МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Львівський національний університет імені Івана Франка

Факультет електроніки та комп'ютерних технологій

Кафедра оптоелектроніки та інформаційних технологій

Допустити до захисту

Завідувач кафедри
_____ проф. Кушнір О. С.
«___» _____2023 р.

Кваліфікаційна робота

Бакалавр
(освітній ступінь)

Веб-сайт для завантаження та перегляду користувацького відео

Виконав:
студент IV курсу групи ФЕП– 41с
спеціальності 121 – Інженерія
програмного забезпечення

_____ В.В. Скребуха

Науковий керівник:
проф. С.А. Свелеба
_____
«____» _____2023 р.

Рецензент:
асистент. Я.А. Шмигельський
_____

Львів 2023

ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені ІВАНА ФРАНКА

Факультет_____Електроніки та комп'ютерних технологій_____
Кафедра_____Оптоелектроніки та інформаційних технологій____
Освітній ступінь _____бакалавр_____
Галузь знань_____12 "Інформаційні технології"_____
(шифр і назва)
Спеціальність _____121 «Інженерія програмного забезпечення»_____
(шифр і назва)

«ЗАТВЕРДЖУЮ»
Завідувач кафедри__*Оптоелектроніки та інформаційних технологій*
_____О.С. Кушнір___
"____" _____20___року

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ (БАКАЛАВРСЬКУ) РОБОТУ СТУДЕНТУ

_____Скребуха Володимир Володимирович_____
(прізвище, ім'я, по батькові)

1.Тема роботи: **Веб-сайт для завантаження та перегляду користувацького відео.**
керівник роботи ____Свелеба Сергій Андрійович д.ф.-м.н., професор,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені Вченою радою факультету від "31" жовтня 2022 року №__30/22_
2. Строк подання студентом роботи _____30 травня 2023 року_____
3. Вихідні дані до роботи

- ✓ *Davidson, D. (2020). "Vimeo and TikTok: Exploring Alternative Video-Sharing Platforms." In "Social Media and the Transformation of Interaction in Society" (pp. 113-135). IGI Global.*
- ✓ *Burgess, J., & Green, J. (2013). "YouTube and the Mainstream Media." In "YouTube: Online Video and Participatory Culture" (pp. 39-58). John Wiley & Sons.*
- ✓ *. Gill, P., & Kaur, S. (2020). "A comparative study of video-sharing platforms: YouTube, Vimeo, and Dailymotion." In 2020 International Conference on Intelligent Sustainable Systems (ICISS) (pp. 1252-1257). IEEE.*
- ✓ *Visual Studio Code: End-to-End Editing and Debugging Tools for Web Developers 1st Edition by Bruce Johnson (Author) Wiley; 1st edition (August 23, 2019) ISBN-13: 978-1119588184*
- ✓ *React Documentation - Офіційна документація React (https://reactjs.org/docs/)*
- ✓ *The Firebase Developer Documentation - Офіційна документація Google Firebase (https://firebase.google.com/docs)*

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити):

1. Огляд літератури по розробці веб-сайтів для завантаження відео.
2. Розробити веб-сайт для завантаження користувацьких відео, що базується на аналізі проблемної області.
3. Перевірка правильності роботи програми.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
____Таблиці та рисунки, знімки екрану та лістинг коду в результатах роботи.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання _1 листопада 2022 року__

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної (бакалаврської) роботи	Строк виконання етапів роботи	Примітка
1	*Аналітичний огляд літератури*	*01.11.2022 р. – 30.11.2022 р.*	
2	*Розробка веб-сайту для завантаження користувацьких відео, що базується на аналізі проблемної області.*	*01.12.2022 р. – 06.03.2023 р.*	
3	*Тестування та налагодження програми*	*20.03.2023 р. – 06.05.2023 р.*	
4	*Оформлення роботи*	*08.05.2023 р. – 30.05.2023 р.*	
5	*Попередній захист дипломної роботи*	*05.06.2023 р.*	

Студент _____ *Скребуха В.В*
(підпис) (прізвище та ініціали)

Керівник роботи _____ *Свелеба С.А.*
(підпис) (прізвище та ініціали)

Анотація

В цій дипломній роботі було розроблено веб-сторінку з наступними функціями: реєстрація профілю користувача, вхід у систему, вхід через сторонній обліковий запис google, відновлення пароля, завантаження відео на веб-сторінку, редагування завантаженого відео, можливістю обирати чи буде завантажене відео публічним чи приватним, система лайків та дизлайків, функціонал підписок, фільтрування відео по категоріям, система коментарів.

ABSTRACT

In this thesis, a web page was developed with the following functions: user profile registration, login, login through a third-party google account, password recovery, uploading a video to a web page, editing an uploaded video, the ability to choose whether the uploaded video will be public or private. private, a system of likes and dislikes subscription functionality, video filtering by category, comment system.

Зміст

ВСТУП.....6
РОЗДІЛ 1. ТЕОРЕТИЧНІ ВІДОМОСТІ. АНАЛІЗ ПРОБЛЕМНОЇ.....7
ОБЛАСТІ.....7
1.1 Огляд існуючих веб-сайтів з можливістю завантаження відео.....7
1.1.1 Tik Tok.....7
1.1.1.1 Недоліки Tik Tok.....8
1.1.2 Vimeo.....9
1.1.2.1 Недоліки Vimeo.....10
1.1.3 YouTube.....11
1.1.3.1 Недоліки YouTube.....12
1.2 Підсумок проблем у існуючих Веб-сайтів.....13
РОЗДІЛ 2. ОПИС ПРОГРАМНОГО СЕРЕДОВИЩА. ФРЕЙМВОРКІВ.....15
ВИКОРИСТАНИХ У РОЗРОБЦІ.ПОСТАНОВКА ЗАВДАННЯ.....15
2.1 Мова програмування Javascript.....15
2.1.1 Безпека.....16
2.1.2 Продуктивність.....17
2.2 Мова програмування TypeScript.....20
2.2.1 Переваги.....21
2.3 Visual Studio Code.....23
2.3.1 Огляд інтерфейсу.....24
2.3.2 Розширеня.....26
2.4 Платформа Node.js.....27
2.4.1. Переваги NodeJS.....28
2.4.2 Пакетний менеджер Npm.....29
2.5 Google Firebase.....30
2.6 Фреймворк React.....32
2.7 Фреймворк axios.....34
2.8 Фреймворк bootstrap4.....35
2.9 Програма FFmpeg.....37

2.10 Фреймворк express…….…....39
2.11 Постановка завдання.....40
2.11.1 Вхідні дані для виконання проекту.....40
2.11.2 Що планується отримати в результаті виконання проекту.....40
2.11.3 Архітектура рішення.....41
2.11.4 Вибір і обгрунтування засобів та технологій використаних для.....42
виконання проекту.....42
РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ ВЕБ-САЙТУ ДЛЯ ЗАВАНТАЖЕННЯ ТА ПЕРЕГЛЯДУ КОРИСТУВАЦЬКИХ ВІДЕО.....44
3.1 Реалізація серверної частини.....44
3.1.1 Структура серверу 44
3.1.2 Серверний скрипт та Корневий скрипт React.....45
3.1.3 Реалізація API завантаження відео та мініатюр.....47
3.1.3.1 реалізація завантаження відео на сервер.....48
3.1.3.2 реалізація створення прев'ю для відео.....49
3.2 Реалізація клієнтської частини.....50
3.2.1 Структура клієнту.....50
3.2.2 Структура бази даних.....54
3.2.2.1 Вхід в обліковий запис,реєстрація,відновлення парлю…….56
3.2.3 Реалізація коментарів.....60
3.2.4 Реалізація функції лайк/дизлайк.....62
3.2.5 Реалізація функції підписки.....63
3.2.6 Реалізація кількості переглядів.....65
3.2.7 Реалізація пошуку відео 66
3.3 Опис реалізованого функціоналу. Тестування.....68
3.3.1 Вхід у обліковий запис.....68
3.3.2 Сторінка “Реєстрація нового користувача”……..69
3.3.3 Сторінка “Домашня сторінка”……69
3.3.4 Сторінка “Завантаження відео”……..70
3.3.5 Сторінка “Перегляд відео”……73

3.3.6 Сторінка “Редагування відео”……75
3.3.7 Сторінка “Мої підписки”……76
3.3.8 Поле пошуку відео.....77
3.3.9 Сортування по категоріям.....78
ВИСНОВОК.....80
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....81
ДОДАТКИ.....83

Вступ

Наше життя складно уявити без наших улюблених гаджетів, кожен ранок встаючи з ліжка ми перше що робимо - беремо до рук телефон та дивимось чи хтось нам телефонував поки ми спали? Що нам написали у месенджери, пошту поки ми спали? Дальше зазвичай ми йдемо на кухню та робимо сніданок з кавою/чаєм і під час сніданку заходимо в наш улюблений youtube, netflix, Instagram, tik tok, або ж будь який інший додаток для поглинання відео контенту і це стало невідємною частиною нашого життя.

Завдяки цим популярним сервісам ми можемо сидячи у своїх чотирьох стінах насолоджуватись як розважальним, науковим, так і кінематографічним Деякі системи такі як youtube, Instagram, tik tok дозволяють користувачам цього сервісу самим завантажувати відео-аудіо контент та його ж переглядати.

У цій дипломній роботі було взято за мету – створити власний веб-сайт для користувацьких відео, з метою оцініки важкості такої розробки як одинієї з можливих застосувань в контексті використання в України адже нажаль, в України немає свого сервісу завантаження відео-контенту через це ми не можемо контролювати контент на не наших сервісах і вчасно його видаляти.

Також існуючі веб-сайти не є доскональними і у кожного з них є проблеми, ці проблеми я й обрав для вирішення створюючи свій веб-сайт, окрім цього також в якості побудови загально скелета додатку яку би у подальшому користувачі github могли б підтримувати та вдосконалювати платформу.

РОЗДІЛ 1. ТЕОРЕТИЧНІ ВІДОМОСТІ. АНАЛІЗ ПРОБЛЕМНОЇ ОБЛАСТІ

1.1 Огляд існуючих веб-сайтів з можливістю завантаження відео

1.1.1 Tik Tok

TikTok — популярний веб-застосунок для відео-контенту, який захопив весь світ. TikTok, запущений у 2016 році, дозволяє користувачам створювати та ділитися короткими відеороликами під музику, Він набув величезної популярності, особливо серед молодшої демографії, і став культурним явищем із величезною базою користувачів по всьому світу (Рис 1.1).

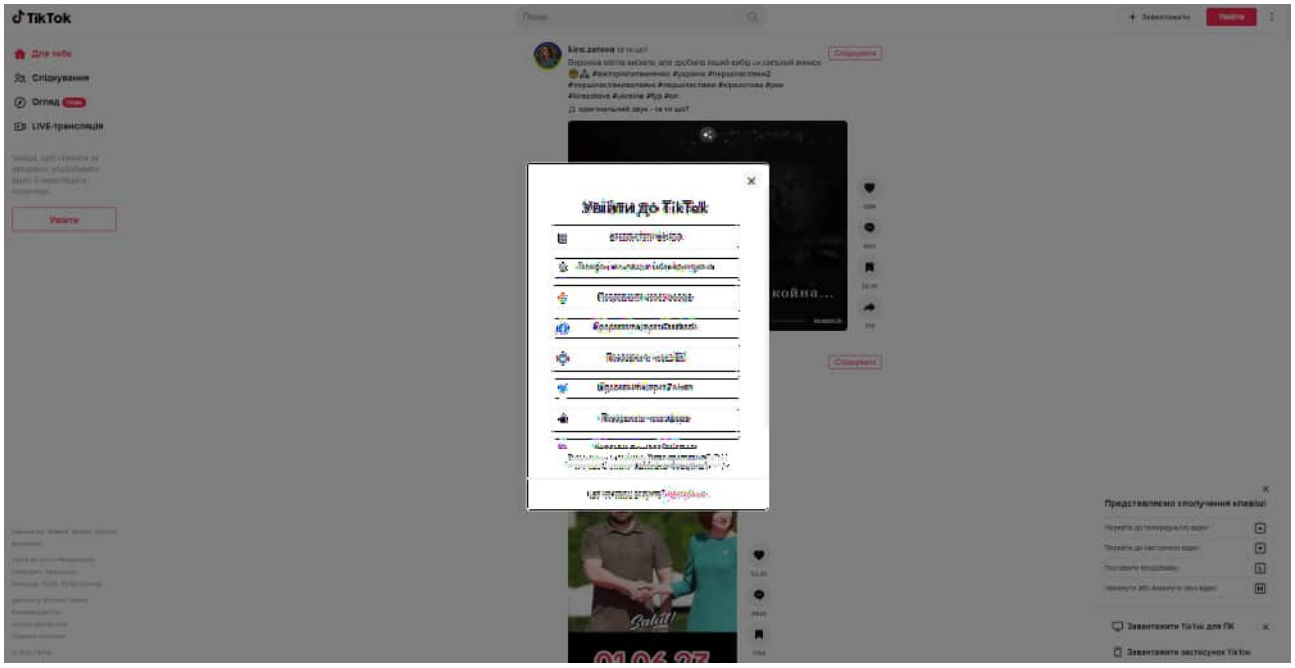

Рис 1.1 Зображення інтерфейсу веб-сайту

Однією з ключових особливостей TikTok є його простота та доступність. Програма надає зручний інтерфейс, який дозволяє будь-кому легко створювати, редагувати та ділитися відео. Завдяки широкому спектру творчих інструментів, фільтрів, ефектів і музичних параметрів TikTok пропонує користувачам можливість створювати візуально привабливий і захоплюючий вміст. Функції редагування дозволяють користувачам додавати текст, наклейки та переходи, покращуючи загальне враження від відео.

Соціальні функції TikTok є невід’ємною частиною програми. Користувачі можуть стежити за іншими, ставити лайки та коментувати відео, а також брати участь у дуетах і співпраці з іншими творцями.

Ця інтерактивна природа заохочує зв’язок і взаємодію між користувачами, сприяючи почуттю причетності та соціальної залученості.

1.1.1.1 Недоліки Tik Tok

Одна з ключових проблем, пов'язаних з TikTok, стосується питань конфіденційності та безпеки даних користувачів. У зв'язку з тим, що компанія-власник TikTok, ByteDance, має штаб-квартиру в Китаї, виникали побоювання стосовно збору даних і можливого доступу до них з боку китайських владних органів.

Хоча TikTok заявляє, що дані користувачів, зокрема їхні відео та особиста інформація, зберігаються на серверах, розташованих за межами Китаю, все ж є питання щодо рівня конфіденційності та можливості неконтрольованого доступу до даних.

Це питання привернуло увагу різних країн, і були запроваджені різні регуляторні заходи, які спрямовані на забезпечення захисту даних користувачів на платформі.

- Проблеми контролю вмісту: TikTok був об'єктом скарг на неприйнятний або шкідливий вміст, включаючи насильство, булінг і експлуатацію

- Проблеми алгоритмічного контролю: Однією з особливостей TikTok є його алгоритмічна система рекомендацій, яка курує вміст, який користувачі бачать у своєму стрічці.

Це може призводити до обмеженої свободи вибору контенту та може бути обтяжливим для творців, які хочуть більшу контроль над своїм вмістом.

- Проблеми залежності від алгоритму: За своєю природою TikTok спрямований на максимальне залучення користувачів та збільшення часу, проведеного на платформі. Це може стати проблемою для користувачів, які хочуть керувати своїм часом та не допустити відволікання.

- Проблеми з аудієнцією та розповсюдженням: Хоча TikTok має велику аудиторію, конкуренція на платформі також є великою.

Крім того, використання TikTok може мати вплив на психічне здоров'я та самопочуття користувачів. Завдяки безперервному засвоєнню контенту та його алгоритмам, користувачі можуть відчувати тиск створювати власний "популярний" вміст або відчувати неадекватну самооцінку порівняно з іншими користувачами.

Такі фактори можуть призводити до стресу, тривоги та незадоволення своїм власним життям. Ці аспекти психологічного впливу TikTok стають предметом досліджень та обговорень.

1.1.2 Vimeo

Vimeo - це популярна веб-платформа для спільного використання та перегляду відео. Заснована у 2004 році, ця платформа надає багато функцій та можливостей для завантаження, перегляду та взаємодії з відео (Рис 1.2).

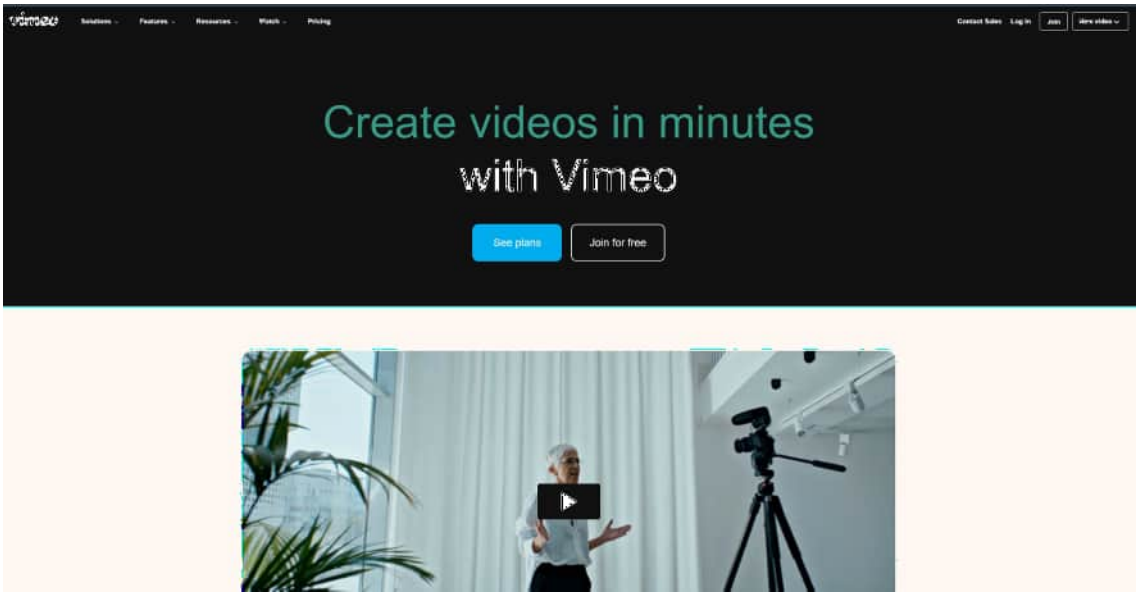

Рис 1.2 Веб-сайт Vimeo

Vimeo є платформою, яка пропонує багато інструментів для творчості. Ці інструменти дозволяють користувачам редагувати свої відео безпосередньо на сайті, додавати підписи, водяні знаки та синхронізувати відео з аудіофайлами. Такі можливості значно полегшують роботу творцям вмісту, оскільки вони

можуть створювати професійно виглядаючі відео, не вдаючись до використання зовнішніх програм для редагування.

Проте, слід відзначити, що Vimeo має обмежену аудиторію порівняно з платформами з великими платформами. Це може впливати на розповсюдження вашого вмісту та досягнення широкої аудиторії. Тому, якщ метою є максимальне поширення і залучення аудиторії, варто врахувати це обмеження.

Незважаючи на це, Vimeo є ідеальною платформою для професіоналів та творців вмісту, які цінують якість інструментів. Вона надає багато корисних можливостей для завантаження, перегляду, редагування та спілкування. Завдяки цим інструментам, ви можете покращити якість свого вмісту, виразити свою унікальність та поділитися своїми творчими ідеями зі світом.

Також варто зазначити, що Vimeo ставить на перший план спілкування з аудиторією. Ви можете створювати спеціальні канали, коментувати відео, обмінюватися думками з іншими користувачами та взаємодіяти зі своїми фанатами та прихильниками.

1.1.2.1 Недоліки Vimeo

Одним з недоліків Vimeo є його обмежена безкоштовна версія. В порівнянні з іншими платформами для завантаження відео, які пропонують безкоштовні плани з більш широким функціоналом, Vimeo має обмежені можливості для безкоштовних користувачів.

Наприклад, у безкоштовній версії Vimeo можуть бути обмеження щодо кількості відео, які можна завантажити, або обмеження щодо розміру файлу. Це може бути недоцільним для тих, хто потребує безкоштовного простору для завантаження великого обсягу відео.

Ще один з недоліків Vimeo є обмежена аудиторія порівняно зі широко популярними платформами, Vimeo не має такої широкої бази активних користувачів, тому розповсюдження вашого вмісту тут може бути менш ефективним, особливо якщо мета - привернення максимальної кількості переглядів і популярність.

Завдяки меншій аудиторії, вміст може мати меншу кількість вподобань, коментарів та розповсюджень порівняно з платформами, які мають широку базу активних користувачів.

Іншим недоліком є обмежена підтримка форматів відео. Vimeo має свої власні вимоги до формату відео, що може створювати проблеми для тих, хто має велику кількість відео в інших форматах.

Наприклад, якщо відео мають формат, який не підтримується Vimeo, вам доведеться конвертувати їх у підтримуваний формат перед завантаженням. Це може забирати більше часу та зусиль.

Додатковим недоліком Vimeo є обмежені можливості соціального взаємодії та спільноти. Хоча платформа має функцію коментування та взаємодії з іншими користувачами, воно не так широко розповсюджене та активне.
Також варто зазначити, що Vimeo має меншу спільноту активних користувачів порівняно з більшими платформами, такими як YouTube.

Це може вплинути на розповсюдження вашого вмісту та взаємодію з іншими користувачами. Якщо вам потрібно широке поширення вмісту та більш активну спільноту, то Vimeo може не задовольнити ваші очікування.
Vimeo спрямований більше на професіональну спільноту та якісний вміст.

Це може бути перевагою для деяких творців, які хочуть зосередитися на якості, але може бути обмеженням для тих, хто прагне досягти максимальної популярності та розповсюдження свого вмісту.

1.1.3 YouTube

YouTube є найбільшою відеоплатформою в Інтернеті, яка пропонує користувачам можливість завантажувати, переглядати та ділитися відеоконтентом (Рис 1.3). Заснований у 2005 році, YouTube став не тільки популярним серед користувачів, але й невід'ємною частиною культури Інтернету, масової комунікації та розваг.

