Вінницький національний технічний університет
(повне найменування вищого навчального закладу)

Факультет інтелектуальних інформаційних технологій та автоматизації
(повне найменування інституту, назва факультету (відділення))

Кафедра комп'ютерних наук
(повна назва кафедри (предметної, циклової комісії))

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Інформаційна технологія організації сервісу відеохостингу»

Виконав: студент 2-го курсу, гр. 3КН-22м
спеціальності 122 «Комп'ютерні науки»
(шифр і назва напряму підготовки, спеціальності)
_____ Кузьменко В.С.
(прізвище та ініціали)

Керівник: д. т. н., проф.
_____ Іванчук Я.В.
(прізвище та ініціали)
«07» 12 2023 р.

Опонент: к.т.н., доцент каф.
_____ Коцюбинський В. Ю.
(прізвище та ініціали)
«07» 12 2023 р.

Допущено до захисту
Завідувач кафедри КН
_____ д.т.н., проф. Яровий А.А.
(прізвище та ініціали)
«08» 12 2023 р.

Вінниця ВНТУ - 2023 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних наук
Рівень вищої освіти ІІ-й (магістерський)
Галузь знань – 12 «Інформаційні технології»
Спеціальність – 122 «Комп'ютерні науки»
Освітньо-професійна програма – «Системи штучного інтелекту»

ЗАТВЕРДЖУЮ
Завідувач кафедри КН
Д.т.н., проф. Яровий А.А.

29.08. 2023 року

З А В Д А Н Н Я

НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТА

Кузьменка Владислава Сергійовича
(прізвище, ім'я, по батькові)

1. Тема роботи Інформаційна технологія організації сервісу відеохостингу
керівник роботи_ д. т. н., проф. Іванчук Я.В.
затверджені наказом вищого навчального закладу від 18.09.2023 року №247.
2. Строк подання студентом роботи 13 листопада 2023 року.
3. Вихідні дані до роботи: діапазон швидкості відтворення відео – 0,5-2х; мова субтитрів – англійські, українські; максимальний об'єм бази даних – 20 ГБт; мінімальна швидкість передачі даних – 500 МБт/с. роздільної здатності – 360р, 480р, 720р, 1080р
4. Зміст текстової частини: вступ, аналіз предметної області організації сервісу відеохостингу, проєктування інформаційної технології організації сервісу відеохостингу, програмна реалізація інформаційної технологій організації відеохостингу, економічна частина, висновки, список використаних джерел.
5. Перелік ілюстративного матеріалу: Схема алгоритму роботи програмного модуля, схема алгоритму роботи рекомендаційного програмного модуля, загальний вигляд головної сторінки додатку, загальний вигляд сторінки авторизації інформаційної технології організації відеохостингу, приклад роботи програми

6. Консультанти розділів роботи

Розділ	Прізвище, ініціалита посада консультанта	Підпис, дата	
		завдання видав	виконанн прийняв
1-3	Іванчук Я.В., д. т. н., проф. каф. КН	19.08.23	10.11.23
4	Кавецький В.В., к. е. н., доц. каф. ЕПВМ	21.10.23	31.10.23

7. Дата видачі завдання 19.08. 2023 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Пр ми
1	Аналіз предметної області організації сервісу відеохостингу	1.09.23 – 07.09.23	
2	Проєктування інформаційної технології організації сервісу відеохостингу	08.09.23 – 20.09.23	
3	Програмна реалізація інформаційної технології організації сервісу	21.09.23 – 20.10.23	
4	Підготовка економічної частини	21.10.23 – 29.10.23	
5	Апробація та/або впровадження результатів дослідження	30.10.23 – 05.11.23	
6	Оформлення пояснювальної записки, графічного матеріалу та презентації	06.11.23 – 10.11.23	

Студент _____ (підпис) Кузьменко В.С.

Керівник роботи _____ (підпис) Іванчук Я.В.

АНОТАЦІЯ

УДК 004.8

Кузьменко В.С. Організація сервісу відеохостингу. Магістерська кваліфікаційна робота зі спеціальності 122 – Комп’ютерні науки, освітня програма – Системи штучного інтелекту. Вінниця: ВНТУ, 2023. 110 с.

На укр. мові. Бібліогр.: 21 назв; рис.: 21; табл. 11.

Магістерська кваліфікаційна робота присвячена розробці програмного модуля для організації сервісу відеохостингу. Програмна реалізація є прикладом використання математичної моделі системи для розв’язання задач рекомендацій. Під час виконання роботи були розроблені та описані структура системи, алгоритм роботи програмного модуля, виконано програмну реалізацію з використання мови програмування JavaScript та середовища розробки Visual Studio Code. Результат магістерської кваліфікаційної роботи може бути використаний як самостійний продукт, що надає змогу переглядати, додавати та отримувати рекомендації щодо перегляду відео.

Ключові слова: відеохостинг, веб-додаток, стримінг, університет, JavaScript, React.

ABSTRACT

Kuzmenko V.S. Organization of video hosting service. Master's thesis on specialty 122 - Computer science, educational program - Artificial intelligence systems. Vinnytsia: VNTU, 2023. 110 p.

In the Ukrainian language. Bibliogr .: 21 titles; fig .: 21; table 11.

The master's thesis is devoted to the development of a software module for the organization of video hosting services. Software implementation is an example of using a mathematical model of the system to solve recommendation problems. During the work, the structure of the system, the algorithm of the software module, and the software implementation using the JavaScript programming language and the Visual Studio Code development environment were developed and described. The result of the master's qualification work can be used as an independent product that allows you to view, add and receive recommendations for watching videos.

Keywords: video hosting, web application, streaming, university, JavaScript, React.

ЗМІСТ

ВСТУП

Актуальність дослідження. У сучасному цифровому світі, де Інтернет є невід'ємною частиною нашого життя, роль відеохостингу [1-3] стає все більш важливою. Їхні можливості включають не тільки завантаження та перегляд відео, а й пряму трансляцію, високоякісні відеоформати, співпрацю зі творцями контенту, а також можливості реклами та монетизації.

Розробка відеохостингу – актуальний та перспективний напрямок у сфері інформаційних технологій, оскільки ця сфера постійно зростає і розвивається. Попит на нові функції, покращена якість відео, безпека користувачів та інтерактивність зростає з кожним днем. споживчі звички та збільшила попит на послуги відеохостингу [4], що призвело до наступних змін.

– Збільшення відеоконтенту: через карантин та обмеження на вихід з дому багато хто звернувся до відеоконтенту як розваги. Я навчаюся і працюю. Відеохостинг став основним джерелом розважальних відеороликів, онлайн-курсів та іншого відеоконтенту.

– Зростання популярності потокового мовлення. Платформи потокового відео в реальному часі, такі як Twitch [5], YouTube Live [6] та Facebook Live [7], стають все більш популярними. Багато хто став споживачами та стрімерами розваг, спілкування та соціальних онлайн-заходів.

– Віддалена робота та навчання: Відеохостинг став корисним інструментом для віддаленої роботи та навчання. Онлайн-конференції, вебінари та відеоуроки стали нормою для багатьох компаній, шкіл та університетів.

– Збільшення кількості користувачів соціальних мереж. Користувачі соціальних мереж, таких як TikTok, Instagram і Snapchat, активно використовували відеохостинги для створення та обміну короткими відеороликами, які набули популярності під час спалаху. Багато компаній використовують відео для презентації своїх продуктів і послуг, а також для відкриття віртуальних магазинів. В цілому з початку пандемії коронавірусу перегляди відео при обміні відео збільшилися на 40%, оскільки розваги,

навчання, спілкування та багато іншого стали невід'ємною частиною життя людей Інтернет. Тому доцільно розвиток інформаційних технологій для організації послуги відеохостингу, що дозволить скоротити час пошуку необхідного контенту та вдосконалити систему видачі рекомендацій для покращення відносин користувача з сервісом.

Зв’язок роботи з науковими програмами, планами, темами. Магістерська робота виконана відповідно до напрямку наукових досліджень кафедри комп’ютерних наук Вінницького національного технічного університету 22 К1 «Розробка прикладних інтелектуальних інформаційних технологій та систем» та плану наукової та навчально-методичної роботи кафедри.

Метою магістерської роботи є підвищення функціональних можливостей сервісу відеохостингу за допомогою розробки рекомендаційної системи, що дозволить підвищити ефективність підбору контенту в залежності від інтересів користувача.

Для досягнення поставленої мети необхідно вирішити наступні **задачі**:

– проаналізувати проблеми надання рекомендацій у відеохостингу;

– виконати аналіз існуючі методи надання рекомендацій для відео-контенту;

– розробка методу надання рекомендацій для сервісу відеохостингу;

– розробка структури роботи програмного модуля надання рекомендацій відеохостингу;

– виконати програмну реалізацію програмного модуля відеохостингу;

– провести тестування програмного модуля надання рекомендацій для сервісу відеохостингу;

– провести розрахунок економічної ефективності науково-технічної розробки за її можливої комерціалізації.

Об’єктом дослідження є процес надання рекомендацій в сервісі відеохостингу.

Предметом дослідження є методи та засоби реалізації програмного модуля сервісу відеохостингу.

Методи дослідження. У роботі використані методи системного аналізу, і об’єктно-орієнтованого програмування, методи дизайну та стилізації веб-додатків, методи оптимізації програмного забезпечення та організації роботи даних та знань.

Наукова новизна отриманих результатів. Розробка інформаційної технології організації сервісу відеохостингу, що включає в собі рекомендаційну систему на основі методу колаборaційної фільтрації даних користувача, що дозволяє значно підвищити ефективність підбору відеопродукції для перегляду.

Практичне значення одержаних результатів. Розроблено алгоритм надання рекомендацій підбору відеопродукції для перегляду користувачам в інформаційній технології організації сервісу відеохостингу.

Достовірність теоретичних положень магістерської кваліфікаційної роботи підтверджується строгістю постановки задач, коректним застосуванням математичних методів під час доведення наукових положень, строгим виведенням аналітичних співвідношень та збіжністю результатів математичного моделювання з результатами, що отримані під час впровадження розроблених програмних засобів.

Апробація результатів роботи. Основні результати досліджень апробовано на LI Науково-технічна конференція підрозділів Вінницького національного технічного університету (10.03.2023 року, м. Вінниця, Україна) [1].

Публікації. За результатами дослідження опубліковано двоє тез доповідей [1, 2].

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ СЕРВІСУ ВІДЕОХОСТИНГУ

1.1 Аналіз історії розвитку сервісів відеохостингу

Відеохостинг – це інтернет-платформа [8], яка надає можливість завантажувати, конвертувати, зберігати, відтворювати та переглядати відео прямо у браузері. Такі сайти зазвичай мають спеціальні програвачі та структуровану систему, яка може приносити дохід. Завантаження відео контенту на відеохостинг здійснюється через веб-сайт, мобільний або настільний додаток або інший інтерфейс (API). Відеохостинги дозволяють завантажувати відео будь-якого типу – від коротких роликів до повнометражних фільмів та шоу. Ці платформи зберігають відео на своїх серверах і надають користувачам можливість використовувати вбудовані коди або посилання для перегляду відео. Зазвичай такі сайти називають веб-сайтами для обміну відео. В останні роки відеохостинги відіграли значну роль у споживанні контенту, рекламній діяльності, освіті та розвагах. Вони стали важливим інструментом для поширення та обміну відео контентом, а їх розвиток торкнувся багатьох сфер нашого життя.

З точки зору бізнесу, відеохостинг є відмінним способом зв'язку з аудиторією, просування власного бренду, реклами, підвищення лояльності клієнтів та виконання інших маркетингових завдань. Відеохостинги користуються як великі корпорації, так і малі підприємства, оскільки ці платформи є ефективним інструментом, здатним приносити доходи при правильному використанні. Для блогерів, музикантів, художників та вчителів відеохостинг представляє чудову нагоду для просування своєї роботи. Більшість платформ для публікації відео пропонують різні способи монетизації контенту, такі як розміщення реклами перед початком відео або всередині нього. Для підприємця публікація контенту на відеохостингу допоможе залучити нових користувачів, підвищити лояльність існуючих клієнтів та забезпечить комунікацію з покупцями, які краще сприймають інформацію у

вигляді відеороликів. Крім того, відеохостинги можна використовувати як сховище, щоб не платити гроші за послуги хмар.

Для роботи з такими майданчиками як глядач не потрібно заздалегідь знати інтерфейс сайту та інформацію про нього. Достатньо відкрити портал і почати перегляд, за бажанням у рядку пошуку можна знайти відеоконтент на певну тематику і іноді є можливість його відсортувати. За бажанням можна зареєструватися, для цього потрібно ввести персональні дані та погодитися з правилами спільноти та політикою конфіденційності даного сервісу.

Для професійної роботи з такими ресурсами як контент-мейкер, користувач достатньо вивчити правила спільноти, основні функції платформи і вибрати собі нішу – тематику, з якою пов'язаний подальший контент. Далі в міру публікації матеріалів, слідує набір аудиторії та переглядів, користувачеві відкриваються функції, пов'язані з монетизацією відеоконтенту.

Щоб працювати з відеохостингами як рекламодавець, необхідно так само вивчити правила та особливості платформи, цільову аудиторію хостингу, дозволені види та тематики реклами, а також ознайомитися з вартістю та способами оплати за розміщення та просування.

З точки зору контент-мейкерів виділяють відеохостинги з ухилом у стрімінг та публікацію відео. Однак можливості даних порталів найчастіше є однаковими, оскільки наявність функцій вибору контенту збільшує цільову аудиторію та кількість проведеного часу на сайті. Найчастіше стримери використовують плагіни та інструменти для рестрінгу, завдяки чому можна одночасно транслювати відео-потік на відеохостинги орієнтовані тільки на стрімінг або лише публікацію відео. Таким чином автор може набрати найбільшу кількість глядачів на трансляції та отримати більше переваг. Існує можливість збереження запису трансляції або її публікації на інші ресурси. Той чи інший відеохостинг став популярним лише завдяки аудиторії, зацікавленій у конкретному контенті.

Під контентом розуміється якісно представлена та упакована корисна інформація. В даний час основними форматами контенту є статті, блоги

експертів, відео огляди, інфографіка, кейси та презентації. Багато фахівців класифікують контент як навчальний, інформаційний та розважальний. Проте мета будь-якого контенту – продаж: товару, себе як автора чи нового медіа ресурсу широкої аудиторії. Тому ефективний контент має бути інформативним, пізнавальним та певною мірою розважальним – все це становить його привабливість. Контент може бути як унікальним, так і ні, а також безкоштовним або платним.

Перевага відеоконтенту в тому, що він сприймається легше і викликає більшу довіру, ніж текстовий формат. Різні відеоролики мають великий попит, від стандартних рекламних роликів, які ми бачимо по телевізору, до навчальних відео, де користувачі шукають відповіді на свої запитання та одержують вирішення проблеми. Відео можна успішно комбінувати з майстер-класом у статті або з оглядовими матеріалами. Тому відеохостинги є важливим інструментом для створення та розповсюдження захоплюючого та корисного контенту.

Сучасні тенденції розвитку відеохостингів:

– Розширення функціональності стало однією з головних тенденцій у розвитку відеохостингів. Популярні платформи, такі як YouTube, Vimeo та Dailymotion, постійно впроваджують нові функції, що полегшують процес створення, редагування та розповсюдження відео. Наприклад, користувачі можуть транслювати відео в реальному часі, використовувати штучний інтелект для автоматичної обробки контенту і навіть зануритися у віртуальну реальність.

– Активна адаптація відеохостингів під мобільні пристрої є ще однією важливою тенденцією. Зі збільшенням популярності смартфонів та планшетів, програми для мобільних пристроїв дозволяють користувачам завантажувати, переглядати та ділитися відео у будь-який час та в будь-якому місці. Це дозволяє розширити аудиторію та збільшити час, який користувачі проводять, переглядаючи відеоконтент.

– Якість відео на відеохостингах також постійно зростає. Відео високої роздільної здатності, такі як 4K і HDR, стають стандартом на багатьох платформах, що забезпечує кращий візуальний досвід для глядачів та дозволяє творцям контенту уявити свою роботу в найкращому світлі.

– Великі відеохостинги активно співпрацюють із креаторами контенту, що сприяє створенню нового та цікавого відеоконтенту. Вони надають інструменти для монетизації контенту, такі як рекламні партнерства та платні передплати фанатам.

– Відеохостинги також стали важливими платформами для реклами та монетизації контенту. Рекламні ролики перед відео та спонсорський контент дозволяють творцям відео отримувати прибуток від своїх робіт. Деякі платформи навіть дозволяють користувачам підписуватись на платні підписки для доступу до ексклюзивного контенту.

– Розвиток відеохостингів в останні роки яскраво показує їхню важливість у сучасному медіапросторі. Вони виявляються не тільки місцем споживання відеоконтенту, а й інструментом самовираження, реклами, монетизації та спілкування. Розширення функціональності, мобільність, збільшення якості відео та співпраця з креаторами контенту роблять відеохостинги невід'ємною частиною нашого життя.

Переваги відеохостингів для користувачів:

– Скорочення витрат за пропускну спроможність і хостинг, що найчастіше остаточно виключає витрати.

– Створення майданчика для обміну та спільного перегляду відеоконтенту.

– Надання зручного середовища для користувачів, де завантаження та потокова передача відео, а також вбудовування відео здійснюються без необхідності знання програмування. Для цього достатньо використати веб-браузер.

Багато відеохостингів не обмежуються у своєму змісті лише однією тематикою. Однак, деякі з них спеціалізуються на певних напрямках,

пропонуючи тематичні портали. Особлива увага приділяється науковому, науково-популярному та навчальному відеоконтенту. Деякі веб-сайти вважають за краще використовувати відеоформати без виплати роялті, такі як Theora (з OGG) та VP8 (з WebM).

1.2 Аналіз відомих технічних рішень систем організації сервісу відеохостингу

Наразі існує безліч сервісів, які надають можливість перегляду відео та рекомендують контент на основі ваших попередніх переглядів. Давайте розглянемо кілька прикладів таких відеохостингів.

YouTube – найбільша відео-спільнота у світі [9]. Основна мета цієї платформи полягає у перегляді та завантаженні відео від користувачів з усього світу в інтернеті. (рис. 1.1).

Технічні особливості для контенту завантажуваного на YouTube, що визначають розмір відео для YouTube за пропорціями, роздільною здатністю та співвідношенням сторін:

- максимальний розмір файлу – 128 ГБ;
- максимальна тривалість відео – 12 год.;
- максимальна роздільна здатність відео – 2160p;
- мінімальна роздільна здатність відео – 240p;
- оптимальне співвідношення сторін – 16:9;
- допустимі формати відео: .mov, .mpeg, .avi, .wmv, .mpegps, .flv, .webM, .3GPP.

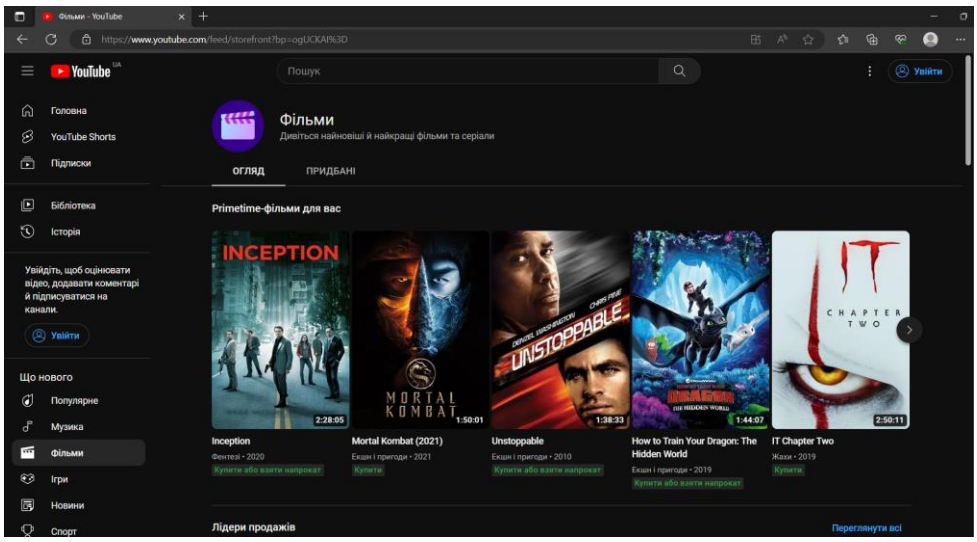

Рисунок 1.1 – Загальний вигляд інтерфейсного вікна сервісу YouTube

Netflix - це американська компанія, що надає стрімінговий відеосервіс і виробляє розважальні медіаконтенти [10]. Минулого року Netflix став найбільшою компанією у сфері розважальних медіа. Крім надання ліцензованого контенту для стрімінгу, Netflix також створює та випускає власні телешоу та повнометражні фільми (рис. 1.2).

Netflix приділила значну увагу розробці алгоритму, здатного визначити переваги користувача і швидко запропонувати йому товар, що цікавить, з каталогу компанії. На початку свого розвитку Netflix не надавав такої особливої значущості даної особливості, оскільки громадськість головним чином говорила про якість продуктів компанії в естетичному та змістовному плані. Однак сьогодні алгоритм рекомендацій Netflix став у центрі уваги. Його система рекомендацій настільки цінна, що в жовтні 2006 року компанія запустила конкурс Netflix Prize, пропонуючи 1 мільйон доларів першій команді, яка розробить алгоритми, здатні передбачати рейтинги фільмів щонайменше на 10% точніше, ніж існуюча система Cinematch. Завершившись 21 вересня 2009 року, конкурс зібрав понад 50 тисяч учасників із 186 країн.

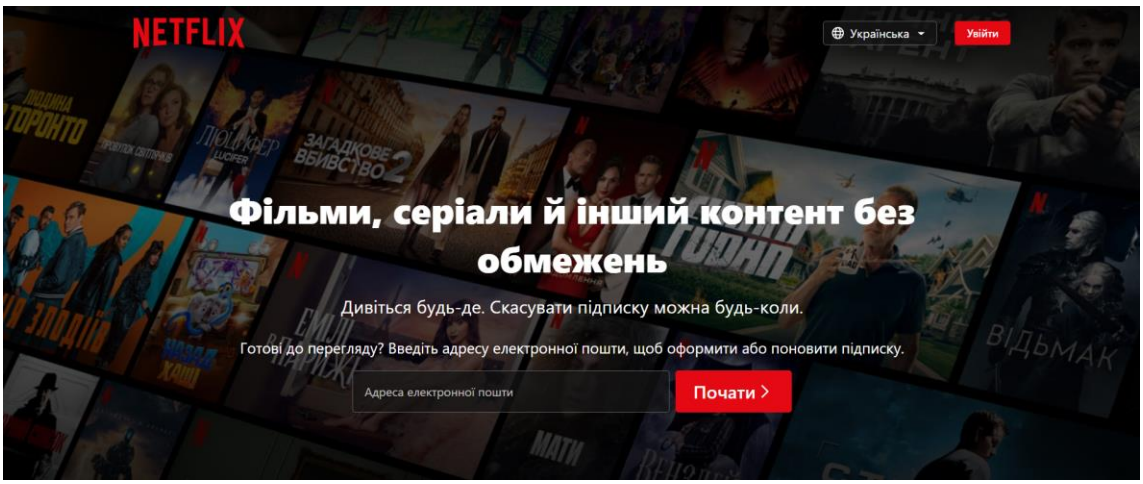

Рисунок 1.2 – Загальний вигляд інтерфейсного вікна сервісу Netflix

З часом система рекомендацій Netflix стала незамінним інструментом для рекомендації контенту користувачам та персоналізації їх досвіду на платформі. Вона є сукупністю різних обчислювальних інструментів, розроблених компанією, починаючи з 2000-х років. Система відповідає за приблизно 80% загального часу перегляду на Netflix, тоді як решта 20% здійснюється через пошук. Для того, щоб запропонувати контент користувачу, система використовує комбінацію фільтрації на основі вмісту та алгоритмів спільної фільтрації.

