкуренція між авторами;

– алгоритм викликає залежність: для глядачів залежність постійного перегляду, для авторів залежність від графіку завантаження відео.

Хоч YouTube і є універсальною платформою з широким функціоналом та різноманітністю контенту, занадто агресивна політика монетизації та жорстка конкуренція спонукає деяких користувачів шукати альтернативні сервіси відеохостингу.

Dailymotion

Dailymotion [5] є відеохостинг-платформою, яка орієнтована на поширення відеоконтенту з акцентом на співпрацю з медіа організаціями та новинними ресурсами. У порівнянні з іншими платформами, сервіс пропонує більш спрощений інтерфейс та меншу кількість рекламного контенту.

Розробники: Бенжамін Бежбом, Олів'є Пуатре.

Власник: Canal+.

Архітектура: Client-server.

Мови реалізації: PHP, JavaScript, Python, Go.

Основні функції Dailymotion:

– завантаження та поширення відео контенту;

– прямі трансляції;

– інтеграція із медія та новинними компаніями;

– персоналізовані рекомендації;

– перегляд у форматі «картинка-в-картинці».

Переваги Dailymotion:

– простий інтерфейс;

– менша кількість реклами ніж у інших платформах;

– орієнтація на професійних медіа та авторів.

Недоліки Dailymotion:

– обмежена кількість доступних тематик;

– обмежені моливості взаємодії із автором (немає можливості додавати коментарі чи «не рекомендувати» відео).

Dailymotion це непогана альтернатива великим платформам, адже достатньо широкий функціонал та простий інтерфейс дозволяє користувачам почати використовувати сервіс без серйозних проблем, а співпраця із медіа та професійними контент-мейкерами забезпечує вищу якість поширюваного відеоконтенту. Менша кількість категорій та тем доступних відео, однак, є серйозним недоліком, що сильно зменшує кількість потенційно зацікавлених користувачів.

Rumble

Rumble [6] є сервісом відеохостингу, який позиціонує себе як альтернатива традиційним сервісам, пропонуючи більш гнучкі умови монетизації та менш жорстку політику модерації контенту. Платформа орієнтована на авторів, які прагнуть швидкого доступу до аудиторії та спрощених механізмів заробітку.

Розробник: Кріс Павловскі.

Власник: Rumble Inc.

Архітектура: Client-server.

Мови реалізації: PHP, JavaScript, Python, Go.

Основні функції Rumble:

– щоденна таблиця лідерів за переглядами;
– прямі трансляції;
– персоналізовані рекомендації;
– окремі рекомендації, підібрані редакторами;
– створення нових категорій відео.

Переваги Rumble:

– легка система монетизації;
– менш жорстка цензура;
– більш різноманітний алгоритм рекомендацій.

Недоліки Rumble:

– однотипний контент, сфокусований на політиці;
– обмежений інструментарій для авторів.

Rumble це одна із більш відомих альтернатив великим платформам відеохостингу і ця популярність була досягнута частково завдяки розслабленій політиці модерації та спрощеній системі монетизації, яка надає можливість авторам створювати більш різноманітний контент і швидше на цьому заробляти. Краща категоризація відеоконтенту на платформі також зацікавлює не малу кількість людей, що прагнуть знайти відео на нові для себе теми. Проте, менш жорстка цензура приваблює на платформу людей з екстремальними поглядами – ультра-праві політики та псевдонауковці використовують цей сервіс для поширення своїх поглядів та пошуку однодумців.

Висновки до розділу 1

У першому розділі проведено системний аналіз предметної області, розкрито об’єкт та предмет кваліфікаційної роботи. Досліджено структурі та функціональні виклики під час розробки вебсервісів відеохостингу, на основі яких обґрунтовано вибір архітектури та інструментарію, що допоможе подолати усі можливі проблеми і створити підґрунтя для подальшого розширення функціоналу застосунку.

Проаналізовано наявні сучасні альтернативні програмні рішення для визначення їх основного функціоналу, переваг та недоліків. На основі цієї інформації виділено основні напрямки розвитку та покращення застосунку, що розробляється. Інкорпорація позитивних аспектів ПЗ-аналогів покращить конкурентоспроможність майбутнього продукту.

2 АНАЛІЗ ІНСТРУМЕНТАРІЮ ТА ФОРМУВАННЯ ВИМОГ

2.1 Аналіз сучасного стану інструментарію

Перед проєктуванням та розробкою застосунку необхідно проаналізувати доступні сучасні інструменти, що дозволяють вирішити поставлені завдання. Аналіз інструментів та методів виконується у два етапи: пошук перевірених та актуальних технологій та їх порівняння між собою.

Технології фронтенд розробки

Більша частина сучасних вебсервісів використовують одну із трьох популярних фронтенд технологій: фреймворк Angular [7], бібліотека React [8] або фреймворк Vue [9]. Усі ці технології мають довгу історію розробки та оновлень, а також підтримують велику кількість сторонніх бібліотек та інструментів. Хоч ці технології мають спільну мету – спрощення розробки та управління користувацької частини застосунку – кожна з них має сильні та слабкі сторони.

Популярність

Кількість користувачів, що зацікавлені або використовують якусь технологію, не є стовідсотковим показником її якості, однак це впливає на кількість документації та доступних сторонніх бібліотек, що розширюють її функціонал.

Найбільш популярною фронтенд бібліотекою вже довгий час є React – станом на 2021 рік 49% розробників регулярно використовують цю бібліотеку. Друге місце за популярністю посідає фреймворк Vue, із яким регулярно працюють 43% розробників. Не зважаючи на те, що Angular це найстарший фреймворк із трьох та має широкий функціонал, їм регулярно користуються лише 18% розробників [10]. З урахуванням кількості вільно доступного стороннього інструментарію та навчальних матеріалів, React є чудовим вибором як для молодих розробників, так і для ветеранів сфери, а отже можна зробити припущення, що популярність цієї бібліотеки буде залишатись висока (рис. 2.1).

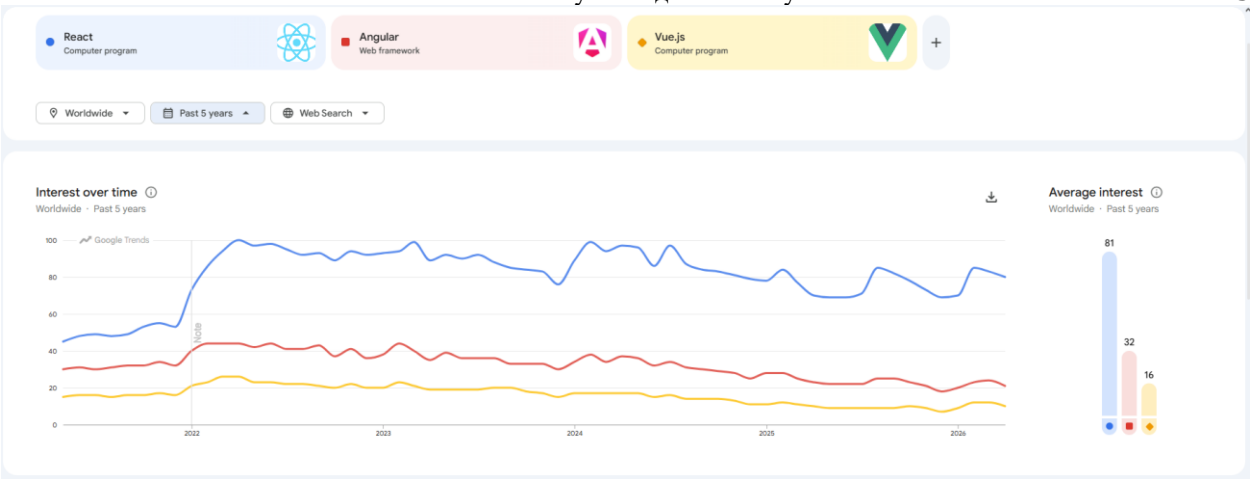

Рисунок 2.1 – Популярність фронтенд технологій на Google Trends

Функції

Angular:

Angular є наймасштабнішим та найстаршим фреймворком із трьох, а отже має і найдовший список вбудованих функцій [11]:

– керування інтерфейсом користувача;

– реакція на введені користувачем дані;

– перевірка введених користувачем даних у формах;

– маршрутизація, керування станами;

– надсилання HTTP-запитів Ajax;

– надання офлайн-підтримки та можливостей PWA (Progressive Web App);

– забезпечення інструментів комплексного тестування;

– керування кількома застосунками та підключенням їх один до одного.

React:

React це open-source JavaScript бібліотека, а отже має меншу кількість вбудованих функцій. Однак, відкритий код та компонентна структура дозволяє легко створювати та поширювати сторонні бібліотеки для додавання нового функціоналу. React надає наступні функції [11]:

– фокус на повторне використання коду;

– простота використання;

– налаштовуваність – вибір бібліотек та інструментарію, що буде використаний;

– оптимізація швидкості та ефективності змін у DOM за допомогою віртуального DOM.

Vue:

Vue це наймолодший фреймворк із трьох. Він має багато спільного із бібліотекою React, однак надає трохи більше вбудованих функцій [11]:

– маршрутизація та керування станами;

– оптимізація роботи з DOM завдяки легшому віртуальному DOM;

– спрощення налагодження інтерфейсу завдяки відображенню змін, що відбувається паралельно процесу написання коду;

– оптимізація часу запуску застосунку.

Порівняння

Для порівняння цих технологій використано їх загальні характеристики та захист від атак [12] (табл. 2.1), а також продуктивність під час роботи застосунку (табл. 2.2).

Таблиця 2.1 – Характеристики фреймворків

Назва	Angular	React	Vue
Архітектура	Component-based	Flux	MVVM
Механізм рендерингу	Real DOM	Virtual DOM	Virtual DOM
Прив’язка даних	One-way from data source to view target, one-way from view target to data source, two-way	One-way from component to view, one-way from view to component	Reactive two-way

Кінець таблиці 2.1

Мова програмування		TypeScript	JSX (JavaScript + HTML/CSS), TSX	HTML, CSS, JavaScript використовуються окремо один від одного у шаблонах
Безпека	XSS	Вбудована	Вбудована	Вбудована
	Clickjacking	Немає	Немає	Немає
	CSRF	Немає	Вбудована	Немає
	Remote Code Execution	Немає	Немає	Вбудована

Для тестування продуктивності фреймворків фахівці створили сторінку, що має велику таблицю рандомізованих даних, над якими виконувались різні операції [10]. Результати операцій порівнювались за швидкістю виконання (у мілісекундах) та використаній пам’яті (у мегабайтах).

Таблиця 2.2 – Продуктивність фреймворків

Назва	Тип результату	Angular	React	Vue
Завантаження сторінки	Час	2.0 мс	1.4 мс	1.3 мс
	Пам’ять	1.57 МБ	1.14 МБ	1.00 МБ
Додавання 1,000 рядків	Час	5.3 мс	5.6 мс	4.4 мс
	Пам’ять	1.22 МБ	1.28 МБ	1.00 МБ
Оновлення кожного десятого рядка 5 раз	Час	5.4 мс	6.1 мс	4.4 мс
	Пам’ять	1.22 МБ	1.39 МБ	1.00 МБ

Кінець таблиці 2.2

Створення та видалення 1,000 рядків 5 раз	Час	2.7 мс	2.2 мс	1.6 мс
	Пам’ять	1.71 МБ	1.39 МБ	1.00 МБ
Додавання 10,000 рядків	Час	32.6 мс	39.5 мс	30.5 мс
	Пам’ять	1.07 МБ	1.30 МБ	1.00 МБ

Хоч результати доволі близькі, але фреймворк Vue показав найкращі результати як по часу, так і по використанню пам’яті під час виконання усіх тестів. При порівнянні React та Angular перший показав кращі результати під час завантаження сторінки та видалення даних, а другий виявився швидшим під час додавання та оновлення рядків.

У підсумку, під час аналізу вибраних технологій Vue показав себе як швидкий та легкий фреймворк, у той час як Angular обмінює більший розмір на ширший функціонал. Бібліотека React надає достатній рівень продуктивності, а великий список доступних сторонніх бібліотек і можливість їх гнучкого налаштування дозволяє легко побудувати модульний проєкт під будь-яку задачу. Враховуючи кількість функцій та компонентів, що будуть наявні у вебзастосунку відеохостингу, можливість знайти перевірене рішення та швидко інтегрувати його у проєкт грає велику роль у масштабованості системи.

Мови програмування для бекенд розробки

Популярність

Сектор бекенд розробки має багату екосистему різних технологій та мов програмування. Найпопулярніша мова програмування за TIOBE index (рис. 2.2) це Python – відносно легкий синтаксис разом із нескінченною кількістю документації та навчальних матеріалів значно полегшує вивчення та початок роботи із цією мовою. Другою за популярністю мовою, що використовується для бекенд розробки, це Java – складніший та більш

громіздкий синтаксис хоч і робить вивчення та використання цієї мови важче, але кількість систем, що залежать від Java, залишається великим. Мова C# [13], яка використовується у платформі .NET, посідає третє місце за популярністю розробки бекенд – хоч у .NET відсутня кросплатформність, з C# простіше працювати ніж з Java, а платформа надає широкий інструментарій та інтеграцію з великою кількістю сервісів. П’ятірку лідерів за популярністю замикають JavaScript та Golang [14].

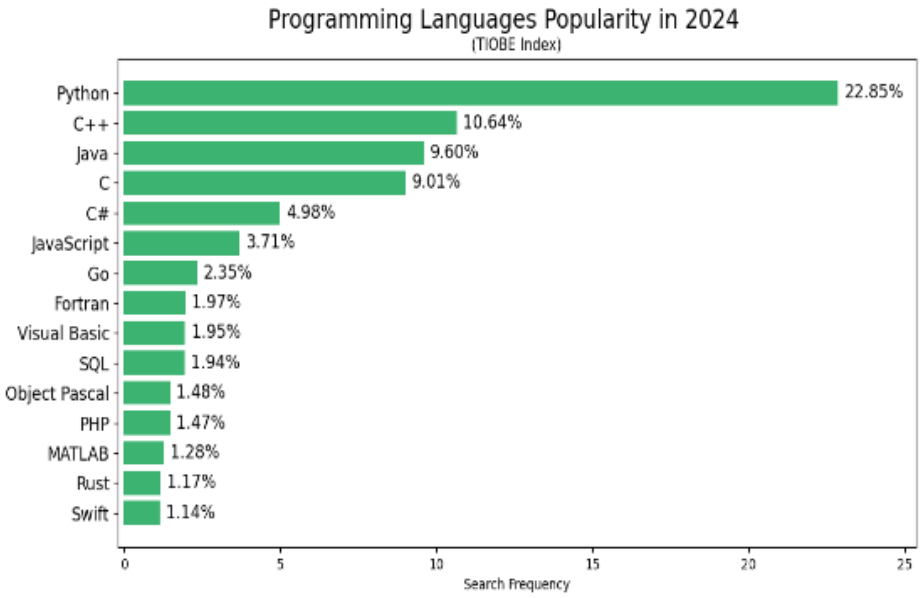

Рисунок 2.2 – TIOBE index популярності мов програмування

Порівняння

Для порівняння продуктивності було вибрано 4 мови програмування: Golang, C#, JavaScript та більш стара, але все ще актуальна мова, PHP [15]. Для тестування ефективності роботи цих мов фахівці створили простий API застосунок, який мав 5 endpoints:

– GET /api/files – повертає дані з 10 файлів. Перевіряє здатність додатка одночасно відкривати та зчитувати багато файлів;

– GET /api/database – повертає дані з бази даних. Перевіряє ефективність з’єднання з базою даних та продуктивність запитів до неї;

– GET /api/calc – повертає результат сортування 5000 елементів за допомогою алгоритму бульбашкового сортування. Вимірює ефективність обробки складних операцій;

– POST /api/task – приймає параметри та записує їх у базу даних. Тестує запис у базу та операції CRUD;

– POST /api/measurement – приймає список чисел з плаваючою комою, сортує їх та записує результат у файл.

Кожен застосунок оцінювався за об’ємом зайнятого дискового простору, загальним часом виконання, обробленими запитами за секунду та затримкою, тестування відбувалось із симуляцією 100 користувачів та 1000 запитів (табл. 2.3).

Таблиця 2.3 – Порівняння продуктивності мов програмування бекенду

Мова	Об’єм зайнятого дискового простору	Загальний час виконання	Оброблені запити за секунду	Затримка
Golang	0.4 МБ	25 с	392	254 с
C#	18.6 МБ	23 с	421	238 с
JavaScript	3.1 МБ	44 с	227	437 с
PHP	20.3 МБ	960 с	10	10,000 с

Із результатів порівняння можна визначити, що PHP показав себе найгірше в усіх аспектах тестування. C#, завдяки своїм асинхронним функціям, виявився найшвидшою з усіх мов та зміг обробити найбільше запитів за секунду, однак програма зайняла 18.6 МБ дискового простору. Golang виявився непоганим конкурентом C#, адже йому знадобилось усього на 2 секунди більше часу для обробки усіх запитів, а за об’ємом використаного дискового простору зайняв перше місце.

Враховуючи можливості інтеграції із різними сервісами та багату екосистему бібліотек, що надає платформа .NET, C# виступає як добрий вибір мови програмування для розробки бекенду вебзастосунку відеохостингу.