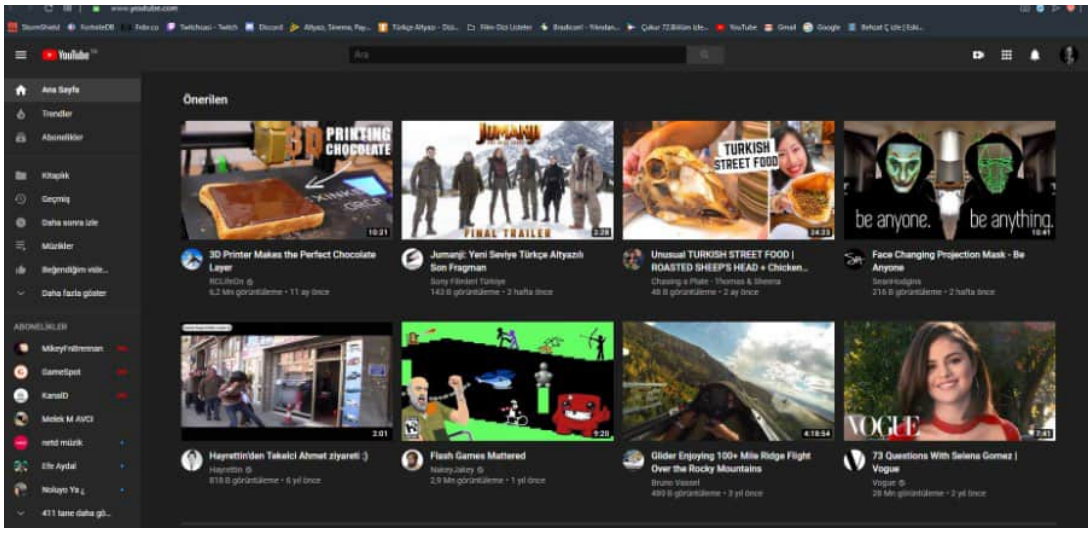

Рис 1.3 Інтерфейс веб-сайту Youtube

YouTube надає безліч можливостей для розповсюдження відеоконтенту. Власники каналів можуть завантажувати відео будь-якої тематики і жанру, що дає їм великий простір для самовираження та творчості.

YouTube є потужним інструментом для популяризації бренду, продукту або послуги. Багато компаній та бізнесів використовують YouTube для створення відеооглядів, рекламних роликів та навчального контенту з метою залучення нових клієнтів та збільшення своєї аудиторії. Завдяки можливості монетизації, власники каналів можуть отримувати дохід від рекламних показів на своїх відео.

Варто зазначити, що YouTube дозволяє взаємодіяти з аудиторією. Глядачі можуть залишати коментарі, ставити лайки та ділитися відео з іншими.

1.1.3.1 Недоліки YouTube

YouTube, будучи однією з найбільших платформ для обміну відео, безсумнівно, революціонізував спосіб споживання та обміну відео. Однак, як і будь-яка інша платформа, вона не позбавлена недоліків і недоліків.

Одна з проблем - поширеність оманливого вмісту або вмісту, що вводить в оману. Деякі творці контенту вдаються до сенсаційних назв, ескізів і оманливих описів, щоб залучити більше переглядів і зацікавленості. Це

підриває довіру до платформи та може призвести до дезінформації або розчарування серед глядачів, які шукають справжній і цінний вміст.

Система монетизації YouTube також зазнала критики. Хоча він пропонує творцям можливість отримувати дохід за допомогою реклами та спонсорства, алгоритмічний характер монетизації може бути непослідовним і несправедливим. Багатьом творцям важко відповідати суворим критеріям монетизації або на їхні прибутки суттєво впливають зміни в політиках чи алгоритмах розміщення реклами.

YouTube збирає величезну кількість даних користувачів для цільової реклами та персоналізації. Це підняло питання про те, як дані користувачів зберігаються, передаються та потенційно доступні третім особам.

Незважаючи на те, що YouTube запровадив функції конфіденційності та керування користувачами, забезпечення повного захисту даних і прозорості залишається постійною проблемою.

Інше занепокоєння пов’язане з алгоритмом рекомендацій платформи. Хоча він спрямований на персоналізацію вмісту та підтримку зацікавленості користувачів, його критикували за просування суперечливого або екстремального вмісту, який може увічнити упередженість, дезінформацію.

1.2 Підсумок проблем у існуючих Веб-сайтів

Підсумуючи проблеми, які виникають на платформах TikTok, Vimeo та YouTube, можна зазначити наступне: Проблема безпеки і конфіденційності даних. Всі три платформи стикаються з питаннями щодо збору, використання та захисту особистої інформації користувачів. Це створює загрозу приватності та можливість несанкціонованого доступу до даних.

Проблема неприпустимого вмісту. Усі три платформи зіткнулися з випадками шкідливого, непристойного або навіть незаконного вмісту. Це може шкодити користувачам, особливо молодій аудиторії, і порушувати етичні та правові норми.

Проблема недостатньої модерації. Усі три платформи мають завдання модерувати вміст, щоб запобігти поширенню неприпустимого або шкідливого контенту. Однак, ця задача виявляється складною через великий обсяг завантажуваного вмісту та недосконаліть автоматизованих систем модерації.

Проблема нерівномірного розподілу доходів. У платформах TikTok, Vimeo та YouTube є можливість монетизувати вміст, але механізм розподілу доходів може бути недостатньо прозорим або несправедливим, особливо для невеликих або початківців каналів.

РОЗДІЛ 2. ОПИС ПРОГРАМНОГО СЕРЕДОВИЩА. ФРЕЙМВОРКІВ ВИКОРИСТАНИХ У РОЗРОБЦІ. ПОСТАНОВКА ЗАВДАННЯ

2.1 Мова програмування Javascript

JavaScript с був створений, щоб «оживити веб-сторінки». Програми цією мовою називаються скриптами. Їх можна записати прямо в HTML веб-сторінки та запускати автоматично під час завантаження сторінки. Сценарії надаються та виконуються як звичайний текст. Для їх запуску не потрібна спеціальна підготовка чи компіляція.

JavaScript може виконуватися не тільки в браузері, але і на сервері, або фактично на будь-якому пристрої, який має спеціальну програму під назвою JavaScript engine. Браузер має вбудований механізм, який іноді називають «віртуальною машиною JavaScript». Двигуни складні. Але основи прості. Двигун (вбудований, якщо це браузер) читає («розбирає») сценарій. Потім він перетворює («компілює») сценарій у машинний код. А потім машинний код виконується досить швидко.

Механізм застосовує оптимізацію на кожному кроці процесу.

Він навіть стежить за скомпільованим сценарієм під час його виконання, аналізує дані, що проходять через нього, і додатково оптимізує машинний код на основі цих знань. Можливості JavaScript значною мірою залежать від середовища, у якому він працює. Внутрішньо-браузерний JavaScript може робити все, що стосується маніпулювання веб-сторінками, взаємодії з користувачем і веб-сервером.

Наприклад у веб-переглядачі може:Додавати новий HTML на сторінку, змініть наявний вміст, змініть стилі. Реагувати на дії користувача, запускати на клацання мишкою, рух вказівника, натискання клавіш. Надсилати запити по мережі на віддалені сервери, завантажувати та завантажувати файли. Отримувати та встановлювати файли cookie, ставити запитання відвідувачу, показувати повідомлення.

2.1.1 Безпека

Безпека JavaScript є головним пріоритетом. Існує величезна кількість речей, які можуть піти не так, від програмних помилок і незахищених даних користувача до зловмисних атак. Хоча моніторинг помилок JavaScript може допомогти і виявити багато з цих проблем, розуміння загальних ризиків безпеки JavaScript і дотримання найкращих практик є не менш важливим.

Найкращі методи безпеки досягаються використанням лінтер JavaScript, аудіт залежностей за допомогою менеджера пакетів та перевірку цілісності підресурсу для зовнішніх сценаріїв.

Почнемо з трьох вразливостей безпеки JavaScript. Під час XSS-атаки зловмисник впроваджує на веб-сторінку шкідливий клієнтський сценарій. Зазвичай вони досягають цього, обходячи політику того самого походження веб-сайту. В результаті зловмисник може отримати доступ до даних користувача та виконувати дії від імені користувача.

Підробка міжсайтового запиту

Атаки CSRF націлені на автентифікованих (увійшли в систему) користувачів, яким програма вже довіряє. Зловмисник отримує доступ до облікового запису користувача, використовуючи інформацію, знайдену в сеансових файлах cookie, і виконує дії від його імені без його відома чи участі. Атаки CSRF також відомі як атаки на сеанс або атаки в один клік.

Уразливості безпеки сторонніх розробників

У інтерфейсній розробці ми використовуємо багато інструментів і бібліотек сторонніх розробників, які відкриті для всіх видів експлойтів JavaScript. Деякі з цих інструментів, як-от React від Facebook, розроблені та підтримуються великими корпораціями, вони надійно виправляють проблеми та дотримуються найкращих практик безпеки JavaScript. Однак багато з них створені незалежними розробниками або невеликими командами, які не завжди мають ресурси для регулярного аудиту або оновлення свого коду. Щоб захистити свій проект слід використовувати лінтер JavaScript

Linters — це інструмент статичного аналізу коду, який перевіряють код на програмні та стилістичні помилки коду та відомі експлойти безпеки. Трьома найвідомішими лінтерами JavaScript є JSHint, JSLint і ESLint. Сучасні редактори вихідного коду, такі як Visual Studio Code та Atom, також мають функцію лінінгування JavaScript.

Щоб контролювати вразливості JavaScript сторонніх розробників потрібно відстежувати всі пакети, які використовуються на веб-сайті. Це можна зробити за допомогою менеджера пакетів, наприклад npm, Yarn або pnpm.

Крім того, що вони дозволяють відстежувати, керувати та оновлювати залежності, ці менеджери пакунків також надають інструменти для аудиту пакунків і пошуку поширених проблем безпеки JavaScript, таких як аудит npm або команди аудиту pnpm які дозволяють запускати аудит коду на різних рівнях.

Оскільки сторонніми або зовнішніми сценаріями можна легко маніпулювати, перевірка їх цілісності перед отриманням із зовнішнього сервера є одним із найважливіших найкращих методів безпеки JavaScript. Перевірка цілісності підресурсу — це функція, вбудована в сучасні веб-браузери яка використовує криптографічний хеш для перевірки цілісності зовнішнього сценарію.

Щоб згенерувати хеш-значення, можна використовувати такий генератор, як SRI Hash Generator, або інструмент командного рядка, наприклад OpenSSL.

2.1.2 Продуктивність

Розробка сайтів на JavaScript відрізняється від інших мов програмування. У типових веб-додатках ми маємо справу з 3–20 елементами одночасно, тому продуктивність незначна. Навіть у Node.js ми розбиваємо дані на сторінки й дозволяємо своїй базі даних виконувати важку роботу. Зважаючи на те, як рідко можна побачити більше кількох елементів.

Продуктивність, пам’ять, розмір вихідного коду – все важливо. Ефективність у програмі значною мірою залежить від інтерпретатора JavaScript. Здебільшого багато речей, які ви написали в минулому, працюватимуть швидше з кожним майбутнім оновленням інтерпретатора але може бути і навпаки (Рис 2.1)

Рис 2.1 Порівняння швидкості коду Javascript у браузерах

Завжди є деякі аспекти продуктивності інтерфейсу, які знаходяться поза контролем особливо при використанні JavaScript – найефективнішими будуть сайти з невеликим або відсутнім JavaScript (хоча це нереально з урахуванням вимог багатьох веб-сайтів).

Ресурси сайту ще повинні бути завантажені клієнтом. Незважаючи на те, що ми можемо пом’якшити це, об’єднавши програму в пакет, продуктивність зрештою залежить від швидкості мережі. JavaScript часто взаємодіє з різноманітними API. Хоча швидкість мережі тут також є фактором, продуктивність також залежить від того, скільки часу потрібно API для обробки запиту та надсилання відповіді.

Одним із найбільш впливових факторів продуктивності веб-сайту є час, який потрібно для завантаження початкових ресурсів. Як правило, що складніша програма, то більші ресурси потрібно завантажити.

Завантаження ресурсів особливо важливе для користувачів у мережах нижчого рівня, які не мають такого ж рівня швидкості та узгодженості, як 4G і 5G. Глобальний індекс Speedtest дає уявлення про різницю між швидкостями мережі в усьому світі. Допомога в покращенні продуктивності та часу завантаження наших додатків може принести значні переваги користувачам із повільнішим мережевим з'єднанням і є важливим кроком до того, щоб зробити Інтернет максимально доступним.

Браузер користувача зберігатиме ресурси в кеші, щоб наступного разу, коли той самий користувач відвідає сайт, він зможе отримати ресурси з локальних даних, а не через HTTP. Оскільки ресурси зберігаються локально, користувач повинен відвідати сайт, перш ніж його ресурси можна буде кешувати. Користувач може будь-коли видалити свій локальний кеш/кеш браузера. Кешування CDN є однією з головних переваг.

Маючи подібність із кешуванням браузера, кеш CDN призначений для зберігання ресурсів для певної програми. Основна відмінність полягає в тому, що CDN зберігає ресурси на сервері в географічно близькому місці до користувача, а не на його локальному комп'ютері. Це означає, що ресурси переміщуються на меншу відстань, надаючи користувачам прискорений доступ до вмісту програми. Використання Web Workers дозволяє запускати сценарії у фонових потоках, знімаючи тиск з основного потоку.

Хоча робочий процес може бути повільним для запуску, міжпотоковий зв'язок надзвичайно швидкий. Їх використання все ще залежить від ситуації, і вони ще не отримали широкого поширення.

Головний потік JS може породжувати необмежену кількість веб-воркерів, доки ресурси користувача не будуть використані повністю. Chrome 80 додав підтримку для роботи з модулями.

Це означає, що робочі версії тепер можуть працювати з усіма перевагами модулів JS: динамічним імпортом, паралельним завантаженням залежностей, оптимізованим виконанням тощо. Оскільки багато нових JS зараз пишуться як модулі, для робітників також приємно мати цю функціональність.

Хоча веб-інтерфейси API були згадані вище, їх також можна використовувати, щоб розвантажити роботу. Існує велика кількість доступних веб-API - вони дозволяють браузеру виконувати завдання, у той час як потік JavaScript продовжує працювати безперервно.

Після виконання завдання може викликати зворотний виклик для повторного входу в потік JavaScript. Наприклад, замість того, щоб писати складну спеціальну частину логіки зберігання та отримання даних на стороні клієнта JS, може мати сенс інтерфейс з API IndexedDB і абстрагувати логіку та продуктивність читання/запису.

Продуктивність веб-додатків на основі JavaScript постійно покращується завдяки прогресу у розвитку браузерних технологій, оптимізаціям і підходам до розробки. На шляху до досягнення високої продуктивності важливо постійно оновлюватись з новими можливостями, використовувати ефективні практики програмування та уважно відстежувати зміни веб-стандартів. Це дозволить розробникам створювати швидкі та ефективні веб-застосунки.

2.2 Мова програмування TypeScript

JavaScript був представлений як мова для клієнтської сторони. Розробка Node.js також відзначила JavaScript як нову серверну технологію. Однак у міру того, як код JavaScript зростає, він має тенденцію ставати брудніше, що ускладнює підтримку та повторне використання коду. Крім того, його неспроможність охопити функції об’єктної орієнтації, надійної перевірки типів і перевірки помилок під час компіляції заважає JavaScript досягти успіху на рівні підприємства як повноцінної серверної технології. Щоб подолати цю прогалину, було представлено TypeScript.

За визначенням, «TypeScript — це JavaScript для розробки програмного масштабу». Це строго типізована, об’єктно-орієнтована, скомпільована мова. Він був розроблений Андерсом Хейлсбергом (розробником C#) у Microsoft, скомпільований у JavaScript. Іншими словами, це JavaScript плюс деякі додаткові функції. Він пропонує всі функції JavaScript і додатковий рівень на

додаток до них: систему типів TypeScript. Наприклад, JavaScript надає такі мовні примітиви, як рядок і число, але не перевіряє, чи ви їх постійно призначали.

TypeScript робить. Це означає, що існуючий робочий код JavaScript також є кодом TypeScript. Основна перевага TypeScript полягає в тому, що він може висвітлювати неочікувану поведінку у вашому коді, знижуючи ймовірність помилок. TypeScript знає мову JavaScript і в багатьох випадках генеруватиме типи змінних самостійно. Наприклад, під час створення змінної та присвоєння їй певного значення TypeScript використовуватиме значення як свій тип.

Розуміючи, як працює JavaScript, TypeScript може створити систему типів, яка приймає код JavaScript, але має типи. Це пропонує систему типів без необхідності додавати додаткові символи, щоб зробити типи явними у вашому коді

2.2.1 Переваги

JavaScript і TypeScript дуже схожі, але є одна важлива відмінність. Ключова відмінність між JavaScript і TypeScript полягає в тому, що JavaScript не має системи типів. У JavaScript змінні можуть безсистемно змінювати форму, тоді як TypeScript у строгому режимі це забороняє. Це полегшує керування та обслуговування TypeScript, особливо з великою кодовою базою.

Система типів TypeScript має незначний вплив на те, як змінні посилаються та оголошуються, але вона має величезний вплив на зручність обслуговування та узгодженість. Наприклад, ось як оголосити числові та текстові змінні в JavaScript(Рис 2.2)

```
let foo = 1
let bar = "text"
bar = 123  // allowed in JavaScript
```

Рис 2.2 Оголошення змінних

Однак використання числових і рядкових типів дозволяє TypeScript застосовувати правило, згідно з яким: Якщо змінну оголошено як певний тип, їй не дозволяється довільно змінювати її тип пізніше в програмі. Це правило є причиною того, що строгий TypeScript не дозволяє панелі змінних переключатися з текстового рядка на число. JavaScript це дозволить. Запровадження системи типів не здається великою проблемою, але воно має великий вплив на те, як керувати великими програмами.

TypeScript не є заміною для JavaScript. це просто більш повнофункціональний і технічно надійний спосіб написання JavaScript. Щоб запустити програму, написану на TypeScript, першим кроком є компіляція коду в JavaScript. TypeScript — це просто кращий спосіб написання коду JavaScript, сумісного з ECMAScript, щоб забезпечити взаємодію між веб-браузерами.
Можна розглядати TypeScript як просто генератор JavaScript. TypeScript допомагає створювати код JavaScript, який можна запускати в будь-якому середовищі, яке підтримує стандарт JavaScript.

Порівняно TypeScript і JavaScript
З точки зору функцій, ось 10 суттєвих відмінностей між JavaScript і TypeScript:

- може бути строго типізований, а JavaScript лише динамічно типізований.
- легше читати та підтримувати.
- підтримує абстракцію через інтерфейси, а JavaScript — ні.
- дозволяє розробникам коментувати код за допомогою декораторів, тоді як JavaScript цього не робить.
- підтримує можливість модульувати та організовувати компоненти за допомогою просторів імен, що не підтримується в JavaScript.
- є більш виразним, ніж JavaScript, завдяки використанню синтаксичних елементів, таких як необов’язкові та іменовані параметри.
- підтримує генерики та функцію виведення типу,
- TypeScript IDE мають більше функцій, оскільки легше створювати плагіни та інструменти для мови зі статичною типізацією.

Код TypeScript легше налагоджувати, оскільки кодова база розширюється, оскільки помилки типу можна виявити під час компіляції, а не під час виконання.

Платформи, керовані JavaScript, такі як NodeJS на сервері та ReactJS на стороні клієнта, продовжують набирати популярність. Можливість писати код на TypeScript і перетворювати його на JavaScript є однією з причин того, чому рівень впровадження обох мов продовжує зростати.

2.3 Visual Studio Code

Visual Studio Code — це безкоштовний, легкий, але потужний редактор вихідного коду, який працює на робочому столі та в Інтернеті та доступний для ОС Windows, macOS, Linux і Raspberry Pi. Він поставляється з вбудованою підтримкою JavaScript, TypeScript і Node.js і має багату екосистему розширень для інших мов програмування (таких як C++, C#, Java, Python, PHP і Go), середовища виконання (таких як .NET і Unity), середовища (такі як Docker і Kubernetes) і хмари (такі як Amazon Web Services, Microsoft Azure і Google Cloud Platform).

Крім загальної ідеї легкості та швидкого запуску, Visual Studio Code має доповнення коду IntelliSense для змінних, методів та імпортованих модулів; графічне налагодження; лінтування, редагування кількома курсорами, підказки параметрів та інші потужні функції редагування; швидка навігація та рефакторинг коду; і вбудований контроль вихідного коду, включаючи підтримку Git. Більшу частину цього було адаптовано з технології Visual Studio.

Власне код Visual Studio створено з використанням оболонки Electron, Node.js, TypeScript і протоколу Language Server Protocol і оновлюється щомісяця. Багато розширень оновлюються так часто, як це необхідно. Різноманітність підтримки різниться в різних мовах програмування та їхніх розширеннях, починаючи від простого підсвічування синтаксису та підбору дужок до налагодження та рефакторингу. Ви можете додати базову підтримку

вашої улюбленої мови за допомогою розфарбовувачів TextMate, якщо мовний сервер недоступний.

Код у репозиторії коду Visual Studio є відкритим вихідним кодом за ліцензією MIT. Сам продукт Visual Studio Code постачається за стандартною ліцензією на продукт Microsoft, оскільки він має невеликий відсоток індивідуальних налаштувань Microsoft. Це безкоштовно, незважаючи на комерційну ліцензію.

2.3.1 Огляд інтерфейсу

Основний інтерфейс Visual Studio Code пропонує кілька ключових елементів, які сприяють продуктивності та ефективності розробки:

Редактор вікна коду: Це головна область інтерфейсу, де ви можете писати код вашої програми. Редактор вікна коду включає в себе функції підсвічування синтаксису, автодоповнення, відступи та багато іншого, що полегшує редагування коду (Рис 2.3).

Бічна панель: Бічна панель містить різноманітні панелі інструментів, такі як панель файлів, панель розширень, панель пошуку та інші. Вона дозволяє швидкий доступ до важливих функцій та налаштувань (Рис 2.3).

Панель розширень: Visual Studio Code має широкий вибір розширень, які додають додаткові функції та можливості до редактора. Панель розширень дозволяє керувати встановленими розширеннями, шукати нові розширення та налаштовувати (Рис 2.3).

Верхнє меню: Верхнє меню містить набір команд та опцій, які охоплюють всі аспекти розробки. Тут ви знайдете можливості редагування, перегляду, відладки, виконання та інші (Рис 2.3).