Фільтрація на основі вмісту спирається на індивідуальні дані користувача, такі як його історія переглядів, поведінка прокручування та тривалість перегляду, щоб запропонувати йому контент, подібний до того, який він вже показав інтерес раніше. Алгоритми спільної фільтрації, на свою чергу, аналізують уподобання та інтереси інших користувачів зі схожими смаками та ґрунтуються на глобальних тенденціях, щоб надати рекомендації контенту.

Vimeo, інтернет-платформа для обміну відео [11], з'явився в 2004 році як побічний проект Джейка Лодвіка та Зака Кляйна - веб-розробників, пов'язаних із засновниками гумористичного сайту CollegeHumor. Вони хотіли створити місце, де користувачі могли б ділитися короткими відео та переглядати їх разом зі своїми друзями. Ідея виникла після того, як деякі відео, розміщені на CollegeHumor, стали користуватися величезною популярністю, включаючи

відеокліп із жовтневого випуску шоу "Суботній вечір у прямому ефірі". У цьому відео випадково синхронізація губ Ашлі Сімпсон викликала величезний резонанс. Спочатку, у спробах залучити аудиторію, Vimeo розглядався як доповнення до CollegeHumor, але з часом розробники сконцентрувалися на покращенні основного сайту.

У перші роки Vimeo приваблював лише невелику аудиторію, переважно завдяки позитивним відгукам. Однак, у міру розвитку, сервіс став набирати популярності та притягувати більше користувачів. Vimeo був унікальним відеохостингом, який пропонував безрекламне перегляд відео, швидке завантаження з мінімальними проблемами, можливість створювати канали та групи, здійснювати приватність відео, мати доступ до покращених статистичних даних про перегляди та навіть заробляти гроші через інтернет. Крім того, відео з Vimeo можна було вбудовувати на різні веб-платформи, включно з блогами та односторінковими сайтами.

Проте, на безкоштовній версії сервісу були деякі обмеження. Відеоролики містили рекламу, користувачі могли завантажувати відео лише один раз на тиждень, причому розмір файлу обмежений 500 МБ. Також були обмеження при створенні груп та каналів. Крім того, не можна було вбудовувати відео з Vimeo на будь-який сайт.

Загальні висновки про Vimeo: платформа пропонує чудові умови для обміну та перегляду відео, проте безкоштовна версія має свої недоліки, які слід враховувати під час використання сервісу.

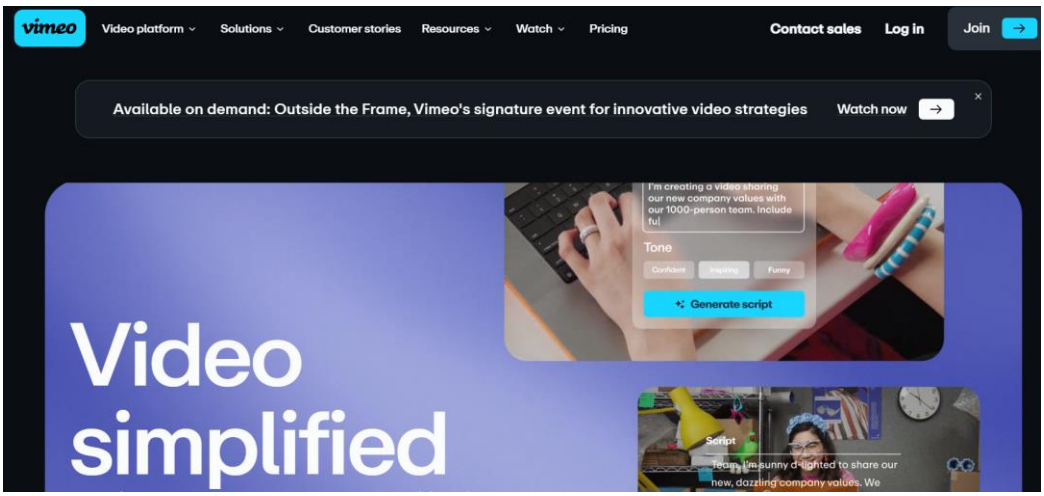

Рисунок 1.3 – Загальний вигляд інтерфейсного вікна сервісу Vimeo

Twitch – це популярна платформа для стримінгу відеоігор та прямих ефірів, яка була заснована у 2011 році. Проте історія її появи почалася задовго до того. У 2007 році Джастін Кан та Емметт Шир запустили веб-платформу, де користувачі могли транслювати своє життя в режимі реального часу. Проте, вже тоді було помічено, що ігрові стрими мають найбільшу популярність. 17 червня 2011 року в офіційному блозі оголосили про створення офіційної платформи Twitch.tv. За рік активність на платформі збільшилась у кілька разів, і вже у 2012 році було здійснено понад 20 мільярдів трансляцій.

Таким чином, платформа Twitch зазнала величезного розвитку за короткий час з моменту її запуску. Сьогодні Twitch є однією з найважливіших платформ для ігрового стримінгу та збирає величезну кількість користувачів з усього світу.

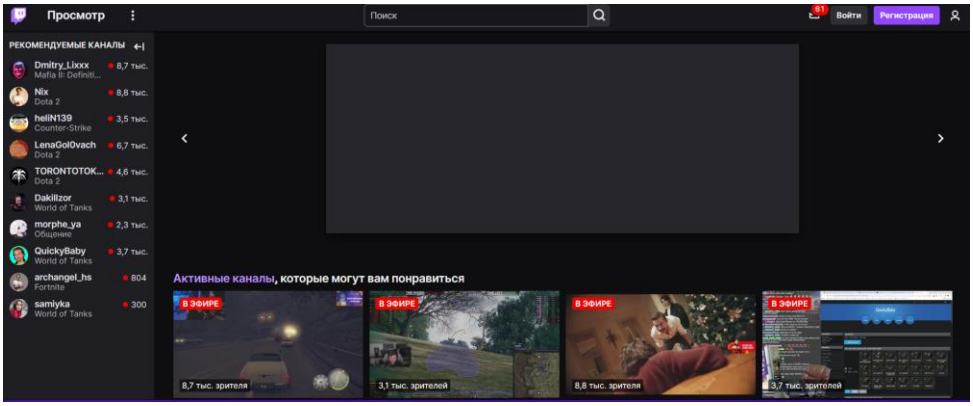

Рисунок 1.4 – Загальний вигляд інтерфейсного вікна сервісу Twitch

Hulu – американська платформа для онлайн-трансляції, орієнтована насамперед на серіали. Створена спільними зусиллями компаній NBC та Comcast, нині її власником є Disney [12]. Назва "Hulu" походить від китайських слів, що мають два значення: "гарбуз" та "інтерактивний запис". Завдяки високому позиціонуванню як сервіс преміум-класу Hulu за 13 років залучив понад 38 мільйонів постійних користувачів (рис. 1.4).

Hulu є одним з небагатьох сервісів у США, що надають доступ до контенту основних телевізійних мереж та можливість переглядати новини та спортивні програми у прямому ефірі. Крім того, Hulu уклав угоди з декількома звукозаписними лейблами, що дозволяє транслювати музичні кліпи та концерти.

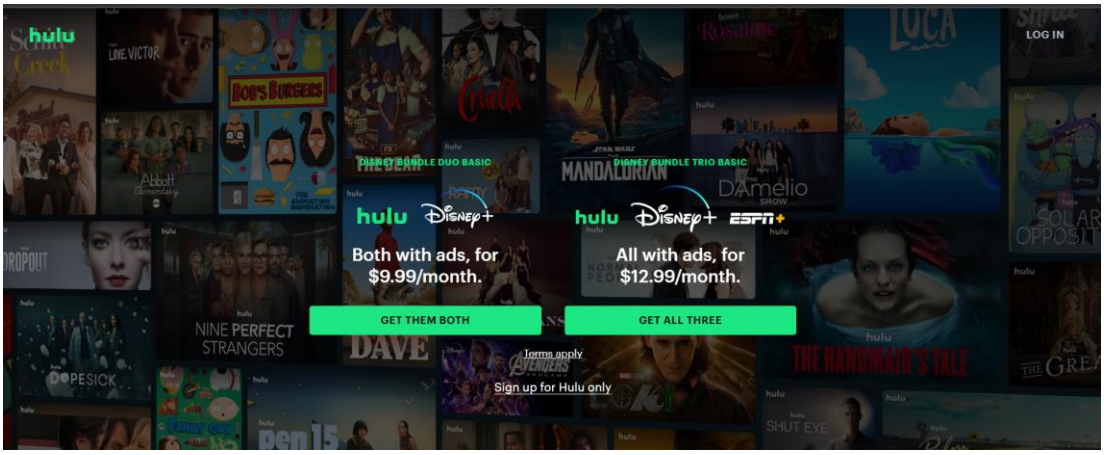

Рисунок 1.5 – Загальний вигляд інтерфейсного вікна сервісу Hulu

Таблиця 1.1 – Порівняльний аналіз сервісів відеохостингу

	YouTube	Netflix	Hulu
Надання рекомендацій	+	+	+
Можливість завантаження власних відео	+	-	-
Безкоштовний перегляд	+	-	+

Серед проаналізованих програмних систем YouTube є максимально наближеним за своїми характеристиками до розроблюваного програмного модуля сервісу відеохостингу. Саме тому було обрано його як прототип.

1.3 Аналіз методів надання рекомендацій в сервісах відеохостингу

Розробка ефективної рекомендаційної системи стає все більш актуальним завданням у різних галузях, таких як інтернет-магазини, пошукові системи, мультимедіа-сервіси та ЗМІ. Основна мета цих систем – персоналізувати контент, тобто адаптувати його автоматично під унікальні потреби кожного користувача. Незважаючи на те, що рекомендаційні системи присутні в онлайн-системах вже понад двадцять років, справді успішних прикладів їх застосування поки що не так багато. На сьогоднішній день активно проводяться дослідження у цій галузі, оскільки все ще існують невирішені питання та невизначеності у методах побудови таких систем.

Рекомендаційна система - це комплекс програм, який визначає інтереси та переваги користувача, а потім формує контентні пропозиції, що відповідають цим інтересам. Такі системи змінили взаємодію між програмними системами та користувачами – замість статичної інформації система може динамічно змінюватися та підлаштовуватися під кожного користувача, що збільшує ступінь взаємодії та надає ширші можливості користувачеві.

Рекомендаційна система включає дві основні сутності - користувача, який є суб'єктом рекомендацій, і товар, який є об'єктом рекомендацій. Залежно від контексту, товаром може бути товар, фільм, трек, книга, новина, онлайн-курс тощо.

Основним завданням рекомендаційної системи є визначення об'єкта, який невідомий користувачеві, але може бути корисним або цікавим для нього в контексті. Рекомендації використовуються у різних галузях, наприклад, в інтернет-магазинах, де пропонується вибрати товари в розділах "з цим товаром купують" або "вам може сподобатися".

Також рекомендаційні системи застосовуються у медіа для показу користувачеві цікавих матеріалів та у соціальних мережах для пропозиції додати до друзів певних людей. Відео та музичні стрімінги також використовують рекомендаційні системи для формування персоналізованих плейлистів або рекомендацій фільмів для конкретного користувача. Наприклад, якщо ви переглянули фільм в онлайн-кінотеатрі, алгоритм підбиратиме для вас нові фільми, враховуючи жанр, акторів, рік випуску, режисера, а також переваги інших користувачів, які також дивилися цей фільм.

Існують такі алгоритми рекомендаційних систем:

Спочатку відбувається збір даних про користувачів, включаючи інформацію про попередні покупки, оцінки, перегляди, а також відвідані сторінки та соціальні контакти. Потім на основі зібраних даних створюються профілі користувачів, в яких відображаються їхні переваги та інтереси.

Потім рекомендаційна система за спеціально розробленим алгоритмом оцінює схожість між споживачами та об'єктами. При цьому аналізується схожість профілів користувачів та схожості між елементами. Наприклад, якщо користувач А і користувач Б мають схожі переваги, то елемент, який подобається користувачеві А, швидше за все, сподобається і користувачеві Б.

На наступному кроці рекомендаційні системи застосовують фільтрацію до величезного обсягу контенту та визначають, які елементи є найбільш релевантними для певного користувача. Можуть бути задіяні фільтри за жанром, часом, місцезнаходженням або іншими параметрами.

Також алгоритми враховують реакцію користувача на пропоновані елементи, щоб працювати над покращенням рекомендацій, оновленням моделі на основі цієї інформації. Наприклад, якщо глядач онлайн кінотеатру позитивно оцінив запропонований фільм, алгоритм врахує це при формуванні рекомендацій у майбутньому.

Серед систем видачі рекомендацій можна виділити чотири основні види [13], що відрізняються наборами алгоритмів та мають свої особливості прийняття рішень:

– Фільтрування на основі контенту. Модель, заснована на контенті, є найпростішим і очевидним методом визначення переваг і інтересів користувача. Суть контентно-орієнтованого підходу полягає в зіставленні користувачів з товарами, послугами або контентом, які їм сподобалися або були придбані раніше. Такі системи добре поводяться, наприклад, для рекомендацій фільмів. Якщо людина переглянула бойовик, сервіс запропонує йому ще кілька фільмів аналогічного жанру.

– Колаборативна фільтрація. Принцип роботи методу колаборативної фільтрації – генерування рекомендацій на основі даних про інших споживачів зі схожими інтересами. Рекомендаційна система, що використовує такий підхід, пропонує продукти як на підставі оцінок самого користувача, так і ґрунтуючись на явних та неявних уподобаннях інших покупців. Наприклад, якщо людина купила пральну машину, алгоритм запропонує товари, які зацікавили інших покупців пральних машин, а не машину іншого бренду.

– Фільтрування, засноване на знаннях. Складнішим способом видачі рекомендацій є фільтрація на основі знань. Цей метод передбачає збирання відомостей про переваги потенційного покупця за допомогою максимальної деталізації його запитів. Такий вид рекомендаційних систем використовується, наприклад, при продажу будинків: щоб система вибрала з каталогу відповідний для покупця варіант, необхідно зібрати дані про бажану кількість поверхів, площу будинку, матеріал стін і безлічі інших параметрів.

– Гібридні рекомендаційні системи. Комбіновані системи рекомендацій передбачають використання різних комбінацій перерахованих вище методів фільтрації. Наприклад, в інтернет-магазинах одягу в рекомендаціях потенційному покупцеві показують і речі, схожі на ті, що він уже дивився, і ті, що замовили користувачі зі схожими уподобаннями. Таким чином, система застосовує механізми відбору на основі контенту та колаборативну фільтрацію одночасно.

Універсального методу видачі рекомендацій не існує, при цьому гібридний підхід дозволяє отримати найточніші рекомендації на основі наявних даних при витраті мінімальної кількості часу.

Одна з найкращих систем надання рекомендацій у відеоплатформах наразі є компанія Netflix. Коли ви заходите в сервіс Netflix, щоб переглянути серіал або фільм, система рекомендацій допомагає зробити вибір без особливих зусиль. Імовірність того, що користувач вирішить подивитися конкретне відео з каталогу, оцінюється на основі багатьох факторів, зокрема:

- особливостей взаємодії із сервісом, наприклад, історії перегляду й оцінок, виставлених іншим відео;
- вибору інших користувачів сервісу зі схожими уподобаннями;
- даних про сам фільм чи серіал, як-от жанр, категорія, акторський склад, рік випуску тощо.

Крім інформації про те, що саме користувач подивився раніше на Netflix, система також використовує дані про те, як користувач це робить, а саме:

- у який час доби ви дивитеся контент;
- на яких пристроях;
- наскільки довго відбувся перегляд відео.

Усі ці дані оцінюються та обробляються за допомогою спеціального алгоритму. Під час прийняття рішень система рекомендацій не використовує демографічну інформацію, наприклад, дані про стать або вік користувача.

Перші результати в списку залежать від дій і вибору інших користувачів, які вводили такі самі або схожі запити. На домашній сторінці Netflix система спеціально для підбирає відео, які будуть включені в кожен рядок, надає цим відео рейтинг, а потім вибудовує порядок показу самих рядів за допомогою складних алгоритмів. Рекомендовані відео системою Netflix у розміщуються так, щоб користувач точно міг знайти для себе щось цікаве.

У кожному рядку є три рівні персоналізації: вибір ряду, який відображатиметься на сторінці; вибір контенту, що міститиметься в ряду; порядок розташування відео.

Для бізнесу системи рекомендацій приносять очевидну вигоду тим, що показують користувачам саме те, що їм цікаво, що призводить до зростання продажів товарів та послуг та збільшення прибутку. У онлайн-магазинах рекомендації працюють на підняття середнього чека та до продажу, а в медіа та онлайн-кінотеатрах – на залучення, оскільки відвідувачу не потрібно шукати, щоб ще подивитися чи прочитати, при цьому він затримується на сайті довше.

Завдяки точним рекомендаціям користувачі відчувають більшу зацікавленість у залишенні на платформі, дивитись більше вмісту та використанні більше сервісів. У бізнес-сфері системи рекомендацій використовуються для підвищення продажів через рекомендації товарів або послуг, які можуть зацікавити конкретного користувача.

Інформація, зібрана системами рекомендацій, може бути використана для аналізу поведінки користувачів та вдосконалення стратегій залучення та утримання аудиторії.

Користь для відвідувачів полягає у комфортніших умовах використання сайту, допомоги з вибором товару чи контенту. За наявності якісних рекомендацій користувачеві не витрачати багато часу, щоб знайти те, що він шукає. Системи рекомендацій дозволяють адаптувати вміст до конкретних інтересів та потреб користувачів. Це забезпечує зручний інтерфейс та покращує взаємодію користувача з платформою.

Системи рекомендацій допомагають користувачам ефективно витрачати свій час, швидко знаходячи те, що їх цікавить, без необхідності витрачати час на пошук.

До мінусів рекомендаційних систем можна віднести звикання користувачів до підказок при виборі книг, фільмів та музики, через що у деяких споживачів може зникнути стимул до вивчення нових незвичних жанрів, оскільки все, що їм пропонується, їх і так влаштовує. Також через брак альтернативної інформації розумна стрічка новин у соцмережах може вплинути на формування світогляду користувачів, позначитися на їхніх поглядах.

Рекомендаційна система відучує замислюватися про свої бажання, покладаючись натомість на алгоритм.

Системи рекомендацій корисні для будь-якого бізнесу з вибором товарів, а також з історією взаємодії користувачів з цими продуктами. Якщо в магазин приходить новий клієнт, який поки не має історії замовлень, алгоритм не знатиме, що йому показувати. Така проблема вирішується накопиченням історії, наприклад, якщо до магазину надійшла нова колекція, з товарами з якої ще ніхто не взаємодіяв, можна створити віджет із цими продуктами, щоб користувачі почали взаємодіяти з ними, при цьому система рекомендацій отримає історію та зможе виділити найпопулярніші товари.

1.4 Висновок до розділу 1

Аналіз розвитку сервісу відеохостингу свідчить про важливість та актуальність цієї галузі інформаційних технологій у сучасному світі. Відеохостинги стали не лише майданчиками для розміщення та спільного ділення відеоконтенту, але й ключовими факторами у розвитку інтернет-економіки, медіа-індустрії, освіти, та багатьох інших сферах. Важливим є розуміння того, що розвиток відеохостингу є динамічним процесом, який вимагає постійної адаптації до змін у технологіях та користувацьких вимогах. Ми порівняли найпопулярніші рішення систем відеохостингу автоматизації, оцінивши їх параметри, плюси і мінуси використання користувачем.

2 ПРОЕКТУВАННЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ ОРГАНІЗАЦІЇ СЕРВІСУ ВІДЕОХОСТИНГУ

2.1 Розробка математичної моделі рекомендаційної системи в сервісах відеохостингу

Основною концепцією у колаборативній фільтрації [13] для формування рекомендацій відео на основі оцінок користувачів, які взаємодіють із системою схожим чином. Рекомендаційна система аналізує профілі користувачів і знаходить подібних користувачів, щоб розрахувати оцінку нового об'єкта з їх оцінок.

Існують два підходи до колаборативної фільтрації: колаборативна фільтрація, заснована на користувачах (user-based) та колаборативна фільтрація, заснована на предметах (item-based). Перший підхід оперує подібністю користувачів, тоді як другий підхід оперує подібністю предметів.

Подібність між користувачами та предметами розраховується через коефіцієнт кореляції між багатовимірними векторами, які представляють рейтинги користувачів для колаборативної фільтрації на основі користувачів та рейтинги предметів для item-based фільтрації. Для цього можна використовувати коефіцієнт кореляції Пірсона чи косинус кута між векторами. Важливою умовою є нормованість, оскільки значення рейтингів перебувають у інтервалі [0,1]. У випадку item-based рекомендаційних методів добре зарекомендував себе уточнений косинус кута, який враховує середні значення рейтингу r^- для даного користувача, як показано у виразі.

Уточнений косинус кута допомагає враховувати різні підходи користувачів до формування рейтингу, де деякі користувачі можуть віддавати перевагу лише високим оцінкам, в той час як інші користувачі можуть рейтингувати, ґрунтуючись на характеристиках предметів.

$$sim(u,i) = \frac{\sum_{u \epsilon U_{ij}} (r_{u,i} - \bar{r}_u)(r_{u,j} - \bar{r}_u)}{\sqrt{\sum_{u \epsilon U} (r_{u,i} - \bar{r}_u)^2} \sqrt{\sum_{u \epsilon U} (r_{u,j} - \bar{r}_u)^2}}, \quad (2.1)$$

де $U_{ij} \subseteq U$ – множина користувачів, оцінивших предмети i, j.

Схема принципу колаборативної фільтрації зображена на рисунку 2.1 та 2.2

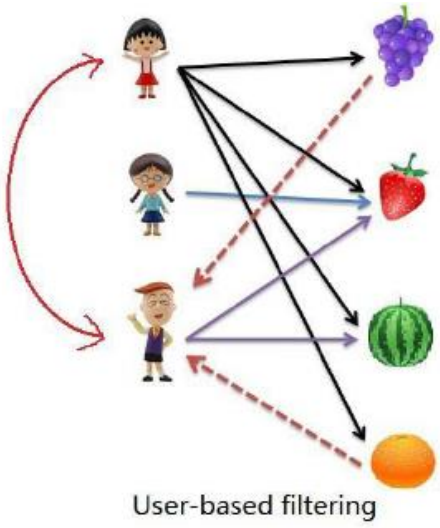

Рисунок 2.1 – Принципова схема User-based колаборативної фільтрації

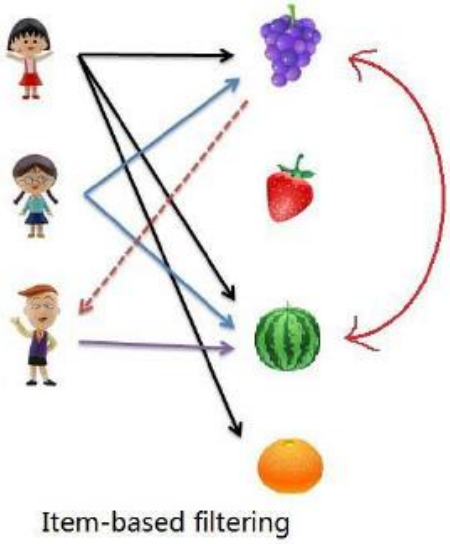

Рисунок 2.2 – Принципова схема Item-based колаборативної фільтрації

Однак у колаборативній фільтрації виникають кілька основних проблем: розмір матриці рейтингів, розрідженість цієї матриці та проблема холодного старту. Тим не менш, облік історії попередніх оцінок при використанні колаборативної фільтрації робить її потужним, ефективним та популярним методом.