Активне використання асинхронності у C# також дозволить оптимізувати обробку запитів, комунікацію із БД та роботу з відеофайлами.

Реляційні системи керування базами даних

Популярність

Для роботи з реляційними БД у розробників є доволі широкий вибір РСКБД. Лідером у списку популярних РСКБД є Oracle DB (рис. 2.3) – enterprise-grade база даних, що обслуговує тисячі підприємств та має великий список функцій. На другому місці за популярністю є MySQL – вона має менше функцій, ніж у Oracle DB, однак її простіше вивчити та з нею легше працювати. Остання РСКБД у трійці лідерів це Microsoft SQL Server – реляційна база даних від Microsoft, що набула популярності завдяки своїй безпеці даних та мові T-SQL, яка надає додаткові можливості для створення і виконання запитів [16].

Rank	DBMS	Score	Type
1	Oracle	1262.66	Relational
2	MySQL	1228.38	Relational
3	Microsoft SQL Server	981.95	Relational
4	PostgreSQL	577.15	Relational
5	MongoDB	496.16	Document
6	Redis	168.31	Key-value
7	IBM Db2	165.15	Relational
8	Elasticsearch	155.76	Search engine
9	SQLite	130.2	Relational
10	Cassandra	114	Wide column
11	Microsoft Access	113.45	Relational
12	MariaDB	97.98	Relational
13	Splunk	90.05	Search engine
14	Hive	82.68	Relational
15	Microsoft Azure SQL Database	75.22	Relational
16	Amazon DynamoDB	75.2	Multi-model
17	Teradata	68.95	Relational
18	Neo4j	57.16	Graph
19	SAP HANA	53.81	Relational
20	Solr	51.79	Search engine

Рисунок 2.3 – Таблиця популярності СКБД

Функції

Для проведення аналізу та порівняння вибрано дві РСКБД: Oracle DB та Microsoft SQL Server.

Функції Oracle DB [17]:

– багатокористувацька архітектура – об’єднання численних баз даних в єдиний екземпляр, що дозволяє мінімізувати використання ресурсів, керуючи ними як повністю окремими об’єктами;

– PL/SQL – власне розширення мови процедур Oracle до SQL, що надає широкі можливості для збережених процедур, вдосконаленого управління помилками та складних операцій із запитами;

– векторний пошук на основі штучного інтелекту – власний векторний тип даних, що забезпечує семантичний пошук за схожістю в неструктурованих даних, поєднуючи моделі на основі трансформерів із традиційними SQL-запитами для мультимодальної аналітики;

– технологія Flashback – перегляд історичних даних без необхідності відновлення на певний момент часу, що забезпечує швидке відновлення після людських помилок або випадків пошкодження даних;

– розширена безпека – віртуальні приватні бази даних, прозоре шифрування даних, брандмауер SQL для запобігання ін'єкціям під час виконання та механізми безпеки міток, розроблені для галузей із суворими нормативними вимогами.

Функції SQL Server [17]:

– шифрування даних – прозоре шифрування даних (TDE) для даних у стані спокою та SSL/TLS для даних під час передачі;

– In-Memory OLTP (online transaction processing) – значно підвищує швидкість обробки транзакцій і скорочує затримку завдяки усуненню вузьких місць у дискових операціях вводу-виводу;

– групи доступності Always On – висока доступність корпоративного рівня та можливості відновлення після збою завдяки використанню декількох реплік з автоматичним або ручним переходом на резервну систему;

– оптимізація плану з урахуванням параметрів – динамічне кешування декількох шляхів виконання для параметризованих запитів, що усуває вузькі місця, пов'язані з аналізом параметрів;

– таблиці реєстру – незмінність у стилі блокчейну завдяки криптографічному хешуванню забезпечує аудит із захистом від несанкціонованого втручання для дотримання нормативних вимог;

– T-SQL – власна процедурна мова SQL Server, що пропонує потужні можливості запитів, комплексну обробку помилок та складні функції маніпулювання даними.

Порівняння

Фахівці провели порівняння між РСКБД Oracle DB та Microsoft SQL Server для аналізу їх безпеки [18]. Порівняння розподіллено на дві частини: аналіз ефективності шифрування даних та кількість виявлених вразливостей.

Для аналізу ефективності шифрування створено таблицю на 507,802 рядки, де кожен рядок необхідно зашифрувати та розшифрувати алгоритмами блочного шифрування AES128 та AES192. Для більшої точності результатів проведено по 5 тестів на кожний алгоритм, після чого РСКБД порівнюються за середнім часом виконання операції (табл. 2.4).

Таблиця 2.4 – Порівняння ефективності шифрування даних

Назва тесту	Microsoft SQL Server	Oracle DB
Отримання усіх рядків таблиці	7263.3 мс	17042.6 мс
Шифрування даних алгоритмом AES128	16861.3 мс	52168.6 мс
Дешифрування даних алгоритмом AES128	8981 мс	60305.3 мс
Шифрування даних алгоритмом AES192	16852.3 мс	59980 мс
Дешифрування даних алгоритмом AES192	10026.3 мс	67352.3 мс

Як можна побачити з результатів у таблиці 2.4, Microsoft SQL Server виконав операції читання, шифрування та дешифрування даних набагато

швидше, ніж Oracle DB. Така велика різниця у часі показує, що при роботі із великим об’ємом чутливих даних Microsoft SQL Server є найкращим вибором серед двох РСКБД.

Для аналізу кількості виявлених вразливостей фахівці проаналізували звіти про вразливості від двох компаній від 1999 року до 2023 року (рис. 2.4 та рис. 2.5) [18].

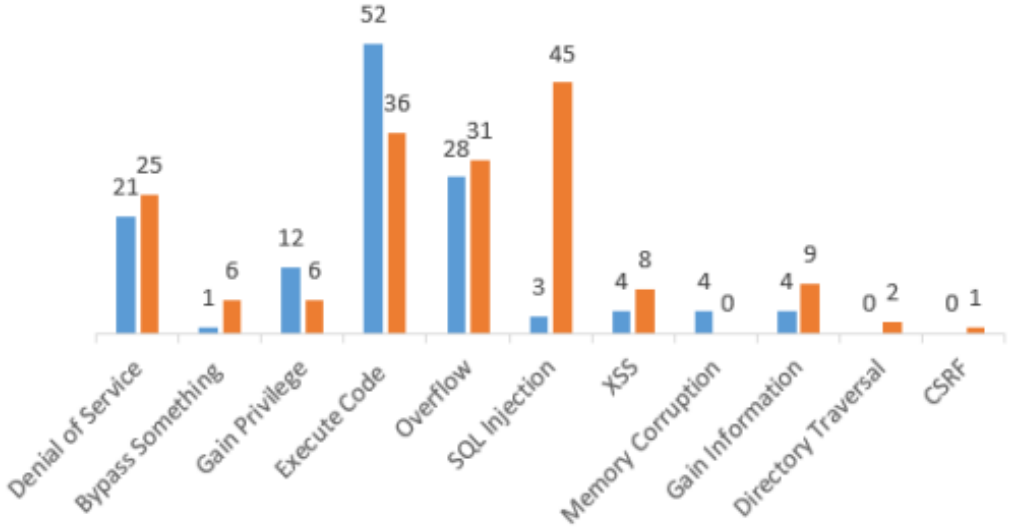

Рисунок 2.4 – Діаграма виявлених вразливостей

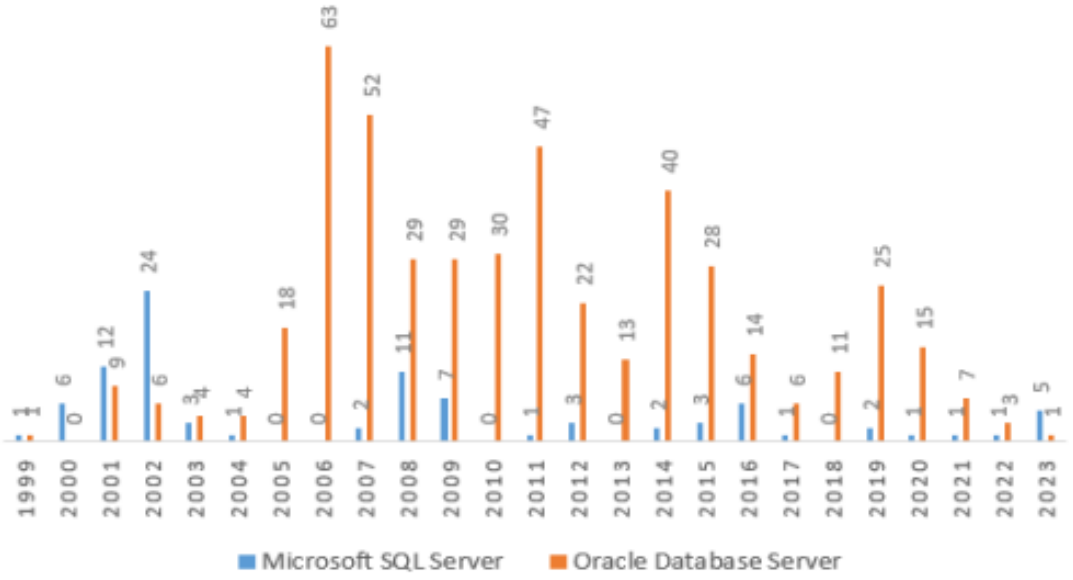

Рисунок 2.5 – Діаграма порічного виявлення вразливостей

У процесі аналізу виявлено, що протягом зазначеного періоду часу в Oracle DB було знайдено більше вразливостей, ніж у Microsoft SQL Server.

Однак, як зазначають фахівці, ці результати не доводять, що Oracle DB менш захищений. Деякі типи вразливостей вказують на те, що Oracle DB піддавався більшій кількості тестувань безпеки, що, ймовірно, зробили його більш захищеним.

Окрім комплексного аналізу безпеки цих РСКБД, фахівці також порівняли продуктивність Oracle DB та Microsoft SQL Server під час запису великої кількості даних [19]. Для тестування використано однаковий набір даних, які записувались зазначену кількість раз (табл. 2.5).

Таблиця 2.5 – Порівняння часу запису даних

Кількість ітерацій запису	Oracle DB	Microsoft SQL Server
1000	0,003 мс	0,009 мс
2000	0,02 мс	0,007 мс
3000	0,02 мс	0,008 мс
4000	0,01 мс	0,009 мс
5000	0,004 мс	0,007 мс
6000	0,005 мс	0,012 мс
7000	0,02 мс	0,01 мс
8000	0,006 мс	0,014 мс
9000	0,004 мс	0,007 мс
10000	0,009 мс	0,008 мс

З результатів аналізу можна визначити, що Microsoft SQL Server працює набагато краще за Oracle DB під час запису меншої кількості даних за раз, однак із збільшенням кількості інформації дві РСКБД починають мати відносно однакову швидкість запису.

Враховуючи можливу кількість даних, що треба зберігати та обробляти під час роботи вебзастосунку відеохостингу, Microsoft SQL Server є кращим варіантом РСКБД, адже він пропонує швидше читання даних, більш ефективне шифрування та достатній функціонал.

2.2 Специфікація вимог до ПЗ

1) Призначення та межі проєкту

1.1) Призначення системи

Вебзастосунок відеохостингу призначений для надання користувачам засобів збереження, структурування та поширення освітнього, наукового та розважального відеоконтенту.

1.2) Погодження, ухвалені в програмній документації

– клієнтська частина реалізовується з використанням React;

– серверна частина реалізовується на платформі .NET Framework (ASP.NET Web API);

– використання системи управління базами даних Microsoft SQL Server;

– збереження відеофайлів у хмарному сховищі Amazon S3;

– використання REST API для комунікації клієнту з сервером.

1.3) Межі проєкту

Проєкт охоплює проєктування та розробку клієнтської та серверної частин вебзастосунку, інтеграцію із хмарними сервісами для збереження даних. Створення окремого застосунку для мобільних пристроїв не передбачено.

2) Загальний опис

2.1) Сфера застосування

Вебзастосунок надає можливість завантажувати, зберігати та поширювати відеоконтент будь-якого типу. Найбільший фокус припадає на наукову, освітню та розважальну сфери, однак це ПЗ є корисним для усіх завдань, у яких використовують відеоматеріали.

2.2) Характеристики користувача

Глядач: базові навички взаємодії з браузером та сучасними вебтехнологіями.

Автор: уміння записувати відео та користуватись застосунками відеомонтажу.

Редактор: професійна підготовка до використання адмін-інструментів для модерації контенту та коментарів.

2.3) Загальна структура та склад системи

Система створюється за тришаровою архітектурою:

2.3.1) Клієнтський рівень (React)

– маршрутизація;
– сторінка каталогу відео;
– сторінка перегляду відео;
– сторінка пошуку відео;
– панель доступних категорій відео;
– панель адміністратора.

2.3.2) Серверний рівень (.NET Framework)

– REST API;
– обробка завантаженого відеоконтенту;
– комунікація з SQL Server;
– підключення до Amazon S3.

2.3.3) Рівень даних

Microsoft SQL Server – зберігання:

– даних користувачів;
– метаданих відео та посилання на відеофайл;
– статистики переглядів;
– реакцій глядачів (лайки, коментарі).

Amazon S3 – зберігання:

– відеофайлів;
– зображень-прев’ю.

3) Функції системи

3.1) Функція реєстрації та авторизації

3.1.1) Опис функції:

Надає можливість створення акаунту та вхід у систему.

3.1.2) Функціональні вимоги

– реєстрація користувача;

– хешування паролів;

– надання доступу до елементів ПЗ в залежності від ролі користувача (авторизація);

3.2) Функція завантаження відео

3.2.1) Опис функції:

Надає функціонал для завантаження, налаштування та персоналізації відеоконтенту.

3.2.2) Вхідні дані

– відеофайл;

– назва;

– опис;

– категорія;

3.2.3) Вихідні дані

– URL відео, що посилається на файл в Amazon S3;

– запис у базі даних;

3.2.4) Функціональні вимоги

– обробка різних форматів;

– перевірка максимального розміру;

– завантаження файлу в Amazon S3;

– збереження даних у SQL Server;

– створення та збереження прев’ю.

3.3) Функція перегляду відео

3.3.1) Опис функції:

Надає можливість переглядати відео, що завантажені на платформі.

3.3.2) Функціональні вимоги:

– отримання даних з SQL Server;
– комунікація з Amazon S3 для стримінгу відеофайлу;
– фіксація перегляду в базі даних;
– збереження реакції користувача.

3.4) Функція пошуку та фільтрації

3.4.1) Опис функції:

Надає можливість шукати відео за назвою та категорією.

3.4.2) Функціональні вимоги:

– пошук за назвою;
– фільтрація за категорією.

3.5) Адміністративна функціональність

3.5.1) Опис функції:

Надає інструменти для модерації контенту на платформі.

3.5.2) Функціональні вимоги:

– управління користувачами;
– блокування акаунтів;
– видалення відео та коментарів.

4) Функціональні вимоги програмного забезпечення

– API повинно повертати відповіді у форматі JSON;
– кожне відео повинно мати унікальний ідентифікатор;
– система повинна підтримувати пагінацію відповідей серверу;
– усі операції повинні логуватись.

5) Нефункціональні вимоги програмного забезпечення

– ПЗ повинно працювати на платформах Windows, MacOS, Linux;
– ПЗ повинно працювати у сучасних браузерах (Google Chrome, Opera, Edge, Firefox і т.д.);

– час відповіді API: на прості запити – не більше 1 секунди; на складні запити (завантаження відео, пошук/фільтрація відео) – не більше 3 секунди;

– система повинна працювати 24/7 за винятком технічного обслуговування, про яке необхідно попереджати користувачів;

– система повинна мати хороший рівень безпеки для захисту від SQL-ін’єкцій та XSS скриптингу;

– інтерфейс системи має бути однаково зрозумілим як для користувача, так і для редактора.

6) Вимоги до технічного забезпечення

Користувач: сучасний браузер (Chrome, Edge, Firefox), стабільне інтернет-з’єднання.

Сервер: підтримка .NET Framework, наявність Microsoft SQL Server, стабільне інтернет-з’єднання для комунікації з клієнтом та доступом до Amazon S3.

7) Вимоги до програмного забезпечення

7.1) Архітектура програмної системи

Система складається клієнтської частини (фронтенд), серверної частини (бекенд) та бази даних.

7.2) Програмне забезпечення ведення інформаційної бази

Всі операції виконуються через Microsoft SQL Server, з використанням інтерфейсу розроблюваного програмного забезпечення.

7.3) Мова і технологія розробки програмного забезпечення

Фронтенд: використання фреймворку React, використання мови JSX.

Бекенд: використання платформи .NET, використання мови C#.

8) Вимоги до зовнішніх інтерфейсів

8.1) Інтерфейс користувача

Інтерфейс має адаптивний дизайн, підтримує роздільні здатності від 320x480 до 1920x1080.

8.2) Апаратний інтерфейс

В якості апаратного інтерфейсу має виступати будь який пристрій, який підтримує вихід в мережу Інтернет.

8.3) Програмний інтерфейс

REST API для комунікації між клієнтом та сервером.

8.4) Комунікаційний інтерфейс

HTTPS для всіх запитів.