Рядок стану: Рядок стану, розташований у нижній частині вікна, надає корисну інформацію про ваш проект та поточну дію. Він може включати індикатори стану файлу, інформацію про виконання коду, помилки та інші повідомлення (Рис 2.3).

Панель налаштувань: в Visual Studio Code надає зручний спосіб налаштування різних аспектів редактора, розширень та поведінки. Вона дозволяє вам змінювати параметри, настроювати різні опції та адаптувати редактор до своїх потреб. (Рис 2.4)

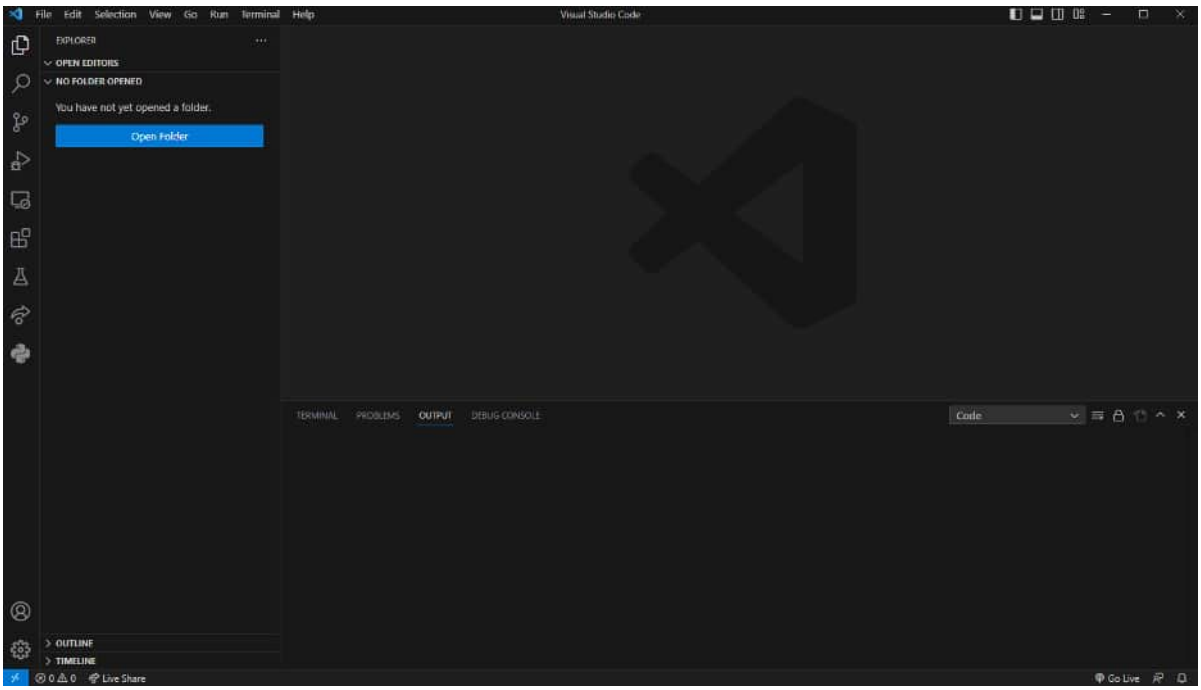

Рис 2.3 Інтерфейс Visual Studio Code

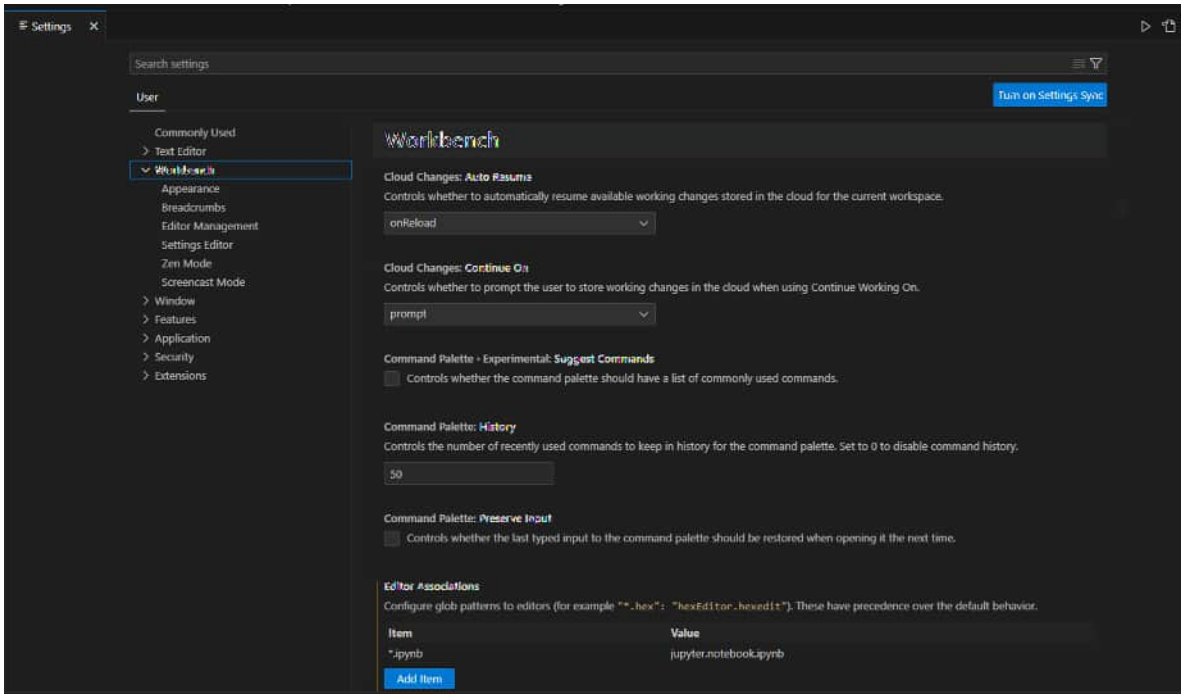

Рис 2.4 Інтерфейс налаштування Visual Studio Code

2.3.2 Розширеня

Швидкий пошук у Visual Studio Code Marketplace дає приблизно 38 000 результатів із підтримкою сотень мов програмування. Керувати розширеннями можна з Marketplace, з бічної панелі розширень у VS Code та з панелі команд VS Code (Рис 2.5)

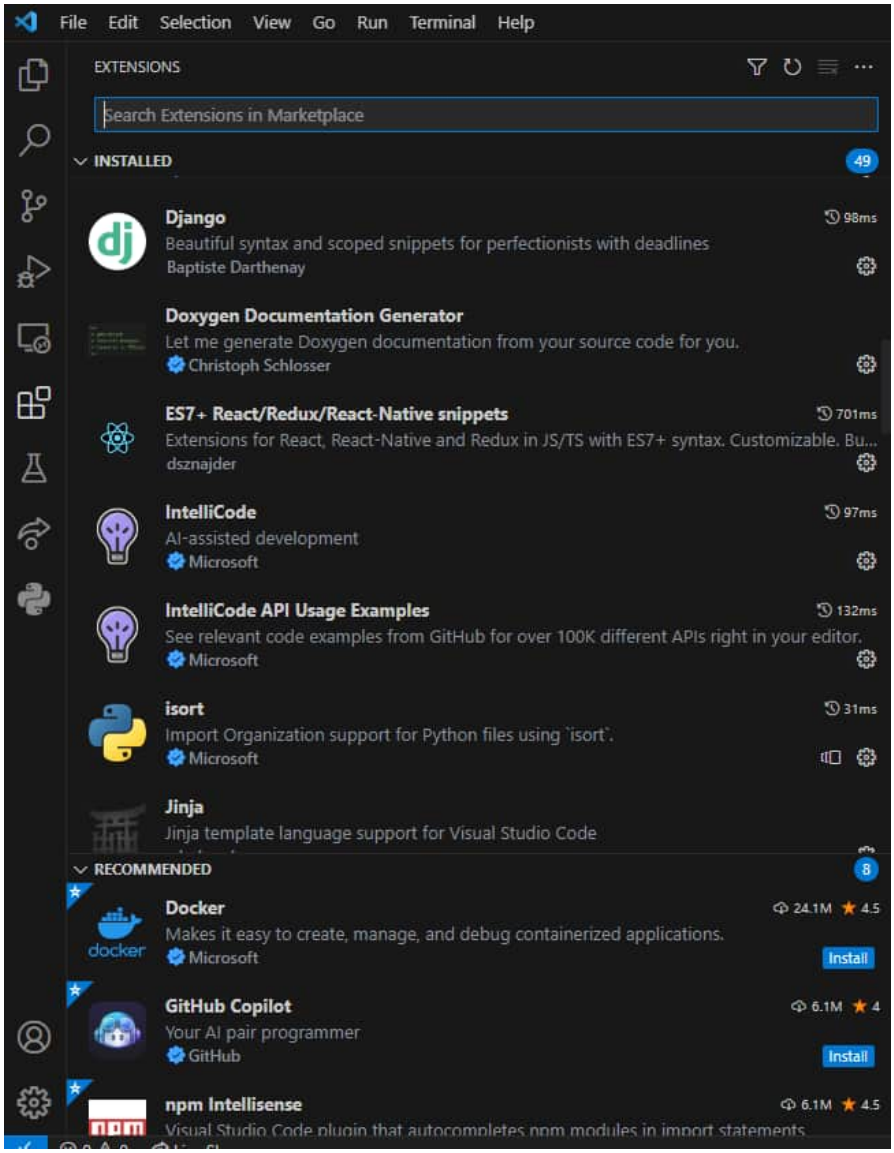

Рис 2.5 Бічна панель керування розширеннями

Найкраще розширення, для Python, було встановлено майже 60 мільйонів. Окрім підтримки кодування Python 3.7+, налагодження та рефакторизму, розширення Python автоматично встановлюватиме Pylance (IntelliSense) і Jupyter (ноутбук) розширення.

2.4 Платформа Node.js

Node.js — це кросплатформне середовище виконання з відкритим кодом для виконання коду JavaScript поза браузером. NodeJS не є фреймворком і не є мовою програмування. Більшість людей плутають і розуміють, що це фреймворк або мова програмування. Ми часто використовуємо Node.js для створення внутрішніх служб, таких як API, наприклад Web App або Mobile App.

Node.js = середовище виконання + бібліотека JavaScript. Особливості NodeJS. Існують також інші мови програмування, які ми можемо використовувати для створення. На наступній діаграмі (Рис 2.6) зображено деякі важливі частини Node.js, які є корисними та допомагають нам краще зрозуміти його серверних служб, тому я збираюся пояснити, чим відрізняється Node.js.

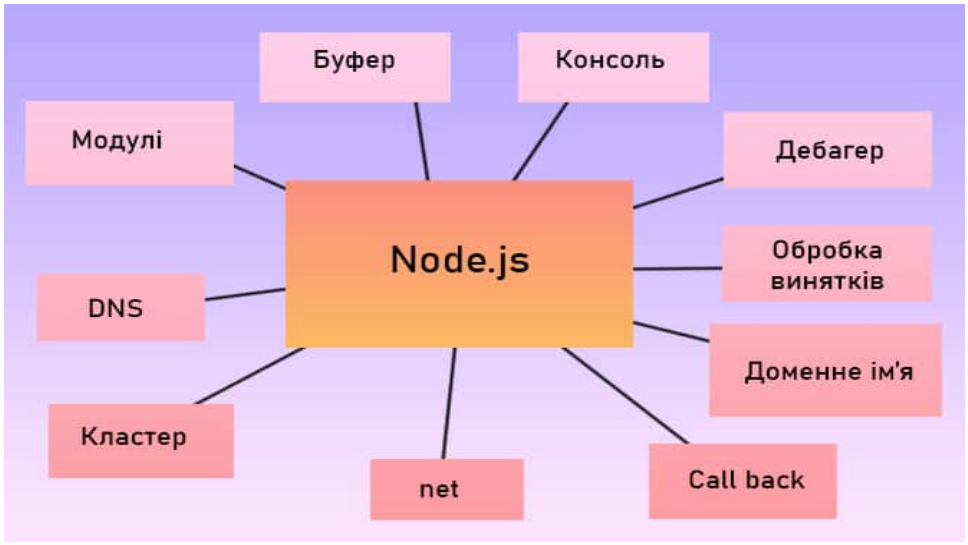

Рис 2.6 Діаграма платформи Node.js

Його легко вивчати, його можна використовувати для створення прототипів і гнучкої розробки. Він використовує JavaScript усюди, тому програмісту JavaScript легко створювати серверні служби за допомогою Node.js

- Вихідний код чистіший і послідовніший.
- Велика екосистема для бібліотеки з відкритим кодом.
- Він має асинхронний або неблокуючий характер.

2.4.1 Переваги NodeJS

Переваги NodeJS

- Легка масштабованість: розробники вважають за краще використовувати Node.js, оскільки він легко масштабує програму як у горизонтальному, так і у вертикальному напрямках. Ми також можемо додати додаткові ресурси під час масштабування програми.
- Веб-програми в режимі реального часу: якщо ви створюєте веб-програму, ви також можете використовувати PHP, і це займе стільки ж часу, коли ви використовуєте Node.js. Але якщо я говорю про створення програм для чату чи програм для ігор, Node.js набагато краще через швидшу синхронізацію. Крім того, цикл подій дозволяє уникнути перевантаження HTTP для розробки Node.js.
- Fast Suite: NodeJs працює на двигуні V8, розробленому Google. Цикл подій у NodeJs обробляє всі асинхронні операції, тому NodeJs працює як швидкий пакет, і всі операції можна виконувати швидко, як-от читання або запис у базі даних, мережеве підключення або файлова система
- Легко вивчати та кодувати: NodeJs легко вивчати та кодувати, оскільки він використовує JavaScript. Якщо ви фронтенд-розробник і добре знаєте JavaScript, ви можете легко вивчити та створити програму на NodeJS

NodeJS також забезпечує кешування одного модуля. Коли надходить будь-який запит для першого модуля, він кешується в пам’яті програми, тому вам не потрібно повторно виконувати код.

У NodeJs HTTP-запит і відповідь розглядаються як дві окремі події. Вони являють собою потік даних, тому коли ви обробляєте файл під час завантаження, це зменшить загальний час і зробить його швидшим, коли дані представлені у формі передачі. Це також дозволяє передавати аудіо- та відеофайли з блискавичною швидкістю.

2.4.2 Пакетний менеджер Npm

В мережі npm називають по різному, хоча абревіатура розшифровується як «Менеджер пакетів вузлів».

npm — це менеджер пакетів для проектів Node.js, доступний для загального використання. Проекти, доступні в реєстрі npm, називаються «пакетами». Він дозволяє нам легко використовувати код, написаний іншими, без необхідності писати його самим під час розробки. Реєстр npm містить понад 1,3 мільйона пакетів, які використовують понад 11 мільйонів розробників у всьому світі. (Ми поговоримо більше про пакети пізніше в цьому посібнику.) Розміщуйте особисті проекти за допомогою Hobby Tier

Навіщо використовувати npm?

Він дає змогу встановлювати бібліотеки, фреймворки та інші засоби розробки для вашого проекту, подібно до встановлення мобільного додатку з магазину програм. Надає доступ до безпечних проектів Node.js для розробки. Це допомагає пришвидшити етап розробки за допомогою попередньо створених залежностей.

Використання команд npm не потребує багато навчання, оскільки їх легко зрозуміти та використовувати. Далі ми поговоримо про інтерфейс командного рядка npm.

Інтерфейс командного рядка npm (CLI)

Інтерфейс командного рядка для npm використовується для запуску різних команд, таких як встановлення та видалення пакетів, перевірка версії npm, запуск сценаріїв пакетів, створення файлу package.json і багато іншого.
Основні команди npm:

1. **npm update**: Ця команда оновлює всі пакети до їх останніх версій, якщо доступні нові версії. Вона перевіряє ваш файл **package.json** на наявність описаних пакетів та оновлює їх до останніх стабільних версій.
2. **npm restart**: Ця команда перезапускає ваш застосунок після внесення змін. Вона може бути корисною, коли ви змінили код або конфігурацію проекту і хочете перезавантажити його для застосування змін.

3. **npm start**: Ця команда запускає ваш застосунок або сервер. Ви визначаєте, який сценарій **start** використовувати в файлі **package.json**. Це часто використовується для запуску веб-сервера або іншої програми.
4. **npm stop**: Ця команда зупиняє запущений застосунок або сервер, якщо він використовується. Вона зазвичай використовується для зупинки веб-сервера або іншої програми, яку ви запустили за допомогою **npm start**.
5. **npm version**: Ця команда дозволяє встановити або оновити версію вашого проекту. Вона може бути використана для автоматичного оновлення номера версії в файлі **package.json** або для створення нового тегу в системі контролю версій.
6. **npm publish**: Ця команда дозволяє опублікувати ваш пакет в репозиторії npm, щоб інші користувачі могли його встановити. Перед використанням цієї команди вам потрібно мати обліковий запис npm і зареєструватись в ньому.

Використання цих команд дозволить ефективно керувати залежностями, виконувати розгортання застосунків та керувати версіями проекту в середовищі Node.js.

Зазвичай окремо інсталювати Npm не потрібно, він йде у комплекті інсталяційного пакету Node.js

2.5 Google Firebase

Що таке Google Firebase?

Google Firebase — це набір хмарних інструментів розробки, які допомагають розробникам мобільних додатків створювати, розгортати та масштабувати свої додатки.

Firebase надає різноманітні функції, зокрема такі:

- Аутентифікація. Firebase надає користувачам безпечний і простий спосіб входу в свою програму. Розробники можуть використовувати

автентифікацію Firebase для підтримки входу електронною поштою та паролем, входу в Google, Facebook тощо.

- База даних у реальному часі. Firebase Realtime Database — це хмарна база даних NoSQL, яка дозволяє організаціям зберігати та синхронізувати дані в реальному часі на всіх пристроях своїх користувачів. Це дозволяє легко створювати програми, які завжди оновлюються, навіть коли користувачі перебувають у режимі офлайн.
- Хмарний обмін повідомленнями. Firebase Cloud Messaging (FCM) — це служба, яка дозволяє компаніям надсилати повідомлення на пристрої своїх користувачів, навіть якщо вони не використовують додаток. Розробники можуть використовувати FCM для надсилання push-сповіщень, оновлення вмісту програми тощо.

Firebase пропонує безкоштовний план із 1 ГБ пам’яті бази даних у реальному часі та один план платної підписки, який включає: Усі дані Firebase шифруються під час передачі та передачі. Контроль доступу на основі ролей. Firebase використовує керування доступом на основі ролей (RBAC), щоб забезпечити детальний контроль над тим, хто може отриматидоступ до даних програми.

Візуалізація де видно що саме виконує Firebase у порівнянні з окремими хостингами (Рис 2.7)

Журнал аудиту реєструє всі доступи до даних, щоб компанії могли відстежувати, хто і коли отримував доступ до даних програми.

Керування доступом на основі ролей у Google Firebase забезпечує детальний контроль доступу до даних.

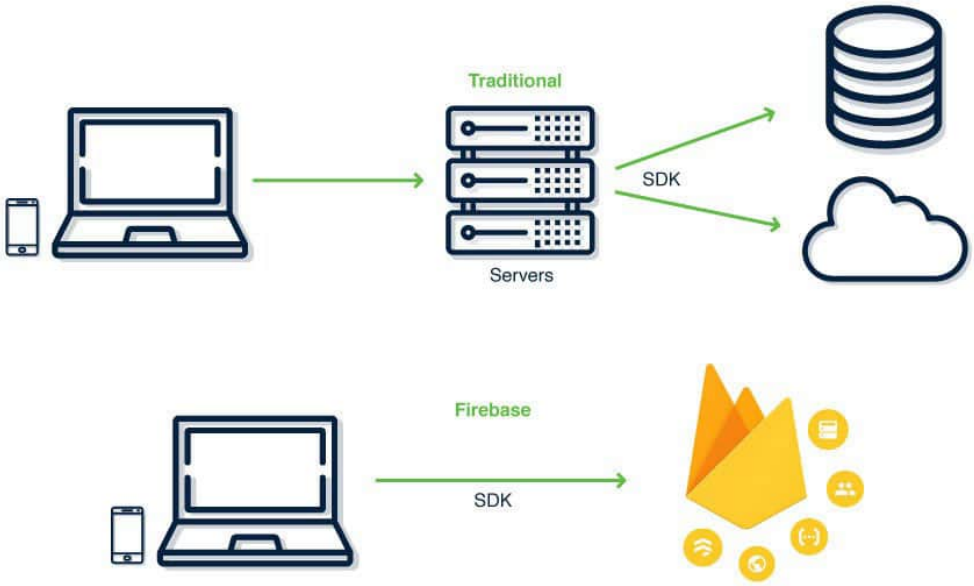

Рис 2.7 Діаграма Firebase

Firebase використовується для автентифікації користувачів у таких самописаних програмах та популярних сервісах База даних Firebase Realtime використовується для зберігання та синхронізації даних у режимі реального часу

2.6 Фреймворк React

React, також відомий як React.js, — це бібліотека JavaScript з відкритим кодом, розроблена Facebook, яка використовується для створення інтерфейсів користувача або компонентів інтерфейсу користувача (Рис 2.8).

React, звичайно, не єдина бібліотека інтерфейсу користувача. Preact, Vue, Angular, Svelte, Lit і багато інших також чудово підходять для створення інтерфейсів з багаторазових елементів. Враховуючи популярність React, варто ознайомитися з тим, як він працює, оскільки ми будемо використовувати його.