Нехай було задано певну множину користувачів $u \in U$, множину об'єктів $i \in I$ та множину подій(діяльність, що виконують над об'єктами) $(r_{ui}, r, i) \in D$. Кожній з подій у системі відповідає результат r_{ui} та інші характеристики. Матриця результатів має розріджену структуру, оскільки кількість предметів у системі велика, а користувальницькі предмети, відомі заздалегідь, становлять лише невелику частину. Завдання рекомендаційної системи полягає у обчисленні прогнозу та рекомендацій для конкретного користувача (user based recommendations) або товару (item based recommendations).

Мета функціонування рекомендаційної системи:

- передбачити перевагу:

$$\hat{r_{ui}} = Predict(u, i, \dots) \approx r_{ui}; \quad (2.2)$$

- персональні рекомендації:

$$u \to \left(i_{1,\dots,} i_k\right) = Recommend_k(u, \dots); \quad (2.3)$$

- схожі об'єкти:

$$u \to \left(i_{1,\dots,} i_M\right) = Similar_m(i, \dots)\,. \quad (2.4)$$

Зобразимо зведені таблиці формулювання постановки задачі(таблиці 2.1 та 2.2):

Таблиця 2.1 – Оцінка користувачів

	Item 1	Item 2	Item 3	Item 4	Item 5	Item 6
User 1	5	4	5			
User 2	4		5			
User 3		3	5		4	
User 4				3	4	
User 5			4	2	4	
User 6	3					5

Таблиця 2.2 – Завдання для прогнозування

	Item 1	Item 2	Item 3	Item 4	Item 5	Item 6
User 1	5	4	5			
User 2	4	?	5			
User 3		3	5	?	4	
User 4			?	3	4	
User 5	?		4	2	4	?
User 6	3					5

Основна ідея колаборативної фільтрації полягає в тому, що користувачі, які взаємодіють з системою подібним чином, зазвичай отримують рекомендації, що відповідають їхнім вподобанням. Для досягнення цієї мети використовуються такі кроки:

Вибір умовної міри схожості між користувачами на основі їхньої історії рейтингів (sim(u,v)).

Розділення користувачів на групи або кластери так, щоб схожі користувачі знаходилися в одному кластері (u→F(u)).

Прогноз оцінки користувача для конкретного об'єкта як середню оцінку кластера за визначеною формулою.

$$\widehat{r_{ui}} = \frac{1}{|F(u)|} \cdot \sum_{v \in F(u)} r_{vi} \ . \qquad (2.5)$$

Недоліки методу алгоритму:

Проблема створення рекомендацій для нових користувачів системи полягає у відсутності відповідного кластера з користувачами, які мають схожі інтереси та переваги. Усі користувачі, які у систему, повинні відповідати розробленим шаблонам, що робить створення рекомендацій для нетипового користувача складним.

Більше того, за відсутності оцінок об'єкта в конкретному кластері неможливо передбачити рекомендації щодо даного об'єкта.

Два підходи щодо покращення роботи методу:

Система, що ґрунтується на користувачах, виходить з припущення, що вподобання користувача подібні до вподобань інших схожих на нього користувачів. Замість жорсткої кластеризації використовується метод кластеризації за визначеною формулою:

$$\widehat{r_{ui}} = \underline{r_u} + \frac{\sum_{v \in U_i} sim(u,v)(r_{vi} - \underline{r_v})}{\sum_{v \in U_i} sim(u,v)} . \quad (2.6)$$

Вади підходу такі як проблема нових або нетипових користувачів, проблема різноманіття.

Система, що базується на предметах рекомендацій, ґрунтується на припущенні, що користувач зацікавлений у схожих товарах до тих, які він вже обрав. Замість жорсткої кластеризації використовується метод кластеризації за визначеною формулою:

$$\widehat{r_{ui}} = \underline{r_i} + \frac{\sum_{j \in Iu} sim(i,j)(r_{ui} - \underline{r_j})}{\sum_{j \in I_u} sim(i,j)}. \quad (2.7)$$

До недоліків цього підходу відносять проблему холодного старту, при якій рекомендації можуть бути тривіальними.

SVD (Singular Value Decomposition) - це метод сингулярного розкладу матриці. Теорема про сингулярний розклад матриці стверджує, що для будь-якої матриці A розміру n × m існує розклад у добуток трьох матриць:

$$\underset{n \times m}{A} = \underset{n \times n}{U} \times \underset{n \times m}{\Sigma} \times \underset{m \times m}{V^T} . \quad (2.8)$$

Матриці U і V – ортогональні, Σ – діагональна(хоч і не квадратна).

$$UU^T = I_n , \quad (2.9)$$

$$VV^T = I_m , \quad (2.10)$$

$$\Sigma = diag(\lambda_1, \dots, \lambda_{\min(n,m)}), \quad (2.11)$$

$$\lambda_1 \geq \cdots \geq \lambda_{\min(n,m)} \geq 0. \quad (2.12)$$

Значення параметрів λ в формулі розташовані за спаданням. Доведення теореми буде опущене.

Крім звичайного розкладання існує також урізане, де з усіх значень λ залишаються лише перші d чисел, а інші вважаються рівними нулю.

$$\lambda_1, \dots, \lambda_{\min(n,m):=0.} \tag{2.13}$$

Це рівносильне тому, що в матрицях U і V залишати лише d стовпців, $\sum$ −обрізати до квадратної розміром dxd

$$\underset{n \times m}{A'} = \underset{n \times d}{U'} \times \underset{d \times d}{\Sigma'} \times \underset{d \times m}{V'^T}. \tag{2.14}$$

На практиці виявляється, що нова матриця A' ефективно наближає вихідну матрицю A і є оптимальним низькоранговим наближенням з точки зору середньоквадратичного відхилення за методом сингулярного розкладу (SVD) для рекомендацій.

Для прогнозу оцінки користувача U для предмету I обираємо вектор p_u (вектор параметрів для даного користувача) та вектор параметрів q_i для даного предмету. Їхній скалярний добуток використовується як шукане передбачення:

$$\widehat{r_{ui}}(\Theta) = p_u^T q_i, \tag{2.15}$$

$$\Theta = \{p_u, q_i | u \in U, i \in I\}. \tag{2.16}$$

Але так як вектори невідомі, їх ще необхідно обрахувати. Ідея полягає у тому, що у нас є оцінки, надані користувачами, за допомогою яких ми можемо знайти такі оптимальні параметри, при яких модель передбачала би ці оцінки якнайкраще:

$$E_{(u,i)}(\hat{r}_{ui}(\Theta) - r_{ui})^2 \rightarrow \min_{\Theta}. \tag{2.17}$$

Отже, завданням є знаходження таких параметрів Θ, які мінімізують квадрат помилки. Тут виникає парадокс: хочемо зменшити помилки в майбутньому, але нам невідомі майбутні оцінки. Оптимізувати це безпосередньо неможливо. Проте у нас є наявні користувацькі оцінки на даний момент. Тож ми можемо підібрати параметри так, щоб помилка на цих вже відомих оцінках була мінімальною. Додатково введемо регуляризатор.

Для оптимізації описаної моделі ми будемо використовувати наступний функціонал:

$$J(\Theta) = \sum_{(u,i)\epsilon D}(p_u^T q_i - r_{ui})^2 + \lambda(\sum_u \|p_u\|^2 + \sum_i \|q_i\|^2). \quad (2.18)$$

Параметрів дійсно велика кількість: для кожного користувача та для кожного об'єкта існує власний вектор, який ми намагаємося оптимізувати. У нас є функція, яка залежить від значної кількості змінних.

Для пошуку її мінімуму можна використовувати метод найменших квадратів, який послідовно оптимізує параметри, як зображено на рисунку 2.3:

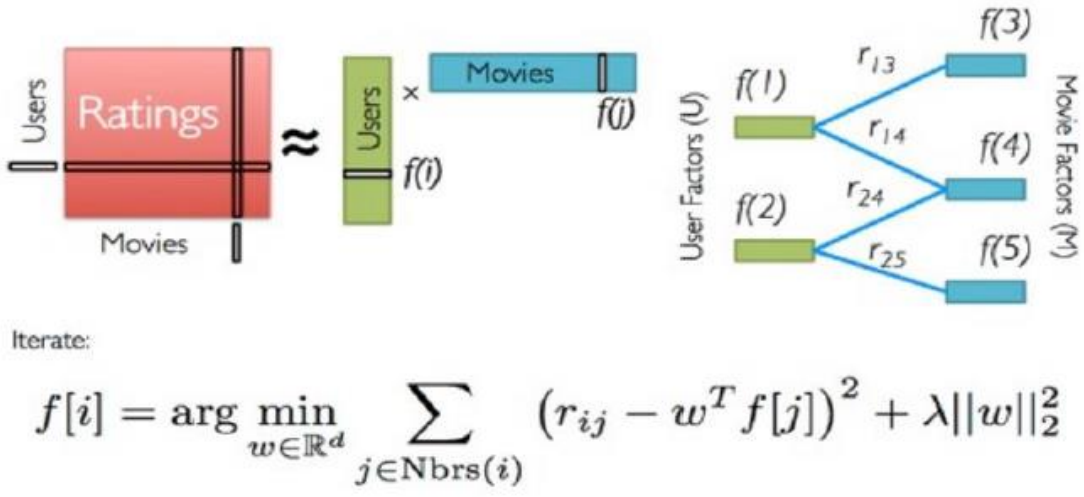

Рисунок 2.3 – Принципова схема методу квадратів, що чергуються ALS

Якщо ми хочемо знайти мінімум для параболи, ми відразу знаємо, де він знаходиться. Цікаво, що функціонал помилки, який ми намагаємося оптимізувати, має схожу форму з параболою. Це призводить до методу найменших квадратів, що чергуються - ми фіксуємо всі параметри об'єктів і оптимізуємо параметри користувачів, потім змінюємо ролі і фіксуємо параметри користувачів, мінімізуючи параметри об'єктів. Ми це робимо ітеративно, крок за кроком для досягнення оптимального рішення.

Одним з шляхів підвищення точності рекомендаційної системи є розширення переліку використовуваної для формування рекомендації інформації, особливо використання контексту.

Під контекстом розумітимемо атрибути, які так чи інакше описують ситуацію, у рамках якої користувач оцінив об’єкт або отримує рекомендації. Тобто у рамках рекомендаційної системи можуть бути враховані два види контексту:

- Контекст, при якому відбувається фіксація інтересів користувача та його вподобань;
- Контекст, при якому відбувається формування рекомендацій.

Можна виділити наступні види контексту, які можуть бутивикористані у рекомендаційній системі:

- Фізичний контекст: час, місцеположення, вид діяльності користувача, погода, освітленість і так далі;
- Соціальний контекст: наявність і роль оточуючих користувача людей;
- Пристрієвий контекст: вид і характеристики пристрою, з якого здійснюється доступ до інформації;
- Модальний контекст: настрій користувача, мету, досвід, когнітивні особливості.

Таким чином, врахування контексту при формуванні пропозицій являє собою один з найперспективніших напрямків розвитку рекомендаційних систем. Використання інформації про контекст може допомогти при вирішенні проблем, що характерні для більшості існуючих рекомендаційних систем: проблеми «холодного старту»(при появі нового користувача вносити пропозицію можна як на основі даних стосовно його соціально-демографічного портрету, так і про його поточний стан у вигляді діяльності), передбачення для нетипових користувачів(врахування великої кількості індивідуальних характеристик дозволяє краще персоніфікувати рекомендації), тривіальність рекомендацій, фільтри(врахування контексту дозволяє не тільки не обмежуватись минулими точками зору користувачів).

2.2 Розробка структури програмного модуля рекомендаційної системи сервісу відеохостингу

У сучасному цифровому світі модулі надання рекомендацій стають все більш популярним і важливим для користувачів різноманітних систем. Відеохостинги, такі як YouTube, Vimeo, та інші, надають зручні засоби для отримання персоніфікованих відео для користувачів. Розробка та підтримка систем рекомендації контенту вимагає вивчення і обрання найкращих компонентів інформаційної технології для забезпечення оптимального функціонування та задоволення потреб користувачів.

Збір інформації про користувачів реалізується засобами відстеження активності користувачів, таких як перегляд відео, залишені відгуки, вподобання та інші взаємодії з платформою. Збір контенту втримання та аналіз властивостей відео, включаючи тематику, ключові слова, категорії та метадані, які допомагають побудувати більш детальний опис контенту.

Нормалізація та очищення даних полягає в обробка даних для видалення дублікатів, виправлення помилок та приведення до стандартного формату для подальшого використання. Створення унікальних профілів на основі зібраних даних, що відображають інтереси та вподобання кожного користувача.

Метрики ефективності полягають у визначенні та валідації метрик, таких як точність рекомендацій, покриття та різноманітність, для оцінки якості системи.

Побудова рекомендацій відбувається за допомогою генерації персоналізованих рекомендацій та їх відображення у зручному інтерфейсі. Можливість користувачів змінювати свої вподобання та отримувати більш персоналізовані рекомендації.

Використання технологій розподіленого зберігання для ефективного зберігання та доступу до великого обсягу даних. Вдосконалення алгоритмів для забезпечення ефективності та швидкодії обчислень рекомендацій.

Взаємодія з пошуковою системою для забезпечення користувачам повного та різноманітного доступу до контенту. Забезпечення можливості аналізу взаємодій користувачів з рекомендаційною системою та створення звітів для покращення ефективності.

Ці компоненти взаємодіють між собою для створення ефективної та персоналізованої рекомендаційної системи для сервісу відеохостингу. Аналіз та обґрунтування вибору складових компонентів є критичними для створення ефективного та конкурентоздатного продукту. Кожна складова має свої обґрунтування, спираючись на технічні вимоги, потреби користувачів та бізнес-цілі. Грамотний вибір цих складових допомагає створити рекомендаційну систему для відеохостингу, яка забезпечує якість, надійність і зручність для користувачів, а також сприяє досягненню бізнес-цілей.

Оскільки модуль рекомендаційної системи належить сервісу відеохостингу який є веб-додатком, потрібно налагодити клієнт-серверну архітектуру для коректної роботи додатку. Веб-додаток запускається за допомогою браузера, на теперішній час найвідомішими є: Google Chrome, Safari, Opera. Інтерфейс програми будується за допомогою мови розмітки документів HTML (Hypertext Markup Language). Веб-браузери отримують HTML документ від сервера за протоколами HTTP/HTTPS або відкривають з локального диска, які потім браузер інтерпретує та відображає їх. За допомогою смартфону чи комп’ютеру відображається інтерфейс, через нього клієнт має змогу надсилати запит сервери та відображати результати, які сервер повертає. Коли до серверу надходить запит, то він обробляє його і зберігає результат в базі-даних або повертає його на сторону клієнта.

За допомогою протоколу HTTP [14] здійснюється обмін даними, що дозволяє отримувати різні ресурси, такі як HTML-документи. HTTP є основою для обміну даними в Інтернеті. Цей протокол є механізмом клієнт-серверної взаємодії, де одержувач, зазвичай веб-браузер, ініціює запити до сервера. Повертаний документ може складатися з різних піддокументів, які є частиною

підсумкового документа, наприклад, тексту, опис структури, зображень, відеофайлів, скриптів та інших елементів.

Незважаючи на те, що HTTP було розроблено на початку 1990-х років, завдяки своїй розширюваності він продовжував розвиватися. HTTP є протоколом прикладного рівня і часто використовує протокол TCP (або захищений TCP - TLS) передачі своїх повідомлень, хоча теоретично може використовуватися й інший надійний транспортний протокол. Завдяки можливості розширення, HTTP застосовується не тільки для отримання гіпертекстових документів, зображень та відео клієнтом, а й для передачі контенту на сервери, наприклад, за допомогою HTML-форм. HTTP також може використовуватися для отримання лише частини документа для оновлення веб-сторінки на запит, наприклад, за допомогою AJAX-запиту.

У HTTP є поняття умовних запитів, у яких результат, і навіть успіх запиту, можуть бути змінені за допомогою порівняння порушених ресурсів зі значенням валідатора. Такі запити можуть бути корисними для валідації контенту в кеші, і позбавлення від марного контролю, щоб перевірити цілісність документа, наприклад, поки триває завантаження, або поки запобігає втраті оновлень, поки вивантажуються або змінюються файли на сервері.

Умовні запити HTTP це запити, які виконуються по-різному, залежно від значення особливих заголовків. Ці заголовки визначають передумову і результат запиту буде різним, якщо умова узгоджена чи ні.

Всі умовні заголовки намагаються перевірити, чи ресурс, що зберігається на сервері, певної версії. Для цього в умовних запитах необхідно вказати версію ресурсу. Оскільки порівняння байта з байтом всього ресурсу є недоцільним і не завжди відповідає бажанню, у запиті передається значення, що описує версію. Такі значення називаються валідаторами та бувають двох видів: дата останньої зміни документа, дата останньої зміни,

непрозорий рядок, що унікально ідентифікує кожну версію, звану тегом об'єкта або etag.

Порівнювати версії одного і того ж ресурсу трохи складніше: залежно від контексту існує два види перевірок на рівність: сувора перевірка використовується, коли очікується побайтова ідентичність, наприклад, при відновленні завантаження. Слабка перевірка використовується, коли користувача агенту потрібно тільки визначити, чи мають два ресурси однаковий контент. І це навіть якщо це незначні відмінності; наприклад, різні оголошення або нижній колонтитул з іншою датою.

Вигляд перевірки не залежить від використовуваного валідатора. Обидва Last-Modified і ETag допускають обидва типи перевірки, хоча складність її реалізації за сервера може різнитися. HTTP за промовчанням використовує строгу перевірку та визначає, коли можна використовувати слабку перевірку.

Приклад роботи HTTP запиту до серверу та його відповіді на клієнтську сторону представлено на рисунку 2.4.

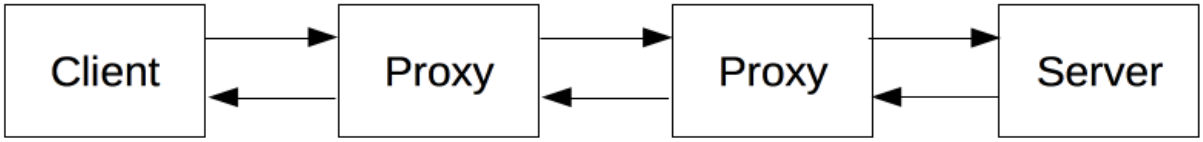

Рисунок 2.4 – Принципова схема роботи запиту до серверу за допомогою протоколу HTTP

У сучасних умовах клієнтська сторона часто зберігається на персональних комп'ютерах, смартфонах та ноутбуках, тоді як сервери розміщуються на потужніших машинах. Подібна модель зберігання дозволяє підвищити продуктивність програми, оскільки клієнт та сервер виконують свої завдання незалежно один від одного. Крім того, вона допомагає скоротити обсяг необхідних даних, що зберігаються на пристрої користувача, і забезпечує конфіденційність даних програми.

При розробці модуля управління платформою для зберігання відео використовується трирівнева архітектура проектування, що передбачає

створення екосистеми, що включає клієнтську частину, серверну частину, побудовану за принципом RESTful API, та шар даних.

REST (Representational State Transfer) [15], або «передача стану уявлення», є архітектурним стилем проектування API з використанням протоколу HTTP. Головною перевагою REST є його гнучкість. REST API застосовується повсюдно там, де потрібне надання даних із сервера користувачеві веб-програми або сайту.

Головними компонентами REST API є:

Client - клієнт або програма, яка запущена на стороні користувача (на його девайсі) та ініціює комунікацію.

Server — сервер, який надає API як доступ до своїх даних та функцій.

Resource - ресурс є будь-яким видом контенту (відео, текст, картинка), який сервер може передати клієнту.

REST API взаємодіє з допомогою HTTP запитів, виконуючи стандартні функції: створення, оновлення, читання, видалення записів у ресурсі. Існує чотири методи, що описують, що потрібно робити з ресурсом:

– POST - створення ресурсу;
– GET – отримання ресурсу;
– PUT – оновлення ресурсу;
– DELETE – видалення ресурсу.

Ресурс – це ключова абстракція інформації. Будь-який вид інформації, який ми можемо використовувати, може бути ресурсом: документ, зображення, часовий сервіс.

Стан ресурсу будь-який певний момент називається уявленням ресурсу. Воно складається з:

– даних;
– метаданих, що описують дані;
– гіпермедіа посилань, які допомагають клієнтам перейти до наступного стану.

Інформація для клієнта може бути надана у різних форматах, таких як JavaScript (JSON), HTML, XLT, Python, PHP або просто текст. Найпопулярнішим форматом є JSON, тому що він є читабельним як для машин, так і для людей, і не залежить від мови програмування. Щоб отримати доступ до ресурсу, клієнт надсилає запит, а сервер генерує відповідь із закодованими даними про ресурс.

Структура будь-якого запиту включає чотири основні компоненти: HTTP-метод, ендпоінти, заголовки і тіло. HTTP метод визначає, які операції повинні бути виконані з ресурсом. Як згадувалося раніше, доступні чотири методи: POST, GET, PUT, DELETE.

Ендпоінт містить URI (Uniform Resource Identifier), який вказує, де та як знайти ресурс в Інтернеті. Найпоширеніший тип URI - це URL (Uniform Resource Location), що представляє повноцінну веб-адресу.

Заголовки зберігають інформацію, що стосується як клієнта, і сервера. Основна мета заголовків - надати автентифікаційні дані, такі як API-ключ, назву або IP-адресу сервера, а також інформацію про формат відповіді.

Тіло запиту необхідне передачі додаткової інформації на сервер, наприклад, дані, які потрібно додати чи замінити. Схема взаємодії клієнт-серверної частини модуля відео платформи з RESTful API наведена на рисунку 2.5.

Розробляючи модуль відеохостингу, REST використовується в чистому вигляді, тобто слідує чотирьом базовим принципам проектування:

– явне використання HTTP-методів;

– незбереження стану;

– надання URI, які відповідають структурі каталогів;

– передача даних в XML, JavaScript Object Notation (JSON) або в обох форматах.