9) Властивості програмного забезпечення

9.1) Доступність

Програмне забезпечення повинно забезпечувати безперервний доступ до функціоналу вебзастосунку 24 години на добу, 7 днів на тиждень, за винятком періодів планового технічного обслуговування або аварійних ситуацій на стороні серверної інфраструктури чи хмарного сховища. Інтерфейс вебзастосунку має коректно працювати на різних типах пристроїв (персональні комп’ютери, ноутбуки, планшети, смартфони) без необхідності встановлення додаткового програмного забезпечення.

9.2) Супроводжуваність

Програмне забезпечення повинно проєктуватися таким чином, щоб забезпечити зручність подальшого супроводу, оновлення та розширення функціоналу. Супроводжуваність забезпечується за рахунок:

– використання модульної архітектури з чітким розподілом відповідальностей;

– логічного розділення frontend та backend частин системи;

– документованого REST API з описом endpoints і форматів запитів/відповідей (наприклад із використанням Swagger);

– оптимізація структури БД і індексація полів.

Код системи повинен бути структурованим, читабельним та відповідати загальноприйнятим стандартам розробки для платформи .NET та застосунків на React.

9.3) Переносимість

Програмне забезпечення повинно мати можливість розгортання на іншому серверному середовищі без внесення суттєвих змін у програмний код.

Переносимість забезпечується за рахунок:

– використання стандартизованого стеку технологій (.NET Framework, Microsoft SQL Server, React);

– зберігання конфігураційних параметрів (рядки підключення до БД, доступ до Amazon S3, ключі JWT) у файлах конфігурації або змінних середовища;

– можливості міграції бази даних на інший сервер Microsoft SQL Server.

Система повинна підтримувати можливість розгортання як у локальному серверному середовищі, так і у хмарній інфраструктурі.

9.4) Продуктивність

Програмне забезпечення повинно забезпечувати стабільну та швидку роботу при нормальному та підвищеному навантаженні.

Основні вимоги до продуктивності:

– час відповіді API на стандартні запити користувача не повинен перевищувати 1 секунди, на складні – 3 секунди;

– система повинна підтримувати одночасну роботу щонайменше 500 активних користувачів без суттєвого погіршення продуктивності;

– реалізація пагінації при відображенні списків відео та коментарів;

– оптимізація SQL-запитів шляхом створення індексів для полів пошуку;

– реалізація асинхронності для операцій читання/запису даних;

– впровадження механізмів кешування для зменшення навантаження на базу даних.

9.5) Надійність

Система повинна функціонувати стабільно протягом тривалого часу без критичних збоїв.

Надійність забезпечується:

– обробки помилок на серверному рівні;

– використання транзакцій при виконанні операцій запису до бази даних;

– резервного копіювання бази даних;

– логування помилок та інших подій.

У разі виникнення помилки система повинна повертати коректне повідомлення користувачу без розкриття внутрішньої структури або конфігурації сервера.

9.6) Безпека

Програмне забезпечення повинно забезпечувати належний рівень інформаційної безпеки як на рівні клієнтської, так і серверної частини.

Основні вимоги до безпеки:

– розмежування прав доступу між редакторами, авторами та глядачами;

– захист від SQL-ін’єкцій шляхом використання параметризованих запитів або ORM;

– захист від XSS та CSRF-атак;

– належне логування дій користувачів.

10) Інші вимоги

– програмне забезпечення повинно відповідати сучасним стандартам та рекомендаціям щодо розробки вебзастосунків і REST-сервісів;

– інтерфейс користувача повинен бути адаптивним та коректно відображатися на екранах з різною роздільною здатністю;

– система повинна підтримувати можливість подальшого функціонального розширення, зокрема додавання нових типів реакцій, розширеної аналітики та нових категорій;

– архітектура системи повинна, за необхідністю, дозволяти інтеграцію з зовнішніми сервісами.

Висновки до розділу 2

У другому розділі проведено аналіз сучасного стану інструментарію, що може бути використаний для реалізації вебсистеми відеохостингу. У ході аналізу розглянуто та порівняно технології, які використовуються для розробки фронтенд та бекенд частин, а також порівняно доступні РСКБД. Результати, отримані після аналізу, використано для обґрунтування вибраного технологічного стеку як оптимального для виконання поставлених завдань.

Також, для вебзастосунку відеохостингу сформовано специфікації вимог. Ці вимоги допоможуть оптимально виконати проєктування архітектури програмного забезпечення та визначити можливі напрямки оптимізації та розширення.

3 МОДЕЛЮВАННЯ СТРУКТУРИ ТА ФУНКЦІОНАЛУ СИСТЕМИ

3.1 Створення UML-діаграм

Розробка застосунків часто включає в себе реалізацію великої кількості функціоналу та компонентів – без планування та попередньої структуризації система може мати крихку та немасштабовану архітектуру, адже будь-яке доповнення чи зміна призведе до необхідності переробки безлічі інших елементів. Для уникнути цієї проблеми використовується UML, що надає стандартизовані методи моделювання та візуалізації комп’ютерних систем.

Беручи до уваги вибраний тип архітектури вебзастосунку відеохостингу та кількість необхідних для його роботи компонентів, створення UML-діаграм є необхідним етапом для забезпечення масштабованості та стійкості системи.

Діаграма варіантів використання (usecase)

Завданням usecase діаграми є візуалізація взаємодій користувачів із функціоналом застосунку. Цей тип діаграм відповідає на запитання «Хто? - Що може робити?», а отже ідеально підходить для моделювання рольової системи допуску. Для вебзастосунку відеохостингу передбачено три основні типи користувачів: Глядач, Автор та Редактор (рис. 3.1). Кожна роль надає доступ до окремої частини функціоналу, однак Глядач виступає як загальний користувач – Автор та Редактор можуть виконувати усі дії, що доступні Глядачу.

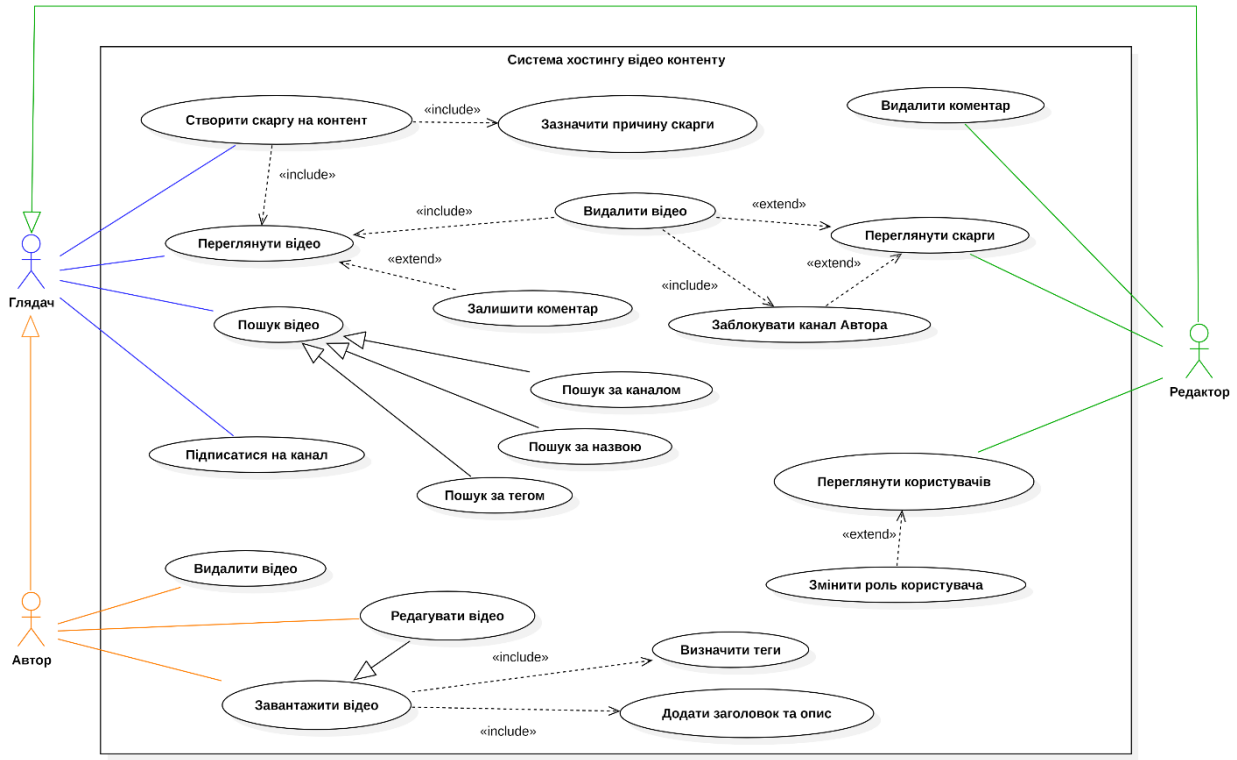

Рисунок 3.1 – Usecase діаграма

Із цієї діаграми можна визначити, які можливості мають користувачі із різними ролями:

– Глядач: перегляд відео контенту, взаємодія з Автором за допомогою коментарів, взаємодія з Редактором за допомогою створення скарг на недопустимий контент;

– Автор: наповнення системи новим відео контентом, редагування існуючого контенту для підтримки його якості, аналіз трендів за допомогою відслідковування кількості підписників та переглядів на відео;

– Редактор: контроль за якістю контенту на платформі, видалення недоречних відео та коментарів, блокування Авторів-порушників, контроль за розподілом ролей.

Діаграма класів

Діаграма класів це один із типів статичних структурних діаграм, що описує структуру системи за допомогою її класів. Вона візуально відображає атрибути та методи класів, а також відносини між їх об’єктами. Створення такої діаграми під час моделювання системи допомагає уникнути

неконтрольованого збільшення кількості класів під час розробки та створення «божественних об’єктів» шляхом попереднього розподілу завдань між заздалегідь визначеними класами.

У контексті проєкту вебзастосунку відеохостингу діаграма класів відображає структуру бекенду, який можна розділити на три основні частини: контекст БД, моделі та контролери.

Контекст БД є центральною частиною діаграми – він надає можливість отримувати, змінювати та видаляти інформацію з бази даних. Вбудовані можливості створення асинхронних запитів та перевірка на їх атомарність забезпечують легку та безпечну роботу з чутливими даними.

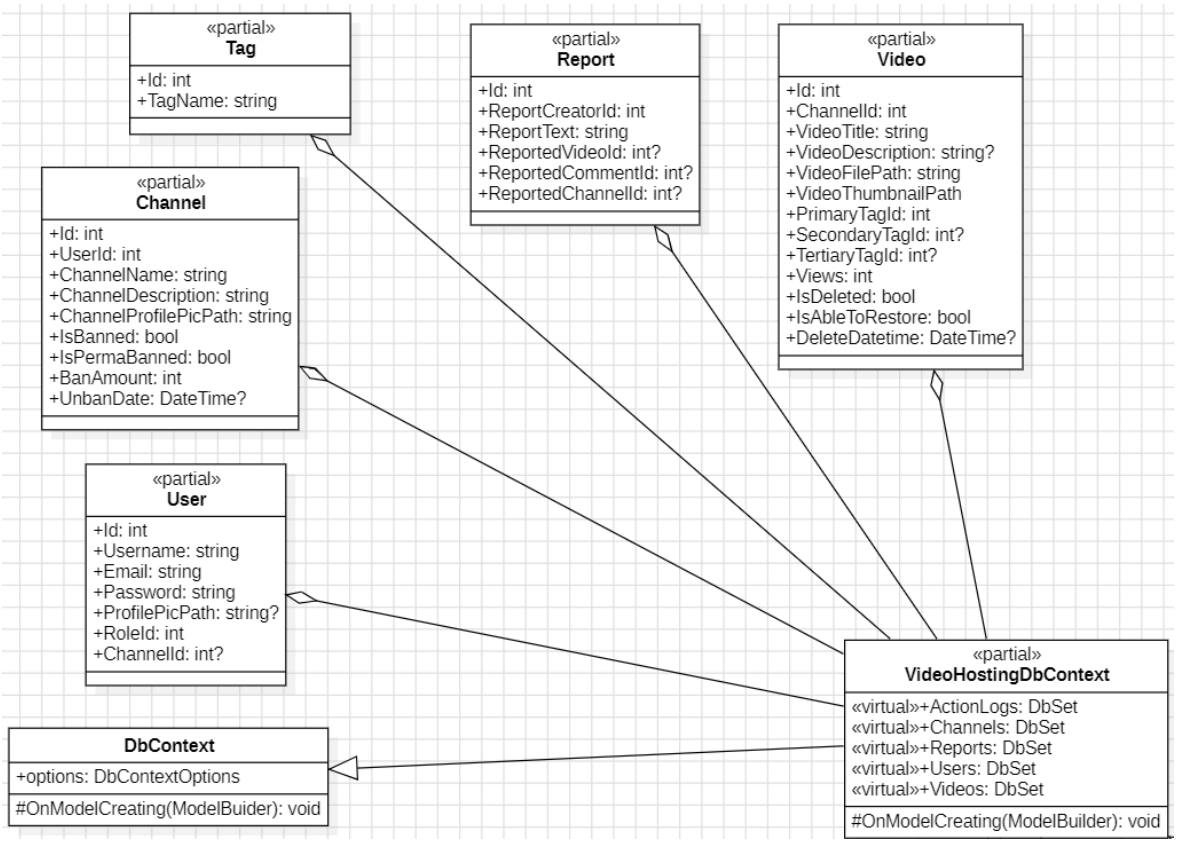

Рисунок 3.2 – Діаграма класів (моделі)

Моделі – це спеціальні класи, які використовуються для реалізації ORM (рис. 3.2). Їх об’єкти зберігають дані з відповідних колонок таблиці БД, а також тимчасово утримують всі зміни, що застосовуються до цих даних. Основними моделями у проєкті виступають User, Video, Channel, Report та Tag, адже вони використовуються у більшості запитів.

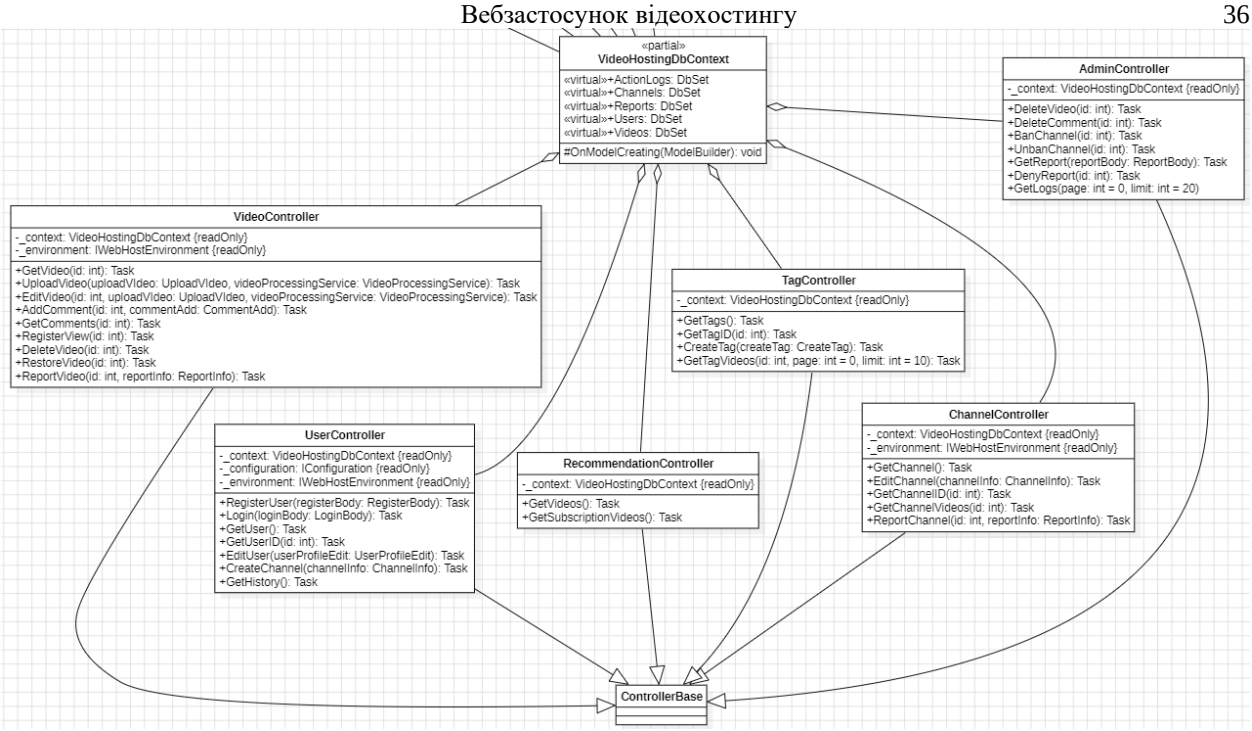

Рисунок 3.3 – Діаграма класів (контролери)

Контролери – це класи, що відповідають за комунікацію між фронтендом та бекендом, надаючи endpoints для двосторонньої передачі даних (рис. 3.3). Окрім простої передачі даних, контролери відповідають за аутентифікацію та авторизацію користувача, а також обробку і верифікацію інформації. Кожен контролер групує доступні методи за принципами «Хто робить запит?» або «З чим працює користувач?» – такий підхід дозволяє контролювати та логічно розподіляти обов’язки між компонентами системи.