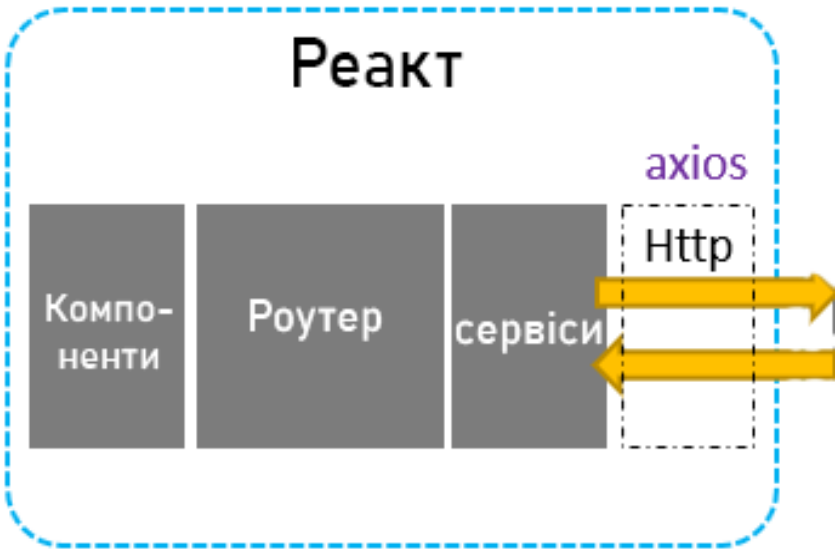

Рис 2.8 Діаграма роботи React Framework

Коли інтерфейсні розробники говорять про код, найчастіше це стосується розробки інтерфейсів для Інтернету. І те, як ми думаємо про композицію інтерфейсу, полягає в таких елементах, як кнопки, списки, навігація тощо. React забезпечує оптимізований і спрощений спосіб вираження інтерфейсів у цих елементах. Це також допомагає створювати складні та складні інтерфейси, організовуючи ваш інтерфейс за трьома ключовими поняттями — компонентами, атрибутами та станом.

Оскільки React зосереджений на композиції, він може ідеально відображати елементи вашої системи дизайну. Отже, по суті, проектування для React насправді винагороджує вас за модульне мислення. Це дає змогу розробляти окремі компоненти перед тим, як об’єднувати сторінку чи перегляд, щоб ви повністю зрозуміли обсяг і призначення кожного компонента — процес, який називають компонентизацією.

Хуки — новий спосіб використання стану та інших функцій React без написання класу.

JSX — це розширення JavaScript, яке вбудовує шаблон HTML у JS за допомогою XML-подібного синтаксису. Він призначений для перетворення в

дійсний JavaScript, хоча семантика цього перетворення залежить від реалізації. JSX здобув популярність завдяки бібліотеці React, але з того часу з'явився й інші реалізації.

Як працює JSX

Компоненти, властивості та стан — це три ключові поняття в React. Практично все, що ви збираєтеся бачити або робити в React, можна класифікувати принаймні за одним із цих ключових понять, і ось короткий огляд цих ключових понять:

Компоненти є будівельними блоками будь-якої програми React. Вони схожі на функції JavaScript, які приймають довільний вхід (Props) і повертають елементи React, що описують те, що має відображатися на екрані.

Перше, що потрібно зрозуміти, це те, що все на екрані програми React є частиною компонента. По суті, додаток React — це лише компоненти в компонентах у компонентах. Отже, розробники не створюють сторінки в React; вони будують компоненти. Компоненти дозволяють розділити інтерфейс користувача на незалежні частини, які можна багаторазово використовувати.

2.7 **Фреймворк axios**

Axios – це досить популярна бібліотека JavaScript, яка використовується для виконання HTTP-запитів з браузера або середовища Node.js. Вона забезпечує простий та зрозумілий інтерфейс для взаємодії з веб-серверами та отримання або відправки даних за допомогою протоколу HTTP.

Основні особливості та переваги Axios:

1. **Простота використання**: Axios пропонує простий та лаконічний API для виконання HTTP-запитів. Вона надає методи для різних типів запитів, таких як GET, POST, PUT, DELETE, та інших.
2. **Підтримка промісів**: Axios використовує проміси, що дозволяє зручно працювати з асинхронними операціями та обробляти їхні результати або помилки.

3. **Підтримка перехоплювачів запитів та відповідей**: Axios дозволяє встановлювати перехоплювачі для обробки запитів та відповідей. Це може бути корисно для встановлення заголовків, обробки помилок або виконання певних дій перед або після виконання запиту.
4. **Автоматична серіалізація та десеріалізація даних**: Axios може автоматично серіалізувати дані в JSON-формат перед відправкою і десеріалізувати відповідь JSON-даних в об'єкт JavaScript.
5. **Підтримка відміни запитів**: Axios надає можливість відміняти запити за допомогою об'єктів CancelToken. Це корисно, коли потрібно припинити виконання запиту або уникнути обробки його відповіді.
6. **Широкий спектр функціональних можливостей**: Axios підтримує такі функціональні можливості, як завантаження та вивантаження файлів, встановлення таймаутів для запитів, керування заголовками та багато іншого.

Axios є популярним вибором для виконання HTTP-запитів у JavaScript-розробці, оскільки вона проста використовувати, надійна та має багато корисних функцій.

2.8 Фреймворк bootstrap4

Bootstrap 4 — це популярний і широко використовуваний інтерфейсний фреймворк для розробки адаптивних і орієнтованих на мобільні пристрої веб-сайтів і веб-додатків. Він надає набір компонентів CSS і JavaScript, які можна легко інтегрувати у ваш проект, заощаджуючи ваш час і зусилля на створенні адаптивного макета та реалізації загальних елементів інтерфейсу користувача.

Однією з ключових особливостей Bootstrap 4 є його орієнтація на мобільний дизайн. Це гарантує, що ваш веб-сайт або програма виглядатимуть чудово та добре функціонуватимуть на мобільних пристроях, забезпечуючи безперебійну роботу на великих екранах. Цей підхід досягається за допомогою гнучкої системи сітки, яка адаптується до різних розмірів і орієнтацій екрана.

Система сітки в Bootstrap 4 заснована на макеті з 12 стовпців, що дозволяє створювати адаптивні дизайни, розділяючи доступний простір на кілька стовпців. Ви можете легко визначити макет для різних розмірів екрана, використовуючи попередньо визначені класи CSS, що спрощує створення адаптивного та плавного дизайну (Рис 2.9) .

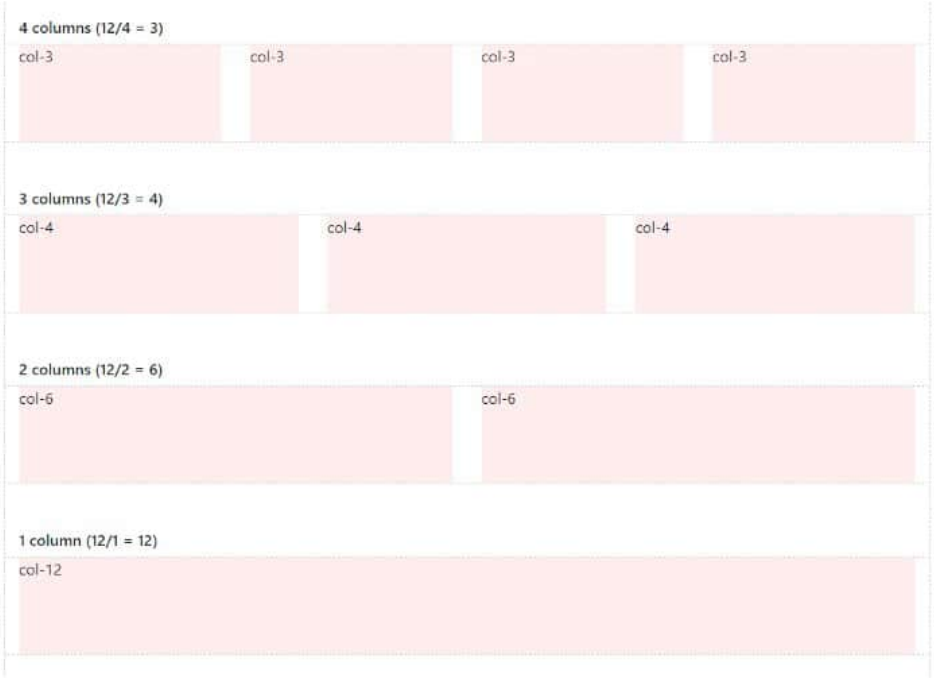

Рис 2.9 Приклад стовпців

Bootstrap 4 також містить широкий набір готових до стилів компонентів, таких як кнопки, форми, панелі навігації, каруселі, моди тощо. Ці компоненти розроблено таким чином, щоб їх можна було налаштовувати та прості у використанні, а також мати постійний і сучасний вигляд. Використовуючи ці компоненти, ви можете швидко додавати інтерактивні елементи до свого проекту без необхідності писати спеціальний код CSS або JavaScript.

Крім того, Bootstrap 4 надає набір службових класів, які можна використовувати для застосування загальних стилів і модифікацій до ваших елементів. Ці корисні класи дозволяють легко налаштовувати поля, відступи, вирівнювання, кольори та інші візуальні властивості, даючи вам більше контролю над зовнішнім виглядом вашого веб-сайту чи програми.

Іншою важливою особливістю Bootstrap 4 є його велика документація та підтримка спільноти. Офіційна документація Bootstrap містить детальну інформацію про те, як використовувати кожен компонент, а також приклади та фрагменти коду, які допоможуть вам швидко розпочати роботу. Також є численні онлайн-ресурси, навчальні посібники та форуми, де ви можете знайти відповіді на свої запитання та повчитися в інших розробників.

Підсумовуючи, Bootstrap 4 — це потужний і гнучкий інтерфейсний фреймворк, який спрощує процес створення адаптивних і орієнтованих на мобільні пристрої веб-сайтів і веб-додатків. Завдяки сітковій системі, попередньо оформленим компонентам, службовим класам і вичерпній документації Bootstrap 4 дає змогу розробникам легко створювати сучасні та візуально привабливі проекти.

2.9 Програма FFmpeg

FFmpeg — це потужна та універсальна мультимедійна платформа, яка надає інструмент командного рядка для обробки аудіо- та відеофайлів. Він широко використовується в різних програмах і проектах для таких завдань, як кодування відео, декодування, перекодування, потокова передача та обробка аудіо (Рис 2.10).

Рис 2.10 Діаграма праці FFmpeg

Однією з головних сильних сторін FFmpeg є широка підтримка форматів. Він може читати та записувати широкий спектр мультимедійних форматів, включаючи популярні контейнерні формати, такі як MP4, AVI та MKV, а також аудіо- та відеокодеки, такі як H.264, AAC, MP3 та багато інших. Це робить FFmpeg цінним інструментом для роботи з різними мультимедійними форматами та конвертації між ними.

Він надає повний набір опцій і параметрів командного рядка, які дозволяють виконувати різноманітні завдання. FFmpeg використовують для маніпулювання та редагування аудіо- та відеофайлів, застосування фільтрів і ефектів, налаштування таких параметрів, як роздільна здатність, частота кадрів і бітрейт, вилучення аудіо з відео, створення ескізів, об’єднання файлів і багато іншого. Гнучкість і багатство функцій роблять FFmpeg придатним для широкого діапазону потреб обробки мультимедіа.

Іншим ключовим аспектом FFmpeg є його висока продуктивність і ефективність. Він розроблений для використання апаратного прискорення та можливостей багатопоточності, що забезпечує швидшу обробку та краще використання системних ресурсів. Це особливо важливо під час роботи з великими мультимедійними файлами або файлами високої роздільної здатності, оскільки FFmpeg може ефективно їх обробляти з мінімальними витратами.

Він також є відкритим додатком з вихідним кодом із живою й активною спільнотою. Це означає, що в мережі доступна велика кількість документації, навчальних посібників і підтримки користувачів. Спільнота постійно робить внесок у розвиток і вдосконалення FFmpeg, гарантуючи, що він залишається в курсі останніх мультимедійних технологій і стандартів.

На додаток до інструменту командного рядка, FFmpeg також надає бібліотеки та API, які дозволяють розробникам інтегрувати мультимедійні функції у власні програми. Ці бібліотеки забезпечують вищий рівень контролю та налаштування, дозволяючи розробникам створювати мультимедійні рішення, адаптовані до їхніх конкретних вимог.

2.10 Фреймворк express

Express — це швидка та мінімалістична структура веб-додатків для Node.js. Це один із найпопулярніших фреймворків для створення веб-серверів і API в екосистемі Node.js. Express забезпечує простий і гнучкий спосіб обробки HTTP-запитів, визначення маршрутів і керування проміжним програмним забезпеченням.

Маршрутизація: Express дозволяє визначати маршрути для обробки різних методів HTTP (GET, POST, PUT, DELETE тощо) і шаблонів URL-адрес. Ви можете вказати функцію обробки маршруту, яка виконується, коли запит відповідає певному маршруту.

Проміжне програмне забезпечення: функції проміжного програмного забезпечення є основною концепцією Express. Це функції, які мають доступ до об’єктів запиту та відповіді та можуть виконувати такі завдання, як розбір тіл запиту, обробка автентифікації, журналювання тощо. Функції проміжного програмного забезпечення можна використовувати глобально для всіх маршрутів або для певних маршрутів.

Запит і відповідь: Express надає об’єкти запиту та відповіді, які інкапсулюють вхідний запит HTTP та вихідну відповідь HTTP відповідно. Ці об’єкти надають різні властивості та методи для маніпулювання та доступу до даних запиту та відповіді.

Інтеграція з іншими бібліотеками: Express можна легко інтегрувати з іншими бібліотеками та фреймворками Node.js. Наприклад, ви можете використовувати Express з такими базами даних, як MongoDB, MySQL або PostgreSQL, або поєднувати його з зовнішніми фреймворками, такими як React або Angular.

Express відомий своєю простотою, гнучкістю та продуктивністю. Він забезпечує легку основу для створення веб-додатків і API, дозволяючи розробникам зосередитися на написанні бізнес-логіки, а не працювати з деталями HTTP низького рівня.

2.11 Постановка завдання

2.11.1 Вхідні дані для виконання проекту

- Вимоги користувачів щодо функціональності веб-сайту для завантаження та перегляду відео.
    - Вимоги можуть включати можливість створення облікового запису користувача, завантаження відео, перегляду відео, пошуку та фільтрації відео, коментування та оцінювання відео.
- Технічні вимоги до веб-сайту, такі як платформа, мови програмування, база даних тощо.
    - Використання Node.js як середовища виконання на серверній стороні для створення веб-додатків.
    - Використання React як фреймворку для створення користувацького інтерфейсу на клієнтській стороні.
    - Використання JavaScript як основної мови програмування для розробки функціональності веб-сайту.
    - Використання HTML та CSS для створення розмітки та стилізації веб-сторінок.
- Зовнішні API або сервіси, які можна використовувати для завантаження або обробки відео.
    - Використання Firebase у якості бази даних та методу авторизації

2.11.2 Що планується отримати в результаті виконання проекту

- Функціональний веб-сайт з можливістю реєстрації:
    - Користувачі зможуть створювати облікові записи на веб-сайті шляхом реєстрації.
    - Реєстрація може включати введення обов'язкових полів, таких як ім'я користувача, електронна пошта та пароль.
- Підписки на користувачів:
    - Користувачі зможуть підписуватися на інших користувачів, щоб отримувати оновлення про їхні нові відео.

    - Підписки можуть бути реалізовані через систему сповіщень або електронну пошту.
- Коментарі, лайки та дизлайки:
    - Користувачі зможуть залишати коментарі під відео, виражаючи свою думку або висловлюючи питання.
    - Користувачі зможуть виразити свою симпатію до відео шляхом постановки лайка або невподобства відео за допомогою дизлайка.
    - Кількість лайків та дизлайків може відображатися на веб-сайті, дозволяючи користувачам оцінювати популярність відео.
- Категорії відео:
    - Відео можуть бути класифіковані за різними категоріями, такими як музика, спорт, розваги, освіта тощо.
    - Користувачі зможуть шукати відео за категоріями або використовувати фільтри для точнішого пошуку.

2.11.3 Архітектура рішення:

- Клієнтська сторона:
    - Використання React для розробки користувацького інтерфейсу на клієнтській стороні.
    - Створення компонентів, які відповідають за відображення відео, коментарів, кнопок лайків/дизлайків та категорій.
    - Використання JavaScript для реалізації функціональності, такої як відправка коментарів, встановлення лайків/дизлайків та фільтрація відео за категоріями.
- Серверна сторона:
    - Використання Node.js для створення серверної частини додатку.
    - Забезпечення API для реєстрації користувачів, завантаження відео, отримання списку відео, коментарів, категорій та взаємодії з ними.
    - Збереження користувацьких даних у базі даних Firebase
    - Інтеграція з FFmpeg для генерації рисунків відео.

2.11. 4. Вибір і обгрунтування засобів та технологій, використаних для виконання проекту

Node.js: Використання Node.js як середовища виконання на серверній стороні є доцільним, оскільки воно дозволяє розробляти швидкі та ефективні веб-додатки за допомогою JavaScript. Node.js також забезпечує високу продуктивність та масштабованість, що дуже важливо для веб-сайту з обробкою великого обсягу відеофайлів та взаємодії з користувачами.

React: Використання React як фреймворку на клієнтській стороні дозволяє побудувати інтерактивний та швидкодіючий користувацький інтерфейс. React використовує компонентний підхід, що спрощує розробку та підтримку коду, а також дозволяє ефективно керувати станом додатку. Це особливо важливо для створення зручного та відзивчивого інтерфейсу веб-сайту для завантаження та перегляду відео.

JavaScript: Використання JavaScript як основної мови програмування дозволяє розробляти різноманітну функціональність веб-сайту. JavaScript є широко підтримуваною та розповсюдженою мовою, яка має багато корисних бібліотек та інструментів для розробки веб-додатків. Вона також підтримує асинхронний підхід, що дозволяє ефективно працювати зі збереженням та обробкою великого обсягу даних, що потрібні для веб-сайту з відео.

HTML та CSS: Використання HTML та CSS для створення розмітки та стилізації веб-сторінок є стандартними підходами в веб-розробці. HTML забезпечує структуру сторінки, а CSS дозволяє змінювати її зовнішній вигляд. Вони допоможуть створити зрозумілий та привабливий інтерфейс для користувачів, що є важливим аспектом веб-сайту для завантаження та перегляду відео.

База даних: Для зберігання і керування користувацькими даними, відеофайлами, коментарями та іншою інформацією, пов'язаною з веб-сайтом, можна використовувати базу даних, таку як Firebase. Обрана база даних повинна бути надійною, масштабованою та забезпечувати ефективну роботу з великим обсягом даних.

Ці технології та засоби були вибрані з огляду на їх ефективність, популярність в галузі веб-розробки та здатність задовольняти вимоги проекту щодо функціональності, продуктивності та безпеки. Вони допоможуть забезпечити стабільну та ефективну роботу веб-сайту для завантаження та перегляду користувацького відео.

РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ ВЕБ-САЙТУ ДЛЯ ЗАВАНТАЖЕННЯ ТА ПЕРЕГЛЯДУ КОРИСТУВАЦЬКОГО ВІДЕО

3.1 Реалізація серверної частини

3.1.1 Структура серверу

Структура серверу має наступний вид (Рис 3.1)

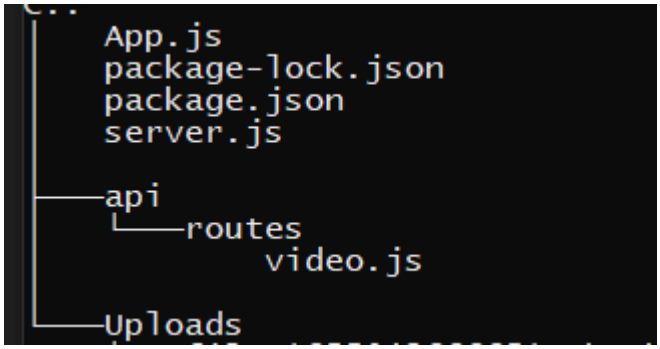

Рис 3.1 Структура серверу

Опис структури:

- server.js: Це основний файл сервера, де відбувається налаштування і запуск серверу. В ньому ви визначаєте маршрути, обробляються HTTP-запити.
- package-lock.json та package.json: Ці файли використовуються для керування залежностями сервера.
- package-lock.json містить точні версії всіх залежностей, встановлених у проекті, тоді як package.json містить перелік залежностей та інформацію про проект.
- App.js: Цей файл це частина фреймворку React який ми використовуємо на сервері. Він містить код, пов'язаний з логікою вашого сервера, обробкою маршрутів та керуванням запитами.
- api/routes/video.js: Це файл, який містить основні маршрути визначені за допомогою фреймворку Express.js. Основна логіка маршрутів пов'язана з завантаженням відео і створенням ескізу відео. Далі ми розберемо його

- Uploads (папка): Це папка, де зберігаються завантажені відео. Коли користувач завантажує відео на веб-сайт, ця папка буде використовуватись для збереження цього відео та його ескізів.

3.1.2 Серверний скрипт та Корневий скрипт React

У серверному скрипті використовується модуль http для створення HTTP-сервера, який слухатиме запити із веб-браузера. Ось опис кожної функції в цьому скрипті:

require("http"): Тут ми підключаємо вбудований модуль http, який дозволяє створювати HTTP-сервери та обробляти HTTP-запити.

require("./App"): Цей рядок підключає модуль, який експортує додаткову логіку серверної частини, що міститься у файлі "App.js".

const port = process.env.PORT || 5000;: Тут ми встановлюємо порт, на якому сервер буде слухати вхідні запити. Він спочатку перевіряє значення змінної оточення process.env.PORT і якщо таке значення існує, використовує його. Якщо змінна оточення process.env.PORT не встановлена, буде використовуватися порт 5000 який ми вказали.

const server = http.createServer(app);: В цій стрічці ми створюємо сам HTTP-сервер за допомогою функції createServer з модуля http. В якості параметра передається app, що є експортованим об'єктом з файлу "App.js". Це означає, що сервер буде використовувати логіку, визначену в "App.js", для обробки вхідних запитів.

server.listen(port, () => { console.log(Listening on port ${port}!); });: Ця функція викликається для запуску сервера на вказаному порту. Вона встановлює слухача на порту port і виводить повідомлення в консоль, що сервер запущений і слухає вказаний порт.

Самий код відповідно (Рис 3.2)

```
const http = require("http");
const app = require("./App");

const port = process.env.PORT || 5000;

const server = http.createServer(app);

server.listen(port, () => {
  console.log(`Listening on port ${port}!`);
});

```

Рис 3.2 Лістинг серверного коду

Весь цей серверний скрипт відповідає за створення HTTP-сервера і налаштування його на прослуховування вхідних запитів на вказаному порту. Коли сервер запущений, він використовує логіку, визначену в "App.js", для обробки запитів.

Тепер наступна частина – корневий скрипт або ж app.js (Рис 3.3).