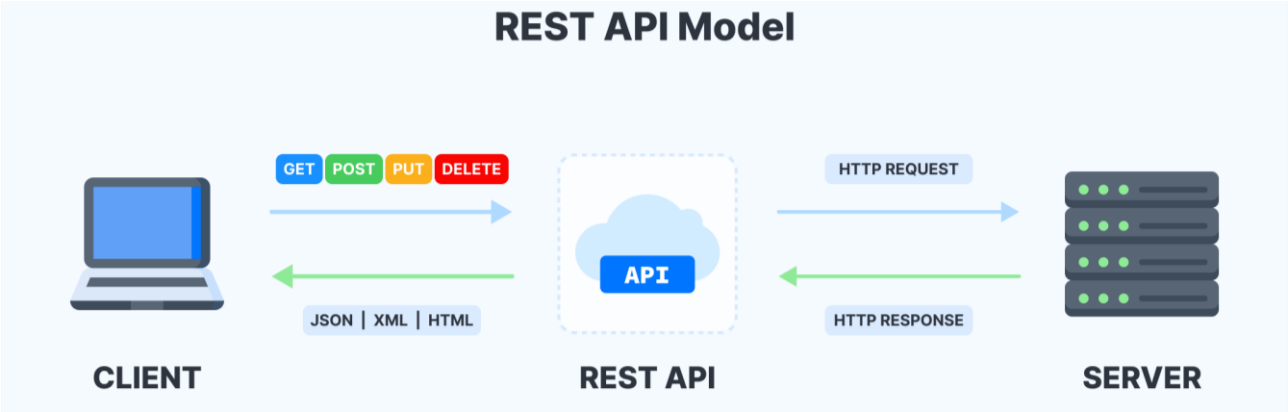

Рисунок 2.5 – Принципова схема взаємодії клієнт-серверної частини з RESTful API

Програма REST побудована на клієнт-серверній архітектурі. Клієнтом є той, хто надсилає запити на ресурси, і він не пов'язаний із сховищем даних. Зберігання даних залишається всередині сервера. У свою чергу, сервери не пов'язані з інтерфейсом користувача. Іншими словами, клієнт і сервер можуть розвиватися незалежно один від одного, що робить REST API більш гнучким та масштабованим.

Основні функції сервера включають створення відповідей, що містять посилання на інші ресурси для навігації по пов'язаних ресурсах додатків. Такі відповіді містять інтегровані посилання. Крім того, при запиті батьківського або контейнерного ресурсу RESTful-відповідь містить посилання на нащадків батьківського елемента або підпорядковані ресурси, щоб зберегти зв'язок з ними.

Сервер також генерує відповіді, що містять інформацію щодо можливості кешування з метою підвищення продуктивності. Для цього сервер включає в HTTP-заголовки відповідні значення cache-control і last-modified (дата зміни даних).

Основні функції клієнтської частини включають перевірку заголовка cache-control у відповідях визначення можливості кешування ресурсу. Клієнтська програма також читає значення last-modified та надсилає заголовок if-modified-since на сервер із запитом про зміну ресурсу. Такий запит, відомий

як conditional get, використовує обидва заголовки. Якщо ресурс не змінився з вказаного часу, сервер повертає стандартний код 304 (not modified) і запитаний ресурс не надсилається. Код 304 означає, що клієнт може використовувати кешовану локальну копію ресурсу як актуальну версію, уникаючи наступних запитів GET до зміни ресурсу.

Клієнтська програма надсилає повні запити, які можуть опрацьовуватися незалежно від інших запитів. Для цього клієнтська програма використовує всі заголовки HTTP, визначені інтерфейсом веб-сервісу, і відправляє повні представлення ресурсів у тілі запиту. Клієнтська програма майже нічого не знає про попередні запити, наявність сеансу на сервері, про можливість сервера додавати контекст у запит і про стан програми, що зберігається між запитами.

Така спільна робота клієнтської програми та сервісу є важливою особливістю RESTful веб-сервісів. Відмова від збереження стану призводить до підвищення продуктивності за рахунок зменшення трафіку та мінімізації стану серверної програми.

2.3 Розробка алгоритму роботи рекомендаційної системи сервісу відеохостингу

Готовий попередній список рекомендацій, який базується на методі колаборативної фільтрації для надання рекомендації, містить відео, сортовані за їхніми оцінками.

Для запуску колаборативної фільтрації створена модель отримує дані від серверу з користувацькими вподобаннями. Як тільки користувач оцінив один з відео, модель зберігає дані і використовує їх при надані наступних рекомендацій, змінюючи порядок контенту, який користувач бачить на екрані.

Рисунок 2.6 – Принципова схема алгоритму роботи рекомендаційного програмного модуля

2.4 Розробка алгоритму роботи системи сервісу відеохостингу

Загальний алгоритм роботи програмного модуля управління сервісом відеохостингу передбачає виконання наступних кроків:

1. Завантаження додатку – користувачі відкривають веб-додаток. Програмний модуль надає функціонал, відповідно до бажання користувача.
2. Якщо було обрано додавання відео – відбувається введення даних про контент.
3. Відео вноситься до бази даних.
4. Якщо користувач бажає переглянути відео.
5. Формується список доступних відео для перегляду. Цей список передається для відображення до клієнтської частини програмного модуля.
6. Коли користувач обрав бажане відео, він може його переглянути
7. Якщо він хоче змінити гучність програвання відео, чи перемотати його він може використовувати повзунки в плеєрі.
8. Після прослуховування відео змінюється кількість прослуховувань у для даного вмісту.
9. Якщо користувач бажає залишити коментар до відео.
10. Він переходить на сторінку з детальною інформацією про відео, вводить коментар в поле та відправляє його.
11. Відбувається запит до серверу та коментар зберігається в базі даних.

Надсилання відео для перегляду відбувається за допомогою потоку HLS який розбиває відео на більш дрібні завантаження HTTP і доставляє їх з використанням протоколу HTTP. Оскільки HLS заснований на HTTP, будь-який звичайний веб-сервер може створити потік.

Схему алгоритму роботи серверної частини програмного модуля управління товарними запасами зображено на рисунку 2.7.

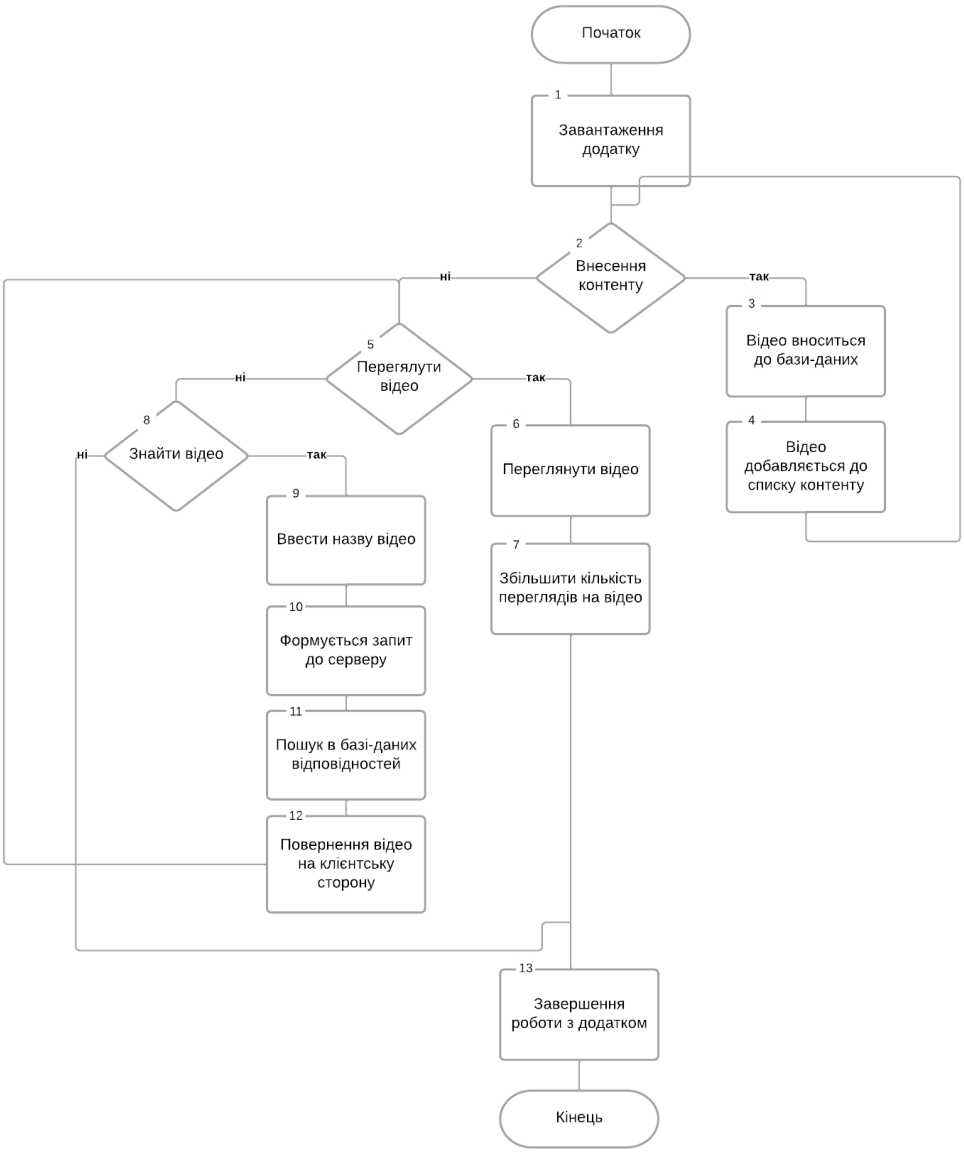

Рисунок 2.7 – Принципова схема алгоритму роботи програмного модуля сервісу відеохостингу

2.4 Висновок до розділу 2

У першому підрозділі роботи було розроблено і досліджено метод «колаборативної фільтрації» для надання рекомендацій користувачам. Було встановлено недоліки підходу та способи їх усунення. Було виконано розробку клієнт-серверної моделі організації відеохостингу відповідно до розглянутої моделі «колаборативної фільтрації». Також було розроблено метод передачі контенту від серверу до користувача. Було розроблено алгоритм роботи програмного додатку.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ ОРГАНІЗАЦІЇ СЕРВІСУ ВІДЕОХОСТИНГУ

3.1 Вибір інструментарію при розробці системи сервісу відеохостингу

Для програмної реалізації програмного модуля сервісу організації відеохостингу оберемо JavaScript. JavaScript [16] – це високорівнева мова програмування, яка підтримує різні підходи: імперативний, функціональний та подієво-орієнтований. Він відрізняється динамічною типізацією і використовується для написання скриптів, що є послідовними операціями. Такі скрипти зазвичай інтерпретуються, а не компілюються, тому для їх роботи не потрібні додаткові програми або інструменти для перетворення в інші формати кодування.

Кожна веб-програма або сайт будується з використанням трьох основних технологій – HTML, CSS та JavaScript. При цьому JavaScript виступає в ролі "мозку" розробки та відповідає за інтерактивність та взаємодію з користувачем.

JavaScript це мова, яка дозволяє вам застосовувати складні речі на web сторінці - кожен раз, коли на web сторінці відбувається щось більше, ніж просто її статичне відображення - відображення контенту, що періодично оновлюється, або інтерактивних карт, або анімація 2D/3D графіки, або прокручування відео в плеєрі, і т.д. - можете бути впевнені, що швидше за все не обійшлося без JavaScript. Це третій шар пирога стандартних web технологій, два з яких (HTML і CSS) ми детально розкрили в інших частинах навчального посібника.

Ще більш цікавою є функціональність, створена поверх основної мови JavaScript. Так звані інтерфейси прикладного програмування (API) надають вам додаткові надздібності для використання у вашому коді JavaScript.

API – це готові набори блоків коду, які дозволяють розробнику реалізовувати програми, які інакше було б важко чи неможливо реалізувати. Вони роблять те саме для програмування, що готові комплекти меблів роблять для домашнього будівництва - набагато простіше брати готові панелі і

скручувати їх разом, щоб зробити книжкову полицю, ніж самому розробляти дизайн, ходити в пошуках правильної деревини, вирізати всі панелі необхідного розміру та форми знайти підходящі гвинти, а потім зібрати їх разом, щоб зробити книжкову полицю.

Особливості JavaScript включають динамічність, гнучкість роботи з функціями і універсальність. Мова підтримується всіма сучасними браузерами, легко інтегрується з розміткою HTML та дозволяє настроювати комунікацію із сервером. Серед інших переваг:

– тип даних визначається, коли змінній або константі присвоюється значення;

– JavaScript функції можна як виконувати, так і повертати, передавати їх як параметри іншим функціями і привласнювати як значення змінних;

– методологія об’єктно-орієнтованого програмування дає змогу представити програму у вигляді сукупності об’єктів;

– дозволяє частково перенести бізнес-логіку із сервера на сторону користувача, тобто виконувати код в браузері, що своєю чергою зменшує навантаження на сервери;

– JavaScript має розвинену інфраструктуру та активну спільноту. Так, веб-розробники можуть працювати з великою кількістю бібліотек і фреймворків як React, Angular і Vue, декількома пакувальниками, як Webpack, Gulp, та допоміжними бібліотеками як Lodash, axios, та іншими.

React JS - це відкритий фреймворк на мові JavaScript, який служить бібліотекою для створення інтерфейсів користувача. Розроблена компанією Facebook ця технологія швидко завоювала популярність серед розробників з усього світу [17]. Основною перевагою React JS є його здатність створювати високопродуктивні та масштабовані додатки. Ключовим поняттям в React JS є компоненти - незалежні блоки коду, що відповідають за відображення певної частини інтерфейсу користувача. React використовується для розробки веб-додатків, де його основною метою є створення інтерактивних, динамічних і чуйних інтерфейсів користувача. Frontend-частина React дозволяє створювати

багатофункціональні та інтерактивні програми зі швидким відображенням та навігацією між сторінками. Також він має такі переваги:

– використання віртуального DOM та ефективного алгоритму оновлення дозволяє робити мінімальні зміни в реальному DOM, що покращує продуктивність застосунків;

– JS React базується на компонентній архітектурі, що дозволяє розбивати інтерфейс на незалежні компоненти. Це спрощує розробку, тестування та підтримку коду, оскільки компоненти можуть бути повторно використані та легко змінюються.

– React пропагує односторонній потік даних, що сприяє простоті та передбачуваності управління станом додатків, полегшує відлагодження та тестування застосунків;

– спільнота розробників, що використовує React велика та активна. Є безліч ресурсів, бібліотек та інструментів для розробки. Також існує багато сторонніх бібліотек, які підтримують React і допомагають розширити його функціональність.

React є ідеальним інструментом для розробки будь-яких проектів. Він забезпечує гнучке розширення та використання компонентів, а також інтеграцію з іншими бібліотеками та фреймворками. Більше того, React підтримує серверний рендеринг, що дозволяє значно покращити швидкість завантаження сторінок та оптимізувати пошукову оптимізацію.

Для зберігання даних на відеохостингу було обрано MongoDB [18] , крос-платформну, потужну, гнучку та легко масштабовану документно-орієнтовану базу даних. MongoDB є ідеальним рішенням для зберігання великих обсягів неструктурованих даних, таких як відео та медіафайли. Вона ефективно зберігає та оптимізовано взаємодіє з великою кількістю даних, характерних для відеохостингів.

Додатково MongoDB надає вбудовану підтримку MapReduce-style Aggregation, що значно спрощує аналіз та обробку даних у розподілених середовищах. Наявність Geospatial індексів робить MongoDB привабливим

вибором для додатків, що потребують геопросторового аналізу чи взаємодії з місцезнаходженням.

MongoDB складається з баз даних, які містять колекції. Кожна колекція є набором документів, де кожен документ складається з пари ключ-значення. Це дозволяє ефективно моделювати структуру даних на відеохостингу, де кожен відеофайл може мати різні атрибути та властивості. Більш того, можливість індексування колекцій допомагає покращити швидкодію вибірки та сортування даних, що важливо для ефективного відображення відповідей користувачам на відеохостинговій платформі.

MongoDB використовує автентифікацію, контроль доступу та шифрування для гарантії безпеки своїх баз даних. Вони використовують шифрування TLS/SSL для захисту даних у русі та місцях зберігання. Вони також дозволяють визначити різні рівні доступу користувачів. Документи MongoDB наслідують ієрархічну модель даних і зберігають більшу частину даних в одному документі, що знижує необхідність об'єднання кількох документів. З'єднання підтримуються операцією $lookup, але вони не оптимізовані підвищення продуктивності. Однак MongoDB пропонує API insertMany() для швидкої вставки даних із пріоритетною увагою продуктивності запису. Зберігаючи дані у вигляді документів JSON, MongoDB дозволяє створювати складні програми з різними типами даних. Наприклад, ви можете створювати нові поля, оновлюючи поля вкладеного масиву. Можна також використовувати конвеєр агрегування – функцію MongoDB, яка дозволяє конвертувати дані, поєднуючи кілька операцій в один робочий процес.

3.2 Програмна реалізація рекомендаційної системи сервісу організації відеохостингу

Для отримання списку індексів предметів, які були оцінені певним користувачем у системі рекомендацій використовується функція getRatedItemsForUser:

```
function getRatedItemsForUser(ratings, userIndex, numItems) {
  const ratedItems = [];
  for (let index = 0; index < numItems; index += 1) {
    if (ratings[userIndex][index] !== 0) {
      ratedItems.push(index);
    }
  }
  return ratedItems;
}.
```

Функція typeCheckCoOccurrenceMatrix використовується для перевірки типу та розмірності матриці, яка представляє собою матрицю співвідношення. Загалом ця функція забезпечує, що матриця співвідношення відповідає очікуваному типу та розмірам, необхідним для подальшого коректного використання в системі рекомендацій. Вона служить як захист від можливих помилок в програмі, пов'язаних із некоректним використанням матриці:

```
function typeCheckCoOccurrenceMatrix(coMatrix, numItems) {
  if (!(coMatrix instanceof math.Matrix)) {
    throw new TypeError('The occurrence matrix should be a mathJS Matrix object generated by createCoMatrix');
  }
  if (!arraysAreEqual(coMatrix.size(), [numItems, numItems])) {
    throw new RangeError('Co matrix has wrong dimensions. Make sure to generate it using createCoMatrix');
  }
}.
```

Функція checkRatingValues використовується для перевірки значень в матриці рейтингів (rating matrix). Ця функція служить для перевірки коректності значень в матриці рейтингів та використовуватися для забезпечення правильного функціонування інших частин системи рекомендацій, які використовують цю матрицю:

```
function checkRatingValues(ratingMatrix) {
  const allowedRatings = [0, 1];
  ratingMatrix.forEach((value) => {
    if ((!Number.isInteger(value)) || (!allowedRatings.includes(value))) {
      throw new TypeError('Wrong rating in rating array. Currently permitted values are 0 and 1');
    }
  });
  return true;
}.
```

Надання рекомендацій відбувається на основі ваг подібності та оцінок інших користувачів. Коли користувач взаємодіє з системою, його або її вподобання та оцінки аналізуються для виявлення схожих користувачів або схожих предметів інтересу.

Ваги подібності визначаються на основі спільних вподобань та дій користувачів. Якщо два користувачі мають схожі вподобання або взаємодіють із схожим контентом, їхній внесок у рекомендації може бути високим. Оцінки користувачів для конкретних об'єктів також враховуються при формуванні рекомендацій.

```
function getRecommendations(ratings, coMatrix, userIndex) {
  typeCheckRatings(ratings);
  let ratingsMatrix;
  try {
    ratingsMatrix = math.matrix(ratings);
  } catch (error) {
    throw new RangeError('Dimension error in ratings matrix');
  }
  const numItems = ratingsMatrix.size()[1];
  typeCheckCoOccurrenceMatrix(coMatrix, numItems);
```

```
    typeCheckUserIndex(userIndex, ratings);
    const ratedItemsForUser = getRatedItemsForUser(ratings, userIndex, numItems);
    const numRatedItems = ratedItemsForUser.length;
    const similarities = math.zeros(numRatedItems, numItems);
    for (let rated = 0; rated < numRatedItems; rated += 1) {
     for (let item = 0; item < numItems; item += 1) {
      similarities.set([rated, item], coMatrix.get([ratedItemsForUser[rated], item])
                                + similarities.get([rated, item]));
     }
    }
    let recommendations = math.zeros(numItems);
    for (let y = 0; y < numRatedItems; y += 1) {
     for (let x = 0; x < numItems; x += 1) {
      recommendations.set([x], recommendations.get([x]) + similarities.get([y, x]));
     }
    }
    recommendations = math.dotDivide(recommendations, numRatedItems);
    const rec = recommendations.toArray();
    let recSorted = recommendations.toArray();
    recSorted.sort((a, b) => b - a);
    if (ONLY_RECOMMEND_FROM_SIMILAR_TASTE) {
     recSorted = recSorted.filter((element) => element !== 0);
    }
    let recOrder = recSorted.map((element) => {
     const index = rec.indexOf(element);
     rec[index] = null;
     return index;
```

```
    });
    recOrder = recOrder.filter((index) => !ratedItemsForUser.includes(index));
    return recOrder;
  }.
```

Реєстрація та авторизація користувача здійснюється на клієнтській стороні за допомогою відповідних механізмів, таких як форми та автентифікаційні токени. У момент реєстрації, користувач подає необхідні відомості, які відправляються на сервер для збереження в базі даних. Після цього, при авторизації, сервер перевіряє введені користувачем дані та надає або відмовляє в доступі.

Обробка інформації відбувається на сервері, де відбувається перевірка логіну та пароля, необхідних для ідентифікації користувача. У випадку успішної авторизації, сервер генерує токен, який надсилається клієнту для подальшого використання при кожному запиті до сервера.

Нижче наведено функції необхідні для процесу реєстрації або входу користувача:

```
export const signup = async (req, res, next) => {
  try {
    const salt = bcrypt.genSaltSync(10);
    const hash = bcrypt.hashSync(req.body.password, salt);
    const newUser = new User({ ...req.body, password: hash });
    await newUser.save();
    res.status(200).send("User has been created!");
  } catch (err) {
    next(err);
  }
};
export const signin = async (req, res, next) => {
  try {
    const user = await User.findOne({ name: req.body.name });
```

```
    if (!user) return next(createError(404, "User not found!"));
    const isCorrect = await bcrypt.compare(req.body.password, user.password);
    if (!isCorrect) return next(createError(400, "Wrong Credentials!"));
    const token = jwt.sign({ id: user._id }, process.env.JWT);
    const { password, ...others } = user._doc;
    res
      .cookie("access_token", token, {
        httpOnly: true,
      })
      .status(200)
      .json(others);
  } catch (err) {
    next(err);
  }
};
export const googleAuth = async (req, res, next) => {
  try {
    const user = await User.findOne({ email: req.body.email });
    if (user) {
      const token = jwt.sign({ id: user._id }, process.env.JWT);
      res
        .cookie("access_token", token, {
          httpOnly: true,
        })
        .status(200)
        .json(user._doc);
    } else {
      const newUser = new User({
        ...req.body,
        fromGoogle: true,
```

```
    });
    const savedUser = await newUser.save();
    const token = jwt.sign({ id: savedUser._id }, process.env.JWT);
    res
      .cookie("access_token", token, {
        httpOnly: true,
      })
      .status(200)
      .json(savedUser._doc);
  }
 } catch (err) {
  next(err);
 }
}.
```

Процес додавання нового відео до бази даних виконується за допомогою спеціальної форми. Користувач заповнює такі поля як назва, опис, додання картинки та завантаження необхідного відео:

```
export const addVideo = async (req, res, next) => {
  const newVideo = new Video({ userId: req.user.id, ...req.body });
  try {
    const savedVideo = await newVideo.save();
    res.status(200).json(savedVideo);
  } catch (err) {
    next(err);
  }
 }.
```

Процес додавання нових коментарів про відео відео до бази даних виконується було реалізовано за допомогою наступних функцій. Компонент отримує пропс videoId, який використовується для отримання коментарів до конкретного відео. Використовуються хуки useState та useEffect для створення

стану для збереження коментарів і здійснення асинхронного запиту на сервер. У функції useEffect викликається асинхронна функція fetchComments. Після отримання відповіді з сервера, коментарі зберігаються у стані компонента за допомогою функції setComments.:

```
const Recommendation = ({ tags }) => {
  const [videos, setVideos] = useState([]);
  useEffect(() => {
    const fetchVideos = async () => {
      const res = await axios.get(`/videos/tags?tags=${tags}`);
      setVideos(res.data);
    };
    fetchVideos();
  }, [tags]);
  return (
    <Container>
      {videos.map((video) => (
        <Card type="sm" key={video._id} video={video} />
      ))}
    </Container>
  );
};
export default Recommendation;
const Comments = ({videoId}) => {
  const { currentUser } = useSelector((state) => state.user);
  const [comments, setComments] = useState([]);
  useEffect(() => {
    const fetchComments = async () => {
      try {
        const res = await axios.get(`/comments/${videoId}`);
```

```
        setComments(res.data);
      } catch (err) {}
    };
    fetchComments();
  }, [videoId]);
  return (
    <Container>
      <NewComment>
        <Avatar src={currentUser.img} />
        <Input placeholder="Add a comment..." />
      </NewComment>
      {comments.map(comment=>(
        <Comment key={comment._id} comment={comment}/>
      ))}
    </Container>
  );
};
export default Comments.
```

3.3 Тестування програмного модуля сервісу відеохостингу

Для проведення тестування програмного додатку спочатку необхідно запустити React-проект на локальній мережі. Це можна зробити, використовуючи командний рядок та викликаючи команду npm start у контексті проекту. Після введення цієї команди, термінал виведе повідомлення, подібне до того, яке зображено на рисунку 3.2.