Діаграма діяльності (activity)

Діаграми діяльності це один із типів поведінкових діаграм, які відображають послідовність дій, що виконується в процесі реалізації певного варіанту використання або функціонування системи в цілому. Такі діаграми дозволяють детально продумати цикл «запит-реакція-відповідь», аналізувати потік даних та визначити критичні точки системи, які необхідно ретельно тестувати чи переробити.

У рамках проєкту вебзастосунку відеохостингу такі діаграми можливо використати для моделювання складного функціоналу, як, на приклад, завантаження відео (рис. 3.4) та обробка скарги (рис. 3.5). Візуалізація роботи

користувача із системою у майбутньому допоможе із моделюванням інтерфейсу та визначенням місць детальної перевірки цілісності даних.

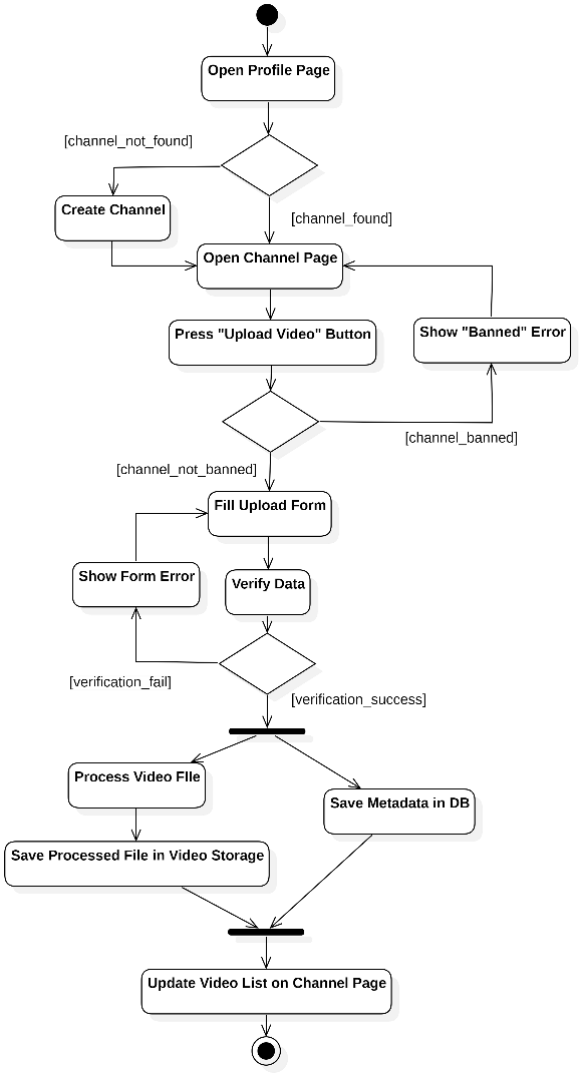

Рисунок 3.4 – Діаграма діяльності «Завантаження відео»

Перша діаграма відображає процес завантаження нового відео. Із важливих елементів можна виділити розгалуження на початку діаграми, а саме різні варіанти розвитку процесу якщо у користувача немає активного акаунту чи його акаунт був заблокований через попередні порушення. Інша частина діаграми, що заслуговує на увагу, це паралельне виконання процесів після успішної верифікації завантаження нового відео – одночасно метадані записуються у БД та відеофайл проходить обробку та зберігається в окремому відео сховищі. Із цієї діаграми можна визначити дві точки фокусу: верифікація даних з форми та паралельна обробка відеоданих. Під час перевірки форми,

особливу увагу необхідно надати файлу, що передається – він повинен мати одне із допустимих відео-розширень, а також він не повинен перевищувати допустимий розмір. Під час паралельної обробки відеоданих алгоритм повинен дотримуватися принципу атомарності – якщо на якомусь етапі сталася помилка, то уся операція має бути скасована.

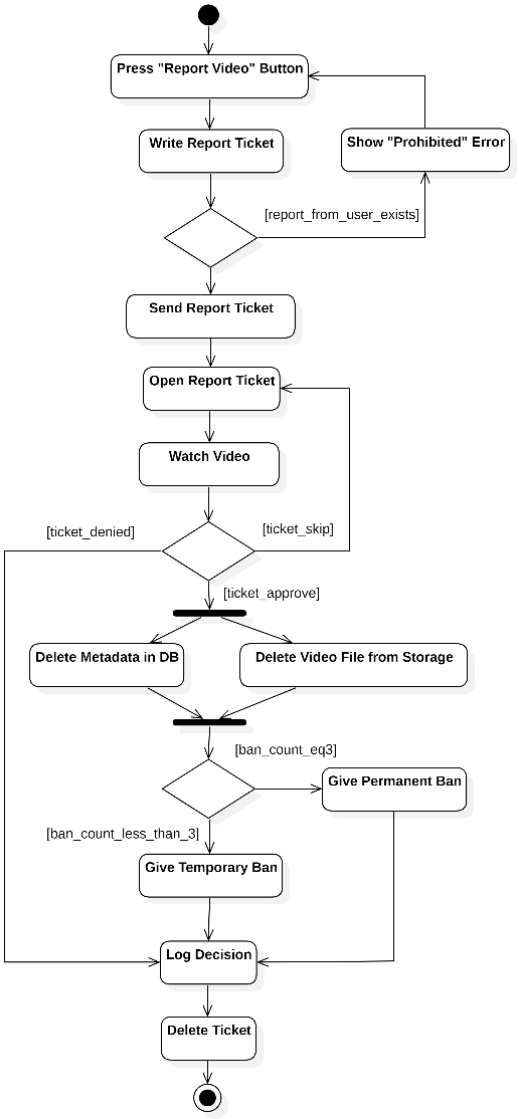

Рисунок 3.5 – Діаграма діяльності «Обробка скарги»

Друга діаграма описує процес обробки скарги на відео Редактором. Діаграма має багато розгалужень, бо наступні рішення часто залежать від попередніх дій користувача. Із важливих моментів можна визначити розгалуження після перегляду відео, що показує на рішення Редактора щодо скарги: якщо Редактор не знайшов нічого, що порушує правила платформи, то

скарга видаляється без виконання додаткових дій; якщо Редактор не впевнений, що може дати правильний вердикт, то він може пропустити скаргу. Це дає можливість переглянути скаргу іншим Редакторам, або повернутися до неї самому після роздумів. Остання дія це підтвердження скарги – у такому випадку система починає паралельний процес видалення відео, що схожий на процес завантаження, а також видає один із типів блокування Автору: якщо Автор має 3 попередні підтверджені скарги, то його канал повністю блокується; якщо менше ніж три, то видається блокування на тиждень.

Діаграма розгортання

Діаграма розгортання це особливий тип структурної діаграми, що відображає конфігурацію фізичних елементів системи під час її розгортання. Вона часто використовується для моделювання «фізичного розташування» програмного забезпечення на серверах. Діаграми розгортання допомагають аналізувати стан фізичного обладнання, визначати шляхи інтегрування нової системи у існуючу інфраструктуру та підібрати правильні технології/протоколи спілкування між вузлами.

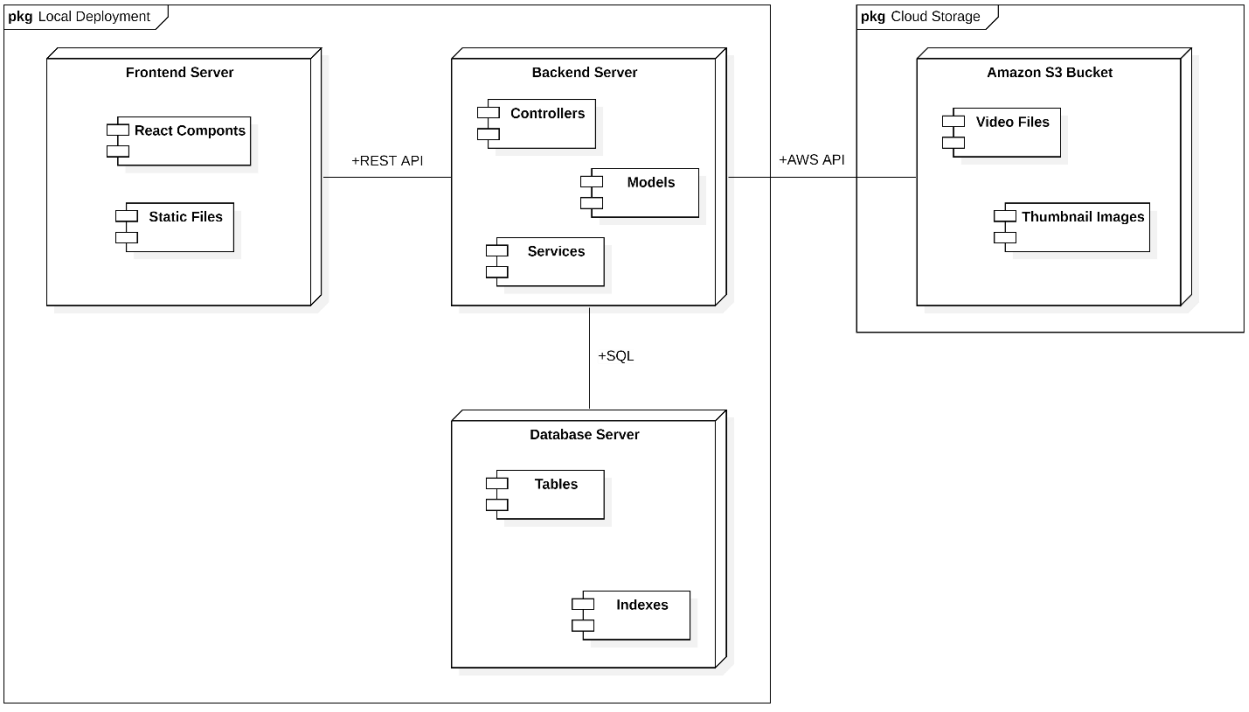

Рисунок 3.6 – Діаграма розгортання

Діаграма розгортання вебзастосунку відеохостингу має в собі дві частини: локальне розгортання та хмарне сховище (рис. 3.6). В своїй основі система має тришарову архітектуру, тобто вона поділяється на фронтенд (presentation layer), бекенд (business layer) та базу даних (data layer), де кожен шар має свій сервер. Фронтенд зберігає усі компоненти React та статичні файли, керує інтерфейсом користувача та оброблює запити до бекенду. Бекенд налаштовує доступні endpoints, оброблює та перевіряє дані, отримані з запитів, та керує транзитом інформації між користувачем і БД. Шар бази даних охоплює дві елементи системи: локальну базу даних та хмарне сховище файлів. Локальна БД відповідає за збереження текстових даних та метаданих відеофайлів, у той час як хмарне сховище зберігає самі відеофайли. Така структура зменшує навантаження на диск та пам’ять локального серверу, адже йому не треба зберігати великий обсяг даних та стриміти його користувачеві напряму.

3.2 Вибір технологій та додаткових компонентів розробки

На основі спроєктованої архітектури вебзастосунку відеохостингу необхідно вибрати технології розробки, що дозволять реалізувати увесь необхідний функціонал без зайвого перевантаження системи. Безпека, цілісність даних та можливості подальшого розширення функціоналу також є важливими факторами під час формулювання технологічного стеку.

У процесі вибору технологій для розробки фронтенду основними критеріями були простора використання та широка підтримка додаткових компонентів. Враховуючи ці критерії вибір був зроблений на користь бібліотеки React – вона активно підтримується і оновлюється, а користувачі постійно створюють безліч додаткових пакетів, що розширюють її функціонал. Мова JSX, що використовується під час розробки у React, є розширенням звичайного JavaScript, а отже не потребує багато сил і часу для вивчення.

До цієї бібліотеки було додано низку сторонніх пакетів, що надають новий функціонал або спрощують розробку інтерфейсу. Нові можливості додають пакети react-cookie (керування браузерними кукі), react-router-dom (маршрутизація) та Yup (валідація форм). Пакети, що допомагають у розробці UI це Formik (нові компоненти для створення форм), reactjs-popup (компонент для легкого створення popup), react-tooltip (компонент для створення і налаштування підказок) та react-icons (пакет з великою кількістю безкоштовних svg іконок).

Під час вибору технологічного стеку для реалізації бекенду особливу увагу було надано критеріям безпеки, швидкості обробки даних та можливості легкого доповнення функціоналу. Після огляду доступних технологій, вибір впав на .NET Framework – фреймворк для розробки вебзастосунків мовою C#. Мова програмування C# надає широкі можливості для асинхронної роботи із даними, що дозволяє оброблювати декілька запитів одночасно без значного впливу на швидкість виконання. Сам фреймворк має достатньо функціоналу щоб задовольнити дві інші критерії: вбудована підтримка створення та обробки JWT разом із можливостями автоматичної авторизації/аутентифікації дозволяє захистити будь-які endpoints від несанкціонованого доступу, а використання атрибутів класів та методів для створення самих endpoints робить створення API легким і швидким завданням.

Хоч наявний стек і надає широкий функціонал, деякі необхідні елементи необхідно брати з офіційних або сторонніх бібліотек. EF Core це офіційна Майкрософт бібліотека, що додає підтримку ORM для .NET застосунків. З її допомогою можна автоматично побудувати необхідні ORM моделі, а також легко комунікувати з БД використовуючи LINQ-запити. Для забезпечення безпеки чутливих даних користувачів, такі як паролі, використано бібліотеку bcrypt.net, що є портом jBCrypt для .NET фреймворку. Ця бібліотека надає функціонал хешування даних із додаванням солі та верифікацію паролів за створеним хешем – паролі у текстовому вигляді ніде не зберігаються, а отже дані користувачів у безпеці. Останні додаткові компоненти це інструмент

FFmpeg та бібліотека Xabe.FFmpeg. FFmpeg це інструмент, що надає можливість працювати з файлами мультимедіа, а саме конвертувати в інші формати, програвати, виокремлювати потоки та інше. Xabe.FFmpeg це бібліотека-обгортка FFmpeg, зроблена для .NET. Вона надає спеціальні вбудовані функції для роботи з цим інструментом, що значно спрощує розробку застосунків, що працюють з мультимедіа файлами.

Як РСКБД, що буде підходити до вже існуючого стеку, було вибрано Microsoft SQL Server – підтримка ACID-транзакцій, шифрування даних та швидкість читання/запису інформації дозволяє безпечно та ефективно використовувати цю базу даних для роботи з великою кількістю даних.

Одним із основних викликів розробки застосунку відеохостингу є логістика збереження та стримінгу контенту – деякі відеофайли, навіть після обробки та стиснення, можуть займати сотні мегабайтів дискового простору, а їх перегляд користувачами навантажує сервер постійними запитами, що вичерпують його оперативну пам’ять. Для вирішення цих проблем вебзастосунок було підключено до хмарного сховища об’єктів Amazon S3, де зберігаються усі відеофайли. Для комунікації із цим сервісом на бекенді встановлено офіційну бібліотеку Amazon.S3, що має функціонал керування завантаженими файлами та доступом до них.

Для контроля над якістю розробленої системи та її компонентів проведено різні типи тестувань, що охоплюють більшу частину функціоналу та перевіряють можливі сценарії роботи користувачів. Integration тести виконано за допомогою xUnit, для E2E тестування використано Playwright, а тестування навантаження реалізовано у Apache JMeter. Такий стек дозволяє ретельно перевірити, чи готовий вебзастосунок до розгортання у виробничому середовищі.

3.3 Моделювання інтерфейсу вебзастосунку відеохостингу

Моделювання інтерфейсу є не менш важливим етапом проєктування ніж моделювання архітектури застосунку – створення мокапів дозволяє

перевірити, наскільки логічно та інтуїтивно будуть розташовуватись різні елементи на сторінці, а також чи не буде сторінка переповнена зайвою інформацією чи функціоналом. Для моделювання інтерфейсу системи відеохостингу створено п’ять мокапів, що слугують прототипами до найбільш важливих сторінок та елементів вебзастосунку.