Файл app.js відповідає за налаштування та конфігурацію серверної частини веб-додатку. Основні дії, які відбуваються в цьому файлі, включають: Підключення необхідних модулів після чогоСтворення екземпляру додатку за допомогою const app = express();.

```
C: > Users > NekoAngelNya > Desktop > дипломка моя > maybe > Server > server > JS App.js > ...
const express = require("express");
const bodyParser = require("body-parser");
const cors = require("cors");
const path = require("path");

const app = express();
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: true }));
app.use(cors());

app.use("/api/video", require("./api/routes/video"));
app.use("/Uploads", express.static("Uploads"));

if (process.env.NODE_ENV === "production") {
  // Set static folder
  // All the javascript and css files will be read and served from this folder
  app.use(express.static("Client/build"));

  // index.html for all page routes    html or routing and naviagtion
  app.get("*", (req, res) => {
    res.sendFile(path.resolve(__dirname, "../Client", "build", "index.html"));
  });
}

module.exports = app;

```

Рис 3.3 Вміст скрипта app.js

Далі ми налаштовуємо middleware для обробки JSON- та URL-даних за допомогою bodyParser.json() і bodyParser.urlencoded(). Встановлюємо middleware cors() для того щоб була можливість конектитись до серверної частини через клієнтську Далі ми налаштовуємо маршрутизації для шляху "/api/video" та підключаємо маршрути описані в файлі "./api/routes/video.js".

Налаштування middleware express.static(), щоб дозволити доступ до статичних файлів з директорії "Uploads" через шлях "/Uploads" щоб у подальшому не було проблем з переглядом відео коли сервер буде завантажувати їх. Далі ми робимо встановлення обробника маршруту для всіх GET-запитів, які не відповідають жодному іншому визначеному маршруту. В цьому обробнику відправляється файл "index.html" з директорії "Client/build".

Експорт об'єкту додатку за допомогою module.exports = app;, щоб його можна було використовувати в інших модулях у подальшому

Отже, app.js відповідає за налаштування сервера, маршрутизацію, обробку запитів та надання статичних файлів у залежності від поточного середовища виконання.

3.1.3 Реалізація API завантаження відео та мініатюр

Наше api за допомогою якого буде здійснюватись робота з відео-файлами описана як було згадано вище у директорії \api\routes у файлі video.js

Давайте розглянемо, за що відповідає кожна реалізація у наведеному коді(Рис 3.4): const express = require("express"); - підключення модуля Express для створення та налаштування сервера. const router = express.Router(); - створення нового маршрутизатора для обробки маршрутів. const multer = require("multer"); - підключення модуля Multer для обробки завантаження файлів.

const ffmpeg = require("fluent-ffmpeg"); - підключення модуля Fluent-ffmpeg для обробки відео.

```
1   const express = require("express");
2   const router = express.Router();
3   const multer = require("multer");
4   const ffmpeg = require("fluent-ffmpeg");
```

Рис 3.4 Підключення модулів

3.1.3.1 Реалізація завантаження відео на сервер

Функція fileFilter (Рис 3.5) визначає правила фільтрації відео-файлів, які дозволені для завантаження. Вона перевіряє тип файлу (mimetype) і визначає, чи є він допустимим для завантаження.

```
6   const fileFilter = (req, file, cb) => {
7     console.log("filefilter executed");
8     if (file.mimetype === "video/mp4" || file.mimetype === "video/mpeg") {
9       cb(null, true);
10    } else {
11      console.log("image type incorrect");
12      cb("only Video files is allowed", false);
13    }
14  };
```

(Рис 3.5) Код який відповідає за функцію fileFilter

Об'єкт storage (Рис 3.6) визначає, куди та як зберігати завантажені файли. Він вказує шлях для збереження файлів та приписує імена файлів на основі їх під час завантаження.

```
16  const storage = multer.diskStorage({
17    destination: function (req, file, cb) {
18      cb(null, "Uploads/");
19    },
20    filename: function (req, file, cb) {
21      const uniqueSuffix = Date.now() + "-" + file.originalname;
22      cb(null, file.fieldname + "-" + uniqueSuffix);
23    },
24  });
```

Рис 3.6 об’єкт storage

Далі маршрути, вони використовуються для обробки HTTP-запитів. В даному випадку, визначені такі маршрути:

router.post("/uploadfiles", ...)(Рис 3.7) - маршрут для завантаження відеофайлу. Використовує middleware upload для обробки завантаження файлу та повертає відповідь з даними про успішність завантаження та шляхом і іменем завантаженого файлу.

```
30  router.post("/uploadfiles", (req, res) => {
31    upload(req, res, (err) => {
32      if (err) {
33        return res.json({
34          success: false,
35          err,
36        });
37      }
38      return res.json({
39        success: true,
40        filepath: res.req.file.path,
41        fileName: res.req.file.filename,
42      });
43    });
44  });
```

Рис 3.7 маршрут router.post/uploadfiles

3.1.3.2 реалізація створення прев'ю для відео

router.post("/uploadThumbnail", ...) (Рис 3.8) - маршрут для завантаження мініатюри (зображення) для відеофайлу. Використовується middleware uploadThumbnail для завантаження зображення та повертає відповідь з даними про успішність завантаження та шляхом і іменем завантаженого файлу.

```
46  router.post("/uploadThumbnail", (req, res) => {
47    const fileFilterThumbnail = (req, file, cb) => {
48      if (
49        file.mimetype === "image/jpeg" ||
50        file.mimetype === "image/apng" ||
51        file.mimetype === "image/png"
52      ) {
53        cb(null, true);
54      } else {
55        console.log("image type incorrect");
56        cb("Sorry bruh You cant have a **not image** thumbnail(for now)*", false);
57      }
58    };
```

Рис 3.8 – маршрут router.post("/uploadThumbnail")

І наступний маршрут для створення мініатюри router.post("/thumbnail", ...) (Рис 3.9) - маршрут для створення мініатюр (знімків) з відеофайлу. Використовує модуль fluent-ffmpeg при взаємодії з FFmpeg у системі про який

згадувалось у другому розділ, необхідний для витягування інформації про відео та автоматизованої створення мініатюр. Повертає відповідь з даними про успішність створення мініатюр, шляхами до мініатюр та тривалістю відео.

```
router.post("/thumbnail", (req, res) => {
  let Thumbsfilepath = [];
  let fileDuration = "";

  ffmpeg.ffprobe(req.body.filePath, (err, metadata) => {
    console.log(metadata.format.duration);
    fileDuration = metadata.format.duration;
  });

  ffmpeg(req.body.filePath)
    .on("filenames", function (filenames) {
      for (names in filenames) {
        Thumbsfilepath.push("Uploads/thumbnails/" + filenames[names]);
      }
    })
    .on("end", function () {
      console.log("Screenshots taken");
      return res.json({
        success: true,
        Thumbsfilepath: Thumbsfilepath,
        fileDuration: fileDuration,
      });
    })
    .screenshots({
      // Will take screens at 20%, 40%, 60% and 80% of the video
      count: 1,
      folder: "Uploads/thumbnails",
      size: "320x240",
      filename: "thumbnail-%b.png",
    });
});
```

Рис 3.9 router.post("/thumbnail",) маршрут створення мініатюр для відео

3.2 Реалізація клієнтської частини

3.2.1 Структура клієнту

У клієнтській частині структура набагато складніша оскільки всі основні фунції пов'язані саме з нею.

Структура має наступний вигляд (Рис 3.10 та Рис 3.11)

```
├──public
└──src
   ├──components
   │  ├──Categoriesbar
   │  ├──comments
   │  ├──EmailVerification
   │  ├──header
   │  ├──horizontalVideocomponent
   │  ├──Input
   │  ├──ResetPassword
   │  ├──sidebar
   │  ├──SideBarRow
   │  ├──SingleComment
   │  ├──SubscribedUser
   │  ├──Suscribe
   │  ├──UserAuthBox
   │  ├──Video
   │  ├──VideoHorizontal
   │  ├──VideoMetaData
   │  └──VideoPlayer
   ├──Config
```

```
├──Context
├──model
├──Screens
│  ├──channelScreen
│  ├──EditVideoScreen
│  ├──ForgotPassword
│  ├──Homescreen
│  ├──LoginScreen
│  ├──PageNotFound
│  ├──SearchScreen
│  ├──subscriptions
│  ├──Upload
│  ├──UserActions
│  ├──VerifyEmail.js
│  └──WatchScreen
└──Store
   ├──actions
   └──reducers
```

Рис 3.10 та Рис 3.11 Структура клієнтської частини

Тепер перейдемо до опису структури, що за що відповідає та де зберігається

components (Компоненти):

Categoriesbar: Компонент, який відображає панель категорій для навігації по різних категоріях відео.

Comments: Компонент, що відповідає за відображення коментарів під відео.

EmailVerification: Компонент, який відображає форму для підтвердження електронної пошти користувача.

Header: Компонент, який відображає заголовок сторінки та навігаційне меню.

HorizontalVideoComponent: Компонент, що відображає відео в горизонтальному форматі.

Input: Компонент, який відображає поле введення тексту та форму для вводу даних.

ResetPassword: Компонент, що відповідає за форму відновлення пароля користувача коли той забув.

Sidebar: Компонент, який відображає бічну панель зі списком підписок, категорій.

SideBarRow: Компонент, який відображає окремий рядок у бічній панелі зі специфічними елементами.

SingleComment: Компонент, що відповідає за відображення окремого коментаря під відео.

SubscribedUser: Компонент, що відображає інформацію про користувача, на якого підписано.

Subscribe: Компонент, що дозволяє користувачам підписуватися на користувачів.

UserAuthBox: Компонент, який відображає блок автентифікації (вхід, реєстрація) користувача.

Video: Компонент, що відповідає за відображення окремого відео.

VideoHorizontal: Компонент, який відображає відео у горизонтальному форматі з додатковою інформацією.

VideoMetaData: Компонент, який відображає метадані про відео, такі як назва, автор, час, кількість переглядів, кількість вподобайок.

VideoPlayer: Компонент, що відтворює відео на сторінці.

Config (Конфігурація):

В цьому каталозі знаходиться конфігураційний файл для бази даних Firebase у якому приватні ключі для підключення до бази.

Context (Контексти):

В цьому каталозі збережені файли, які використовуються для реалізації контексту в додатку.

model (Моделі):

У цьому каталозі розміщені моделі даних, моделі користувача, відео, коментарів.

Screens (Екрани):

channelScreen: Екран, який відображає інформацію про канал користувача.

EditVideoScreen: Екран, на якому користувач може редагувати відео.

ForgotPassword: Екран, що відповідає за відновлення забутого пароля.

Homescreen: Головний екран, на якому відображаються відео з різних категорій.

LoginScreen: Екран, на якому користувач може увійти в свій акаунт.

PageNotFound: Екран, який відображається, коли сторінка не знайдена.

SearchScreen: Екран, який дозволяє користувачу здійснювати пошук відео.

subscriptions: Екран, на якому відображаються підписки користувача.

Upload: Екран, на якому користувач може завантажувати відео.

UserActions: Екран, який дозволяє користувачу виконувати різні дії з відео

VerifyEmail: Екран, на якому користувач може підтвердити свою електронну пошту.

WatchScreen: Екран, на якому відображається відео для перегляду.

Store (Сховище):

actions: Каталог, де зберігаються файли, що містять функції-дії для взаємодії зі станом додатку.

reducers: Каталог, де знаходяться файли, що містять редуктори для обробки дій та зміни стану додатку.

Це загальна структура клієнтського проекту з розподілом функціональних компонентів та логічних частин.

Тепер трошки детальніше про деякі окремі файли

У файлі app.js (Рис 3.12) описано основний файл додатку. Основна ідея цього файлу полягає в налаштуванні маршрутизації та відображенні різних екранів додатку за допомогою react-router-dom, налаштуванні стану додатку та підключенні Redux для керування станом за допомогою react-redux.

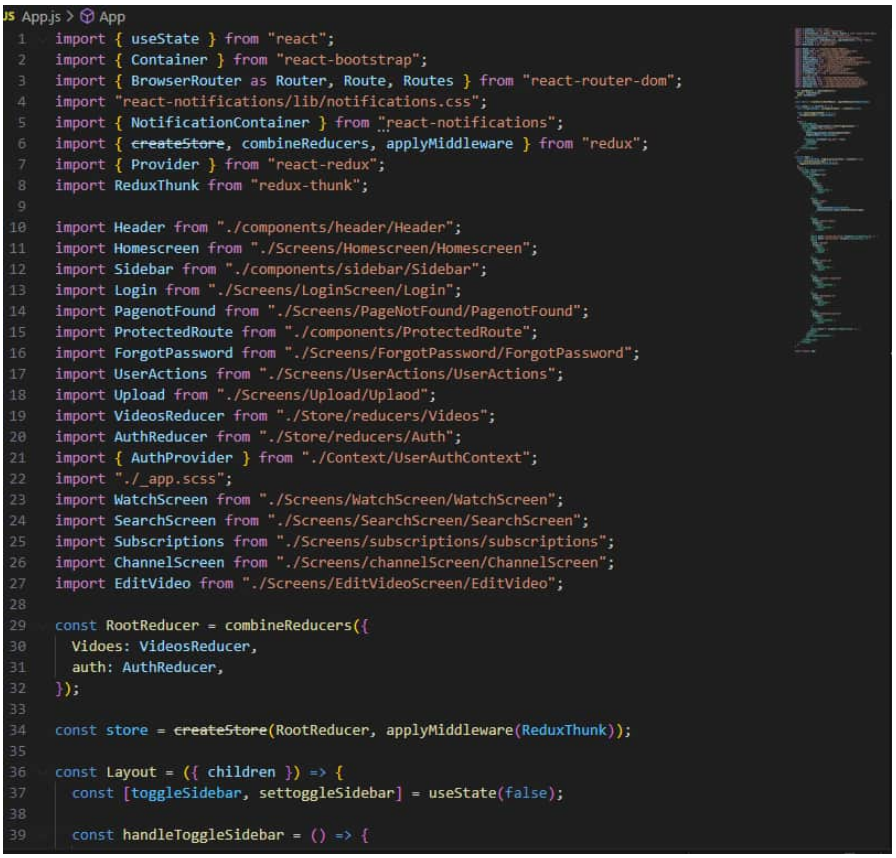

Рис 3.12 Вміст головного файлу app.js

Файл categories.js в свою чергу відповідальний за категорії у які відео можна відносити, виглядає він так (Рис 3.13)

Рис 3.13 вміст файлу з переліком категорій

Відповідно редагуючи цей файл ми можемо добавляти або ж наоборот видаляти категорії у які можна добавляти відео

3.2.2 Структура бази даних

У якості бази даних у нас виступає база даних від компанії google firebase Сама база даних має вид наступної структури під дані наших користувачів, коментарів, відео (Рис 3.14)

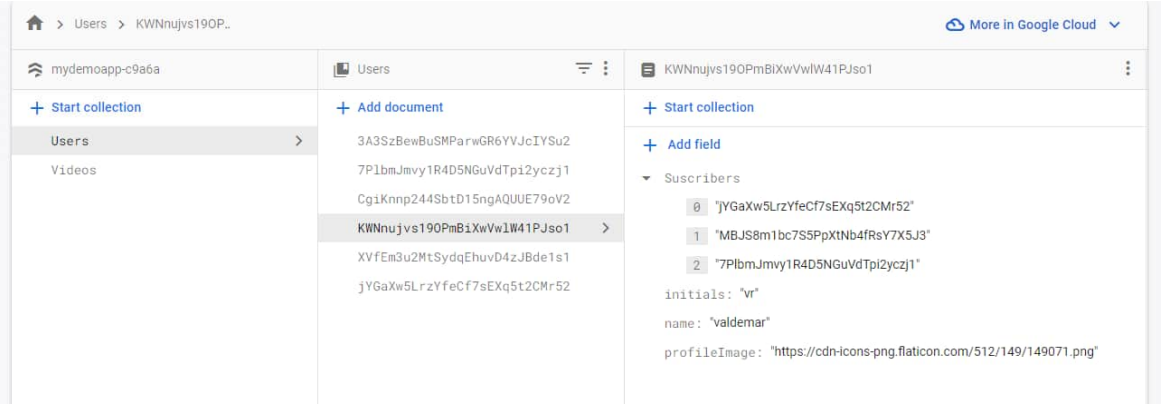

Рис 3.14 Структура бази даних користувачів

Це структура users, в ній зберігається інформація про канали користувачів, можна побачити такі поля як “suscribers” це поле відповідає за підписників, як можна помітити у користувача Valdemar вже є 3 підписники, кожен з них має унікальний id який посилається на їх профілі
Також є стрічка під ініціали, ім’я та зображення профілю (Рис 3.15)

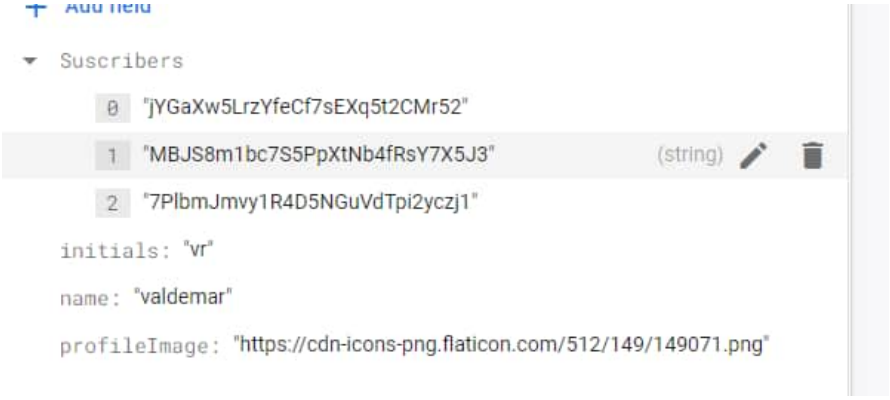

Рис 3.15 структура баиз користувача

Із структурою відео тут цікавіше (Рис 3.16), у нас є такі поля як категорії, знову посилання на зображення профілю, ім’я користувача який завантажив відео, також структура коментарів яка містить самий вміст коментаря, id відео до якого воно залишене, ім’я хто залишив, дата, id користувача який залишив коментар, та його зображення, окрім цього є також опис у якому поміщається опис відео, структура у якій поміщаються дизлайки та лайки, довжина відео, id хто це відео завантажив, його тип приватне чи публічне та назва, у полі views зберігаються ip адреси з яких відео було переглянуте, система не враховує нові перегляди з одного й того ж самого ip адреса щоб відео не накручувало перегляди

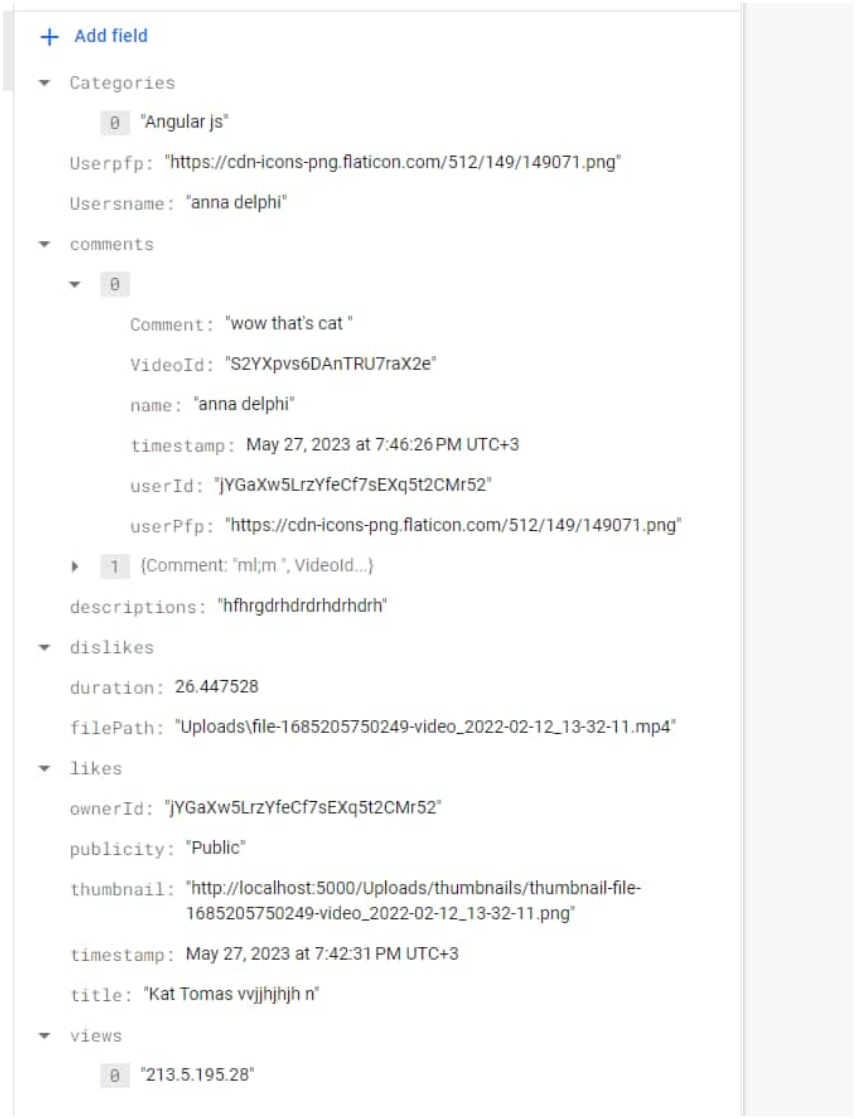

Рис 3.16 структура бази відео

3.2.2.1 Вхід в обліковий запис,реєстрація,відновлення паролю

Реалізація функції входу, створення нового аккаунту, первірки емейлу та скидання пароля а також вихід із системи реалізований у скрипті UserAuthContext.jsФункція useAuth(Рис 3.17): Ця функція використовує useContext для отримання даних автентифікації з AuthContext. Вона повертає об'єкт, що містить дані автентифікації та вказує, чи є користувач аутентифікованим.