Запуск React-проекту [19] за допомогою команди npm start дозволяє ініціювати локальний сервер для перегляду та випробування додатку. Отримане

повідомлення в терміналі містить інформацію про стан запуску, а також URL-адресу, за якою можна перейти для перегляду проекту у веб-браузері.

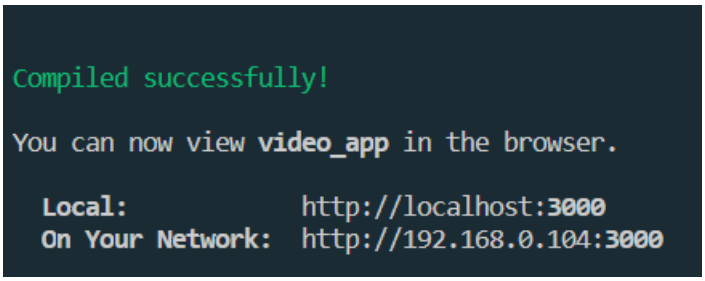

Рисунок 3.2 – Загальний вигляд типового повідомлення в системі про початок процесу завантаження проекту у локальній мережі

Через 2-3 хвилини, проект повністю завантажиться, а веб-додаток автоматично відкриється за посиланням у локальній мережі, як показано на рисунку 3.3. Після завантаження користувач автоматично перенаправлений на головну сторінку додатку.

На цій головній сторінці доступна вся необхідна функціональність та інтерфейс додатку. Користувач має можливість взаємодіяти з різними елементами, виконувати необхідні завдання та оцінювати його функціональні можливості.

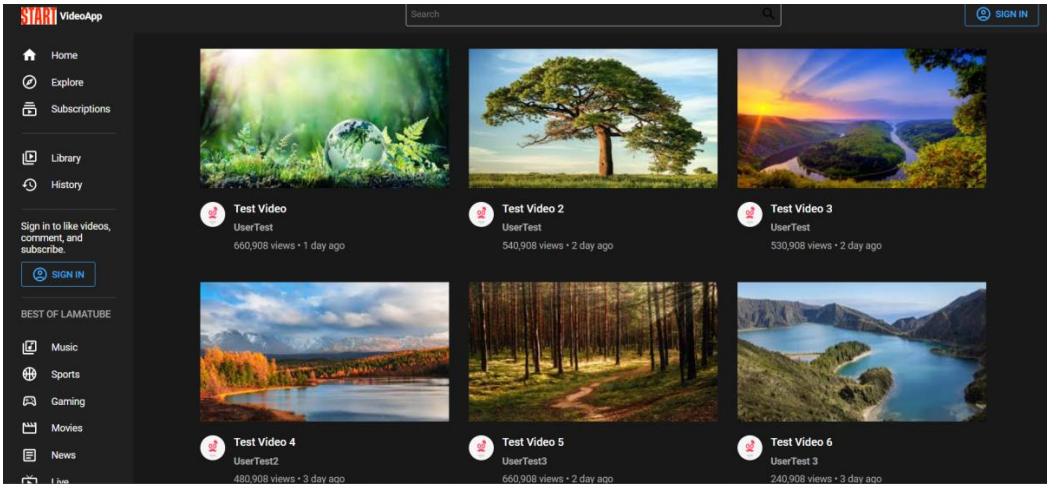

Рисунок 3.2 – Загальний вигляд головної сторінки додатку

Наступним кроком необхідно перейти до форми реєстрації користувача в системі (рис 3.3).

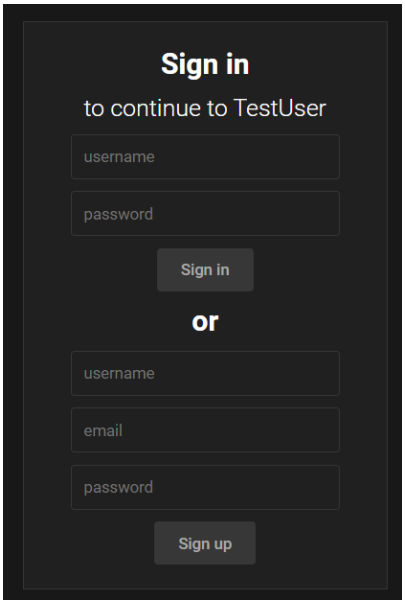

Рисунок 3.3 – Загальний вигляд сторінки авторизації інформаційної технології організації відеохостингу

Далі необхідно зареєструвати користувача до системи, для чого використовується спеціальна форма реєстрації. У цій формі користувач має вказати необхідну інформацію, таку як ім'я, електронну адресу та обрати безпечний пароль для входу до облікового запису. Після успішної реєстрації ці дані зберігаються у базі даних для подальшого використання.

Якщо користувач вже зареєстрований, йому потрібно лише ввести свої дані для входу в систему. Це включає в себе введення електронної адреси або ідентифікатора користувача, а також паролю, який він обрав під час реєстрації. Після введення коректних даних система перевіряє їх на вірність, а в разі успіху користувач отримує доступ до свого особистого облікового запису (рис 3.4).

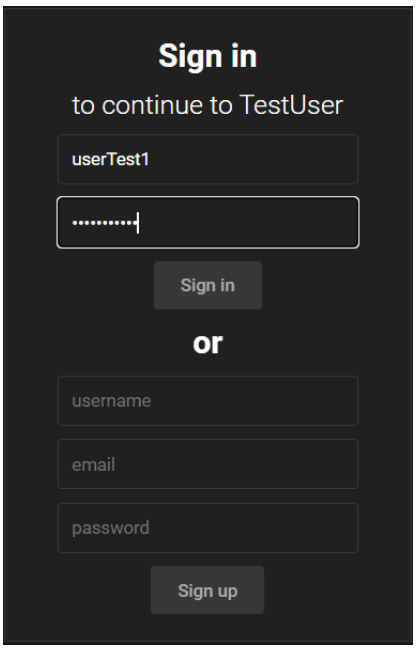

Рисунок 3.4 – Загальний вигляд заповненої форми реєстрації користувача інформаційної технології організації відеохостингу

Після реєстрації відбудеться перенаправлення на сторінку для отримання рекомендації для користувача на отримання відео для перегляду, для цього потрібно ввести в поле бажану тематику для перегляду відео та обрати час завантаження відео, приклад заповненої форми для отримання рекомендації наведено на рисунку 3.5.

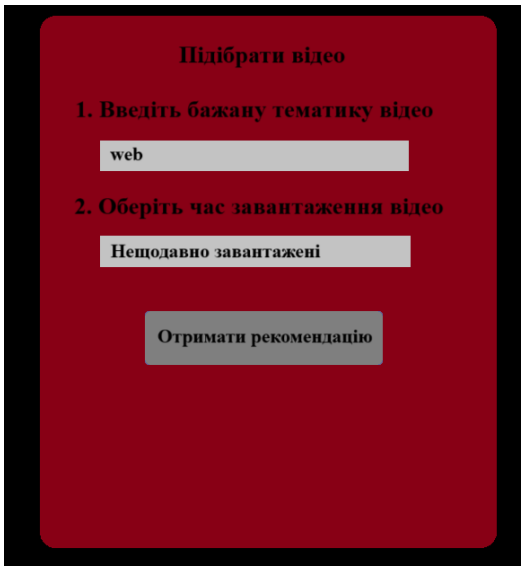

Рисунок 3.5 – Загальний вигляд заповненої форми для отримання рекомендації для перегляду відео

Результат отримання рекомендацій щодо перегляду відео наведено на рисунку 3.6. Користувач отримав відео стосовно web тематики, що було завантажено найближчим часом.

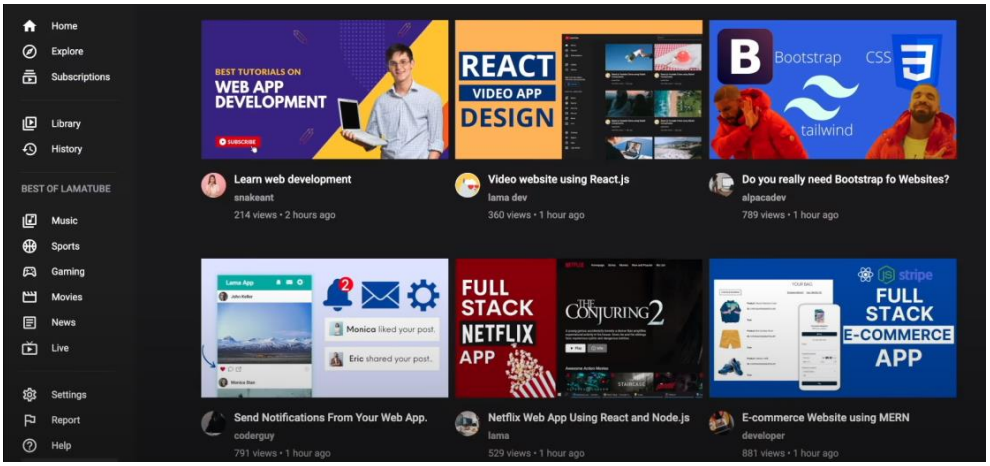

Рисунок 3.6 – Загальний вигляд результату отримання рекомендацій інформаційної технології організації відеохостингу

Після перегляду відео користувачеві запропоновано заповнити форму для оцінки переглянутого відео, де користувач може відмітити чи правильно було надано відео для тієї тематики, що він запросив, та відзначити чи порекомендує це відео він іншим користувачам системи, приклад заповненої форми для оцінки переглянутого відео наведено на рисунку 3.7.

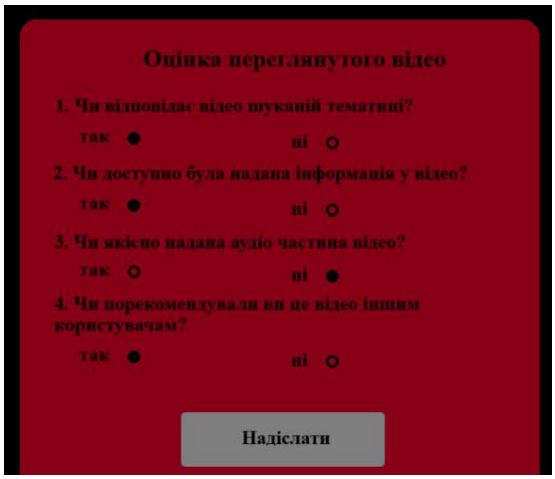

Рисунок 3.7 – Загальний вигляд заповненої форми для оцінки переглянутого відео в системі відеохостингу

Після реєстрації відбудеться перенаправлення на головну сторінку, наступним кроком є перевірка завантаження власного відео до системи відеохостингу, завантаження відбувається за допомогою спеціальної форми де необхідно заповнити всі поля, форма додання відео буде виглядати як зображено на рисунку 3.8.

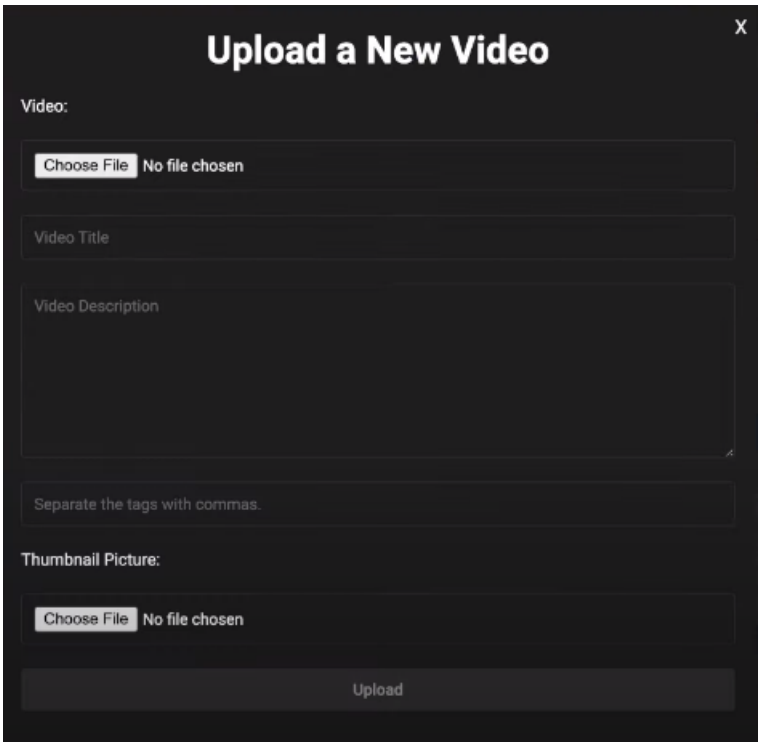

Рисунок 3.8 – Загальний вигляд форми завантаження нового відео до інформаційної технології організації відеохостингу

Для додання бажаного відео до бази даних відеохостингу, потрібно заповнити форму на сторінці «Upload a new Video» (рис. 3.9). Поле «Video» відповідає за відкриття необхідного відео користувача, яке обмежене обсягом не більше 20Гб. Поле «Video Title» відповідає за назву відео, під введеною назвою даний тест буде збережено до бази даних. «Video Description» – відповідає за опис відео для завантаження користувачем до системи. «Separate Tag» – вид міток які використовуються при подальшій рекомендації відео користувачам. «Thumbnail Picture» – відповідає за завантаження невеликої

картинки обсягом до 15Мб яка використовується для попереднього позначення відео.

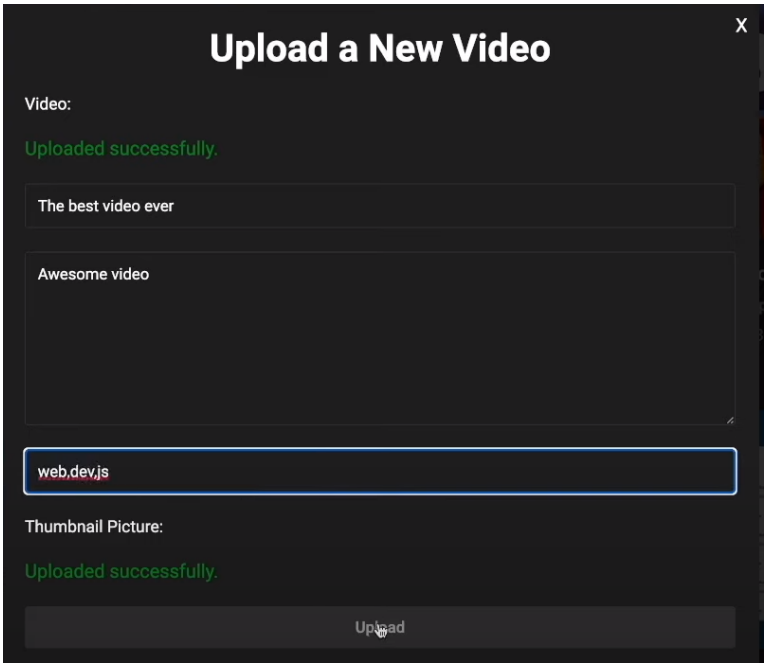

Рисунок 3.9 – Загальний вигляд заповненої форми завантаження нового відео до інформаційної технології організації відеохостингу

У меті була поставлена ціль є підвищення функціональних можливостей сервісу відеохостингу за допомогою розробки рекомендаційної системи, що дозволить підвищити ефективність підбору контенту в залежності від інтересів користувача. Для перевірки її досягнення потрібно порівняти характеристики розробленої інформаційної технології організації відеохостингу з існуючими аналогами.

Сутність кожного експерименту полягала у перегляді 10 різних відео та подальшої рекомендації схожих за інтересами відео для користуачеві. Отже, як показали дослідження, точність процесу рекомендації відео підвищиться, в середньому, не менш як на 7 відсотків. Для аналізу точності було взято за результат відсоткове відношення кількості бажаної рекомендації (спільні теги, схожий вміст та оцінка відео користувачем) до загальної виведених відео для користувача. Результати роботи інформаційної технології організації системи відеохостингу та її аналогів наведені в таблиці 3.1.

Таблиця 3.1 – Результати аналізу точності надання рекомендацій з використанням запропонованої інформаційної технології організації системи відеохостингу

	Vevo	VidLii	Vimeo	Інформаційна технологія організації системи відеохостингу
Точність,%	87	86	89	93

За результатами проведених досліджень, проведене тестування інформаційної технології організації системи відеохостингу. Рекомендація відео на основі методу колаборативної фільтрації забезпечує кращий процес надання відео, що дозволило підвищити точність процесу надання рекомендації, в середньому, не менш як на 7 відсотків в порівняні з аналогічними системами відеохостингу.

3.4 Висновок до розділу 3

У даному розділі обґрунтовано вибір інструментарію, що використовувався при розробці системи організації відеохостингу до якиго відноситься мова програмування JavaScript, бібліотека React для розробки інтерфейсів користувача та база даних MongoDB Для зберігання даних відеохостингу.

Описаний загальний алгоритм реєстрації та автеніфікації користувача. Розроблено основні програмні модулі інформаційної технології: модуль користувач, модуль обробки відео, модуль зберігання даних та модуль автентифікації.

Проведено тестування та аналіз результатів роботи програмного забезпечення для організації системи відеохостингу, яке показало, що інформаційна технологія організації системи відеохостингу задовільняє поставлену мету та процес надання рекомендацій користувачам відбувається точніше на 7% за існуючі аналоги, що доводить ефективність розробленої інформаційної технології.

4 ЕКОНОМІЧНА ЧАСТИНА

4.1 Оцінювання науково-технічного рівня та комерційного потенціалу розробки

Для оцінювання науково-технічного рівня та комерційного потенціалу розробки, проведемо комерційний та технологічний аудит розробки створеної в результаті виконання магістерської кваліфікаційної роботи інформаційної технології організації сервісу відеохостингу. Дана робота допомагає організувати такі процеси, як перегляд відео контенту та завантаження власного контенту до платформи, допомога з вибором бажаного відео. Особливістю програми є те, що дана технологія використовує інтеграцію експертної системи для вдалого підбору відео контенту для користувача сервісу.

Отримані результати комерційного та технологічного аудиту створеної розробки необхідно порівняти з обраним аналогом OnlineVideoService, що має ціну 1900 гривень.

Для проведення комерційного та технологічного аудиту було залучено 3-х незалежних експертів. Результати оцінювання науково-технічного рівня та комерційного потенціалу розробки представлено в таблиці 4.1 [20].

Таблиця 4.1 – Результати оцінювання науково-технічного рівня та комерційного потенціалу розробки

Критерії	Експерт (ПІБ, посада)		
	1	2	3
	Бали:		
1. Технічна здійсненність концепції	2	3	2
2. Ринкові переваги (наявність аналогів)	2	3	3
3. Ринкові переваги (ціна продукту)	3	4	3
4. Ринкові переваги (технічні властивості)	3	3	2

Продовження таблиці 4.1

5. Ринкові переваги (експлуатаційні витрати)	2	3	2
6. Ринкові перспективи (розмір ринку)	3	2	3
7. Ринкові перспективи (конкуренція)	3	3	3
8. Практична здійсненність (наявність фахівців)	4	4	3
9. Практична здійсненність (наявність фінансів)	3	2	2
10. Практична здійсненність (необхідність нових матеріалів)	3	3	4
11. Практична здійсненність (термін реалізації)	3	4	4
12. Практична здійсненність (розробка документів)	3	4	3
Сума балів	34	38	34
Середньоарифметична сума балів $СБ_с$	35,3		

Відповідно до отриманих результатів науково-технічний рівень та комерційний потенціал розробки є вище середнього, такий рівень був досягнутий шляхом покращення та/або розширенням функціональних можливостей нової розробки в порівнянні з аналогом, а також за рахунок збільшення показника точності надання рекомендацій.

Проаналізуємо рівень конкурентоспроможності розробки та аналога відповідно до технічних параметрів [21]. Технічні параметри розробки та аналога представлені в таблиці 4.2

Таблиця 4.2 – Технічні параметри аналога та розробки

Параметр	Одиниця виміру	Аналог	Нова розробка	Індекс зміни значення параметра	Коефіцієнт вагомості
Технічні					
Інтерфейс	-	6	7	1,16	0,25

Продовження таблиці 4.2

Функціональні можливості	-	7	9	1,29	0,15
Кількість класів рекомендацій	-	3	4	1,33	0,25
Оперування стандартними даними	-	7	8	1,14	0,2
Стабільний доступ в інтернет в інтернет	-	9	9	1	0,15
Економічні					
Ціна товару	грн	1900	1500	0,79	0,3
Транспортування	грн	600	500	0,83	0,1
Встановлення	грн	1100	700	0,63	0,15
Експлуатаційні витрати	грн	1050	1150	1,1	0,45

Розрахуємо нормативні параметри конкурентоспроможності за нормативними параметрами за формулою:

$$I_{нп} = \prod_{i=1}^{n} q_i \quad (4.1)$$

де $I_{нп}$ – загальний показник конкурентоспроможності за нормативними параметрами; q_i – одиничний показник за нормативним параметром.

$$I_{нп} = 1. \quad (4.2)$$

Знайдемо значення групового параметричного індексу за технічними параметрами з урахуванням вагомості кожного параметра:

$$I_{тп} = \sum_{i=1}^{n} q_i * \alpha_i, \quad (4.3)$$

де $I_{тп}$ – груповий параметричний індекс за технічними параметрами; q_i – одиничний параметричний показник параметра; α_i – вагомість параметричного показника.

$$I_{тп} = 1{,}16 * 0{,}25 + 1{,}29 * 0{,}15 + 1{,}33 * 0{,}25 + 1{,}4 * 0{,}2 + 1 * 0{,}15 = 1{,}25. \quad (4.4)$$

Знайдемо значення групового параметричного індексу за економічними параметрами:

$$I_{еп} = \sum_{i=1}^{n} q_i * \beta_i. \quad (4.5)$$

де $I_{еп}$ – груповий параметричний індекс за технічними параметрами; q_i – одиничний параметричний показник параметра; α_i – частка економічного параметра.

$$I_{еп} = 0{,}79 * 0{,}3 + 0{,}83 * 0{,}1 + 0{,}63 * 0{,}15 + 1{,}1 * 0{,}45 = 0{,}91. \quad (4.6)$$

На основі групових параметричних індексів за нормативними, технічними та економічними показниками розраховують інтегральний показник конкурентоспроможності за формулою:

$$К_{інт} = I_{нп} * \frac{I_{тп}}{I_{еп}}, \quad (4.7)$$

де $К_{інт}$ – інтегральний показник конкурентоспроможності; $I_{нп}$ – загальний показник конкурентоспроможності за нормативними параметрами; $I_{тп}$ – груповий параметричний індекс за технічними параметрами; $I_{еп}$ – груповий параметричний індекс за технічними параметрами.

$$К_{інт} = 1 * \frac{1{,}25}{0{,}91} = 1{,}37. \quad (4.8)$$

Отже, $К_{інт}$ > 1, це означає що нова розробка перевищує аналог за конкурентоспроможністю і є дуже перспективною при виведенні нової розробки на ринок.