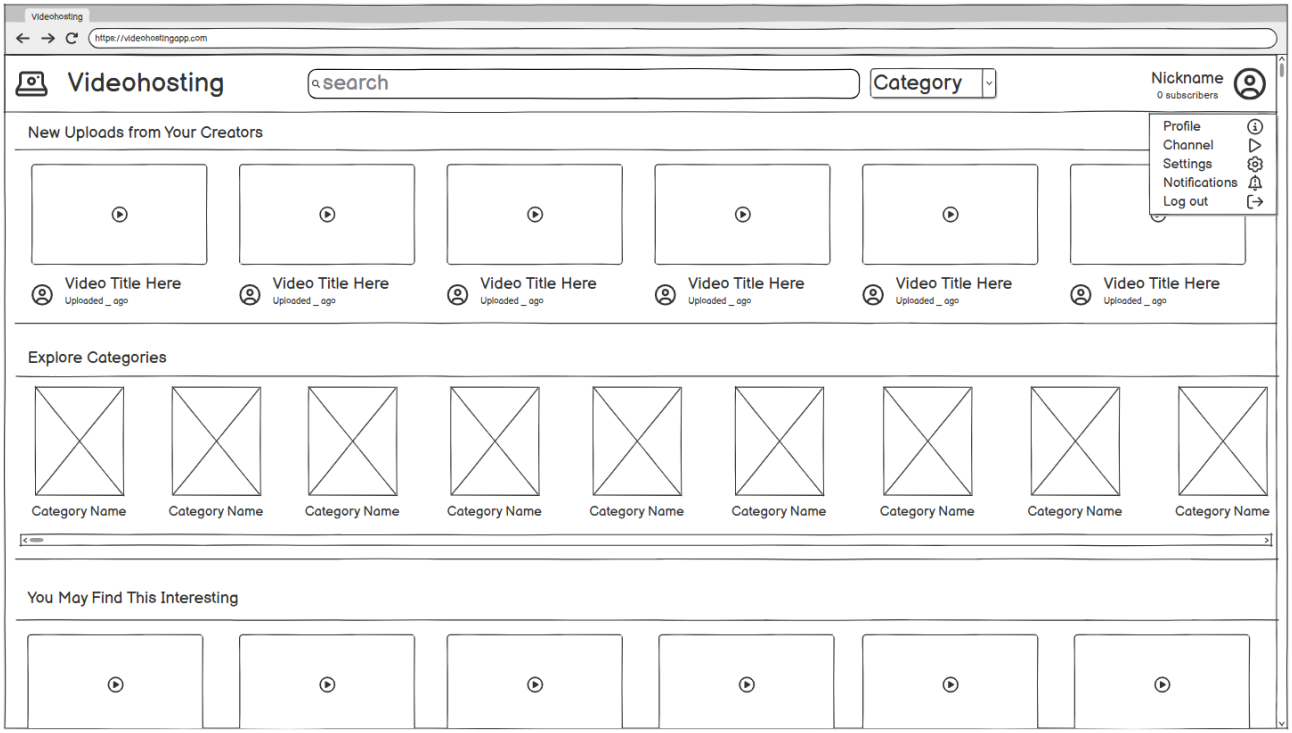

Рисунок 3.7 – Мокап головної сторінки

Перший мокап відображає інтерфейс головної сторінки вебзастосунку (рис. 3.7). Сторінка має шапку, де знаходиться назва та іконка сервісу, поле пошуку та інформація про активний акаунт користувача.

Основне тіло сторінки наповнене списком доступних для перегляду відео та категоріями. Перша секція відображає нові відео від авторів, на які підписан користувач. Друга секція відображає перелік доступних категорій, які користувач може вибрати для фільтрації відео за категорією. Остання секція відображає перелік рекомендованих відео для користувача.

Компонент відео складається з титульного зображення, ціль якого привернути увагу користувача, та короткої інформації про канал та відео, що розташовані під ним.

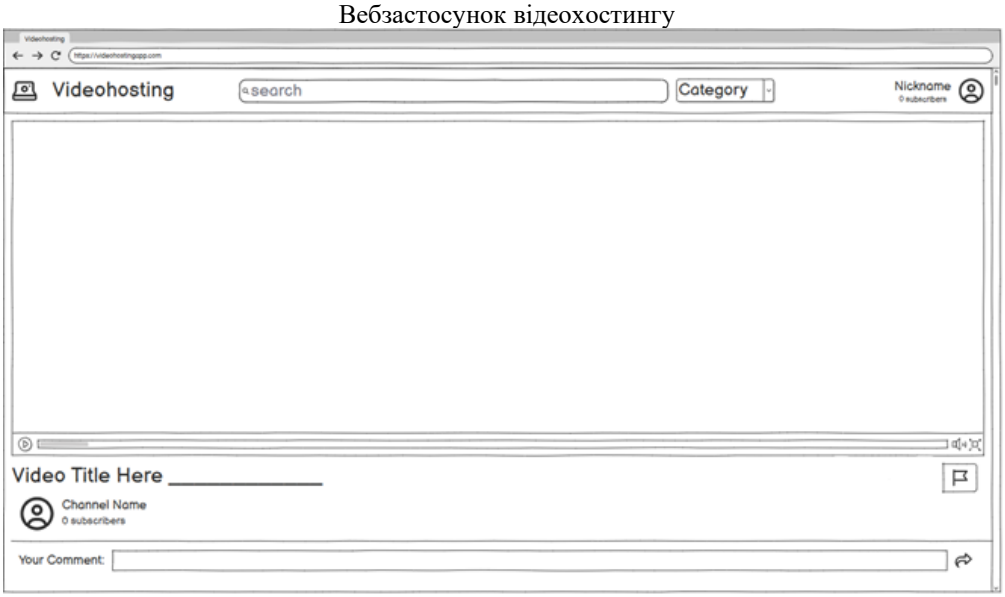

Рисунок 3.8 – Мокап сторінки перегляду відео

Другий мокап відображає інтерфейс сторінки перегляду відео (рис. 3.8). Шапка залишається незмінною з попередньої сторінки. Основна секція сторінки це відеоплеєр, який завантажує та програє відеофайл. Під плеєром глядач може переглянути назву та автора відео. Окрім того, у глядача є можливість залишити коментар під відео, або поскаржитись на відео за недопустимий контент.

Компонент, що відображає інформацію про автора відео, працює як посилання на канал – таким чином глядач може натиснути на нього і переглянути більше детальну інформацію про канал та опубліковані відео.

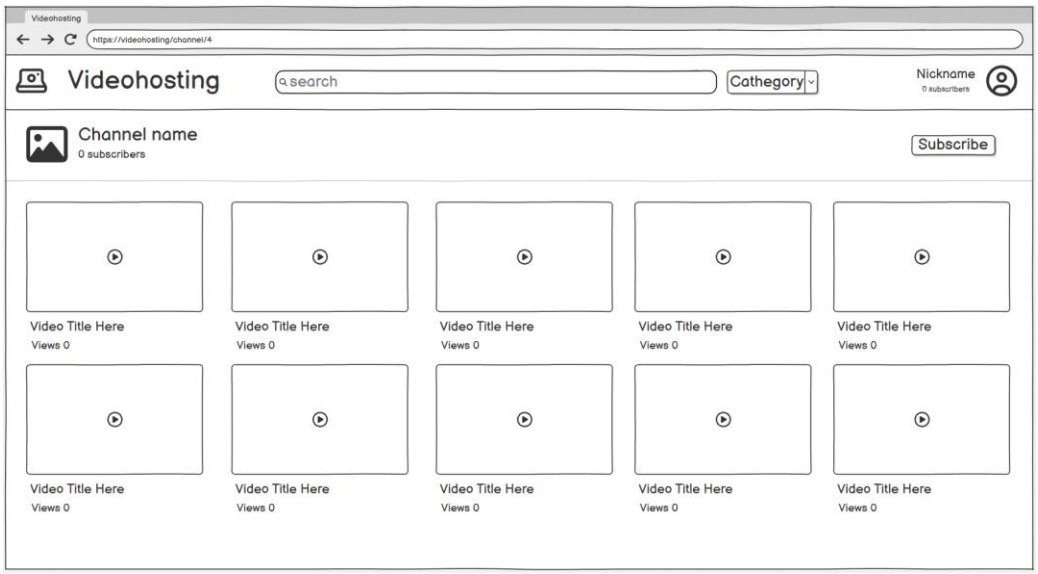

Рисунок 3.9 – Мокап сторінки каналу

Третій мокап відображає інтерфейс сторінки перегляду каналу (рис. 3.9). Тіло сторінки розділено на дві частини: секція інформації та секція опублікованих відео. У секції інформації відображається назва, зображення та кількість підписників каналу, також доступна кнопка підписки на цей канал. Секція опублікованих відео показує усі відео, що були завантажені Автором каналу. Компонент відео з першого мокапу був змінений, а саме прибрано інформацію про канал – це було замінено показником кількості переглядів конкретного відео.

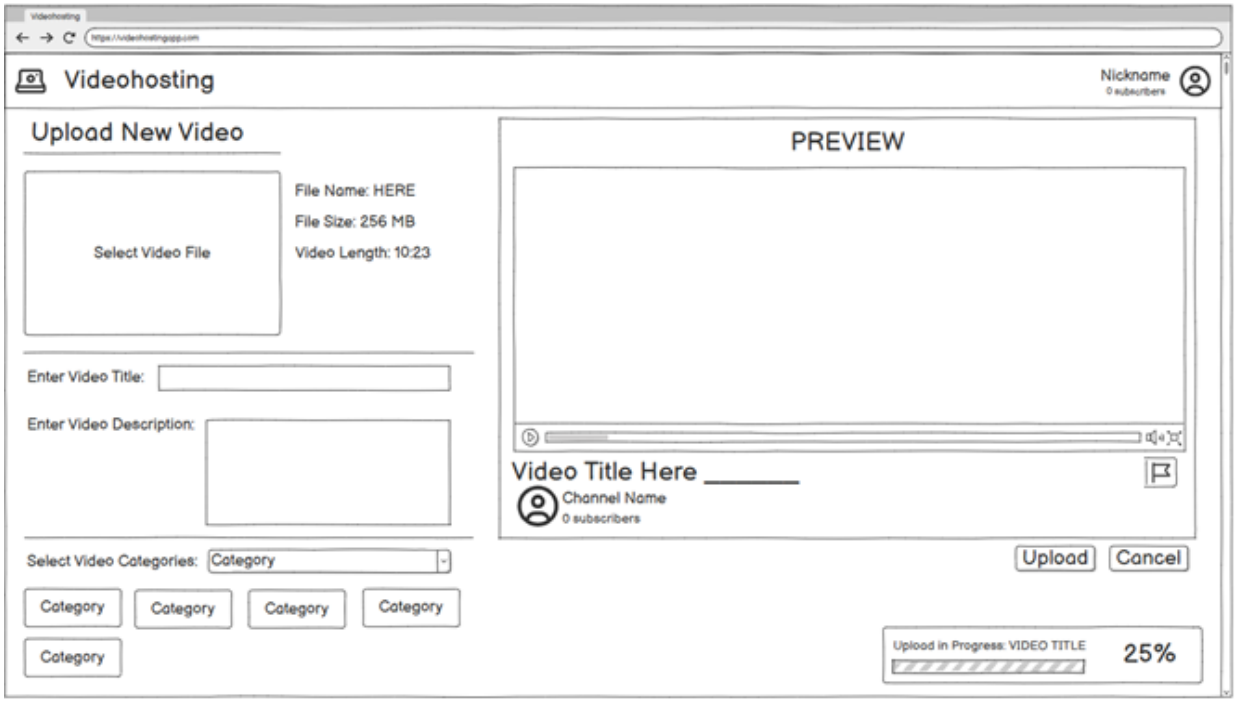

Рисунок 3.10 – Мокап сторінки публікації відео

Четвертий мокап показує інтерфейс сторінки створення нового відео автором (рис. 3.10). Сторінка розділяється на 2 секції: форма завантаження та прев’ю. У формі завантаження автор має можливість вибрати відео файл із локального сховища, вписати назву та опис відео, а також вибрати категорії, до яких це відео відноситься.

Усі дані, які вводить автор, відображаються у секції прев’ю, що надає можливість побачити попередній вигляд сторінки перегляду. Під елементом прев’ю є кнопка «Upload», яка починає процес завантаження і публікації відео, та кнопка «Cancel» яка очищає усі поля форми. Якщо автор має якесь відео у

процесі завантаження, він побачить спливаюче вікно із інформацією про назву відео та прогрес його публікації.

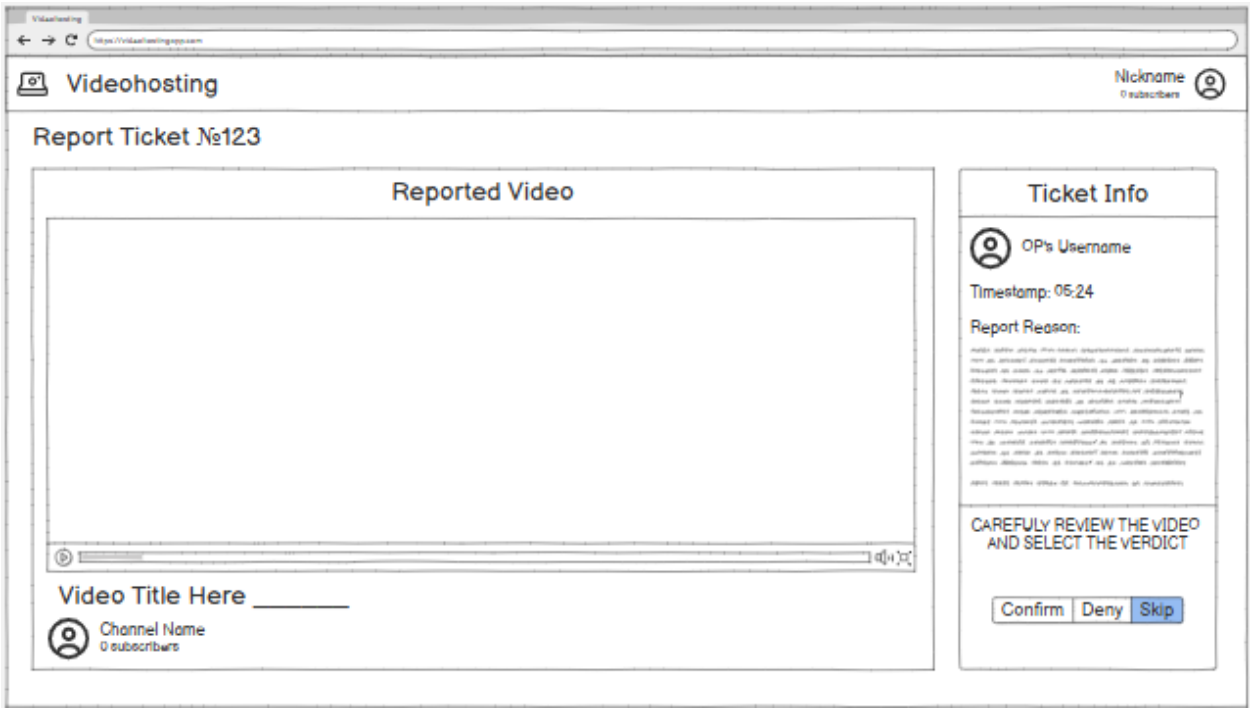

Рисунок 3.11 – Мокап сторінки перегляду скарги

Останній мокап відображає сторінку перегляду скарги редактором. Ця сторінка також має 2 секції: перегляд відео, на яке поскаржилися, та інформація щодо самої скарги. Перша секція схожа на прев'ю при публікації відео – це дає редактору можливість швидко переглянути об'єкт скарги і знайти можливі порушення без необхідності покидати сторінку скарги. Друга секція зберігає інформацію про автора скарги, приблизну позначку часу де є порушення та опис самого порушення. Після цієї інформації редактор має можливість вибрати вердикт: підтвердити порушення, спростувати порушення чи пропустити цю скаргу.

Висновки до розділу 3

У третьому розділі проведено проєктування архітектури, функціоналу та інтерфейсу вебзастосунку відеохостингу. Для моделювання архітектури системи створено набір UML-діаграм, що допомагають спланувати її розробку, тестування та майбутнє розгортання. На основі створених діаграм

та попереднього аналізу стану сучасних технологій у розділі 2 визначено повний технічний стек для розробки вебзастосунку відеохостингу.

Для реалізації фронтенду вибрано бібліотеку React, що супроводжується низкою додаткових пакетів та компонентів: react-cookie, react-router-dom, Formik, react-icons, reactjs-popup, react-tooltip та Yup. Цей набір інструментарію дозволяє легко створювати, змінювати та розширювати компоненти та сторінки, інтегруючи новий функціонал по ходу його розробки.

Для реалізації бекенду вибрано фреймворк .NET, який дозволяє легко створювати endpoints та має широкий вбудований функціонал для забезпечення їх безпеки від несанкціонованого доступу. До фреймворку додано бібліотеки EF Core, що відповідає за комунікацію з базою даних за допомогою ORM, bcrypt.net для хешування паролів користувачів та Xabe.FFmpeg для роботи з інструментом конвертації мультимедіа FFmpeg. Такий технологічний стек відкриває можливість до створення швидкого, безпечного та простого бекенду.

Microsoft SQL Server вибрано як підходящу РСКБД для даного проєкту, базуючись на її швидкості читання/запису великої кількості даних та підтримки ефективного шифрування інформації.

Окрім визначення архітектури та підбору технічного стеку проведено моделювання інтерфейсу – створено декілька мокапів основних сторінок застосунку, що виступають як перші прототипи дизайну. На етапі безпосередньої розробки ці мокапи будуть використані для реалізації інтуїтивного та функціонального кінцевого інтерфейсу.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ

4.1 Розробка вебзастосунку відеохостингу

Фронтенд

Фронтенд вебзастосунку складається з 21 сторінки, із них 8 є основними: головна, перегляд відео, перегляд каналу, перегляд профілю, завантаження відео, перегляд користувачів, перегляд скарг та перегляд логів.