Компонент AuthProvider(Рис 3.17): Цей компонент є постачальником контексту AuthContext для дочірніх компонентів. Він містить основну логіку автентифікації та взаємодії з Firebase.

```
export const useAuth = () => {
  const auth = useContext(AuthContext);
  return { ...auth, isAuthenticated: auth.CurrentUser != null };
};

export function AuthProvider({ children }) {
  const [CurrentUser, setCurrentUser] = useState();
  const [loading, setloading] = useState(true);

  useEffect(() => {
    const unsuscribe = onAuthStateChanged(auth, (user) => {
      console.log(user);
      setCurrentUser(user);
      setloading(false);
    });

    return unsuscribe;
  }, []);
```

Рис 3.17 функції useAuth, AuthContext

Функція SignupUser (Рис 3.18):

Ця функція використовує createUserWithEmailAndPassword з Firebase для реєстрації нового користувача. Після успішної реєстрації, викликається функція CreateFireStoreUser для створення запису користувача у базі даних Firestore.

```
40  const SignupUser = (email, password, name) => {
41    return createUserWithEmailAndPassword(auth, email, password).then(
42      (resp) => {
43        if (resp.user) {
44          CreateFireStoreUser(
45            resp.user,
46            name,
47            "https://cdn-icons-png.flaticon.com/512/149/149071.png"
```

(Рис 3.18) Функція SignupUser

Функція CreateFireStoreUser (Рис 3.19):

Ця функція використовує setDoc з Firebase для створення запису користувача у Firestore. Запис містить дані, такі як ім'я, зображення профілю, ініціали та список підписників.

```
const CreateFireStoreUser = (user, name, profileImage) => {
  const UsersRef = doc(db, "Users", user.uid);
  setDoc(UsersRef, {
    name: name,
    profileImage: profileImage,
    initials: name[0] + name[name.length - 1],
    Suscribers: [],
  }).then(() => {
    console.log("added User");
  });
```

Рис 3.19 Функція CreateFireStoreUser

Функція LogInUser (Рис 3.20):

Ця функція використовує signInWithEmailAndPassword з Firebase для входу користувача за допомогою електронної пошти та пароля.

```
const LogInUser = (email, password) => {
  return signInWithEmailAndPassword(auth, email, password);
};
```

Рис 3.20 Функція LogInUser

Функція SendEmailVerification (Рис 3.21):

Ця функція використовує sendSignInLinkToEmail з Firebase для надсилання посилання на підтвердження електронної пошти для поточного користувача.

```
const SendEmailVerification = () => {
  const actionCodeSettings = {
    url: "http://localhost:3000",
    handleCodeInApp: true,
  };
  return sendSignInLinkToEmail(auth, CurrentUser.email, actionCodeSettings);
};
```

Рис 3.21 Функція SendEmailVerification

Функція VerifyEmailLink (Рис 3.22):

Ця функція перевіряє, чи є поточне посилання на вход за електронною поштою, використовуючи isSignInWithEmailLink з Firebase. Якщо перевірка пройшла успішно, викликається signInWithEmailLink для входу користувача.

```
const VerifyEmailLink = () => {
  if (isSignInWithEmailLink(auth, window.location.href)) {
    return signInWithEmailLink(auth, CurrentUser.email);
  }
};
```

Рис 3.22 Функція VerifyEmailLink

Функція SignInWithGoogle (Рис 3.23):

Ця функція використовує signInWithPopup з Firebase для входу користувача за допомогою облікового запису Google.

```
const SignInWithGoogle = () => {
  const provider = new GoogleAuthProvider();
  return signInWithPopup(auth, provider);
};
```

Рис 3.23 Функція SignInWithGoogle

Функція forgotPassword(Рис 3.24):

Ця функція використовує sendPasswordResetEmail з Firebase для надсилання електронного листа з посиланням для скидання пароля користувача.

```
const forgotPassword = (email) => {
  return sendPasswordResetEmail(auth, email, {
    url: "http://localhost:3000/login",
  });
};
```

Рис 3.24 Функція forgotPassword

3.2.3 Реалізація коментарів

Перше ніж писати реалізацію за допомогою функцій нам необхідно створити клас Comment (Рис 3.25), який використовується для створення об'єктів коментарів.

constructor: Конструктор цього класу приймає різні параметри, такі як VideoId, userId, name, profileImage, Comment і timestamp. При створенні нового об'єкта коментарія, ці параметри передаються в конструктор. this.name, this.userPfp, this.Comment, this.userId, this.VideoId, this.timestamp: Ці рядки коду встановлюють властивості об'єкта коментарія на значення, передані в конструктор. Наприклад, this.name = name встановлює властивість name об'єкта коментарія на значення, передане в параметрі name. Таким чином, коли створюється новий об'єкт коментарія з використанням цього класу, він буде мати властивості, які відповідають переданим значенням в конструкторі. Наприклад, name, userPfp, Comment, userId, VideoId і timestamp будуть доступні як властивості створеного об'єкта коментарія.

```
1  class Comment {
2    constructor(VideoId, userId, name, profileImage, Comment, timestamp) {
3      this.name = name;
4      this.userPfp = profileImage;
5      this.Comment = Comment;
6      this.userId = userId;
7      this.VideoId = VideoId;
8      this.timestamp = timestamp;
9    }
10 }
```

Рис 3.25 Створений клас коментаря

Переходимо до реалізації функції коментування (Рис 3.26) та (Рис 3.27).

Ми використовуємо функцію у вигляді компоненту, працює вона наступним чином. Створюються станові змінні за допомогою useState для loading, error, Comment. Змінна loading використовується для відображення спінера під час завантаження коментарів. Змінна error використовується для відображення повідомлення про помилку, якщо сталася помилка під час

надсилання коментаря. Змінна Comment зберігає значення тексту коментаря, який користувач вводить у полі вводу.

Використовуються хуки useDispatch і useSelector з пакету react-redux для отримання функції диспетчера та вибірки потрібних даних зі стану Redux.
Відбувається визов функцій dispatch та useSelector, щоб отримати дані про коментарі та інформацію про користувача зі стану Redux.

Функція handleCommentSubmit викликається при надсиланні форми коментаря. Вона встановлює значення loading на true та очищує повідомлення про помилку. Потім відбувається виклик функції диспетчера.

VideoActions.Create_Comment, щоб створити новий коментар для відео. Значення коментаря, зображення профілю та ім'я користувача передаються як параметри. Після успішного створення коментаря, значення Comment очищується. Якщо сталася помилка, значення error встановлюється на повідомлення про помилку.

Використовується хук useEffect для отримання деталей користувача за допомогою функції диспетчера UserAuthActions.fetchUserDetails при завантаженні компонента. Залежність dispatch вказує, що ефект має бути виконаний, якщо змінюється значення dispatch.

Компонент повертає JSX-розмітку, яка включає поле для відображення кількості коментарів, форму для надсилання нового коментаря, список коментарів та компонент SingleComment, який відображає окремий коментар. Значення comments мапляться для відображення кожного коментаря у списку.

Компонент Comments експортується за допомогою функції trackWindowScroll з пакету react-lazy-load-image-component, щоб включити прокрутку вікна для компонента.

```jsx
function Comments({ userPfp, VideoId, name, scrollPosition }) {
  const [loading, setloading] = useState(false);
  const [error, seterror] = useState();
  const [Comment, setComment] = useState("");
  const dispatch = useDispatch();
  const comments = useSelector((state) => state.Vidoes.video.comments);
  const Userinfo = useSelector((state) => state.auth.userInfo);
  console.log(Userinfo);

  const handleCommentSubmit = async (e) => {
    e.preventDefault();
    setloading(null);
    seterror(false);
    try {
      await dispatch(
        VideoActions.Create_Comment(
          VideoId,
          Comment,
          Userinfo.profileImage,
          Userinfo.name
        )
      );
      setComment("");
    } catch (err) {
      seterror(err.message);
    }
    setloading(false);
  };

  useEffect(() => {
    const FetchUserdetails = async () => {
      await dispatch(UserAuthActions.fetchUserDetails());
    };
    FetchUserdetails();
  }, [dispatch]);
```

```jsx
return (
  <div className="comments">
    <p>{comments.length} Comments</p>
    <div className="comments_form d-flex w-100 my-2">
      {Userinfo && (
        <LazyLoadImage
          src={Userinfo.profileImage}
          effect="blur"
          scrollPosition={scrollPosition}
          alt="userPfp"
        />
      )}
      <form onSubmit={handleCommentSubmit} className="d-flex flex-grow-1">
        <textarea
          autoFocus
          className="flex-grow-1"
          required
          placeholder="Add a comment..."
          value={Comment}
          onChange={(e) => setComment(e.target.value)}
        />
        {!loading ? (
          <button className="border-0 p-2">Comment</button>
        ) : (
          <SpinnerCircular color="#00BFFF" />
        )}
      </form>
    </div>
    <div className="comments__list">
      {comments.map((comment) => (
        <SingleComment
          userId={comment.userId}
          userPfp={comment.userPfp}
          name={comment.name}
          comment={comment.Comment}
          timestamp={comment.timestamp}
        />
      ))}
    </div>
  </div>
);

export default trackWindowScroll(Comments);
```

Рис 3.26 та Рис 3.27 з лістингом функції коментування

3.2.4 Реалізація функції лайк/дизлайк

Реалізація функції лайк описана у скрипті videoMetaData,js Працює вона наступним чином. Функція handleLike викликається при натисканні на кнопку лайка. Вона встановлює значення loading на true та очищує повідомлення про помилку. Залежно від стану лайку та дизлайку, викликаються відповідні функції диспетчера(Рис 3.28)

```jsx
const handleLike = async () => {
  setloading(null);
  seterror(false);
  try {
    if (!IsLiked && !Isdisliked) {
      await dispatch(VideoActions.Create_Like(VideoId));
    } else if (IsLiked) {
      await dispatch(VideoActions.Remove_Like(VideoId));
    } else if (Isdisliked && !IsLiked) {
      await dispatch(VideoActions.Create_Like(VideoId));
      await dispatch(VideoActions.Remove_DisLike(VideoId));
    }
    setloading(false);
  } catch (err) {
    seterror(err.message);
```

Рис 3.28 Лістинг коду до функції лайку

Функція спочатку визначає, чи користувач ще не поставив лайк та дизлайк до відео (обидва стани false). Якщо так, викликається функція VideoActions.Create_Like для додавання лайка. У разі, якщо користувач вже поставив лайк (стан true), викликається функція VideoActions.Remove_Like для видалення лайка.

Якщо користувач поставив дизлайк(Рис 3.29), але не лайк (стани true та false відповідно), спочатку викликається функція VideoActions.Create_Like для додавання лайка, а потім викликається функція VideoActions.Remove_DisLike для видалення дизлайку.

```
const handleDisLike = async () => {
  setloading(null);
  seterror(false);
  try {
    if (!Isdisliked && !IsLiked) {
      await dispatch(VideoActions.Create_DisLike(VideoId));
    } else if (Isdisliked) {
      await dispatch(VideoActions.Remove_DisLike(VideoId));
    } else if (IsLiked && !Isdisliked) {
      await dispatch(VideoActions.Create_DisLike(VideoId));
      await dispatch(VideoActions.Remove_Like(VideoId));
    }
    setloading(false);
  } catch (err) {
    seterror(err.message);
  }
```

Рис 3.29 Лістинг коду до функції дизлайку

У кінці функції значення loading встановлюється на false, а в разі виникнення помилки встановлюється значення error на повідомлення про помилку.

3.2.5 Реалізація функції підписки

Спочатку ми імпоруємо необхідні залежності.Визначаються стан error для збереження повідомлення про помилку, посилання subscribeRef для отримання посилання на елемент кнопки, диспетчер dispatch для виклику екшенів та отримання інформації про власника відео зі стану.

Функція onSubscribeClick(Рис 3.30) викликається при натисканні на кнопку підписки. Вона перевіряє, чи користувач вже підписаний на канал або

ні. Якщо користувач не підписаний, викликається функція VideoActions.Subscribe для підписки на канал. Після успішної підписки, стан Userinfo.Suscribers оновлюється, текст та клас кнопки змінюються відповідно.

Якщо користувач вже підписаний, викликається функція VideoActions.UnSubscribe для відписки від каналу. Після успішної відписки, текст та клас кнопки знову змінюються.

```
14  const onSuscribeClick = () => {
15    if (
16      Userinfo?.Suscribers.length == 0 ||
17      Userinfo?.Suscribers.indexOf(auth.currentUser.uid) !== 0
18    ) {
19      VideoActions.Subscribe(OwnerId)
20        .then(() => {
21          Userinfo.Suscribers.push(auth.currentUser.uid);
22          subscribeRef.current.innerText = "Subscribed";
23          subscribeRef.current.classList.add("Subscribed");
24        })
25        .catch((error) => {
26          seterror(error.message);
27        });
28    } else {
29      VideoActions.UnSubscribe(OwnerId)
30        .then(() => {
31          subscribeRef.current.innerText = "Subscribe";
32          subscribeRef.current.classList.remove("Subscribed");
33          Userinfo.Suscribers.pop(-1);
34        })
35        .catch((error) => {
36          seterror(error.message);
37        });
38    }
39  };
```

Рис 3.30 Функція підписки

Далі використовуємо useEffect (Рис 3.31) для відслідковування зміни стану error і відображення повідомлення про помилку за допомогою NotificationManager.error. Також використовуємо useEffect(Рис 3.31) для завантаження деталей власника відео при імпорті компонента. Далі викликається функція VideoActions.fetchUserDetails для отримання інформації про власника відео. У разі виникнення помилки, встановлюється стан error і відображається повідомлення про помилку за допомогою NotificationManager.error. Компонент повертає розмітку, яка містить

інформацію про власника відео та кнопку підписки. Залежно від стану підписки, текст і клас кнопки змінюються.

```
useEffect(() => {
  if (error) {
    NotificationManager.error(error, "Error", 10000);
  }
}, [error]);

useEffect(() => {
  const FetchUserdetails = async () => {
    try {
      await dispatch(VideoActions.fetchUserDetails(OwnerId));
    } catch (error) {
      seterror(error.message);
      NotificationManager.error(
        error.message,
        "Unable to get Owner Info :<",
        10000
      );
    }
  };
  FetchUserdetails();
}, [dispatch]);
```

Рис 3.31 функція useEffect

3.2.6 Реалізація кількості переглядів

Сама функція рахування кількості переглядів знаходиться у скрипті дій а саме videos.js функція побудована так, що вона записує ip адресу з якої користувач дивився відео, це необхідно для того щоб у випадку якщо користувач створив декілька аккаунтів – він не міг накручувати перегляди. Сама функція виглядає наступним чином (Рис 3.32)

```
export const SetView = (VideoId, Ip) => {
  return new Promise((resolve, reject) => {
    try {
      const VidRef = doc(db, "Videos", VideoId)
      updateDoc(VidRef, {
        views: arrayUnion(Ip.ip),
      });
      resolve("Suscribed !!!");
    } catch (error) {
      reject(Error("Failed :< didn't work!"));
    }
  });
};
```

Рис 3.32 Функція оновлення переглядів

Функція SetView (Рис 3.32) отримує два параметри: VideoId (ідентифікатор відео) і Ip (IP-адреса переглядача). Відбувається створення нової обіцянки (Promise), яка приймає дві функції зворотного виклику resolve і reject. У блоку try виконується наступне: Створюється посилання VidRef на документ в базі даних, що відповідає відео з ідентифікатором VideoId. Використовуючи функцію updateDoc з Firestore, оновлюється документ VidRef, додавши IP-адресу Ip.ip до масиву views (кількість переглядів). Викликається функція resolve з повідомленням "Suscribed !!!", якщо оновлення пройшло успішно. У разі виникнення помилки, викликається функція reject з помилкою Error("Failed :< didn't work!").

3.2.7 Реалізація пошуку відео

Для пошуку відео у нас написана наступна функція (Рис 3.33) fetchSearchvidoes - вона використовується для отримання відео за результатами пошуку з бази даних. Спочатку експортується функція fetchSearchvidoes, яка приймає параметр searchquery. У функції використовується async підхід, що дозволяє виконувати асинхронні операції. У блоку try виконується наступне: Застосовується функція capitalizeFirstWords до searchquery для форматування пошукового запиту.

Створюється посилання VideosRef на колекцію "Videos" в базі даних Firestore. Створюється запит q за допомогою функції query, який вибирає документи з колекції VideosRef, де поле "title" знаходиться в діапазоні від CaseSearchquery до CaseSearchquery + "\uf8ff" (використовується для пошуку заголовків, що починаються з searchquery), і поле "publicity" має значення "Public" (саме public для того щоб ніхто не міг доступитись до приватного відео користувача).

Виконується отримання даних за допомогою функції getDocs і переданого запиту q. Результат зберігається в змінну querySnapshot. Створюється порожній масив Searchvideostobeloaded. За допомогою методу forEach ітеруються

документи з querySnapshot і для кожного документа створюється новий об'єкт класу Video з відповідними властивостями.

Отримані відео додаються до масиву Searchvideostobeloaded. За допомогою dispatch викликається дія SET_SearchVidoes, що передає масив Searchvideostobeloaded у поле Searchvideos.

У випадку виникнення помилки у блоку catch, помилка виводиться у консоль і викликається throw error для передачі помилки далі.

Отже, функція fetchSearchvidoes виконує пошук відео в базі даних за допомогою переданого пошукового запиту searchquery

```
330 export const fetchSearchvidoes = (searchquery) => {
331   return async (dispatch) => {
332     const CaseSearchquery = capitalizeFirstWords(searchquery);
333     try {
334       const VideosRef = collection(db, "Videos");
335       const q = query(
336         VideosRef,
337         where("title", ">=", CaseSearchquery),
338         where("title", "<=", CaseSearchquery + "\uf8ff"),
339         where("publicity", "==", "Public")
340       );
341       const querySnapshot = await getDocs(q);
342       const Searchvideostobeloaded = [];
343       querySnapshot.forEach((doc) => {
344         Searchvideostobeloaded.push(
345           new Video(
346             doc.id,
347             doc.data().ownerId,
348             doc.data().title,
349             doc.data().thumbnail,
350             doc.data().descriptions,
351             doc.data().duration,
352             doc.data().views,
353             doc.data().Categories,
354             doc.data().publicity,
355             doc.data().filePath,
356             doc.data().Usersname,
357             doc.data().Userpfp,
358             doc.data().timestamp,
359             doc.data().comments,
360             doc.data().likes,
361             doc.data().dislikes
362           )
363         );
364       });
365       dispatch({
366         type: SET_SearchVidoes,
367         Searchvideos: Searchvideostobeloaded,
368       });
369     } catch (error) {
370       console.log(error);
371       throw error;
372     }
373   };
374 };
375
```

Рис 3.33 Функція для пошуку відео

3.3 Опис реалізованого функціоналу. Тестування

3.3.1 Вхід у обліковий запис

При запуску сервера та клієнта відповідно командами npm start у нас запускається сервер на порті 5000 та клієнт на 3000, заходимо ми за допомогою адреси свого локального серверу або ж просто локалхостом localhost:3000 (Рис 3.34)

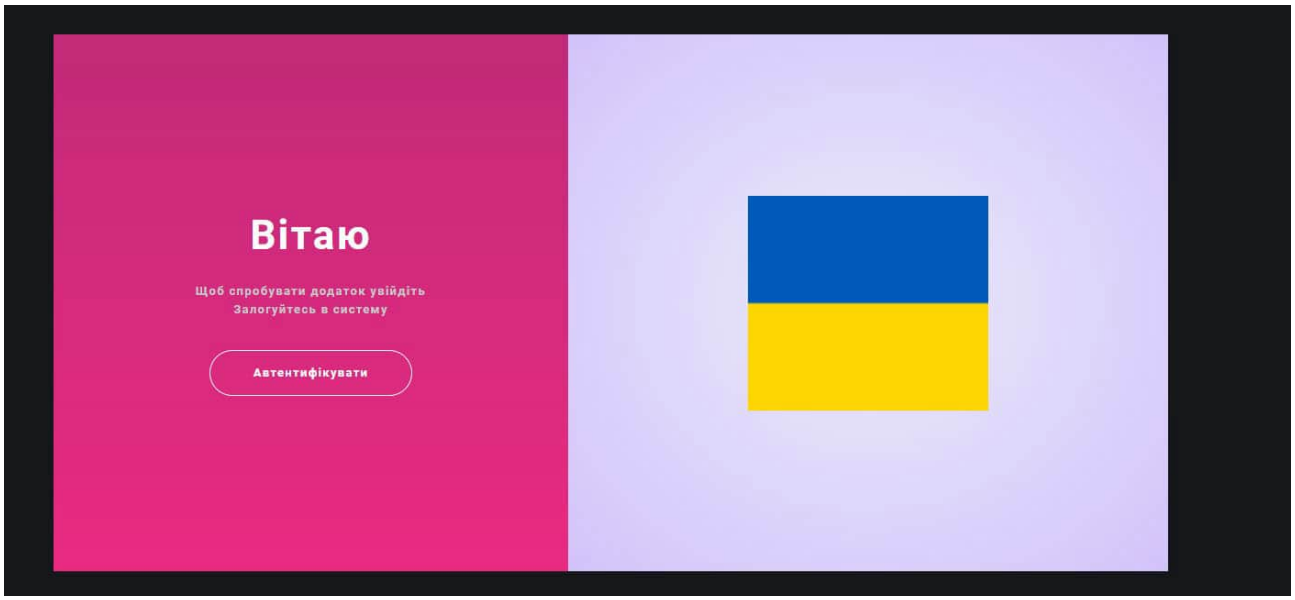

Рис 3.34 Зображення входу в обліковий запис

Отримуємо вікно привітання (Рис 3.34), з кнопкою яка пропонує нам автентифікуватись, натискаємо на неї, після того як ми натиснули оскільки веб-сайт наш динамічний у нас одразу міняється вікно і ми можемо бачити привітання, кнопку входу, кнопку *забув пароль* та поля для вводу нашої інформації,також можна увійти через аккаунт google (Рис 3.35)

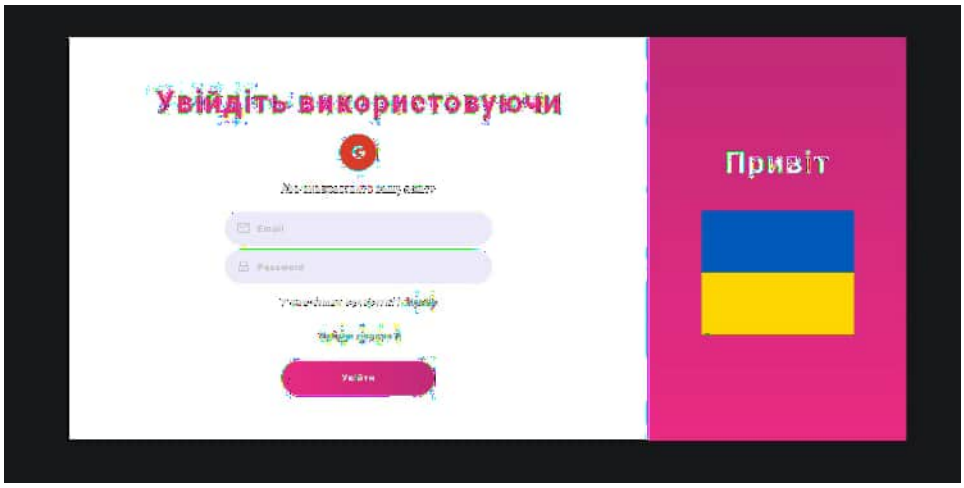

Рис 3.35 Зображення реєстрації/входу

3.3.2 Сторінка “Реєстрація нового користувача”

Натискаємо на кнопку зареєструватись нам відкриваються додаткові текстові поля (Рис 3.36) куди можна ввести своє ім’я, пароль, та електронну пошту та вводимо данні для реєстрації та після увійти. Після того як ми увійдемо в систему нас привітає.