4.2 Прогнозування витрат на виконання науково-дослідної, дослідно-конструкторської та конструкторсько-технологічної роботи

Розрахуємо основну заробітну плату дослідників ($З_о$) відповідно до посадових окладів працівників, за формулою:

$$З_о = \sum_{i=1}^{k} \frac{M_{ni} * t_i}{T_p}, \quad (4.9)$$

де M_{ni} – місячний посадовий оклад конкретного дослідника, грн; t_i – кількість днів роботи конкретного робітника, дн; T_p – середня кількість робочих днів в місяці, T_p=22 дні.

Відповідно до вище описаної формули, знайдемо основну заробітну плату керівника проекту з місячним посадовим окладом 19000 грн та 40 днями роботи.

$$З_о = \frac{19000 * 40}{22} = 34545{,}46 \quad (4.10)$$

Отже, заробітна плата керівника проекту складає 34545,46 гривень, решту розрахунків зводимо до таблиці 4.3.

Таблиця 4.3 – Витрати на заробітні плати дослідників

Найменування посади	Місячний посадовий оклад, грн	Оплата за робочий день, грн	Кількість днів роботи	Витрати на заробітну плату, грн
Керівник проекту	19000	863,64	40	34545,46
Науковий співробітник	17000	772,73	40	30909,1
Всього				65454,56

Розрахуємо додаткову заробітну плату дослідників ($З_{дод}$), за формулою:

$$З_{дод} = (З_о + З_р) * \frac{Н_{дод}}{100\%}, \quad (4.11)$$

де $З_о$ – заробітна плата дослідників; $З_р$ – заробітна плата робітників, $З_р = 0$; $Н_{дод}$ – норма нарахування додаткової заробітної плати, $Н_{дод} = 12\%$.

$$З_{дод} = (65454,56 + 0) * 0,12 = 7854,55. \quad (4.12)$$

Отже, додаткова заробітна плата дослідників складає 7854,55 гривень.

Розрахуємо відрахування на загальнообов’язкове державне соціальне страхування та здійснення заходів соціального захисту населення.

Нарахування ($З_н$) на заробітну плату працівників розраховується як відсоток від суми основної та додаткової заробітної плати за формулою:

$$З_н = (З_о + З_р + З_{дод}) * \frac{Н_{зп}}{100\%}, \quad (4.13)$$

де $З_о$ – заробітна плата дослідників; $З_р$ – заробітна плата робітників, $З_р = 0$; $З_{дод}$ – додаткова заробітна плата; $Н_{зп}$ – норма нарахування на заробітну плату, $Н_{зп} = 22\%$.

$$З_{дод} = (65454,56 + 0 + 7854,55) * 0,22 = 16128,004. \quad (4.13)$$

Отже, відрахування на загальнообов'язкове державне соціальне страхування та здійснення заходів соціального захисту населення складає 16128,004 гривень.

Розрахуємо витрати на матеріали (М) придбані в сторонніх підприємств, установ і організацій та витрачені на проведення досліджень за прямим призначенням згідно з нормами їх витрачання, за формулою:

$$M = \sum_{j=1}^{n} \text{Н}_j * \text{Ц}_j * \text{К}_j - \sum_{j=1}^{n} \text{В}_j * \text{Ц}_{\text{в}j}, \quad (4.14)$$

де Н_j – норма витрат на матеріалу, кг; Ц_j – вартість матеріалу, грн/кг; К_j – коефіцієнт транспортних витрат, $\text{К}_j = 1{,}13$; В_j – маса відходів, кг; $\text{Ц}_{\text{в}j}$ – ціна відходів, грн/кг.

$$M = 167{,}81 + 58{,}08 + 609{,}07 - 100{,}69 = 734{,}27. \quad (4.15)$$

Отже, витрати на матеріали складають 734,27 гривень, зведемо отримані результати до таблиці 4.4.

Таблиця 4.4 – Витрати на матеріали

Найменування матеріалу, марка, тип, сорт	Ціна за 1кг, грн	Норма витрат, кг	Величина відходів, кг	Ціна відходів, грн	Вартість витраченого матеріалу
Офісний папір, Maestro, A4, 80 г/м2	59,4	2,5	1,5	100,69	67,12

Продовження таблиця 4.4

Канцелярське приладдя, Maxflow, кулькова ручка, чорна	25,70	2	0	0	58,08
Картридж, Canon, PG-46, Black	539	1	0	0	609,07
Всього					734,27

Витрати на комплектуючі відсутні.

Витрати на спецустаткування відсутні.

Розрахуємо витрати на розробку та придбання спеціальних програмних засобів і програмного забезпечення ($В_{прг}$) необхідних для проведення досліджень, також витрати на їх проектування, формування та встановлення, за формулою:

$$В_{прг} = \sum_{i=1}^{k} Ц_i * С_i * К_i \ , \quad (4.16)$$

де $Ц_i$ – ціна одиниці відповідного програмного забезпечення, грн; $С_i$ – кількість одиниць відповідного програмного забезпечення, грн; $К_i$ – коефіцієнт, що враховує інсталяцію, налагоджування програмного забезпечення, $К_i = 1{,}11$.

$$В_{прг} = 466{,}6 + 744{,}41 = 1211{,}01 \ . \quad (4.17)$$

Отже, витрати на програмне забезпечення складає 1211,01 гривень, зведемо отримані результати до таблиці 4.5.

Таблиця 4.5 – Витрати на придбання програмного забезпечення

Найменування програмного засобу	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
Прикладне середовище розробки ПЗ	1	420,36	466,6
Хмарне середовище MongoDB	1	670,64	744,41
Разом			1211,01

Розрахуємо амортизаційні відрахування ($А_{обл}$) по кожному виду обладнання, устаткування та інших приладів і пристроїв, а також програмного забезпечення для проведення науково-дослідної роботи, за його наявності в дослідній організації або на підприємстві.

В спрощеному вигляді амортизаційні відрахування по кожному виду обладнання, приміщень та програмному забезпеченню тощо, можуть бути розраховані з використанням прямолінійного методу амортизації за формулою:

$$А_{обл} = \frac{Ц_б}{Т_в} * \frac{t_{вик}}{12}, \quad (4.18)$$

де $Ц_б$ – балансова вартість обладнання, програмних засобів, приміщень тощо, які використовувались для проведення досліджень, грн; $t_{вик}$ – термін використання обладнання, програмних засобів, приміщень під час досліджень, місяців; $Т_в$ – строк корисного використання обладнання, програмних засобів, приміщень тощо, років.

Відповідно до описаної вище формули знайдемо амортизаційні відрахування по приміщенню з балансовою вартістю 55000 гривень з терміном використання 2 місяці.

$$А_{обл} = \frac{550000}{20} * \frac{2}{12} = 4583{,}33\ . \quad (4.19)$$

Отже, амортизаційні відрахування по приміщенню складають 4583,33 гривень, решту розрахунків зводимо до таблиці 4.6.

Таблиця 4.6 – Амортизаційні відрахування по кожному виду обладнання

Найменування обладнання	Балансова вартість, грн	Строк корисного використання, років	Термін використання обладнання, місяців	Амортизаційні відрахування, грн
Приміщення	550000	20	2	4583,33
Комп'ютер та комп'ютерна периферія	45000	2	2	3 750
Офісне обладнання	20000	4	2	833,33
Всього				9166,66

Розрахуємо витрати на електроенергію ($В_е$), що витрачається з технологічною метою на проведення досліджень, за формулою:

$$В_е = \sum_{i=1}^{n} \frac{W_{yi} * t_i * Ц_e * К_{впі}}{\eta_i}, \quad (4.20)$$

де W_{yi} – встановлена потужність обладнання на певному етапі розробки, кВТ; t_i – тривалість обробки обладнання на етапі дослідження, год; $Ц_e$ – вартість 1 кВТ-години електроенергії, 7,50 грн; $К_{впі}$ – коефіцієнт, що враховує використання потужності, $К_{впі}$ = 0,875; η_i – коефіцієнт корисної дії обладнання, η_i = 0,95.

Відповідно до описано вище формули знайдемо витрати на електроенергію для комп'ютера потужністю 0,065 кВт, що пропрацював 266 годин.

$$В_е = \frac{0,065*266*7,50*0,875}{0,95} = 119,44. \quad (4.21)$$

Отже, витрати на електроенергію для комп'ютера на час проведення досліджень складає 119,44 гривень, решту розрахунків зводимо до таблиці 4.7.

Таблиця 4.7 – Витрати на електроенергію

Найменування обладнання	Встановлена потужність, кВт	Тривалість роботи, год	Сума, грн
Комп'ютер	0,065	266	119,44
Обслуговування робочого місця	0,018	266	33,08
Всього			152,52

Витрати на службові відрядження відсутні.

Витрати на роботи, які виконують сторонні підприємства, установи і організації відсутні.

Розрахуємо статтю витрат «Інші витрати» ($І_в$), до неї входять витрати, які не ввійшли в зазначені статті витрат і не відносяться до собівартості досліджень за прямими ознаками. Дані витрати розраховуються як відсоток від основної заробітної плати дослідників та робітників за формулою:

$$І_в = (З_о + З_р) * \frac{Н_{ів}}{100\%}, \quad (4.22)$$

де $З_о$ – заробітна плата дослідників, грн; $З_р$ – заробітна плата робітників, $З_р$ = 0 грн; $Н_{ів}$ – норма нарахування за статтею «Інші витрати».

Відповідно до описаної вище формули знайдемо «Інші витрати» при заробітній платі дослідників 65454,56 і нормі нарахувань 65%.

$$I_в = (65454{,}56 + 0) * 0{,}65 = 42545{,}46. \quad (4.23)$$

Отже, витрати за статтею «Інші витрати» складають 42545,46 гривень.

Розрахуємо статтю витрат «Накладні (загальновиробничі) витрати» ($В_{нзв}$), до них належать: витрати пов'язані з управлінням організацією; витрати на винахідництво та раціоналізацію; витрати на підготовку (перепідготовку) та навчання кадрів; витрати, пов'язані з набором робочої сили; витрати на оплату послуг банків; витрати, пов'язані з освоєнням виробництва продукції; витрати на науково-технічну інформацію та рекламу та ін.

Витрати розраховуються як відсоток від основної заробітної плати дослідників та робітників за формулою:

$$В_{нзв} = \left(З_о + З_р\right) * \frac{Н_{нзв}}{100\%}, \quad (4.24)$$

де $З_о$ – заробітна плата дослідників, грн; $З_р$ – заробітна плата робітників, $З_р$ = 0 грн; $Н_{нзв}$ – норма нарахування за статтею «Накладні (загальновиробничі) витрати».

Відповідно до описаної вище формули знайдемо «Накладні (загальновиробничі) витрати» при заробітній платі дослідників 65454,56 і нормі нарахувань 135%.

$$В_{нзв} = (65454{,}56 + 0) * 1{,}35 = 88363{,}66. \quad (4.25)$$

Отже, витрати за статтею «Накладні (загальновиробничі) витрати» складають 88363,66 гривень.

Знайдемо витрати на проведення науково-дослідної роботи ($В_{заг}$), для цього знайдемо суму всіх попередніх статей витрат:

$$В_{заг} = 65454{,}56 + 0 + 7854{,}55 + 16128{,}004 + 734{,}27 + 0 + 0 + 1211{,}01 + 9166{,}66 + 119{,}44 + 0 + 0 + 42545{,}46 + 88363{,}66 = 231577{,}61. \quad (4.26)$$

Отже, витрати на проведення науково-дослідної роботи складають 231577,61 гривень.

Знайдемо загальні витрати (ЗВ) на завершення науково-дослідної роботи та оформлення її результатів за формулою:

$$ЗВ = \frac{В_{заг}}{\eta}, \quad (4.27)$$

де $В_{заг}$ – витрати на проведення науково-дослідної роботи, грн; η – коефіцієнт, що характеризує етап виконання науково-дослідної роботи, η = 0,5.

Відповідно до описаної вище формули знайдемо загальні витрати на завершення науково-дослідної роботи та оформлення її результатів:

$$ЗВ = \frac{231577{,}61}{0{,}5} = 463155{,}22. \quad (4.28)$$

Отже, загальні витрати на завершення науково-дослідної роботи та оформлення її результатів складають 463155,22 гривень.

4.3 Розрахунок економічної ефективності науково-технічної розробки за її можливої комерціалізації потенційним інвестором

В ринкових умовах узагальнювальним позитивним результатом, що його може отримати потенційний інвестор від можливого впровадження результатів тієї чи іншої науково-технічної розробки, є збільшення у потенційного інвестора величини чистого прибутку. Саме зростання чистого прибутку забезпечить потенційному інвестору надходження додаткових коштів, дозволить покращити фінансові результати його діяльності, підвищить конкурентоспроможність та може позитивно вплинути на ухвалення рішення щодо комерціалізації цієї розробки.

Розрахуємо прибутки інвестора протягом трьох років після впровадження науково-технічної розробки. Для цього розрахуємо можливе збільшення чистого прибутку $\Delta П_i$, для кожного з років протягом яких очікується отримання позитивних результатів від можливого впровадження та комерціалізації науково-технічної розробки за формулою:

$$\Delta П_i = (\pm \Delta Ц_o * N + Ц_o * \Delta N)_i * \lambda * \rho * (1 - \frac{\vartheta}{100}). \qquad (4.29)$$

де і – рік, і = 1,2,3; $\pm \Delta Ц_o$ – зміна основного якісного показника від впровадження результатів науково-технічної розробки в аналізованому році, $\pm \Delta Ц_o = 0$; N – основний кількісний показник, який визначає величину попиту на аналогічні чи подібні розробки у році до впровадження результатів нової науково-технічної розробки, $N = 0$; $Ц_o$ – основний якісний показник, який визначає ціну реалізації нової науково-технічної розробки в аналізованому році; ΔN – зміна основного кількісного показника від впровадження результатів науково-технічної розробки в аналізованому році; λ – коефіцієнт, який враховує сплату потенційним інвестором податку на додану вартість, $\lambda = 0{,}8333$; ρ – коефіцієнт, який враховує рентабельність інноваційного продукту (послуги), ρ

= 0,3; ϑ – ставка податку на прибуток, який має сплачувати потенційний інвестор, ϑ = 18%.

Відповідно до описаної вище формули розрахуємо прибутки інвестора протягом трьох років при ціні реалізованої розробки 1500 і при впровадженні 25000 одиниць реалізованої розробки:

$$\Delta П_1 = (0 * 0 + 1500 * 12000)_1 * 0{,}8333 * 0{,}3 * \left(1 - \frac{18}{100}\right) = 3689852{,}4, \quad (4.30)$$

$$\Delta П_2 = (0 * 0 + 1500 * 8000)_2 * 0{,}8333 * 0{,}3 * \left(1 - \frac{18}{100}\right) = 2459901{,}6, \quad (4.31)$$

$$\Delta П_3 = (0 * 0 + 1500 * 5000)_3 * 0{,}8333 * 0{,}3 * \left(1 - \frac{18}{100}\right) = 1537438{,}5. \quad (4.32)$$

Отже, прибутки інвестора протягом першого року складають 3689852,4 гривень, протягом другого року – 2459901,6 гривень, протягом третього року – 1537438,5 гривень.

Розрахуємо приведену вартість збільшення всіх чистих прибутків ПП, які отримає інвестор від можливого впровадження та комерціалізації науково-технічної розробки:

$$ПП = \sum_{i=1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \quad (4.33)$$

де $\Delta П_i$ – збільшення прибутку для кожного з років протягом яких очікується отримання позитивних результатів від можливого впровадження та комерціалізації науково-технічної розробки, грн; T – період часу, протягом якого очікується отримання позитивних результатів від впровадження та комерціалізації науково-технічної розробки, роки; τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, τ = 0,1; t – період часу (в роках) від моменту початку впровадження науково-технічної розробки до моменту отримання потенційним інвестором додаткових чистих прибутків у цьому році.

$$ПП = \frac{3689852,4}{1,1^1} + \frac{2459901,6}{1,1^2} + \frac{1537438,5}{1,1^3} = 6542488,1. \quad (4.34)$$

Отже, приведена вартість збільшення всіх чистих прибутків, які отримає інвестор від можливого впровадження та комерціалізації науково-технічної розробки складає 6542488,1 гривень.

Розрахуємо величину початкових інвестицій PV, які інвестор повинен вкласти для впровадження та комерціалізації науково-технічної розробки, за формулою:

$$PV = k_{інв} * ЗВ, \quad (4.35)$$

де $k_{інв}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізації, $k_{інв}$ = 2; ЗВ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, грн.

Відповідно до описаної вище формули розрахуємо величину початкових інвестицій при загальних витратах на проведення науково-технічної розробки та оформлення її результатів 463155,22 гривень:

$$PV = 2 * 463155,22 = 926310,44. \quad (4.36)$$

Отже, величина початкових інвестицій, які інвестор повинен вкласти для впровадження та комерціалізації науково-технічної розробки складає 926310,44 гривні.

Розрахуємо абсолютний економічний ефект $Е_{абс}$ для інвестора від можливого впровадження та комерціалізації науково-технічної розробки:

$$Е_{абс} = ПП - PV, \quad (4.37)$$

де ПП – приведена вартість зростання всіх чистих прибутків від можливого впровадження та комерціалізації науково-технічної розробки, грн; PV – теперішня вартість початкових інвестицій, грн.

Відповідно до описаної вище формули розрахуємо абсолютний економічний ефект для інвестора від можливого впровадження та комерціалізації науково-технічної розробки:

$$\text{Е}_{\text{абс}} = 6542488{,}1 - 926310{,}44 = 5616177{,}66. \qquad (4.38)$$

Отже, значення абсолютного економічного ефекту для інвестора від можливого впровадження та комерціалізації науково-технічної розробки складає $5616177{,}66$ гривень. Значення $\text{Е}_{\text{абс}}$ є великим додатнім значенням, що свідчить про потенційну зацікавленість інвестора у впровадженні та комерціалізації науково-технічної розробки.

Для остаточного прийняття рішення по впровадження науково-технічної розробки необхідно розрахувати внутрішню економічну дохідність $\text{Е}_{\text{в}}$ вкладених інвестицій та порівняти її з так званою бар'єрною ставкою дисконтування $\tau_{\text{мін}}$, яка визначає ту мінімальну внутрішню економічну дохідність, нижче якої інвестиції в будь-яку науково-технічну розробку вкладати буде економічно недоцільно.

Внутрішню економічну дохідність $\text{Е}_{\text{в}}$ розраховується за формулою:

$$\text{Е}_{\text{в}} = \sqrt[\text{Т}_{\text{ж}}]{1 + \frac{\text{Е}_{\text{абс}}}{PV}} - 1, \qquad (4.39)$$

де $\text{Е}_{\text{абс}}$ – абсолютний економічний ефект від вкладених інвестицій, грн; PV – теперішня вартість початкових інвестицій, грн; $\text{Т}_{\text{ж}}$ – життєвий цикл науково-технічної розробки, тобто час від початку її розробки до закінчення отримування позитивних результатів від її впровадження, роки.

Відповідно до описаної вище формули розрахуємо внутрішню економічну дохідність при абсолютному економічному ефекту 5616177,66 гривень, теперішній вартості інвестицій 926310,44 гривні і життєвому циклі 3 роки.

$$E_в = \sqrt[3]{1 + \frac{5616177,66}{926310,44}} - 1 = 0,91. \quad (4.40)$$

Отже, внутрішня економічна дохідність вкладених інвестицій дорівнює 0,91.

Визначимо бар'єрну ставку дисконтування $\tau_{мін}$ за формулою:

$$\tau_{мін} = d + f, \quad (4.41)$$

де d – середньозважена ставка за депозитними операціями в комерційних банках, d = 0,35; f – показник, що характеризує ризикованість вкладення інвестицій, f = 0,4.

Відповідно до описаної вище формули розрахуємо бар'єрну ставку дисконтування:

$$\tau_{мін} = 0,35 + 0,4 = 0,75. \quad (4.42)$$

Отже, значення величини $E_в > \tau_{мін}$, тому інвестор може бути зацікавлений у фінансуванні впровадження науково-технічної розробки та виведенні її на ринок.

Розрахуємо період окупності інвестицій $T_{ок}$, які можуть бути вкладені інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$T_{ок} = \frac{1}{E_в}, \quad (4.43)$$

де $E_в$ – внутрішня економічна дохідність вкладених інвестицій, $E_в$ = 0,91.

Відповідно до описаної вище формули розрахуємо період окупності інвестицій:

$$T_{ок} = \frac{1}{0,91} = 1,1. \quad (4.44)$$

Отже, період окупності інвестицій складає 1,1. Значення величини $T_{ок}$ < 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати інвестора профінансувати впровадження цієї розробки та виведення її на ринок.

4.4 Висновок до розділу 4

Економічна частина роботи містить розрахунок витрат на розробку та виготовлення нового технічного рішення, сума яких складає 463155,22 гривень, також спрогнозовано витрати по кожній статті витрат. Також було розраховано чистий прибуток який може отримати інвестор від реалізації нового технічного рішення, розраховано економічний ефект і термін окупності вкладених інвестицій. В результаті розрахунків можна зробити висновок, що розробка є дешевшою та перспективнішою за аналог. Період окупності інвестицій складає 1,1 роки, що менше 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

ВИСНОВКИ

У результаті виконання магістерської кваліфікаційної роботи було проведено аналіз сучасного стану розвитку систем відеохостингів, аналізу відомих технічних рішень програмних систем організації систем відеохостингів, надання рекомендацій для відео-контенту та аналіз існуючі методи надання рекомендацій для відео-контенту.

Було здійснено проектування математичної моделі для надання рекомендацій для інформаційної технології системи організації відеохостингу, розроблено структуру програмного модуля системи сервісу відеохостингу та розроблено структуру роботи програмного модуля надання рекомендацій відеохостингу, а також розроблено алгоритм роботи системи організації відеохостингу.