Майже кожна сторінка представляється окремим компонентом, до якого прив’язаний свій шлях (рис. 4.1). Такий підхід робить сторінки незалежними один від одного, а отже зміни чи доповнення до одного компонента не будуть мати непередбачувані ефекти на інші.

```
function App() {
  return (
    <Routes>
      <Route path='/' element={<Layout />}>
        <Route index element={<Home />} />
        <Route path='/login' element={<Login />} />
        <Route path='/register' element={<Register />} />
        <Route path='/register_success' element={<RegisterSuccess />} />
        <Route path='/tag/:id' element={<Tag />} />
        <Route path='/video/:id' element={<Video />} />
        <Route path="/channel/:id" element={<ChannelPublic />} />
        <Route path="/search" element={<SearchResults />} />

        <Route element={<ProtectedRoute />}>
          <Route path='/profile' element={<Profile />} />
          <Route path='/profile/edit' element={<ProfileEdit />} />
          <Route path='/history' element={<WatchHistory />} />
          <Route path="/channel/create" element={<CreateChannel />} />
        </Route>

        <Route element={<ProtectedRoute allowedRole={"creator"} />}>
          <Route path="/channel" element={<ChannelOwner />} />
          <Route path="/channel/edit" element={<EditChannel />} />
          <Route path="/channel/deleted" element={<DeletedVideos />} />
          <Route path='/video/upload' element={<UploadVideo />} />
          <Route path='/video/edit/:id' element={<EditVideo />} />
        </Route>

        <Route element={<ProtectedRoute allowedRole={"admin"} />}>
          <Route path="/admin" element={<AdminUsers />} />
          <Route path="/admin/user/:id" element={<AdminUser />} />
          <Route path="/admin/reports" element={<AdminReports />} />
          <Route path="/admin/logs" element={<AdminLogs />} />
        </Route>
      </Route>
    </Routes>
  )
}
```

Рисунок 4.1 – Маршрутизація застосунку

Деякі маршрути вебзастосунку захищені від користувачів, у яких не повинно бути доступу до їх переглядання (рис. 4.2). Головна, перегляд відео

та каналу є публічними сторінками, а отже навіть неавторизований користувач може їх переглядати. Сторінки профілю, історії перегляду та створення каналу стають доступними для користувачів, що авторизовані в системі. Після створення каналу користувач автоматично отримує роль Автора, після чого йому доступні сторінки перегляду та зміни свого профілю, а також сторінки роботи з відео. Перегляд усіх зареєстрованих акаунтів, списку скарг та логів системи доступно тільки для користувачів, що мають роль Редактора. Стати Редактором можна тільки якщо користувач не має ролі Автора, та якщо інший Редактор вручну поміняє йому роль. Ця структура сприяє розподілу відповідальностей між користувачами, надаючи їм доступ тільки до необхідного інструментарію та інформації.

```
import { Navigate, Outlet } from "react-router-dom";
import { useAuth } from "../Context/AuthContext";

export default function ProtectedRoute({
    allowedRole = "",
    requireAuth = true
}) {
    const { isAuthenticated, role, loading } = useAuth();

    if (loading) {
        return <div>Loading...</div>;
    }

    if (requireAuth && !isAuthenticated) {
        return <Navigate to="/login" replace />;
    }

    if (
        allowedRole !== "" &&
        allowedRole !== role
    ) {
        return <Navigate to="/" replace />;
    }

    return <Outlet />;
}
```

Рисунок 4.2 – Захист маршруту

Бекенд

Бекенд системи можна розділити на 3 складові: контролери, моделі та сервіси. Контролери відповідають за створення та управління endpoints, обробку даних та комунікацію з БД. Моделі являють собою шаблони

сутностей у базі даних, адже вони використовуються для створення запитів через ORM (табл. 4.1). Сервіси являють собою окремі класи, що додають новий функціонал або керують автоматизованими процесами.

Таблиця 4.1 – Основні моделі та їх типи

Модель	Типи
User	int Id, string Username, string Email, string Password, string? ProfilePicPath, int RoleId, int? ChannelId
Video	int Id, int ChannelId, string VideoTitle, string? VideoDescription, string VideoFilePath, string VideoThumbnailPath, int PrimaryTagId, int? SecontadyTagId, int? TertiaryTagId, int Views, bool IsDeleted, bool IsAbleToRestore, DateTime? DeleteDatetime
Channel	int Id, int UserId, string ChannelName, string ChannelDescription, string ChannelProfilePicPath, bool IsBanned, DateTime? UnbanDate, int BanAmount, bool IsPermaBanned
Report	int Id, int ReportCreatorId, string ReportText, int? ReportedVideoId, int? ReportedCommentId, int? ReportedChannelId
Tag	int Id, string TagName
ActionLog	int Id, int UserIdAction, int? UserIdAffected, int? VideoIdAffected, int? CommentIdAffected, string Action, DateTime ActionTime

Вебзастосунок відеохостингу має 7 контролерів, які керують усіма процесами (табл. 4.2). Для отримання будь-яких даних фронтенду необхідно звернутися до одного з endpoints, які надають ці контролери (рис. 4.3).

Таблиця 4.2 – Список доступних контролерів

Назва	Функції	Основні моделі	Основні методи
UserController	Управління користувачами: реєстрація, авторизація, створення каналу, редагування профілю	User, Channel	RegisterUser, Login, GetUser, CreateChannel

Кінець таблиці 4.2

VideoController	Управління відео: завантаження нових відео, редагування/видалення існуючих відео	Video, Channel, Comment	GetVideo, UploadVideo, AddComment, ReportVideo
ChannelController	Управління каналами: редагування каналу, отримання інформації про канал	Channel, Video, Subscription	GetChannel, GetChannelID, GetChannelVideos, ReportChannel
SearchController	Управління пошуком відео та каналів	Video, Channel	SearchVideos, SearchChannels
RecommendationController	Підбір відео для перегляду	Video	GetVideos, GetSubscriptionVideos
TagController	Фільтрування відео за їх головним тегом	Tag, Video	GetTag, GetTagID, GetTagVideos
AdminController	Керування сайтом: обробка скарг, видалення недопустимого контенту, зберігання логів усіх дій	Video, Channel, Report, Logs	DeleteVideo, BanChannel, GetReport, GetLogs

```
[ApiController]
[Route("[controller]")]
1 reference
public class TagController : ControllerBase
{
    11 references
    private readonly VideoHostingDbContext _context;

    0 references
    public TagController(VideoHostingDbContext context)
    {
        _context = context;
    }

    [HttpGet("tags")]
    0 references
    public async Task<IActionResult> GetTags()
    {
        var tags = await _context.Tags.OrderBy(tag => tag.TagName)
        .Select(tag => new
        {
            tag.Id,
            tag.TagName
        })
        .ToListAsync();

        return Ok(new { message = "Tags fetched", tags });
    }
```

Рисунок 4.3 – Приклад контролеру та endpoint

Вебзастосунок відеохостингу реалізовує та використовує чотири сервіси, які доповнюють роботу контролерів або автоматизують деякі операції. Першим сервісом є VideoProcessingService – він виступає як абстракція для роботи з інструментом FFmpeg (рис. 4.4). Задача цього сервісу полягає у тому, що він оброблює наданий користувачем відео файл, щоб конвертувати його у стандартизований формат .mp4 та створити титульне зображення. Ці вихідні файли зберігаються у хмарному відео сховищі, тоді як метадані записуються у базу даних.

```
2 references
public async Task<string> ConvertToMp4(string inputPath)
{
    var videosDir = Path.Combine(_environment.WebRootPath, "videos");

    if (!Directory.Exists(videosDir))
        Directory.CreateDirectory(videosDir);

    var outputFileName = $"{Guid.NewGuid()}.mp4";
    var outputPath = Path.Combine(videosDir, outputFileName);

    var conversion = FFmpeg.Conversions.New().AddParameter($"-i \"{inputPath}\"")
        .AddParameter("-c:v libx264")
        .AddParameter("-preset medium")
        .AddParameter("-crf 26")
        .AddParameter("-c:a aac")
        .AddParameter("-b:a 128k")
        .AddParameter("-movflags +faststart")
        .SetOutput(outputPath);

    await conversion.Start();

    return $"/videos/{outputFileName}";
}
```

Рисунок 4.4 – Метод конвертування відео

Наступні два сервіси мають схожу логіку виконання, адже вони обидва є сервісами автоматизації процесів. UnbanService та VideoFullDeleteService викликають свої функції кожен день у той самий час, як і час запуску серверу. UnbanService автоматично розблоковує канали, якщо зазначений день розблокування співпадає із сьогоднішнім днем. VideoFullDeleteService виконує повну чистку бази даних та відео сховища від відео, що були видалені місяць тому – цінність цього сервісу полягає у тому, що коли редактор чи автор видаляє відео, то виконується «soft delete», а отже самі файли і записи

залишаються на диску і займають пам’ять. Ця система зроблена для того, щоб автор міг відновити відео, якщо передумав його видаляти.

Останнім сервісом, що використовує бекенд, є S3StorageService – він виступає абстракцією для комунікації з хмарним сховищем об’єктів Amazon S3 (рис. 4.5). Цей сервіс надає методи для завантаження та видалення файлів, самостійно будуючи необхідні запити та керуючи обліковими даними. Додатково, цей сервіс надає функціонал створення presigned URL для будь-якого збереженого об’єкта, що створює тимчасове публічне посилання на нього. Використовуючи ці посилання, браузер користувача робить запит на отримання відео до серверів Amazon, а отже локальний сервер вебзастосунку уникає перевантаження.

```
public string GetPresignedUrl(string key, int expirationMinutes = 30, int expirationHours = 0)
{
    if (string.IsNullOrWhiteSpace(key))
    {
        return string.Empty;
    }

    var urlRequest = new GetPreSignedUrlRequest
    {
        BucketName = _settings.BucketName,
        Key = key,
        Verb = HttpVerb.GET,
        Expires = DateTime.UtcNow.AddMinutes(expirationMinutes).AddHours(expirationHours)
    };

    return _s3.GetPreSignedURL(urlRequest);
}
```

Рисунок 4.5 – Метод створення presigned URL

4.2 Тестування розробленого застосунку

Тестування вебзастосунку відеохостингу проводилось з метою перевірки елементів системи на коректність роботи та їх поведінку під час очікуваного навантаження. Цей процес розділений на три етапи: integration тести, e2e тести та тестування навантаження. Враховуючи структуру бекенду, а саме те, що він повністю API-driven та наявні сервіси тісно пов’язані зі сторонніми залежностями, було вирішено відмовитись від створення Unit тестів.

Integration тести

Задача integration тестів полягає у перевірці роботи контролерів та їх взаємодію із сервісами та базою даних (рис. 4.6). Використовуючи такий підхід можливо комплексно перевірити коректність роботи та взаємодії елементів системи із різними вхідних даних, що спрощує виявлення зон ризику та формування шляхів можливого покращення API на ранніх стадіях.

```
[Fact]
0 references
public async Task AdminDeleteVideo_WhenAuthorized_VideoNotFound()
{
    TestUserContext.Role = "admin";
    TestUserContext.UserId = "2";

    using (var scope = _factory.Services.CreateScope())
    {
        var db = scope.ServiceProvider.GetRequiredService<VideoHostingDbContext>();

        TestSeeder.ResetDB(db);
        TestSeeder.SeedTags(db);
        TestSeeder.SeedChannel(db);
    }

    var response = await _client.DeleteAsync("/Admin/video/1/delete",
    cancellationToken: TestContext.Current.CancellationToken);

    response.StatusCode.Should().Be(HttpStatusCode.NotFound);
}
```

Рисунок 4.6 – Приклад integration тесту

Після успішного виконання усіх тестів, використано інструмент cobertura та reportgenerator для аналізу результатів та відсотку покриття системи (табл. 4.3).

Таблиця 4.3 – Аналіз відсотку покриття тестами

Тип покриття	Значення
Line	88.2%
Branch	77.5%
Method	98.2%

Із результатів аналізу можна визначити, що відсоток покриття тестами є допустим для даної системи – усі основні методи та гілки успішно пройшли тестування, а ті частини, що залишились непокритими, або не є критично важливими, або тестувалися вручну.

E2e тести

End-to-end тестування має на меті перевірку повного робочого циклу застосунку або конкретного користувацького сценарію, а отже надає можливість прослідити як усі компоненти системи взаємодіють між собою у середовищі, наближеному до виробничого. Для застосунку відеохостингу створено два типи таких тестів: контроль виконання окремих операцій (такі як аутентифікація (рис. 4.7), перегляд відео та інше) і проходження повного шляху «реєстрація-створення каналу-завантаження відео».

```
import { test, expect } from '@playwright/test';

test('user can log in successfully', async ({ page }) => {
  await page.goto('http://localhost:5173/login');

  await page.fill('[data-testid="email"]', 'DeathRun@gmail.com');
  await page.fill('[data-testid="password"]', 'testPassword123');

  await page.click('[data-testid="login_button"]');

  await expect(page).toHaveURL('http://localhost:5173/');

  await expect(page.locator('text=Logout')).toBeVisible();
});
```

Рисунок 4.7 – Тестування логіну

Тестування навантаження

Тестування навантаження дозволяє перевірити поведінку та час відгуку системи, коли вона стикається із різною кількістю одночасних запитів. Аналізуючи результати даного тестування можна визначити ступінь оптимізованості застосунку та знайти максимальну кількість можливих користувачів, яку система може обробити. Вебзастосунок відеохостингу має вимогу до підтримки роботи 500 користувачів одночасно, а отже це число було взято за основу тестування навантаження (табл. 4.4).

Таблиця 4.4 – Тестування навантаження

Метрика	Значення
Налаштування тестування	10 хв, 500 запитів/с
Середній час відповіді	143 мс
Мінімальний час відповіді	1 мс
Максимальний час відповіді	431 мс
Відсоток помилок	0.00%

Аналізуючи результати можна визначити, що система комфортно справляється із навантаженням у 500 запитів в секунду – час відповіді не перевищує 0.5с навіть для складних запитів, такі як пошук/фільтрація відео. Як результат, система готова до роботи у виробничому середовищі під час очікуваного навантаження.

Для формування повноцінного вердикту щодо готовності вебзастосунку відеохостингу до експлуатації проведено додаткові тестування навантаження та часу відгуку із різними вхідними даними. Основним напрямком цієї перевірки було визначення максимальної кількості запитів, які система може обробити без катастрофічної втрати продуктивності (табл. 4.5).

Таблиця 4.5 – Контроль продуктивності під час різного навантаження

Кількість запитів	Середній час відповіді	Максимальний час відповіді	Відсоток помилок
250 запитів/с	66 мс	320 мс	0.00%
500 запитів/с	143 мс	431 мс	0.00%
750 запитів/с	215 мс	543 мс	0.00%
1000 запитів/с	271 мс	737 мс	0.00%
1500 запитів/с	429 мс	994 мс	0.00%

Інтерпретація результатів: незважаючи на постійне збільшення кількості запитів система показує стійкість до помилок, адже усі запити були успішно оброблені. Очікувано, із більшою кількістю користувачів обробка запитів стає

повільнішою, однак навіть під час 1000 запитів за секунду, що перевищує очікуване навантаження в два рази, система підтримує час відповіді менше секунди. Серйозна втрата продуктивності починає спостерігатися з 1500 запитів в секунду, адже максимальний час відповіді майже дорівнює 1 с, що є максимально допустимим лімітом для простих запитів.

Таким чином можна впевнено заявити, що розроблений вебзастосунок відеохостингу готовий до розгортання і експлуатації у виробничому середовищі, адже відповідає поставленим нефункціональним вимогам. Продемонстрована стійкість системи під час надмірного навантаження підкреслює її готовність до майбутнього розширення функціоналу та поступового збільшення клієнтської бази.

4.3 Керівництво користувача

У цьому підрозділі розглянуто потенційний шлях користувача від першої реєстрації до роботи з адміністративною панеллю. Проаналізовано кожну дію користувача та показано усі важливі сторінки.

Новий користувач

Користувач вперше заходить на сайт та відкриває головну сторінку (рис. 4.8). Перше, що він бачить це список доступних для перегляду відео, список тегів, поле пошуку та кнопки реєстрації і логіну. Система пошуку доступна усім користувачам та має два режими: пошук відео та пошук каналу. Список тегів виводить перелік усіх доступних категорій. Користувач може натиснути на будь-який тег, після чого вебзастосунок виведе тільки ті відео, які підпадають під цю категорію.

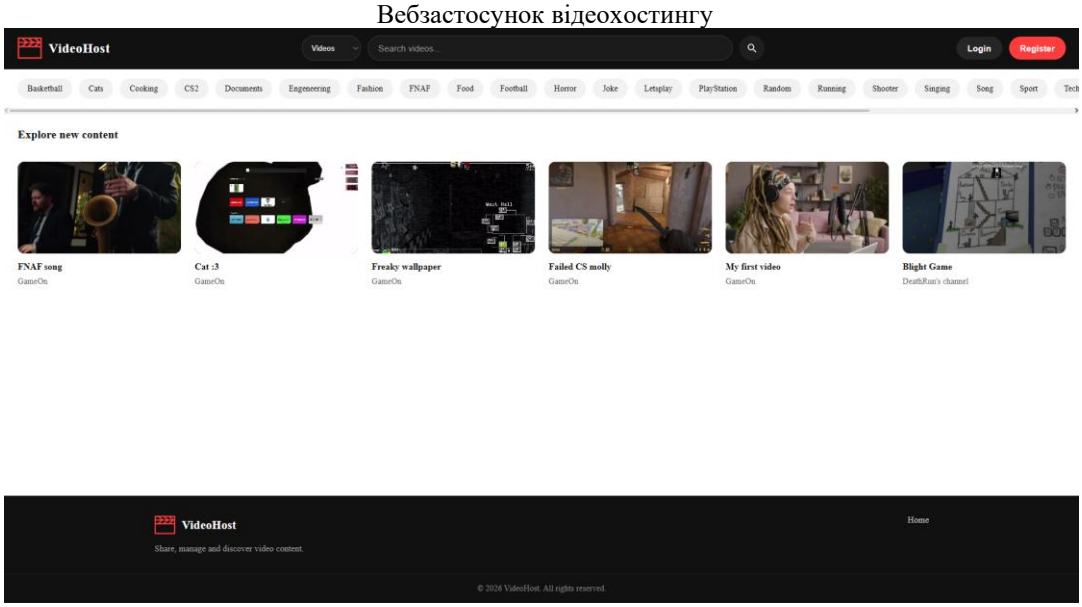

Рисунок 4.8 – Головна сторінка сайту

Користувач переходить на сторінку реєстрації, де він може створити собі акаунт (рис. 4.9). Система просить користувача ввести назву акаунту, зазначити email та придумати пароль. Після заповнення форми система перевіряє чи введено правильні дані та чи не існує вже акаунт з таким email – якщо пошук знайшов існуючий акаунт, то сервер надсилає помилку 400. У разі успіху, сервер використовує дані з форми для створення нового акаунту, автоматично присвоюючи йому роль глядача. По завершенню фіналізації усіх дій, користувача переводять на сторінку успіху (рис. 4.10), звідки він може перейти до сторінки логіну.