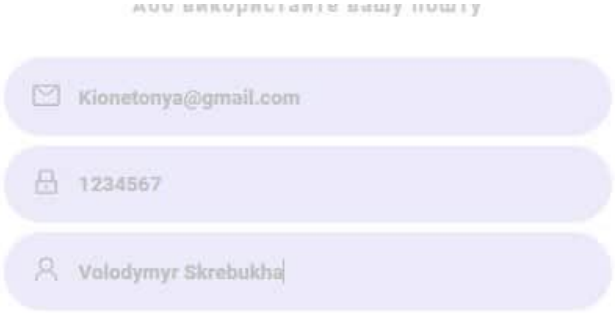

Рис 3.36 Вікно з даними для реєстрації

3.3.3 Сторінка “Домашня сторінка”

Це домашня сторінка (Рис 3.37), на ній будуть показуватись відео нові завантажені відео на сайт, вони будуть сортуватися використовуючи такі параметри як, кількість переглядів, як давно вони були завантажені, які у них теги, зліва на панелі наразі доступно 3 пункти:

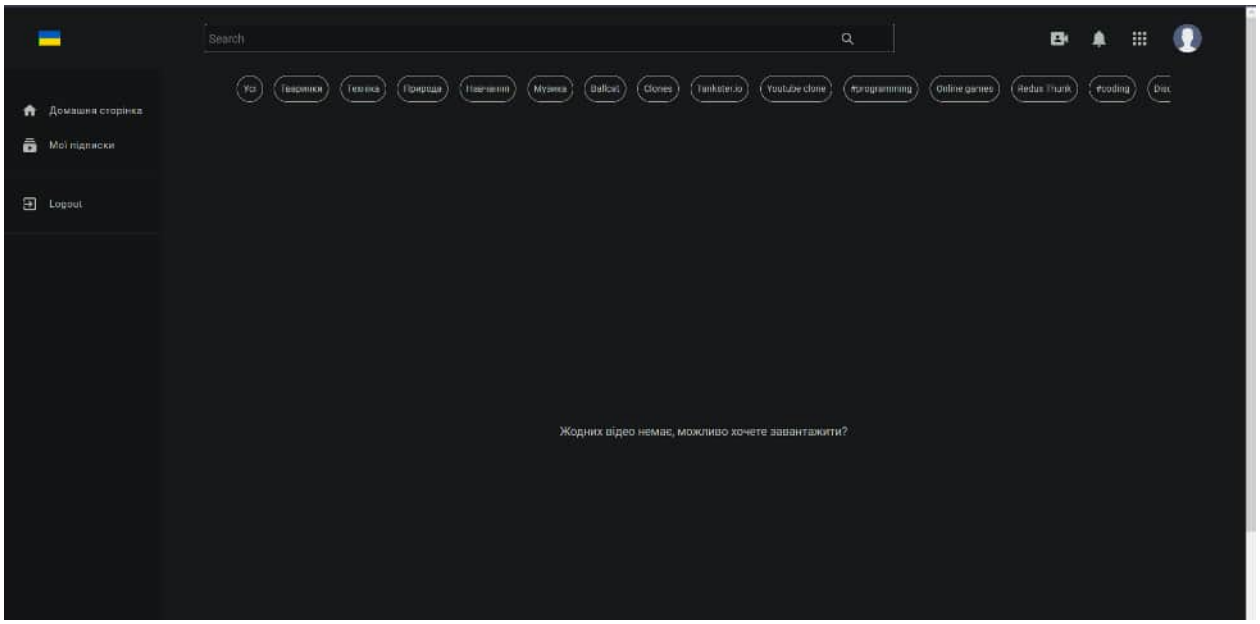

Рис 3.37Домашня сторінка

- пункт “домашня сторінка”
- пункт “мої підписки”
- вихід з аккаунту

Зверху у нас є поле для пошуку відео, підбір відео по категоріям, та кнопка завантаження відео

3.3.4 Сторінка “Завантаження відео”

Перейдемо на сторінку завантажння, натискаємо на іконку відео (Рис 3.38)

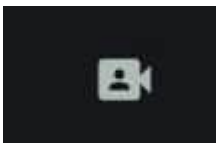

Рис 3.38 Іконка завантаження відео

Нам відкриється поверх вікно(Рис 3.39). У нас є відповідна секція у яку ми можемо завантажити відео або перетягуванням, або натиснувши на неї і вибрав відео

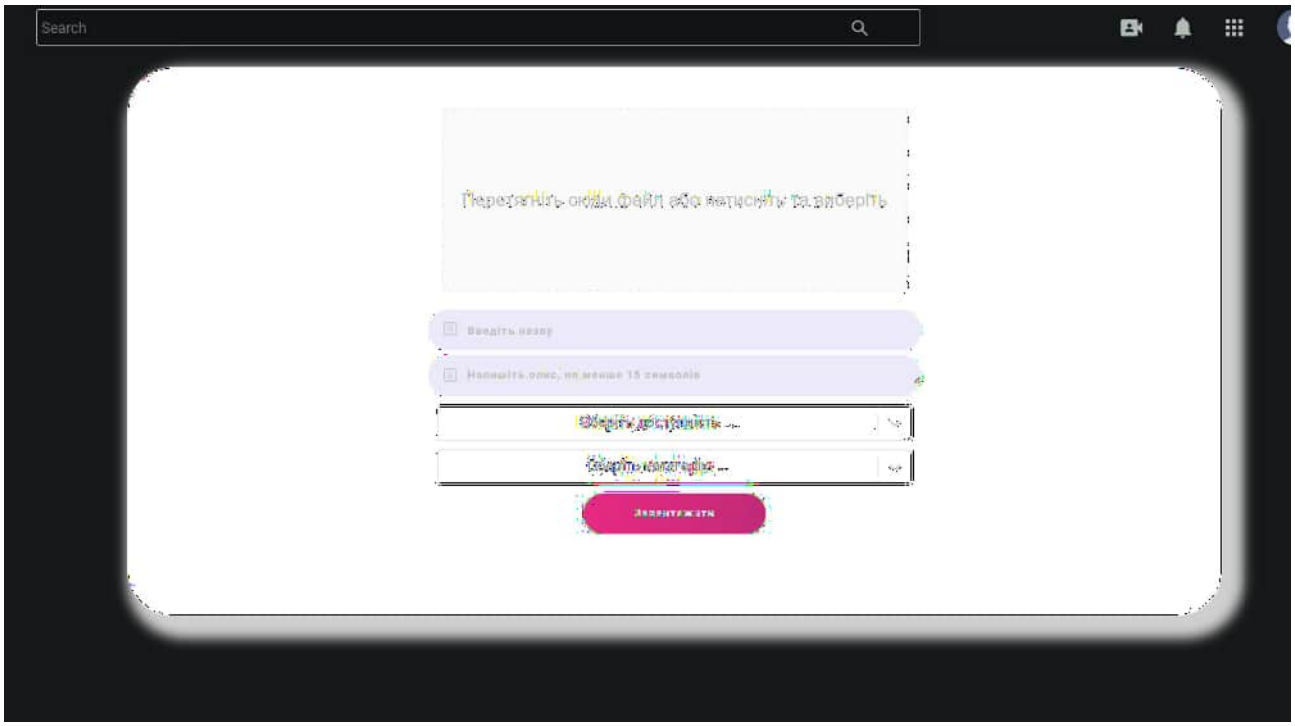

Рис 3.39 Вікно завантаження відео

Також далі йдуть 2 текстові поля (Рис 3.39):

- у першому назва для відео
- у другому ж опис відео

Далі у нас є вибір з якою доступністю буде завантажене відео. У випадку вибору “публічне” наше відео буде у всіх користувачів видно, у випадку ж вибору “приватне” його зможемо дивитися тільки ми або ті хто має на нього посилання (Рис 3.40)

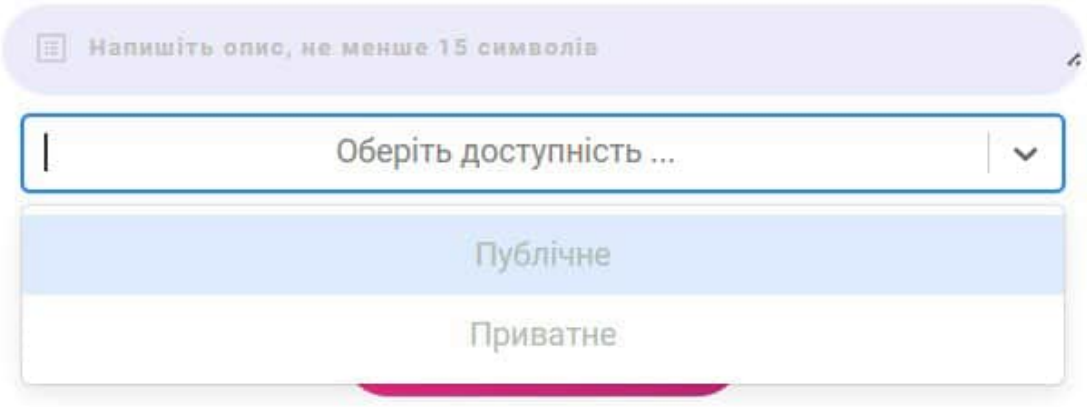

Рис 3.40 Вибір доступності

Наступне поле дозволяє нам обрати до якої категорії відео буде відноситись (Рис 3.41) щоб у подальшому воно попадалось у рекомендаціях зі схожим змістом, ну або можна обрати щоб воно не мало якогось конкретного напрямку і рекомендувалось усім

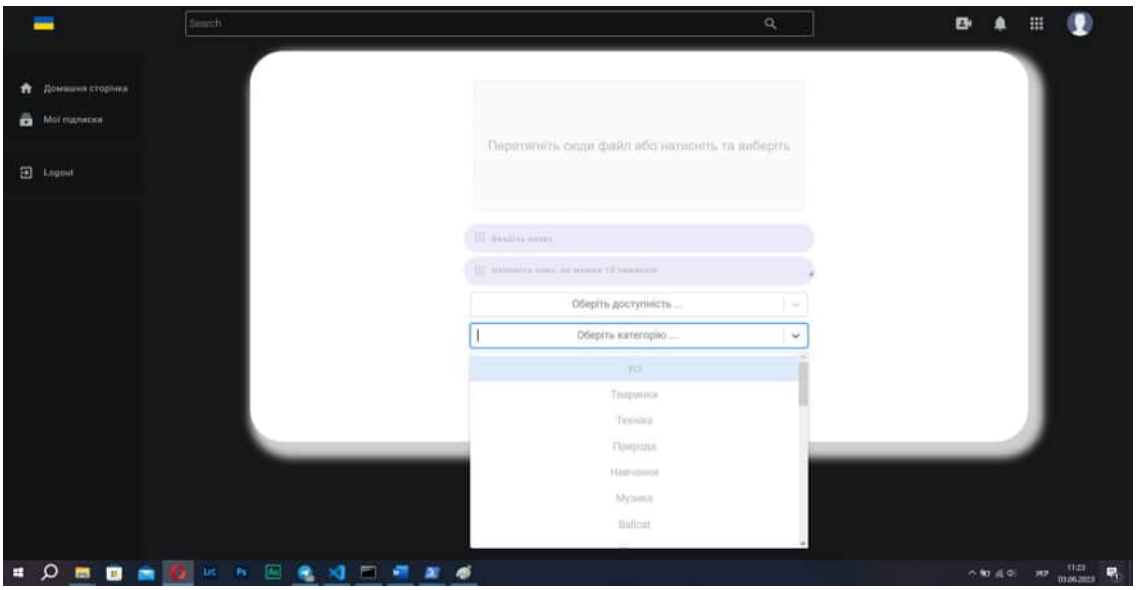

Рис 3.41 Вибір категорії до якої відноситься відео

Спробуємо завантажити якесь відео.

Для цього обираємо саме відео, придумуємо назву, опис, обираємо доступність та категорію для відео (Рис 3.42)

Рис 3.42 Меню завантаження

Та натискаємо “завантажити”. Після натискання кнопки піде процес завантаження, як тільки воно завершиться нас про це повідомить, після завантаження на сервер відео-файлу описаний FFmpeg та наша функція написана розпочне генерацію мініатюри і теж виведе відповідне повідомлення в правому верхньому кутку (Рис 3.43)

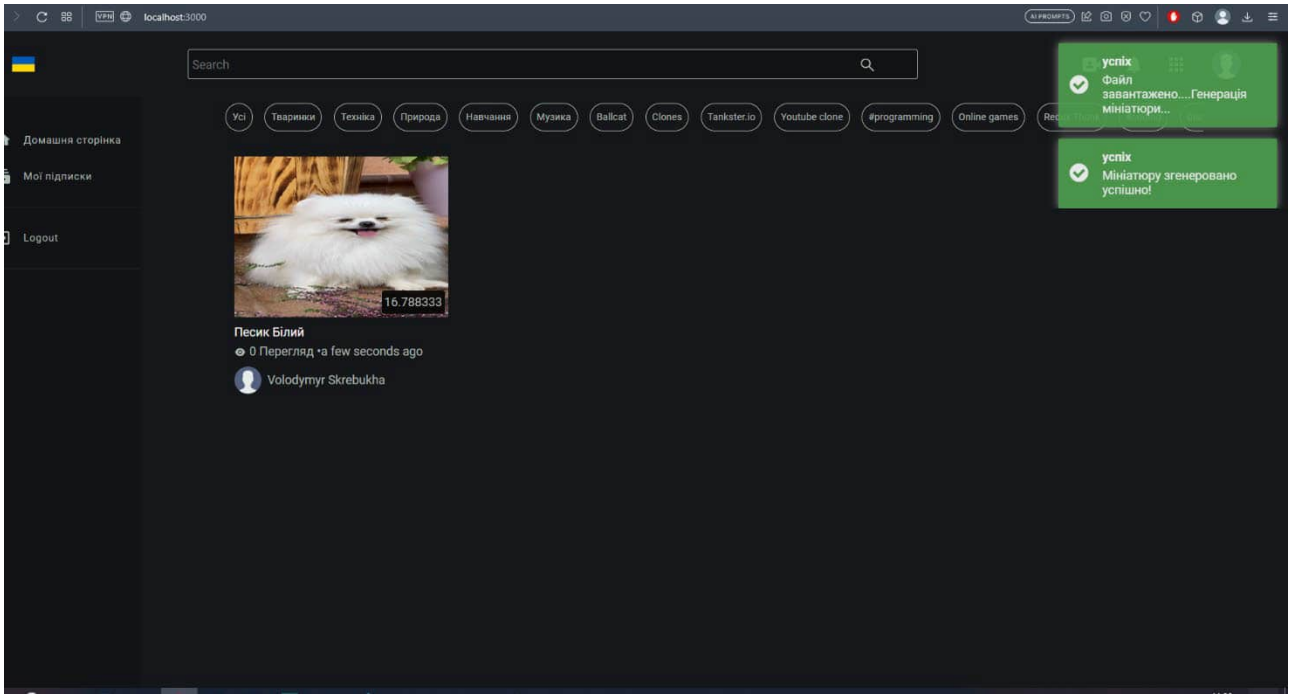

Рис 3.43 Результат завантаження відео

Ми можемо бачити відео яке з’явилось на домашній сторінці, видно назву відео, кількість переглядів (наразі 0) час коли завантажено, хто завантажив (Рис 3.43)

3.3.5 Сторінка “Перегляд відео”

Натискаємо на завантажене відео та нас перекине на нову сторінку. Ми бачимо відео плеєр (Рис 3.44), можливості у нього наступні

- повноекранний режим
- регулювання гучності
- функція зміни швидкості
- функція картинка в картинці

Окрім цього можна побачити зображення користувача, час завантаження, лайки та дизлайки, також кнопку підписки яка наразі не активна бо ми не можемо самі на себе підписатися. Оновимо сторінку та горнемо у низ

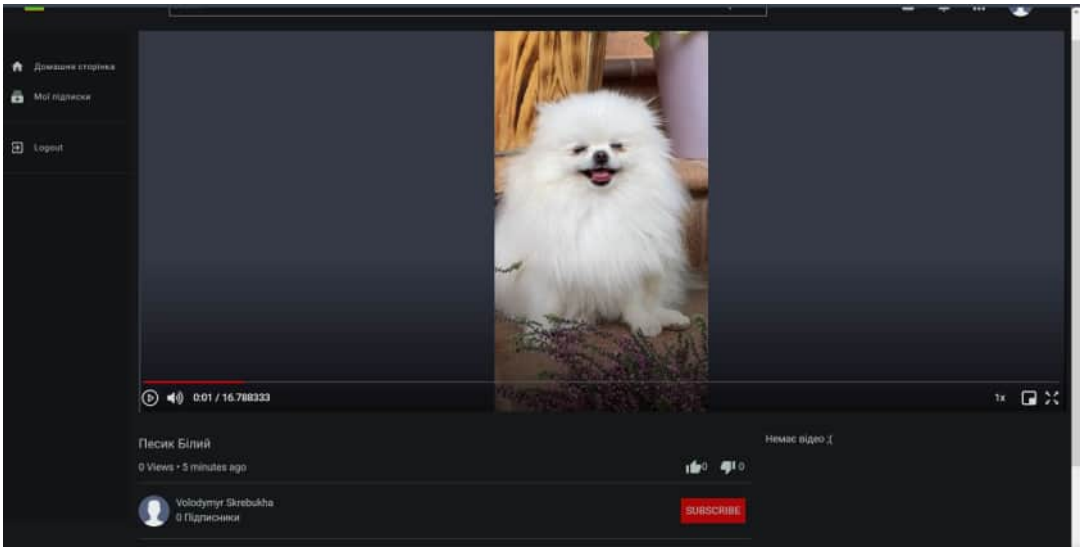

Рис 3.44 Сторінка з відео

Після оновлення інформації у базі даних у нас пявився 1 перегляд, тобто наший

Також для прикладу було оставлено коментар, та поставлений самолайк (Рис 3.45)

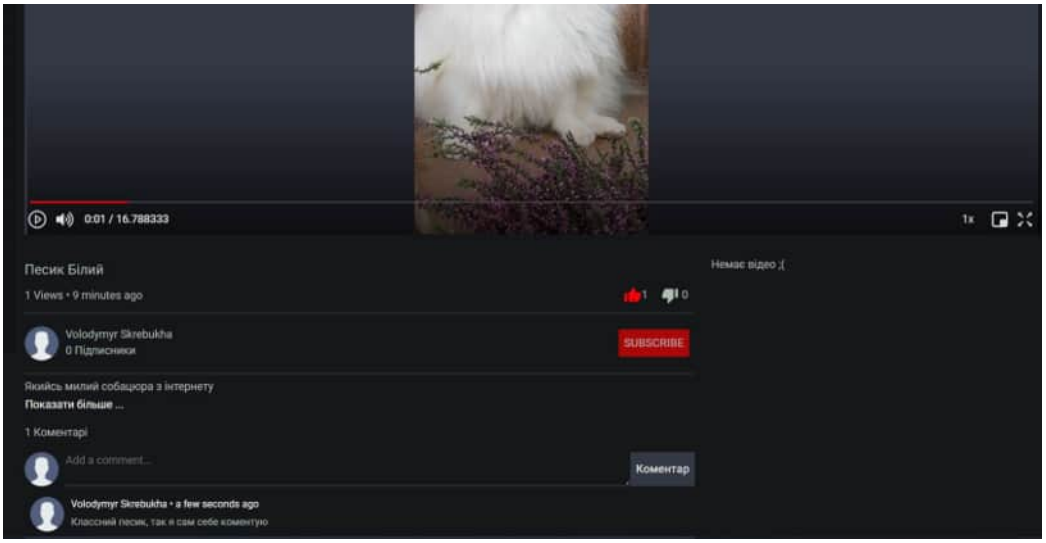

Рис 3.45 Результат після оновлення інформації

3.3.6 Сторінка “Редагування відео”

Переходимо на домашню сторінку, після чого наводимо мишею на наше завантажене відео, у нас появиться кнопка редагувати (Рис 3.46), натиснувши яку нам відкриється нове вікно у якому ми зможемо внести зміни

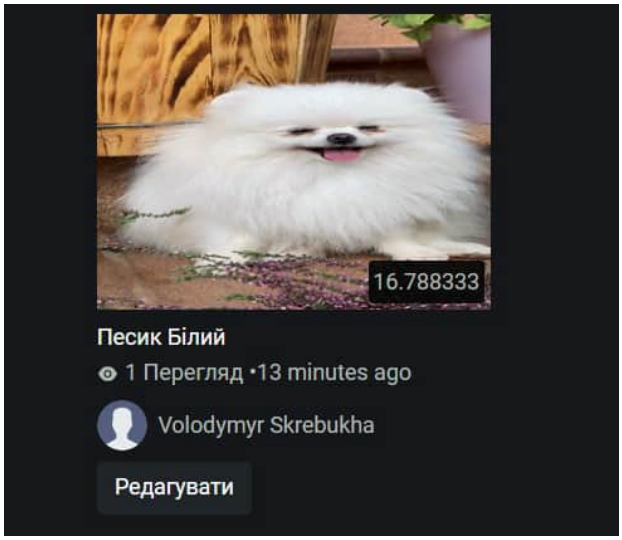

Рис 3.46 кнопка редагування

Як бачимо можна редагувати назву, опис, також можна тепер поставити свою мініатюру яку ми хочемо і в любий час відповідно приховати відео зробивши приватним (Рис 3.47)