Програмно реалізовано програмний модуль для інформаційної технології організації системи відеохостингу. Обґрунтовано вибір інструментарію при розробці системи відеохостингу, проведено тестування програмної частини системи організації відеохостингу яке показало, що інформаційна технологія організації системи відеохостингу задовільняє поставлену мету та процес надання рекомендацій користувачам відбувається точніше на 7% за існуючі аналоги, що доводить ефективність розробленої інформаційної технології.

Виконано розрахунок витрат на розробку нового програмного продукту, сума яких складає 463155,22 гривень. Було прогнозовано орієнтовану величину витрат по кожній з статей витрат. Також розраховано чистий прибуток, який може отримати виробник від реалізації нового технічного рішення, розраховано період окупності витрат для інвестора та економічний ефект при використанні даної розробки. У результаті аналізу розрахунків можна зробити висновок, що розроблений програмний продукт дешевший за аналог і є високо конкурентоспроможним. Період окупності складе близько 1,1 роки що робить розробку комерційно привабливою.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Інформаційна технологія сервісу відеохостинг / В.С. Кузьменко, Я.В. Іванчук – Тези LII науково-технічної конференції підрозділів ВНТУ (НТКП ВНТУ-2023). – [Електронний ресурс]. – Тип доступу: https://conferences.vntu.edu.ua/index.php/all-fksa/all-fksa-2023/paper/view/17382.

2. Розробка веб-додатку музичного стримінгового сервісу/ В.С. Кузьменко, В.І. Месюра – Тези LI науково-технічної конференції підрозділів ВНТУ (НТКП ВНТУ-2022). – [Електронний ресурс]. – Тип доступу: https://conferences.vntu.edu.ua/index.php/all-fksa/all-fksa-2022/paper/view/14831/12500.

3. Сервіси хостингу – [Електронний ресурс]. – Режим доступу: https://habr.com/ru/company/audiomania/blog/473206/.

4. Вплив COVID на споживання контенту – [Електронний ресурс]. – Режим доступу: http://surl.li/gtnp.

5. Twitch – [Електронний ресурс]. – Режим доступу: https://tsn.ua/cybersport/scho-take-twitch-yak-nim-koristuvatisya-i-dlya-chogo-potriben-servis-1810927.html.

6. YouTube Live – [Електронний ресурс]. – Режим доступу: https://www.freeconference.com/uk/blog/broadcast-a-live-video-conference-on-youtube/.

7. Facebook Live – [Електронний ресурс]. – Режим доступу: https://www.facebook.com/formedia/tools/facebook-live.

8. Відеохостинг та його переваги – [Електронний ресурс]. – Режим доступу: https://timeweb.com/ru/community/articles/videohostingi-obzor-samih-populyarnyh-ploshchadok.

9. Відеохостинг YouTube – [Електронний ресурс]. – Режим доступу: https://uk.economy-pedia.com/11030595-youtube.

10. Відеохостинг Netflix – [Електронний ресурс]. – Режим доступу: https://help.netflix.com/uk/node/412.

11. Відеохостинг Vimeo – [Електронний ресурс]. – Режим доступу: https://ms.detector.media/internet/post/32225/2023-06-20-vimeo-zaprovadyla-try-shi-funktsii-dlya-redaguvannya-video/.

12. Відеохостинг Hulu – [Електронний ресурс]. – Режим доступу: https://tebapit.com/%D1%89%D0%BE-%D1%82%D0%B0%D0%BA%D0%B5-hulu-%D1%96-%D1%8F%D0%BA-%D1%86%D0%B5-%D0%BF%D1%80%D0%B0%D1%86%D1%8E%D1%94/.

13. Колаборативна фільтрація– [Електронний ресурс]. – Режим доступу: https://csecurity.kubg.edu.ua/index.php/journal/article/view/101.

14. HTTP протокол – [Електронний ресурс]. – Режим доступу: https://training.qatestlab.com/blog/technical-articles/http-protocol-what-and-where-to-test/.

15. REST – [Електронний ресурс]. – Режим доступу: https://habr.com/ru/articles/38730/.

16. Що таке JavaScript – [Електронний ресурс]. – Режим доступу: https://astwellsoft.com/uk/blog/tehnology/javascript.html.

17. Знайомство з React – [Електронний ресурс]. – Режим доступу: https://uk.legacy.reactjs.org/tutorial/tutorial.html.

18. Основи MongoDB – [Електронний ресурс]. – Режим доступу: https://devzone.org.ua/post/osnovi-mongodb.

19. React-додатки – [Електронний ресурс]. – Режим доступу: https://uk.legacy.reactjs.org/docs/create-a-new-react-app.html.

20. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт / Уклад. : В. О. Козловський, О. Й. Лесько, В. В. Кавецький. – Вінниця : ВНТУ, 2021. – 42 с.

21. Кавецький В. В. Економічне обґрунтування інноваційних рішень: практикум / В. В. Кавецький, В. О. Козловський, І. В. Причепа – Вінниця : ВНТУ, 2016. – 113 с.

Додаток А (обов'язковий)

Протокол перевірки кваліфікаційної роботи на наявність текстових запозичень

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Інформаційна технологія організації сервісу відеохостингу

Тип роботи: магістерська кваліфікаційна робота
(БДР, МКР)

Підрозділ кафедра комп'ютерних наук, ФІІТА
(кафедра, факультет)

Показники звіту подібності Unicheck

Оригінальність 82,6% Схожість 17,4%

Аналіз звіту подібності (відмітити потрібне):

✓ Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.

□ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.

□ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck щодо роботи.

Автор роботи Кузьменко В.С.

Керівник роботи Іванчук Я.В.

Опис прийнятого рішення

Магістерську кваліфікаційну роботу допущено до захисту

Особа, відповідальна за перевірку Озеранський В.С.

Додаток Б (обов'язковий)

Лістинг програми

```
function normalizeCoMatrix(coMatrix, normalizerMatrix) {
  return math.dotDivide(coMatrix, normalizerMatrix);
}
unction getRatedItemsForUser(ratings, userIndex, numItems) {
  const ratedItems = [];
  for (let index = 0; index < numItems; index += 1) {
    if (ratings[userIndex][index] !== 0) {
      ratedItems.push(index);
    }
  }
  return ratedItems;
}
function arraysAreEqual(array1, array2) {
  if (array1.length !== array2.length) {
    return false;
  }
  for (let index = 0; index < array1.length; index += 1) {
    if (array1[index] !== array2[index]) {
      return false;
    }
  }
  return true;
}
function typeCheckRatings(ratings) {
  if (!Array.isArray(ratings)) {
    throw new TypeError('The ratings and coMatrix field should be an array of arrays (matrix)');
  }
}
function typeCheckCoOccurrenceMatrix(coMatrix, numItems) {
  if (!(coMatrix instanceof math.Matrix)) {
    throw new TypeError('The occurrence matrix should be a mathJS Matrix object generated by
createCoMatrix');
  }
  if (!arraysAreEqual(coMatrix.size(), [numItems, numItems])) {
    throw new RangeError('Co matrix has wrong dimensions. Make sure to generate it using
createCoMatrix');
  }
}
function typeCheckUserIndex(userIndex, ratings) {
  if (!Number.isInteger(userIndex)) {
    throw new TypeError('The field userIndex should be an integer');
  }
  if ((userIndex < 0) || (userIndex >= ratings.length)) {
    throw new RangeError('User index out of rage');
  }
}
function checkRatingValues(ratingMatrix) {
```

```
  const allowedRatings = [0, 1];
  ratingMatrix.forEach((value) => {
    if ((!Number.isInteger(value)) || (!allowedRatings.includes(value))) {
      throw new TypeError('Wrong rating in rating array. Currently permitted values are 0 and 1');
    }
  });
  return true;
}
function getRecommendations(ratings, coMatrix, userIndex) {
  typeCheckRatings(ratings);
  let ratingsMatrix;
  try {
    ratingsMatrix = math.matrix(ratings);
  } catch (error) {
    throw new RangeError('Dimension error in ratings matrix');
  }
  const numItems = ratingsMatrix.size()[1];
  typeCheckCoOccurrenceMatrix(coMatrix, numItems);
  typeCheckUserIndex(userIndex, ratings);
  const ratedItemsForUser = getRatedItemsForUser(ratings, userIndex, numItems);
  const numRatedItems = ratedItemsForUser.length;
  const similarities = math.zeros(numRatedItems, numItems);
  for (let rated = 0; rated < numRatedItems; rated += 1) {
    for (let item = 0; item < numItems; item += 1) {
      similarities.set([rated, item], coMatrix.get([ratedItemsForUser[rated], item])
                                + similarities.get([rated, item]));
    }
  }
 let recommendations = math.zeros(numItems);
  for (let y = 0; y < numRatedItems; y += 1) {
    for (let x = 0; x < numItems; x += 1) {
      recommendations.set([x], recommendations.get([x]) + similarities.get([y, x]));
    }
  }
  recommendations = math.dotDivide(recommendations, numRatedItems);
  const rec = recommendations.toArray();
  let recSorted = recommendations.toArray();
  recSorted.sort((a, b) => b - a);
  if (ONLY_RECOMMEND_FROM_SIMILAR_TASTE) {
    recSorted = recSorted.filter((element) => element !== 0);
  }
  let recOrder = recSorted.map((element) => {
    const index = rec.indexOf(element);
    rec[index] = null;
    return index;
  });
  recOrder = recOrder.filter((index) => !ratedItemsForUser.includes(index));
  return recOrder;
}
function createCoMatrix(ratings) {
  typeCheckRatings(ratings);
  let ratingsMatrix;
```

```
  try {
    ratingsMatrix = math.matrix(ratings);
  } catch (error) {
    throw new RangeError('Dimension error in ratings matrix');
  }
  checkRatingValues(ratingsMatrix);
  const nUsers = ratingsMatrix.size()[0];
  const nItems = ratingsMatrix.size()[1];
  const coMatrix = math.zeros(nItems, nItems);
  // const normalizerMatrix = math.zeros(nItems, nItems)
  const normalizerMatrix = math.identity(nItems);
  for (let y = 0; y < nUsers; y += 1) {
    for (let x = 0; x < (nItems - 1); x += 1) {
      for (let index = x + 1; index < nItems; index += 1) {
        if (ratings[y][x] === 1 && ratings[y][index] === 1) {
          coMatrix.set([x, index], coMatrix.get([x, index]) + 1);
          coMatrix.set([index, x], coMatrix.get([index, x]) + 1);
        }
        if (NORMALIZE_ON_POPULARITY && (ratings[y][x] === 1 || ratings[y][index] === 1)) {
          normalizerMatrix.set([x, index], normalizerMatrix.get([x, index]) + 1);
          normalizerMatrix.set([index, x], normalizerMatrix.get([index, x]) + 1);
        }
      }
    }
  }
  return NORMALIZE_ON_POPULARITY ? normalizeCoMatrix(coMatrix, normalizerMatrix) :
coMatrix;
}
function collaborativeFilter(ratings, userIndex) {
  if (!Array.isArray(ratings)) return false;
  const coMatrix = createCoMatrix(ratings);
  const recommendations = getRecommendations(ratings, coMatrix, userIndex);
  return recommendations;
}
module.exports = {
  cFilter: collaborativeFilter,
  getRecommendations,
  coMatrix: createCoMatrix,
};
export const signup = async (req, res, next) => {
  try {
    const salt = bcrypt.genSaltSync(10);
    const hash = bcrypt.hashSync(req.body.password, salt);
    const newUser = new User({ ...req.body, password: hash });

    await newUser.save();
    res.status(200).send("User has been created!");
  } catch (err) {
    next(err);
  }
};
```

```
export const signin = async (req, res, next) => {
  try {
    const user = await User.findOne({ name: req.body.name });
    if (!user) return next(createError(404, "User not found!"));

    const isCorrect = await bcrypt.compare(req.body.password, user.password);

    if (!isCorrect) return next(createError(400, "Wrong Credentials!"));

    const token = jwt.sign({ id: user._id }, process.env.JWT);
    const { password, ...others } = user._doc;

    res
      .cookie("access_token", token, {
        httpOnly: true,
      })
      .status(200)
      .json(others);
  } catch (err) {
    next(err);
  }
};

export const googleAuth = async (req, res, next) => {
  try {
    const user = await User.findOne({ email: req.body.email });
    if (user) {
      const token = jwt.sign({ id: user._id }, process.env.JWT);
      res
        .cookie("access_token", token, {
          httpOnly: true,
        })
        .status(200)
        .json(user._doc);
    } else {
      const newUser = new User({
        ...req.body,
        fromGoogle: true,
      });
      const savedUser = await newUser.save();
      const token = jwt.sign({ id: savedUser._id }, process.env.JWT);
      res
        .cookie("access_token", token, {
          httpOnly: true,
        })
        .status(200)
        .json(savedUser._doc);
    }
  } catch (err) {
    next(err);
  }
};
```

```
export const addComment = async (req, res, next) => {
  const newComment = new Comment({ ...req.body, userId: req.user.id });
  try {
    const savedComment = await newComment.save();
    res.status(200).send(savedComment);
  } catch (err) {
    next(err);
  }
};

export const deleteComment = async (req, res, next) => {
  try {
    const comment = await Comment.findById(res.params.id);
    const video = await Video.findById(res.params.id);
    if (req.user.id === comment.userId || req.user.id === video.userId) {
      await Comment.findByIdAndDelete(req.params.id);
      res.status(200).json("The comment has been deleted.");
    } else {
      return next(createError(403, "You can delete ony your comment!"));
    }
  } catch (err) {
    next(err);
  }
};

export const getComments = async (req, res, next) => {
  try {
    const comments = await Comment.find({ videoId: req.params.videoId });
    res.status(200).json(comments);
  } catch (err) {
    next(err);
  }
};
export const update = async (req, res, next) => {
  if (req.params.id === req.user.id) {
    try {
      const updatedUser = await User.findByIdAndUpdate(
        req.params.id,
        {
          $set: req.body,
        },
        { new: true }
      );
      res.status(200).json(updatedUser);
    } catch (err) {
      next(err);
    }
  } else {
    return next(createError(403, "You can update only your account!"));
  }
};
```

```
export const deleteUser = async (req, res, next) => {
  if (req.params.id === req.user.id) {
    try {
      await User.findByIdAndDelete(req.params.id);
      res.status(200).json("User has been deleted.");
    } catch (err) {
      next(err);
    }
  } else {
    return next(createError(403, "You can delete only your account!"));
  }
};

export const getUser = async (req, res, next) => {
  try {
    const user = await User.findById(req.params.id);
    res.status(200).json(user);
  } catch (err) {
    next(err);
  }
};

export const subscribe = async (req, res, next) => {
  try {
    await User.findByIdAndUpdate(req.user.id, {
      $push: { subscribedUsers: req.params.id },
    });
    await User.findByIdAndUpdate(req.params.id, {
      $inc: { subscribers: 1 },
    });
    res.status(200).json("Subscription successfull.")
  } catch (err) {
    next(err);
  }
};

export const unsubscribe = async (req, res, next) => {
  try {
    try {
      await User.findByIdAndUpdate(req.user.id, {
        $pull: { subscribedUsers: req.params.id },
      });
      await User.findByIdAndUpdate(req.params.id, {
        $inc: { subscribers: -1 },
      });
      res.status(200).json("Unsubscription successfull.")
    } catch (err) {
      next(err);
    }
  } catch (err) {
    next(err);
  }
```

```
};

export const like = async (req, res, next) => {
  const id = req.user.id;
  const videoId = req.params.videoId;
  try {
    await Video.findByIdAndUpdate(videoId,{
      $addToSet:{likes:id},
      $pull:{dislikes:id}
    })
    res.status(200).json("The video has been liked.")
  } catch (err) {
    next(err);
  }
};

export const dislike = async (req, res, next) => {
    const id = req.user.id;
    const videoId = req.params.videoId;
    try {
      await Video.findByIdAndUpdate(videoId,{
        $addToSet:{dislikes:id},
        $pull:{likes:id}
      })
      res.status(200).json("The video has been disliked.")
  } catch (err) {
    next(err);
  }
};
const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <Provider store= {store}>
    <PersistGate loading={null} persistor={persistor}>
      <App />
    </PersistGate>
    </Provider>
  </React.StrictMode>
);
const Container = styled.div`
  display: flex;
`;

const Main = styled.div`
  flex: 7;
  background-color: ${({ theme }) => theme.bg};
`;
const Wrapper = styled.div`
  padding: 22px 96px;
`;

function App() {
```

```
  const [darkMode, setDarkMode] = useState(true);
  const { currentUser } = useSelector((state) => state.user);

  return (
    <ThemeProvider theme={darkMode ? darkTheme : lightTheme}>
      <Container>
        <BrowserRouter>
          <Menu darkMode={darkMode} setDarkMode={setDarkMode} />
          <Main>
            <Navbar />
            <Wrapper>
              <Routes>
                <Route path="/">
                  <Route index element={<Home type="random" />} />
                  <Route path="trends" element={<Home type="trend" />} />
                  <Route path="subscriptions" element={<Home type="sub" />} />
                  <Route path="search" element={<Search />} />
                  <Route
                    path="signin"
                    element={currentUser ? <Home /> : <SignIn />}
                  />
                  <Route path="video">
                    <Route path=":id" element={<Video />} />
                  </Route>
                </Route>
              </Routes>
            </Wrapper>
          </Main>
        </BrowserRouter>
      </Container>
    </ThemeProvider>
  );
}

export default App;
const initialState = {
  currentVideo: null,
  loading: false,
  error: false,
};

export const videoSlice = createSlice({
  name: "video",
  initialState,
  reducers: {
    fetchStart: (state) => {
      state.loading = true;
    },
    fetchSuccess: (state, action) => {
      state.loading = false;
      state.currentVideo = action.payload;
    },
```

```
    fetchFailure: (state) => {
      state.loading = false;
      state.error = true;
    },
    like: (state, action) => {
      if (!state.currentVideo.likes.includes(action.payload)) {
        state.currentVideo.likes.push(action.payload);
        state.currentVideo.dislikes.splice(
          state.currentVideo.dislikes.findIndex(
            (userId) => userId === action.payload
          ),
          1
        );
      }
    },
    dislike: (state, action) => {
      if (!state.currentVideo.dislikes.includes(action.payload)) {
        state.currentVideo.dislikes.push(action.payload);
        state.currentVideo.likes.splice(
          state.currentVideo.likes.findIndex(
            (userId) => userId === action.payload
          ),
          1
        );
      }
    },
  },
});

export const { fetchStart, fetchSuccess, fetchFailure, like, dislike } =
  videoSlice.actions;

export default videoSlice.reducer;
const initialState = {
  currentUser: null,
  loading: false,
  error: false,
};

export const userSlice = createSlice({
  name: "user",
  initialState,
  reducers: {
    loginStart: (state) => {
      state.loading = true;
    },
    loginSuccess: (state, action) => {
      state.loading = false;
      state.currentUser = action.payload;
    },
    loginFailure: (state) => {
      state.loading = false;
```

```
      state.error = true;
    },
    logout: (state) => {
      state.currentUser = null;
      state.loading = false;
      state.error = false;
    },
    subscription: (state, action) => {
      if (state.currentUser.subscribedUsers.includes(action.payload)) {
        state.currentUser.subscribedUsers.splice(
          state.currentUser.subscribedUsers.findIndex(
            (channelId) => channelId === action.payload
          ),
          1
        );
      } else {
        state.currentUser.subscribedUsers.push(action.payload);
      }
    },
  },
});

export const { loginStart, loginSuccess, loginFailure, logout, subscription } =
  userSlice.actions;

export default userSlice.reducer;
const Upload = ({ setOpen }) => {
  const [img, setImg] = useState(undefined);
  const [video, setVideo] = useState(undefined);
  const [imgPerc, setImgPerc] = useState(0);
  const [videoPerc, setVideoPerc] = useState(0);
  const [inputs, setInputs] = useState({});
  const [tags, setTags] = useState([]);

  const navigate = useNavigate()

  const handleChange = (e) => {
    setInputs((prev) => {
      return { ...prev, [e.target.name]: e.target.value };
    });
  };

  const handleTags = (e) => {
    setTags(e.target.value.split(","));
  };

  const uploadFile = (file, urlType) => {
    const storage = getStorage(app);
    const fileName = new Date().getTime() + file.name;
    const storageRef = ref(storage, fileName);
    const uploadTask = uploadBytesResumable(storageRef, file);
```

```
    uploadTask.on(
      "state_changed",
      (snapshot) => {
        const progress =
          (snapshot.bytesTransferred / snapshot.totalBytes) * 100;
        urlType === "imgUrl" ? setImgPerc(Math.round(progress)) :
setVideoPerc(Math.round(progress));
        switch (snapshot.state) {
          case "paused":
            console.log("Upload is paused");
            break;
          case "running":
            console.log("Upload is running");
            break;
          default:
            break;
        }
      },
      (error) => {},
      () => {
        getDownloadURL(uploadTask.snapshot.ref).then((downloadURL) => {
          setInputs((prev) => {
            return { ...prev, [urlType]: downloadURL };
          });
        });
      }
    );
  };

  useEffect(() => {
    video && uploadFile(video , "videoUrl");
  }, [video]);

  useEffect(() => {
    img && uploadFile(img, "imgUrl");
  }, [img]);

  const handleUpload = async (e)=>{
    e.preventDefault();
    const res = await axios.post("/videos", {...inputs, tags})
    setOpen(false)
    res.status===200 && navigate(`/video/${res.data._id}`)
  }

  return (
    <Container>
      <Wrapper>
        <Close onClick={() => setOpen(false)}>X</Close>
        <Title>Upload a New Video</Title>
        <Label>Video:</Label>
        {videoPerc > 0 ? (
          "Uploading:" + videoPerc
```

```
        ) : (
          <Input
            type="file"
            accept="video/*"
            onChange={(e) => setVideo(e.target.files[0])}
          />
        )}
        <Input
          type="text"
          placeholder="Title"
          name="title"
          onChange={handleChange}
        />
        <Desc
          placeholder="Description"
          name="desc"
          rows={8}
          onChange={handleChange}
        />
        <Input
          type="text"
          placeholder="Separate the tags with commas."
          onChance={handleTags}
        />
        <Label>Image:</Label>
        {imgPerc > 0 ? (
          "Uploading:" + imgPerc + "%"
        ) : (
          <Input
            type="file"
            accept="image/*"
            onChange={(e) => setImg(e.target.files[0])}
          />
        )}
        <Button onClick={handleUpload}>Upload</Button>
      </Wrapper>
    </Container>
  );
};

export default Upload;
const Navbar = () => {
  const navigate = useNavigate()
  const [open, setOpen] = useState(false);
  const [q, setQ] = useState("");
  const { currentUser } = useSelector((state) => state.user);
  return (
    <>
      <Container>
        <Wrapper>
          <Search>
            <Input
```

```
            placeholder="Search"
            onChange={(e) => setQ(e.target.value)}
          />
          <SearchOutlinedIcon onClick={()=>navigate(`/search?q=${q}`)}/>
        </Search>
        {currentUser ? (
          <User>
            <VideoCallOutlinedIcon onClick={() => setOpen(true)} />
            <Avatar src={currentUser.img} />
            {currentUser.name}
          </User>
        ) : (
          <Link to="signin" style={{ textDecoration: "none" }}>
            <Button>
              <AccountCircleOutlinedIcon />
              SIGN IN
            </Button>
          </Link>
        )}
      </Wrapper>
    </Container>
    {open && <Upload setOpen={setOpen} />}
  </>
 );
};

export default Navbar;
```

Додаток В (обов'язковий)

ІЛЮСТРАТИВНА ЧАСТИНА

ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ ОРГАНІЗАЦІЇ СЕРВІСУ ВІДЕОХОСТИНГУ

Виконав: студент 2-го курсу, групи 3КН-22м
спеціальності 122 «Комп'ютерні науки»
(шифр і назва напряму підготовки, спеціальності)
Кузьменко В.С.
(прізвище та ініціали)

Керівник: д.т.н., проф. кафедри КН
Іванчук Я.В.
(прізвище та ініціали)
«07» 12 2023 р.