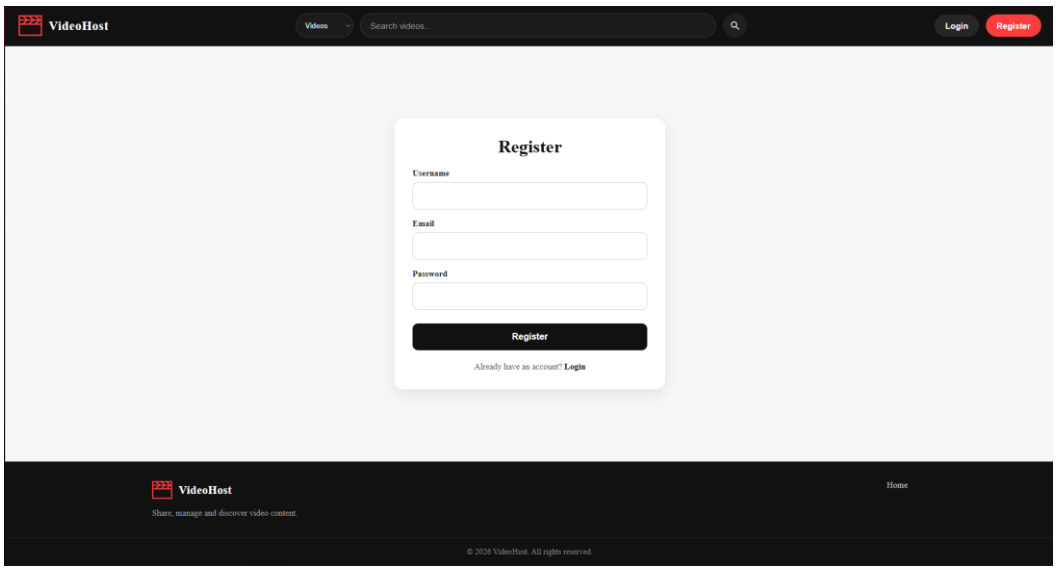

Рисунок 4.9 – Сторінка реєстрації

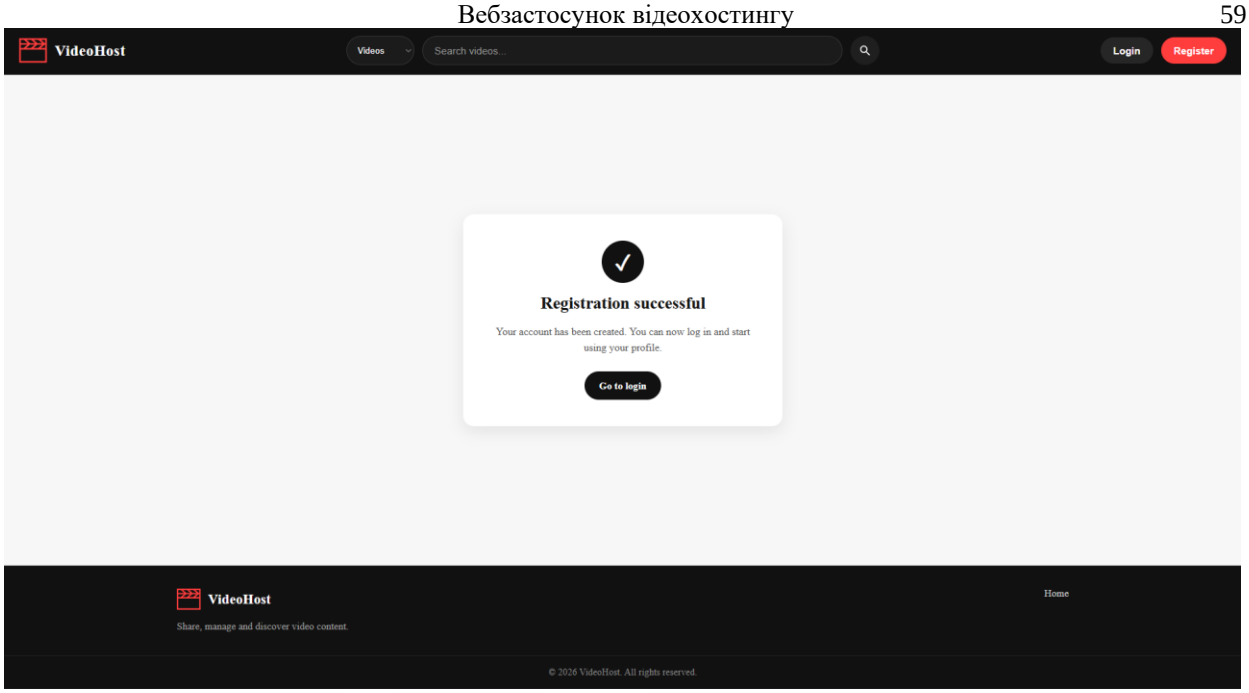

Рисунок 4.10 – Сторінка успіху реєстрації

У разі успішного логіну система створює та надсилає користувачу його JWT токен, що зберігає облікові дані користувача. Сайт отримує цей токен та зберігає його у вигляді кукі – таким чином він автоматично буде доданий до усіх майбутніх запитів до сервера. Кукі та токени залишаються дієвими протягом години, після чого користувачу пропонують заново увійти в свій акаунт.

Глядач

Авторизований користувач натискає на відео, після чого відкривається сторінка перегляду вибраного відео (рис. 4.11). На даний момент у глядача є декілька дій на вибір:

– переглянути відео;
– залишити свій коментар;
– видалити свій коментар;
– перейти на канал автора відео;
– залишити скаргу на відео;
– залишити скаргу на будь-які коментарі від інших користувачів.

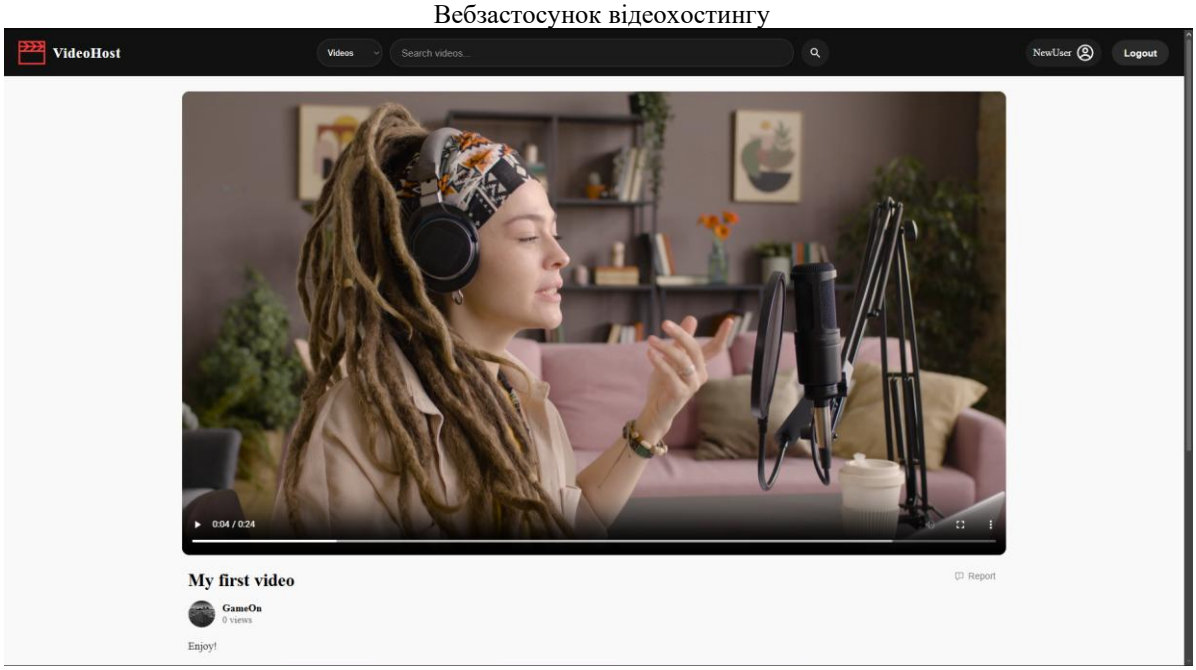

Рисунок 4.11 – Сторінка перегляду відео

Користувач вибирає переглянути канал автора, щоб знайти інші цікаві відео (рис. 4.12). Відкривається сторінка каналу, де Глядач може вибрати відео зі списку, підписатись на канал чи написати скаргу. Глядач вибирає підписатись, після чого система одразу оновлює кількість підписників цього каналу.

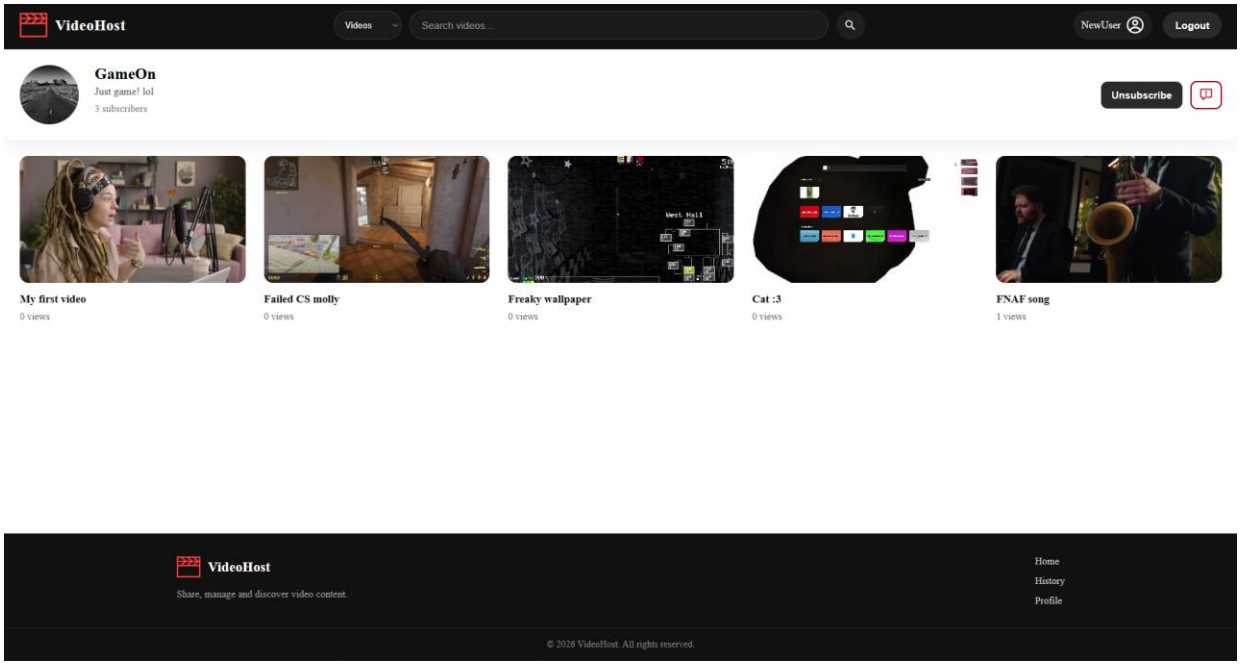

Рисунок 4.12 – Сторінка перегляду каналу

Наступним кроком глядача є створення свого каналу. Він переходить на сторінку свого профілю, де натискає на кнопку «Створити Канал». Система відкриває форму створення каналу (рис. 4.13), де користувач повинен зазначити його назву, опис та вибрати зображення профілю. Усі дані форми система верифікує локально та на сервері, а якщо якесь поле було введено неправильно то користувачу надсилається код помилки. Якщо ж всі дані правильно, то система створює новий канал, прив’язує його до акаунту користувача та міняє його роль на Автор.

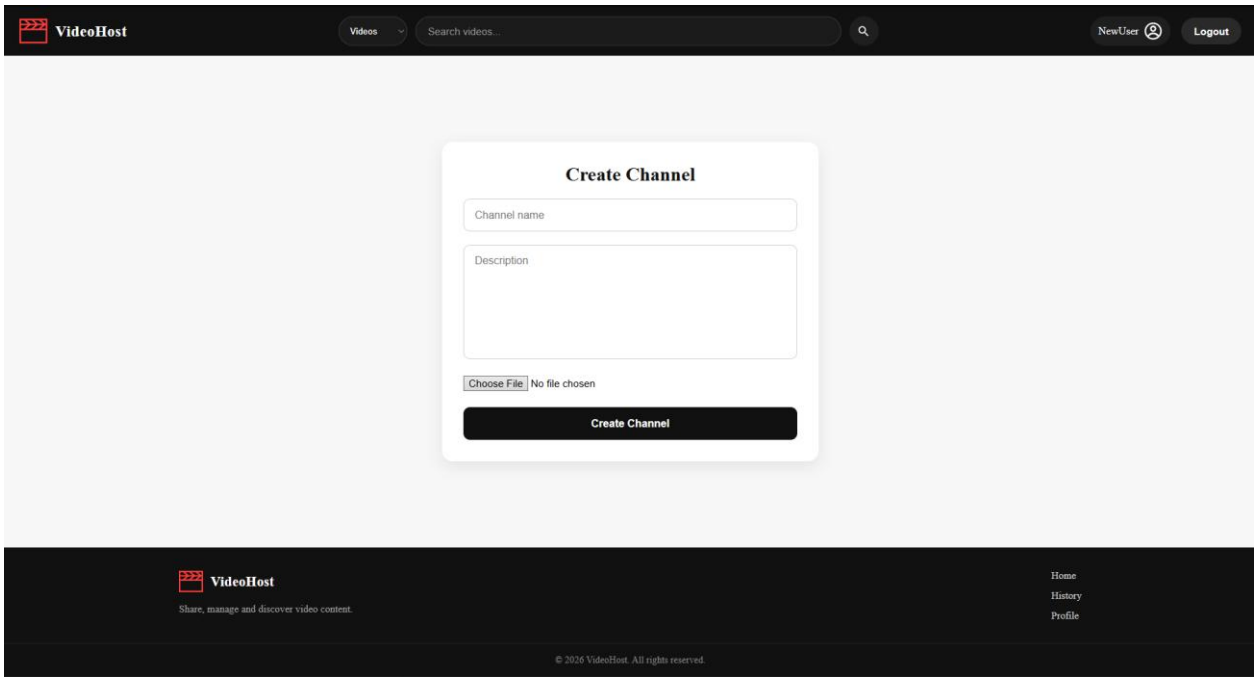

Рисунок 4.13 – Сторінка створення каналу

Автор

Одразу після створення каналу користувач захотів завантажити своє перше відео. На сторінці свого каналу він натискає на кнопку «Завантажити відео», що перенаправляє його до форми створення відео (рис. 4.14). Там він вводить назву та опис відео, обирає підходящі теги та відеофайл. Система перевіряє розмір файлу та його розширення, після чого відправляє відео на обробку. По завершенню обробки, користувача перенаправляє на сторінку свого каналу, де можна побачити нове відео.