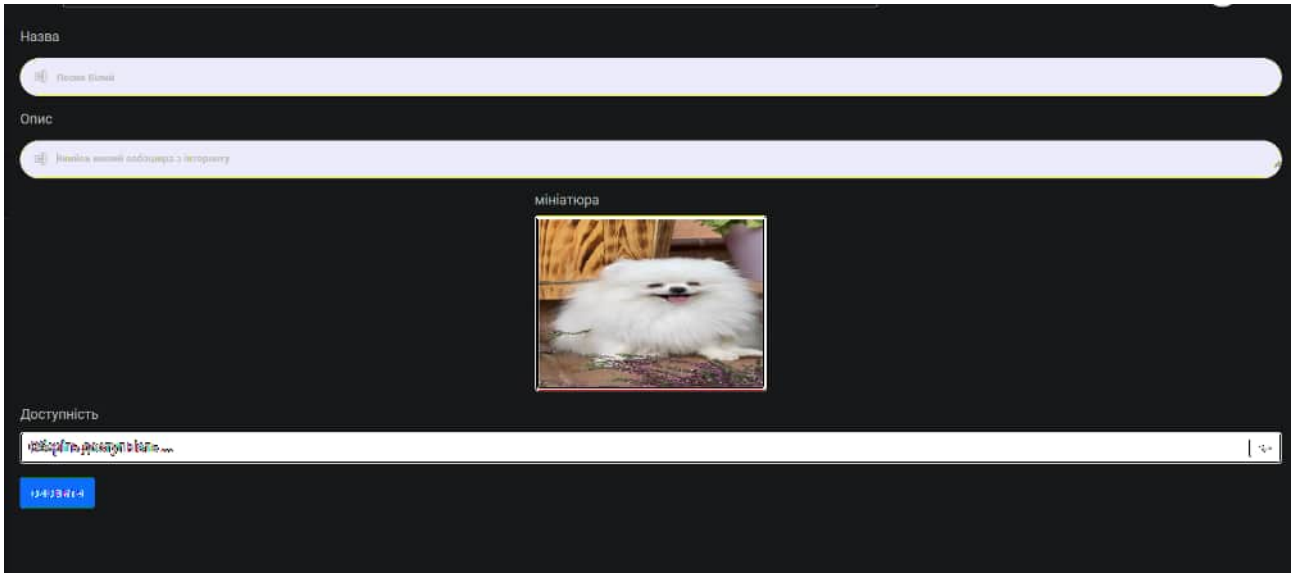

Рис 3.47 Вікно редагування завантаженого відео

3.3.7 Сторінка “Мої підписки”

Для демонстрації працездатності сторінки було створено ще один аккаунт з якого було завантажено декілька відео, перейдемо на одне з них та натиснемо кнопку підписатися

Після натискання на кнопку підписки ми підписуємось на цього користувача, кнопка стає сірою(Рис 3.48)

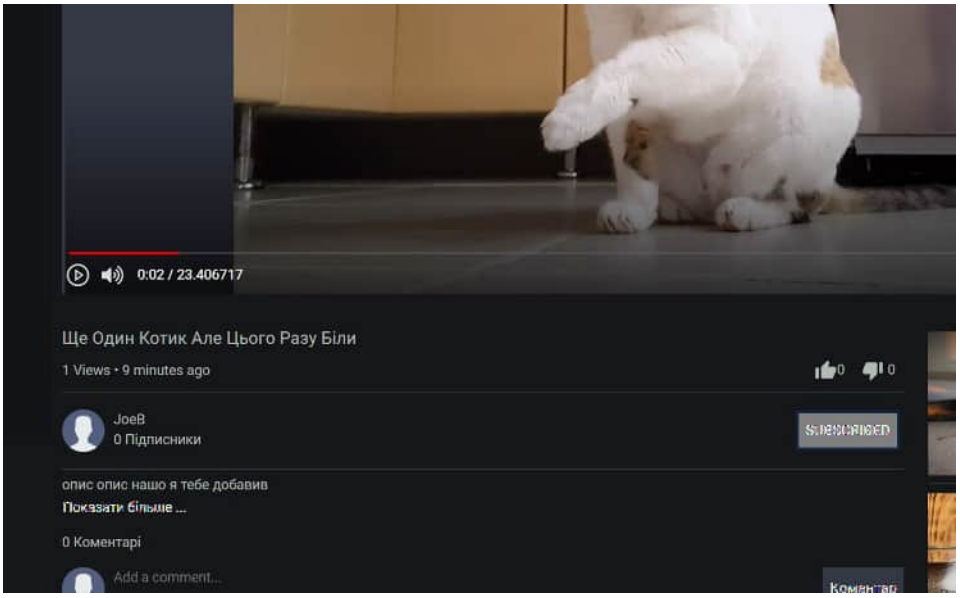

Рис 3.48 Фунціональність кнопки підписки

Тепер перейдемо у сторінку “мої підписки” Як бачимо ми підписані на користувача JoeB (Рис 3.49)

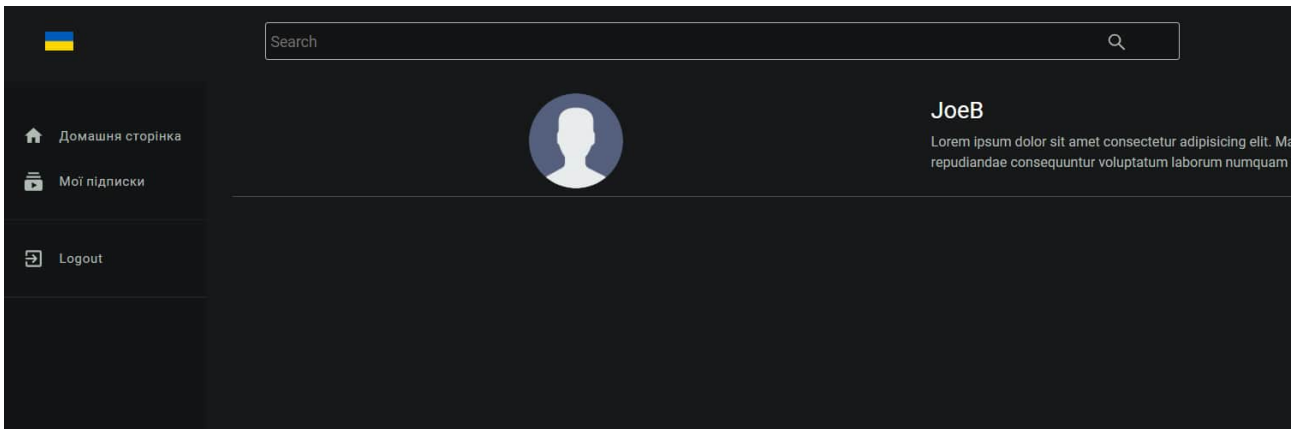

Рис 3.49 Сторінка підписок

Якщо натиснемо на нього – ми перейдемо на його канал (Рис 3.50) та зможемо дивтися тільки його відео. Також можна побачити кількість підписників, наразі ми одні.

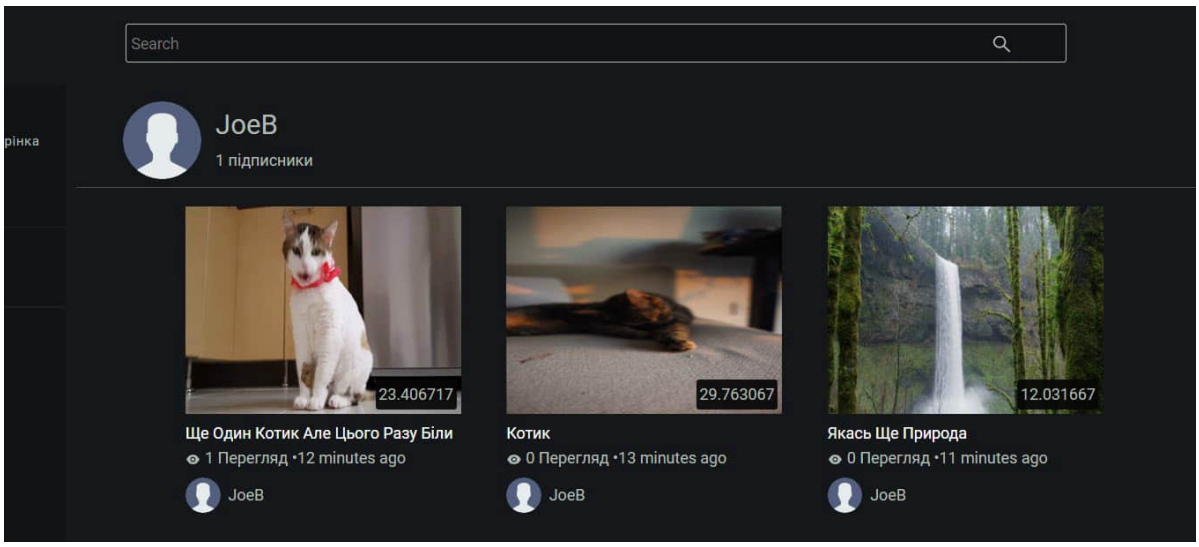

Рис 3.50 Канал користувача

3.3.8 Поле пошуку відео

Продемонструємо поле для пошуку відео, для цього введемо букву або слово якого немає у відео і при спробі пошуку слова “нло” нічого не знайдено (Рис 3.51), воно й не дивно інакше б пошук не коректно працював.

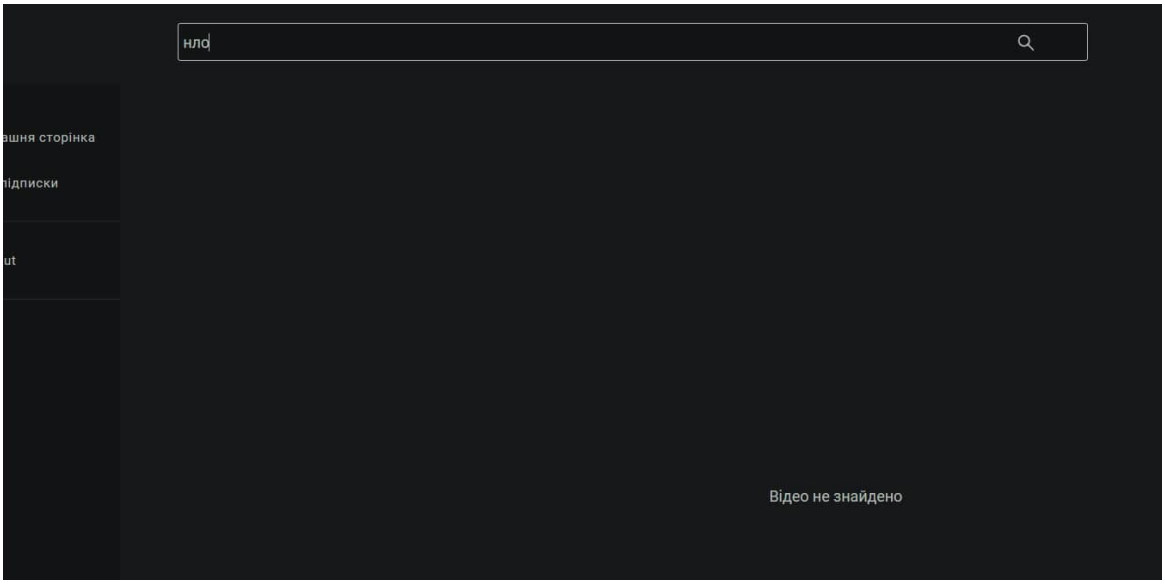

Рис 3.51 Результат введення неіснуючого результату

Тепер введемо запит “котик” І воно знайшло відео з такою назвою (Рис 3.52), а інакше й бути не могло.

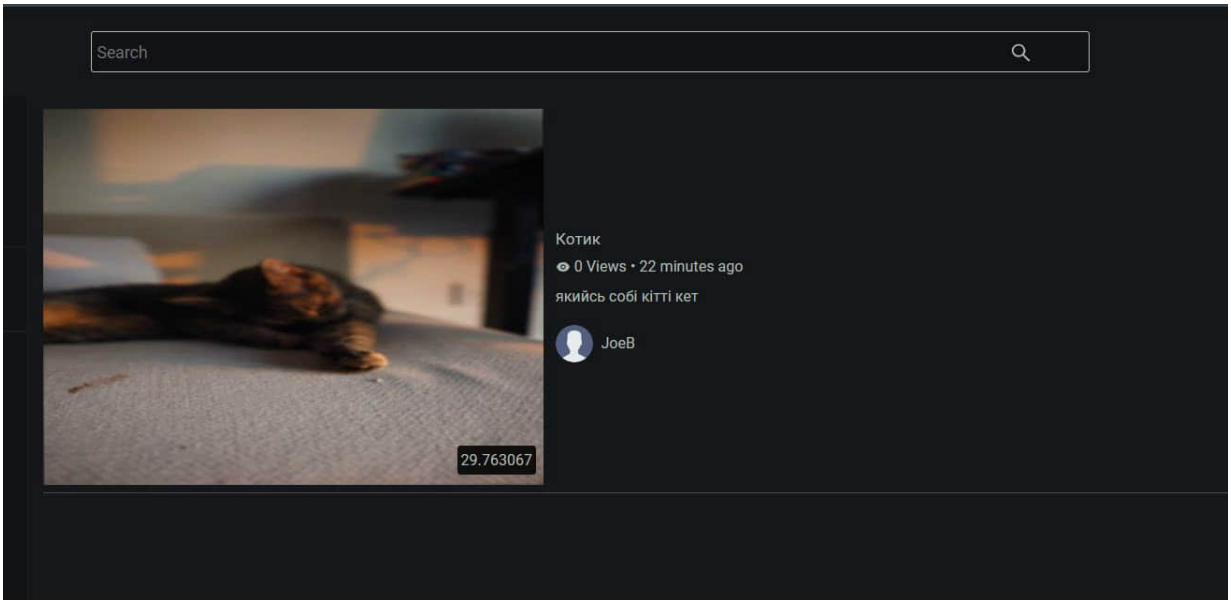

Рис 3.52 Результат за словом “котик”

3.3.9 Сортування по категоріям

Ну і звичайно ж продемонструємо сортування відео по категоріям.

Обираємо категорію “тваринки” (Рис 3.53) після чого обрана опція підсвічується і система нам видає відео які відносяться тільки до тієї категорії при тому ігнорує усі інші. Виберемо категорію “природа” (Рис 3.54) відповідно ми бачимо лише відео яке відноситься до природи

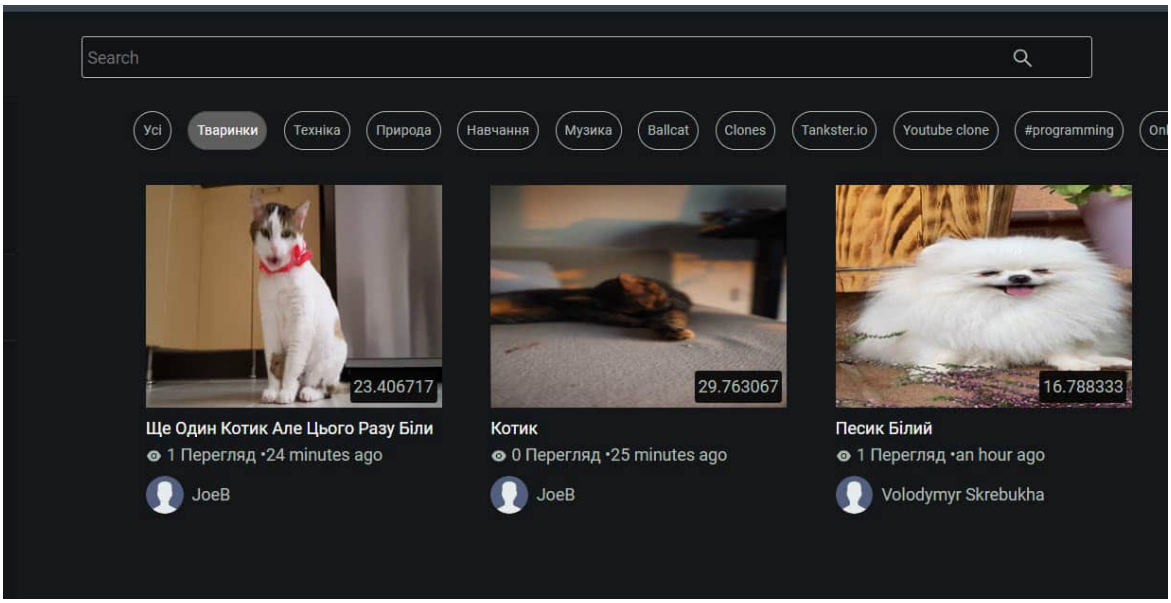

Рис 3.53 Результат вибору категорії “тваринки”

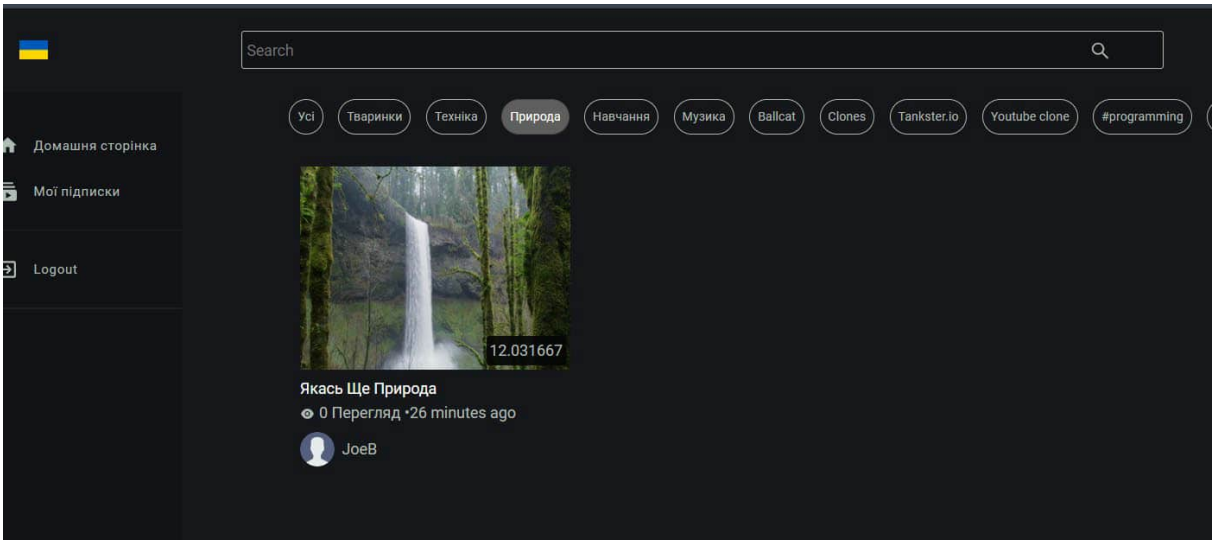

Рис 3.54 Результат вибору категорії “природа”

Висновок

У рамках дипломної роботи було розроблено веб-сайт для завантаження користувацьких відео, що базується на аналізі проблемної області.

Під час аналізу проблемної області були виявлені основні проблеми, з якими зіштовхуються користувачі наявних платформ. До них відносяться обмежена свобода контенту, складний інтерфейс та недостатня конфіденційність.

Враховуючи ці зауваження, веб-сайт був розроблений з урахуванням цих проблем.

В результаті було розроблено веб-сайт для завантаження відео, який надає зручну та просту у використанні платформу.

Забезпечено свободу контенту, де користувачі можуть додавати, переглядати та спільно використовувати відеофайли без обмежень при цьому не порушючи закони моралі та країни у якій розміщується веб-сайт.

Відсутність будь яких засобів для отримання біометричних даних за користувачами в цілях використання продажу, таргетованої реклами то що.

Враховано проблему приватності, де користувачі мають можливість встановлювати рівень приватності для своїх відеоматеріалів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Tripathi, A., & Vasudeva, M. (2020). "TikTok App: A Review of Content, Features, and Privacy Issues." In 2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT) (pp. 1-6). IEEE.

2. Davidson, D. (2020). "Vimeo and TikTok: Exploring Alternative Video-Sharing Platforms." In "Social Media and the Transformation of Interaction in Society" (pp. 113-135). IGI Global.

3. Martindale, M., & Durham, M. G. (2018). "The Vimeo Toolbox: Strategies for Creating Quality Videos." Journal of Electronic Resources in Medical Libraries, 15(3-4), 180-189

4. Hesselberth, P. (2019). "Beyond YouTube: Vimeo as a Platform for Video Activism." In "Vimeo Video School: A Storytelling Guide to Filmmaking" (pp. 131-142). Routledge.

5. Burgess, J., & Green, J. (2013). "YouTube and the Mainstream Media." In "YouTube: Online Video and Participatory Culture" (pp. 39-58). John Wiley & Sons.

6. Gill, P., & Kaur, S. (2020). "A comparative study of video-sharing platforms: YouTube, Vimeo, and Dailymotion." In 2020 International Conference on Intelligent Sustainable Systems (ICISS) (pp. 1252-1257). IEEE.

7. "Web Analytics 2.0: The Art of Online Accountability and Science of Customer Centricity" by Avinash Kaushik Sybex; 1st edition (October 26, 2009)
ISBN-13: 978-0470529393

8. JavaScript: The Good Parts: The Good Parts First Edition O'Reilly Media; First Edition (May 1, 2008) ISBN-13: 978-0596517748

9. TypeScript Documentation - Офіційна документація TypeScript (https://www.typescriptlang.org/docs/)

10. Mastering TypeScript - Second Edition by Nathan Rozentals Packt Publishing; 2nd Revised edition (February 24, 2017) ISBN-13: 978-1786468710

11. Visual Studio Code: End-to-End Editing and Debugging Tools for Web Developers 1st Edition by Bruce Johnson (Author) Wiley; 1st edition (August 23, 2019) ISBN-13: 978-1119588184

12. "Web Development with Node and Express: Leveraging the JavaScript Stack" by Ethan Brown O'Reilly Media; 1st edition (August 5, 2014)
ISBN-13 : 978-1491949306

13. The Firebase Developer Documentation - Офіційна документація Google Firebase (https://firebase.google.com/docs)

14. React Documentation - Офіційна документація React (https://reactjs.org/docs/)

15. "Pro React" by Cassio de Sousa Antonio Publisher : Apress; 1st ed. edition (December 24, 2015) ISBN-13 : 978-1484212615

16. Bootstrap 4 Documentation - Офіційна документація Bootstrap 4 (https://getbootstrap.com/docs/4.0/)

17.Офіційна документація FFmpeg (https://ffmpeg.org/documentation.html)