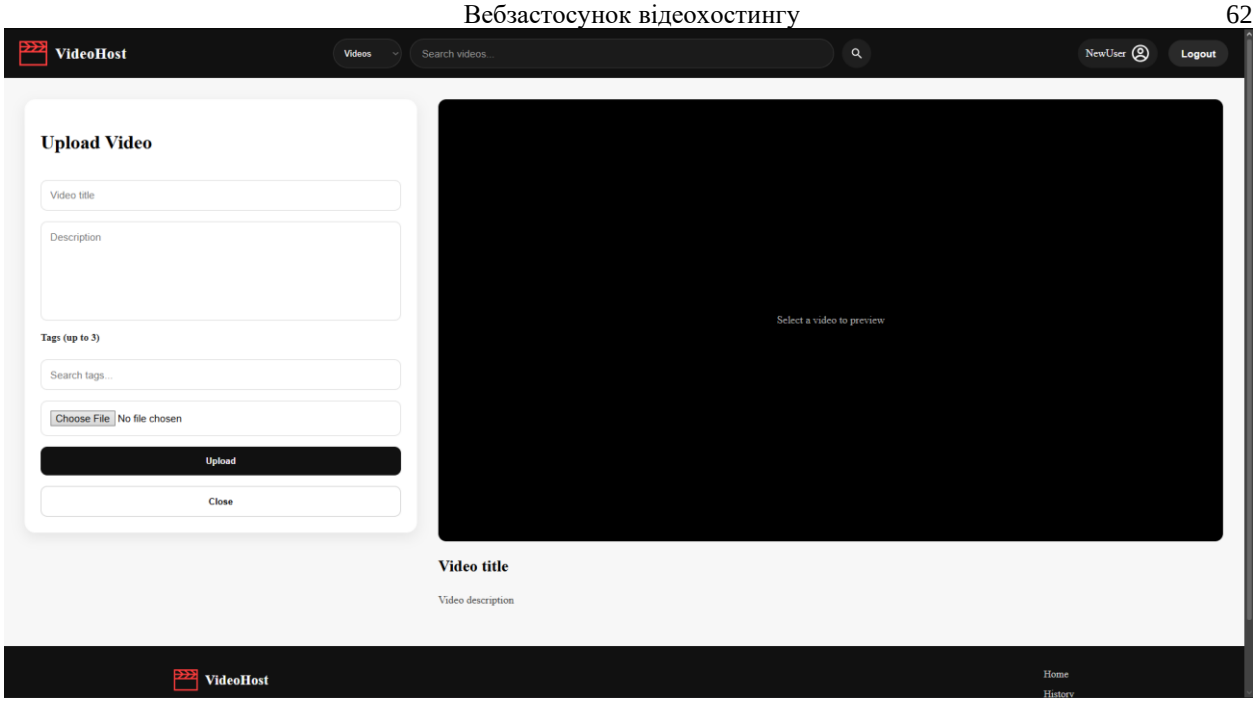

Рисунок 4.14 – Сторінка завантаження відео

Одразу після завантаження відео Автор вирішив його видалити. Він переходить на сторінку перегляду відео, де натискає на кнопку «Видалити», після чого відео зникає зі списку доступних. Повертаючись до свого каналу, Автор натискає на кнопку «Видалені відео», після чого попереднє відео з’являється у списку (рис. 4.15).

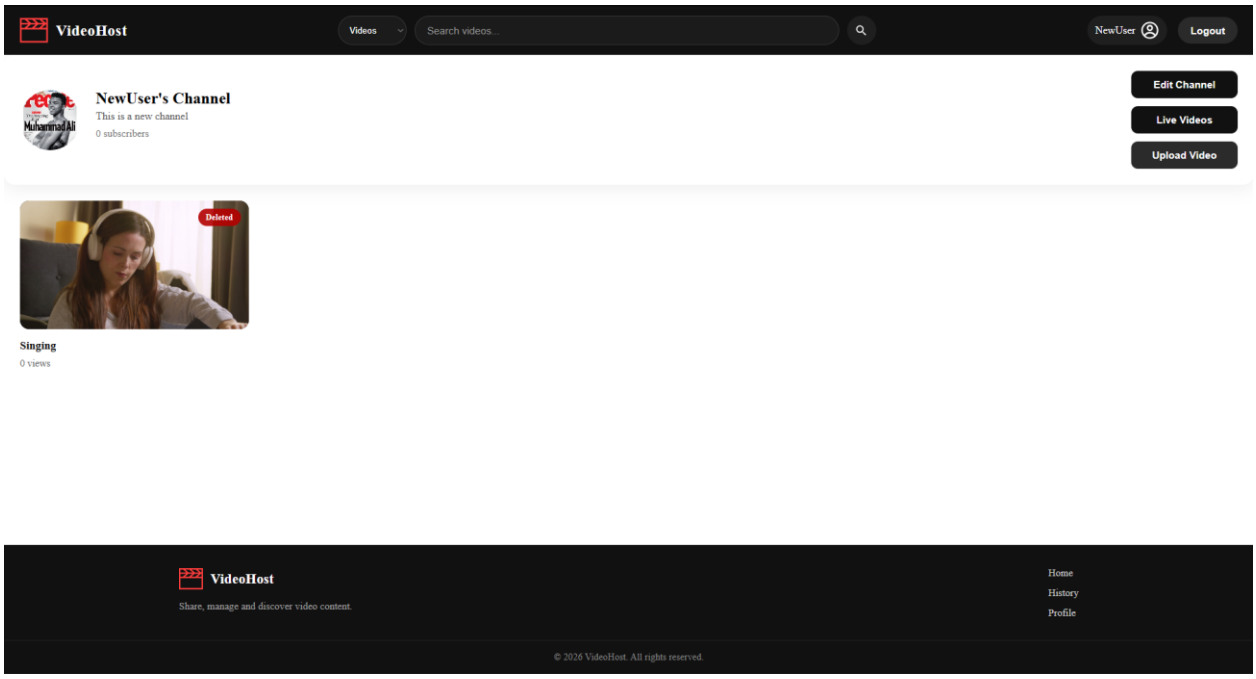

Рисунок 4.15 – Сторінка видалених відео

Редактор

Користувач отримав доступ до акаунту Редактора. Щоб перейти до адмін панелі, Редактор відкриває свій профіль та натискає кнопку «Адмін панель», що відкриває сторінку перегляду зареєстрованих користувачів (рис. 4.16).

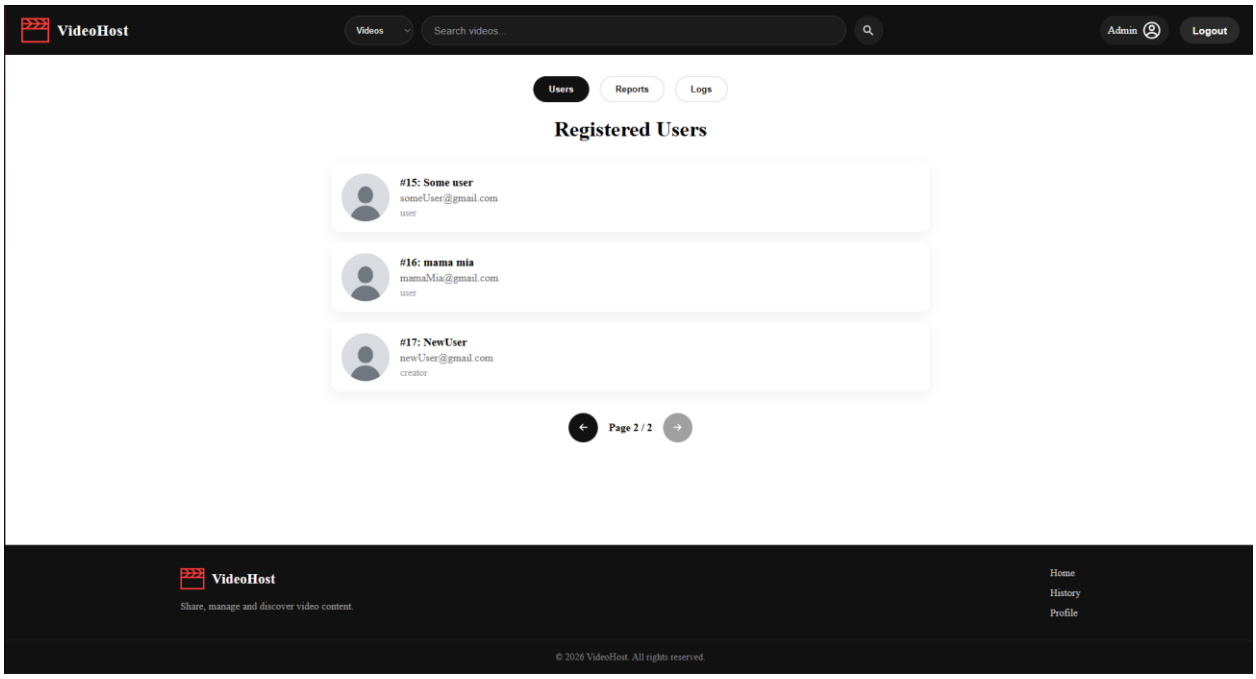

Рисунок 4.16 – Сторінка перегляду зареєстрованих користувачів

Редактор вирішив подивитись які скарги створили інші користувачі. Він натискає на кнопку навігації «Скарги», що перенаправляє його на сторінку перегляду скарг (рис. 4.17). Сторінка складається із двох частин, де перша показує предмет скарги (ним може бути відео, канал або коментар), а друга надає деталі самої скарги – хто її створив та зазначену причину. Така система дозволяє Редактору повністю сфокусуватися на своїй роботі, адже йому не треба переходити на іншу сторінку для виявлення чи підтвердження порушення.

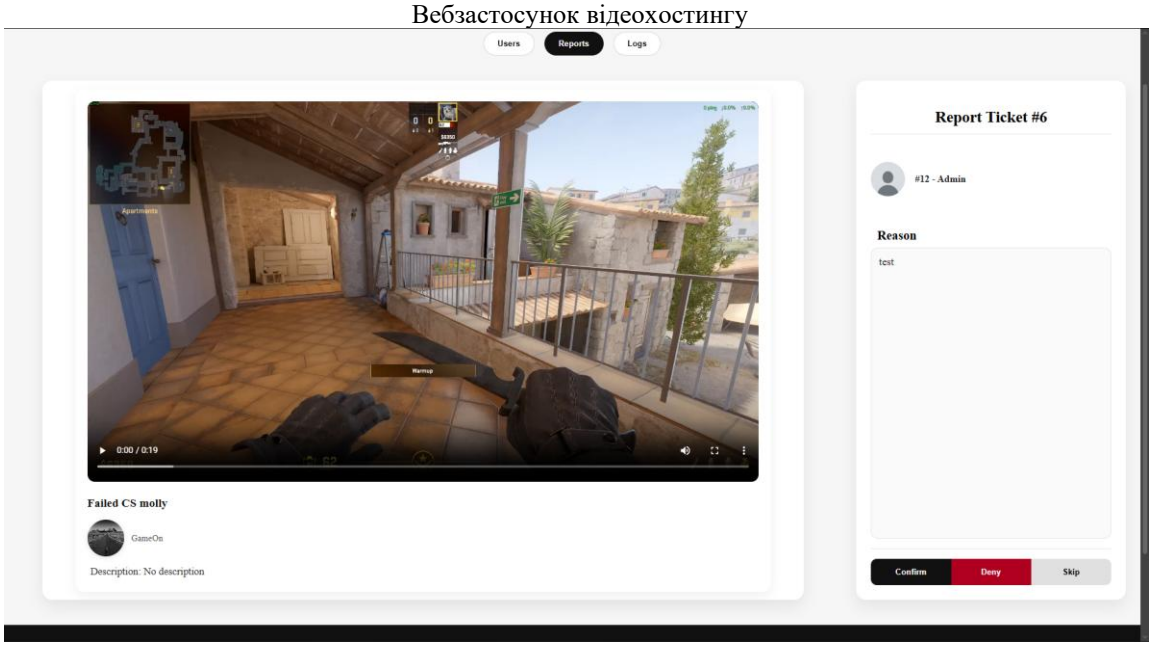

Рисунок 4.17 – Сторінка перегляду скарги

Окрім звичайного перегляду, Редактор має право вибрати дію щодо цієї скарги: якщо Редактор підтвердить скаргу, то відео буде позначено як видалене без можливості відновлення, а канал автора буде тимчасово заблокований; якщо Редактор відхилить скаргу, то вона буде видалена, а відео чи канал не постраждають. Пропускаючи скаргу Редактор переходить до наступної в списку – таким чином це дає можливість Редактору обдумати правильне рішення і винести вердикт пізніше.

Висновки до розділу 4

У четвертому розділі представлено програмну реалізацію та тестування вебзастосунку відеохостингу. Описано структуру фронтенд та бекенд частин із демонстрацією основних компонентів, сутностей, контролерів та сервісів. Розглянуто реалізацію integration та e2e тестів що перевіряли коректність роботи системи та її елементів. Описано виконані навантажувальні тести, результати яких підтвердили відповідність вебзастосунку до поставлених вимог та визначили готовність його розгортання у виробниче середовище. Створено керівництво користувача для демонстрації можливих дій та функціоналу, що доступні користувачам різних ролей. Як результат виконаної роботи, розробка вебзастосунку відеохостингу вважається успішною.

ВИСНОВКИ

У процесі виконання кваліфікаційної бакалаврської роботи досягнуто поставленої мети – розроблено вебзастосунок відеохостингу, що надає інструментарій категоризації контенту для полегшення поширення наукової та освітньої інформації.

Усі завдання. поставлені на початку роботи, успішно виконано в повному обсязі:

– проаналізовано предметну область, розкрито визначені об’єкт та предмет роботи;

– проаналізовано застосунки-аналоги YouTube, Dailymotion та Rumble, визначені переваги та недоліки використано для формування вимог до проєктованої системи;

– сформовано специфікацію вимог та функціоналу вебзастосунку, визначено критерії якості та безпеки;

– спроєктовано архітектуру, інтерфейс бізнес-логіку та фізичне розташування системи за допомогою UML-діаграм варіантів використання, класів, діяльностей, розгортання та мокапів;

– вибрано та обґрунтовано технологічний стек, використаний під час розробки вебзастосунку відеохостингу;

– виконано та описано програмну реалізацію основних елементів системи відеохостингу;

– протестовано розроблений застосунок за допомогою integration, e2e та навантажувальних тестів, проаналізовано результати тестування та зроблено відповідні висновки.

Практичне значення та новизна виконаної роботи полягають у розробці готового до виробничого використання вебзастосунку відеохостингу, що використовує систему категоризації та інтеграцію з сервісом хмарного сховища для сприяння збереження та поширення відеоконтенту різного типу та тематики.

Для вебзастосунку відеохостингу окреслено наступні шляхи подальшого розвитку:

– реалізація системи рекомендацій, що використовує теги та історію перегляду для персоналізованого підбору відеоконтенту;

– розширення системи категоризації для стимуляції більшої різноманітності контенту шляхом підбору менш популярних категорій;

– імплементація аналітичних інструментів та статистики для авторів;

– покращення та розширення системи створення відео: додання можливості вибору свого титульного зображення, розробка інструментарію базового відеомонтажу всередині вебзастосунку.

Як результат проведеної праці, кваліфікацйну бакалаврську роботу виконано у повному обсязі – опрацьовано і занотовано необхідну теорію, проведено проєктування, розробку та тестування готового до виробночої експлуатації вебзастосунку відеохостингу, визначено можливі шляхи його подальшого розвитку.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. FFmpeg. URL: https://www.ffmpeg.org (date of access: 09.08.2026);
2. BCrypt.Net. URL: https://github.com/BcryptNet/bcrypt.net (date of access: 09.08.2026);
3. Microsoft.AspNetCore.Authentication.JwtBearer. URL: https://www.nuget.org/packages/Microsoft.AspNetCore.Authentication.JwtBearer /9.0.16 (date of access: 09.08.2026);
4. YouTube. URL: https://www.youtube.com (date of access: 09.08.2026);
5. Dailymotion. URL: https://www.dailymotion.com/us (date of access: 09.08.2026);
6. Rumble. URL: https://rumble.com (date of access: 09.08.2026);
7. Angular. URL: https://angular.dev (date of access: 09.08.2026);
8. React. URL: https://react.dev (date of access: 09.08.2026);
9. Vue.js. URL: https://vuejs.org (date of access: 09.08.2026);
10. Srikanth Prabhu, Shiva Raj Pokhrel, Gang Li *Applications and Techniques in Information Security*. Springer Nature, 2022. p. 304;
11. IJRASET. *Comparative analysis on front-end frameworks for web applications*. academia.edu. 2022. Vol. 10, № 7. pp. 298–307;
12. Swacha J., Kulpa A. *Evolution of Popularity and Multiaspectual Comparison of Widely Used Web Development Frameworks*. Electronics. 2023. Vol. 12, № 17. p. 23. DOI: https://doi.org/10.3390/electronics12173563 (date of access: 01.05.2026);
13. Ліпський Д. Архітектура та функціональні особливості платформи для автоматизованого тестування вебзастосунків. ВІСНИК Київського національного університету імені Тараса Шевченка. 2025. Вип. 81, № 2. сс. 166–173. DOI: https://doi.org/10.17721/1812-5409.2025/2.26 (дата звернення: 01.05.2026);
14. Đurđev D. *Popularity of programming languages*. AIDASCO Reviews. 2024. Vol. 2, № 2. pp. 24–29;

15. Kastelik G., Gańczarczyk M., Więcław Ł., Gancarczyk T. *Performance comparison of modern backend programming languages*. University of Bielsko-Biała, 2024. pp. 125–136;

16. Akhtar A. *Popularity Ranking of Database Management Systems*. 2023. p. 8;

17. Oracle vs SQL Server - Key Differences. URL: https://airbyte.com/data-engineering-resources/oracle-vs-sql-server (date of access: 01.05.2026);

18. Daniel BĂRBULESCU, Adriana-Cristina ENACHE-DUCOFFE, Mihai TOGAN. *A new comparative study of database security*. Romanian Journal of Information Technology and Automatic Control. 2023. Vol. 33, № 3. pp. 17–28;

19. Yatsyshyn V., Pastukh O., Palamar A., Zharovskyi R. *Technology of relational database management systems performance evaluation during computer systems design*. Scientific journal of the Ternopil national technical university. 2023. Vol. 109, № 1. pp. 54–65. DOI:10.33108/visnyk_tntu2023.01.054 (date of access: 01.05.2026).