Вінниця ВНТУ - 2023 рік

Рис. В.1 – Принципова схема алгоритму роботи рекомендаційного програмного модуля

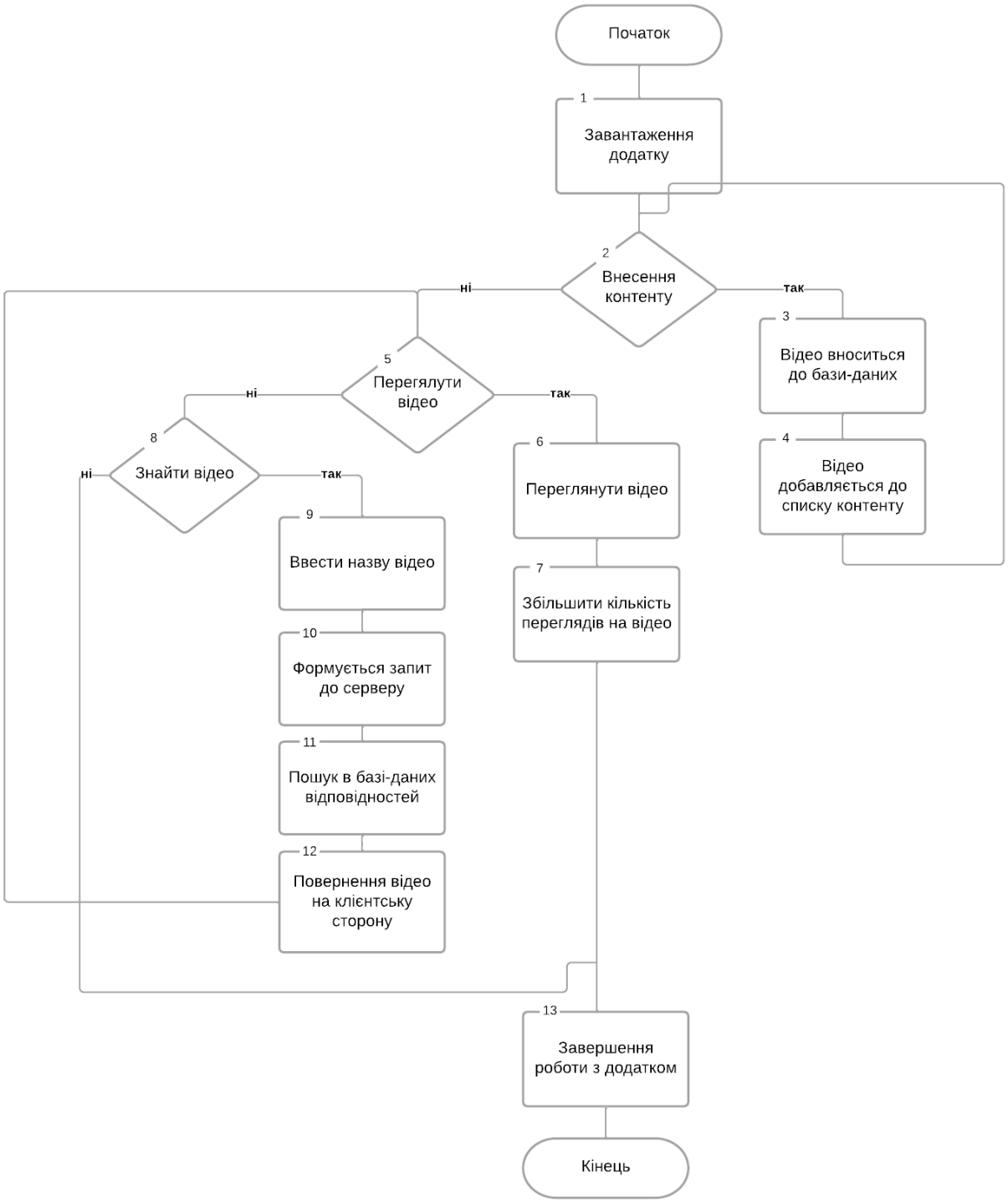

Рис. В.2 – Принципова схема алгоритму роботи програмного модуля сервісу відеохостингу

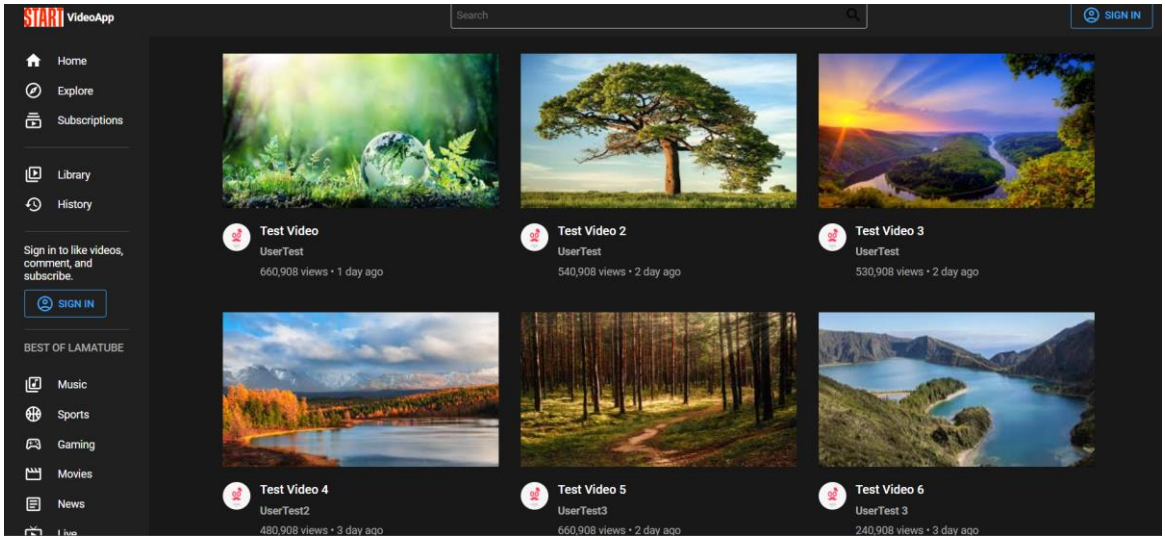

Рис. В.3 – Загальний вигляд головної сторінки програного модуля сервісу відеохостингу

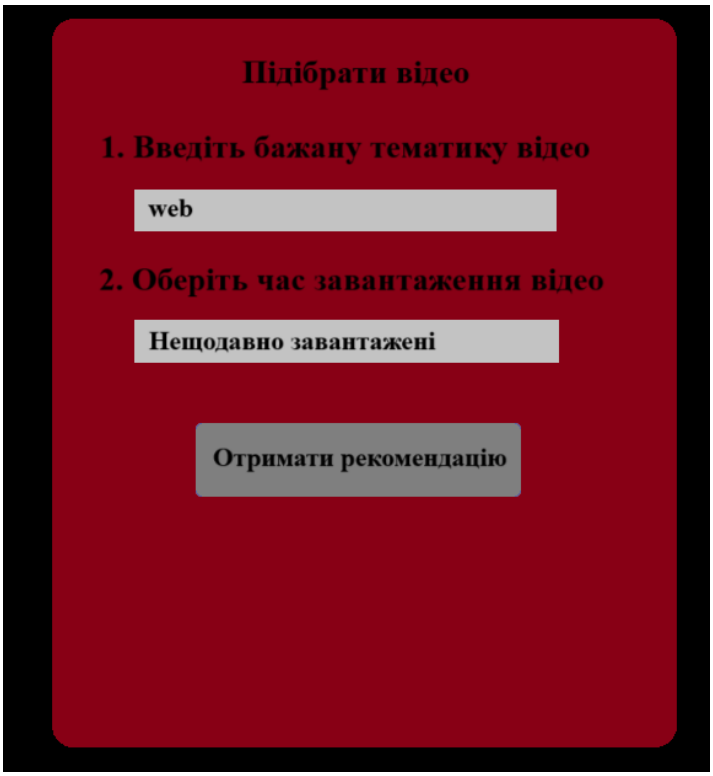

Рис. В.4 – Загальний вигляд прикладу форми надання рекомендацій у інформаційній технології сервісу організації відеохостингу

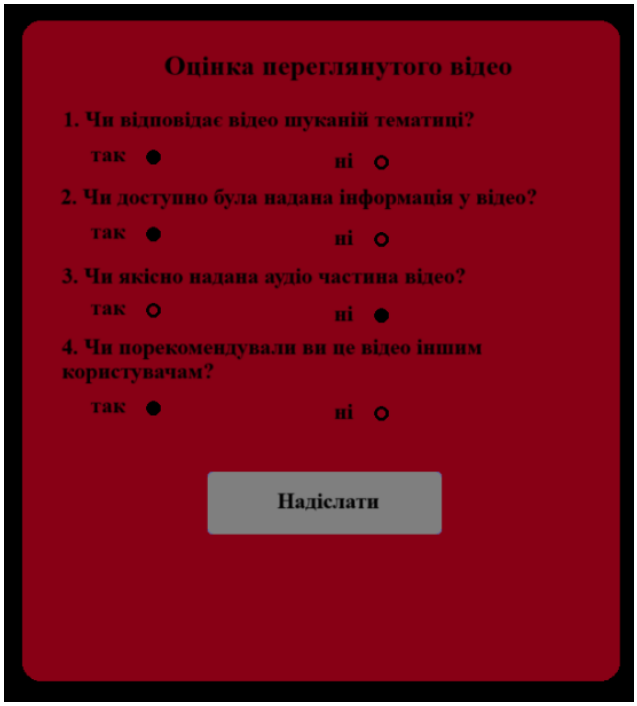

Рис. В.5 – Загальний вигляд прикладу форми оцінки рекомендацій у інформаційній технології сервісу організації відеохостингу

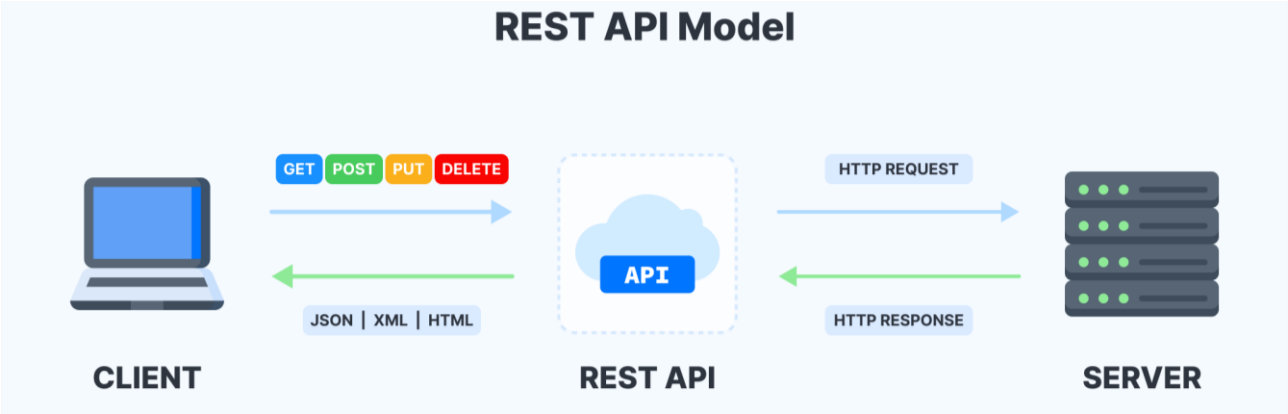

Рисунок В.6 – Принципова схема взаємодії клієнт-серверної частини з RESTful API

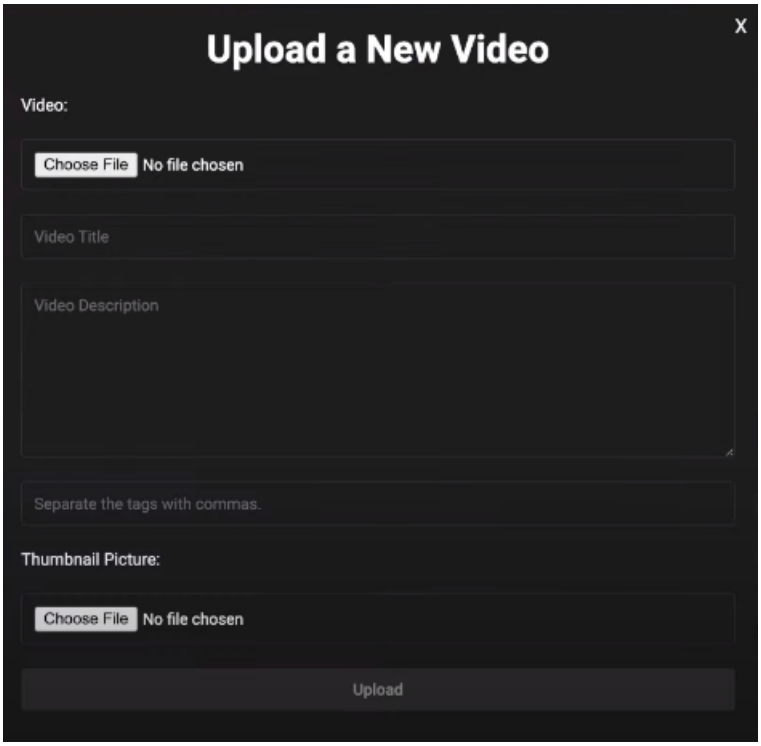

Рисунок В.7 – Загальний вигляд форми завантаження нового відео до інформаційної технології організації відеохостингу

Додаток Г (довідниковий)

Інструкція користувача

Запуск React-проекту відбувається за допомогою введення команди «npm start», яка дозволяє ініціювати локальний сервер для перегляду та випробування додатку. Отримане повідомлення в терміналі містить інформацію про стан запуску, а також URL-адресу, за якою можна перейти для перегляду проекту у веб-браузері. Повідомлення зображене на рисунку Г.1.

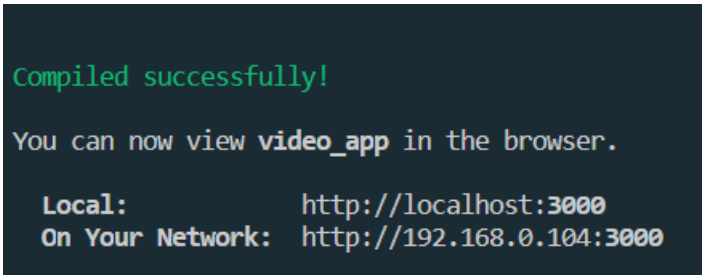

Рисунок Г.1 – Загальний вигляд повідомлення про початок процесу завантаження проекту у локальній мережі

Через 2-3 хвилини, проект повністю завантажиться, а веб-додаток автоматично відкриється за посиланням у локальній мережі, як показано на рисунку Г.2. Після завантаження користувач автоматично буде перенаправлений на головну сторінку додатку.

На цій головній сторінці доступна вся необхідна функціональність та інтерфейс додатку. Користувач має можливість взаємодіяти з різними елементами, виконувати необхідні завдання та оцінювати його функціональні можливості. Цей етап важливий для виявлення можливих проблем та підтвердження коректності роботи веб-додатку в локальному середовищі перед його подальшим впровадженням або тестуванням на інших середовищах.

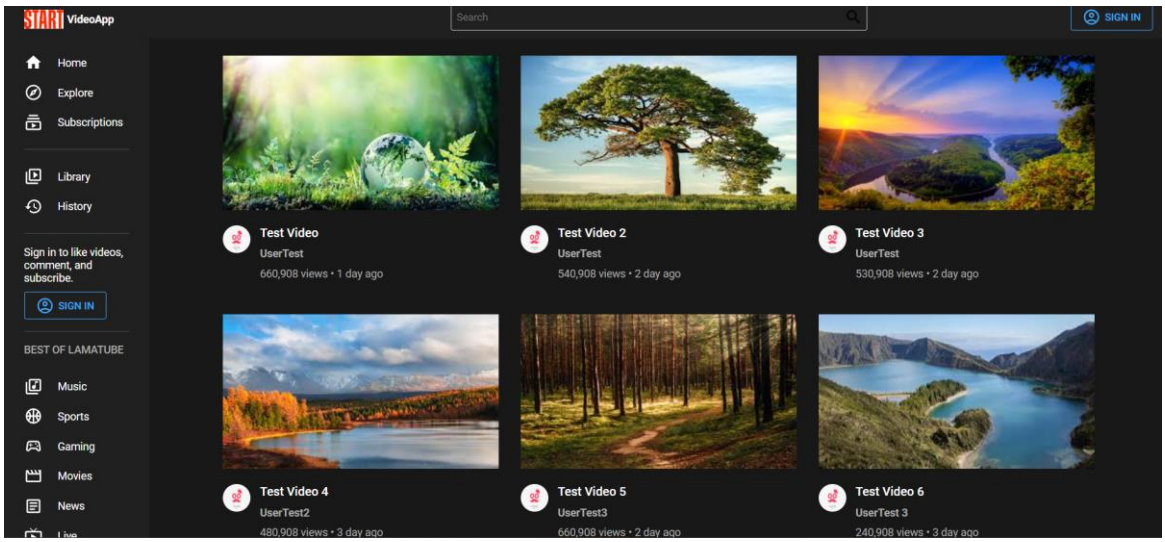

Рисунок Г.2 – Загальний вигляд головної сторінки додатку

Наступним кроком необхідно перейти до форми реєстрації користувача в системі (рис Г.3).

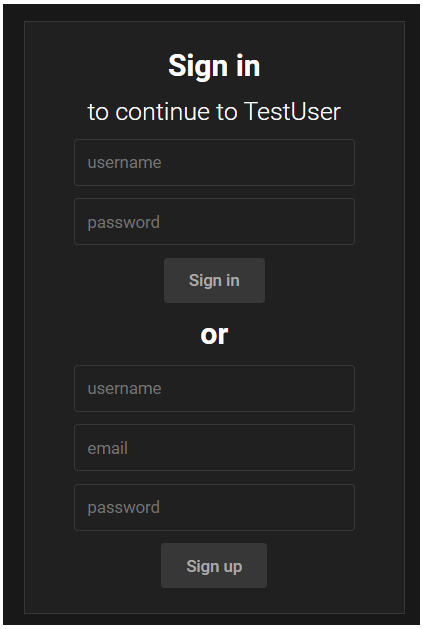

Рисунок Г.3 – Загальний вигляд сторінки авторизації інформаційної технології організації відеохостингу

Далі необхідно зареєструвати користувача до системи за допомогою спеціальної форми для реєстрації або ввести дані для входу якщо користувач вже зареєстрований (рис Г.4).

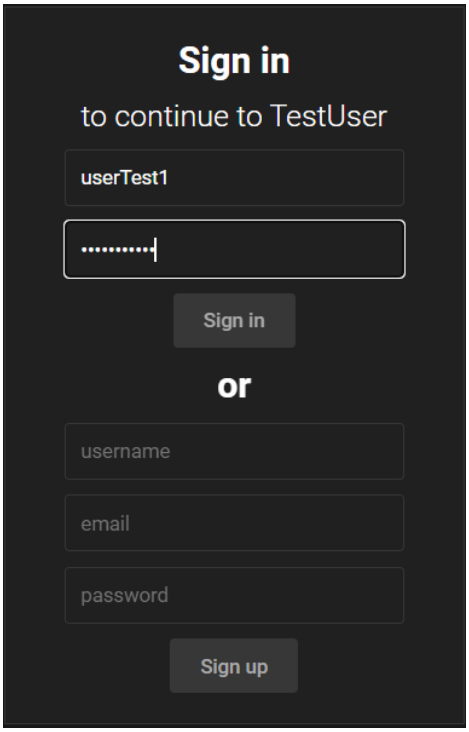

Рисунок Г.4 – Загальний вигляд заповненої форми реєстрації користувача інформаційної технології організації відеохостингу

Після реєстрації відбудеться перенаправлення на головну сторінку, наступним кроком користувач може перейти до завантаження власного відео до системи відеохостингу, завантаження відбувається за допомогою спеціальної форми де необхідно заповнити всі поля, форма додання відео буде виглядати як зображено на рисунку Г.5.

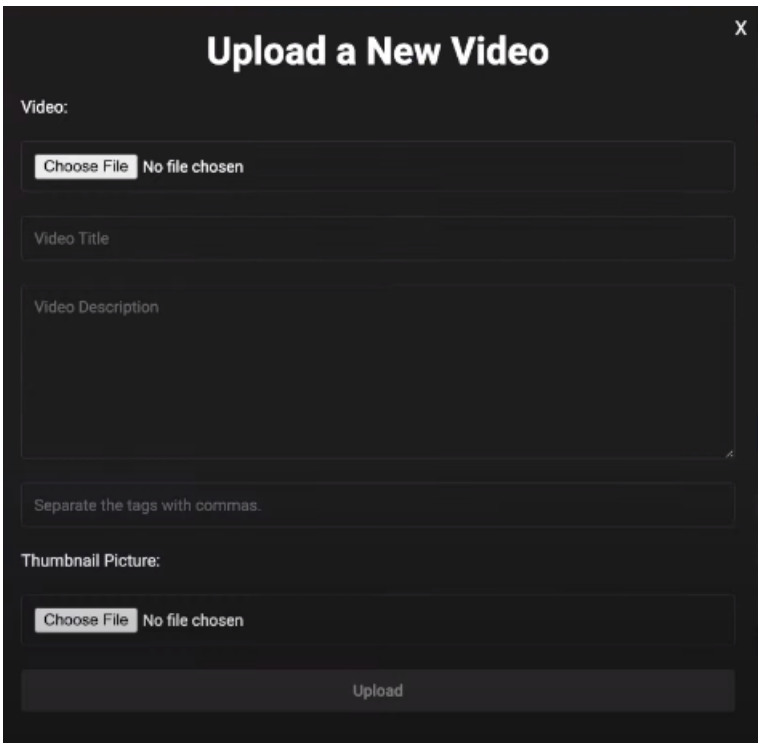

Рисунок Г.5 – Загальний вигляд форми завантаження нового відео до інформаційної технології організації відеохостингу

Для додання бажаного відео до бази даних відеохостингу, потрібно заповнити форму на сторінці «Upload a new Video» (рис. Г.6). Поле «Video» відповідає за відкриття необхідного відео користувача, яке обмежене обсягом не більше 20Гб. Поле «Video Title» відповідає за назву відео, під введеною назвою даний тест буде збережено до бази даних. «Video Description» – відповідає за опис відео для завантаження користувачем до системи. «Separate Tag» – вид міток які використовуються при подальшій рекомендації відео користувачам. «Thumbnail Picture» – відповідає за завантаження невеликої картинки обсягом до 15Мб яка використовується для попереднього позначення відео.

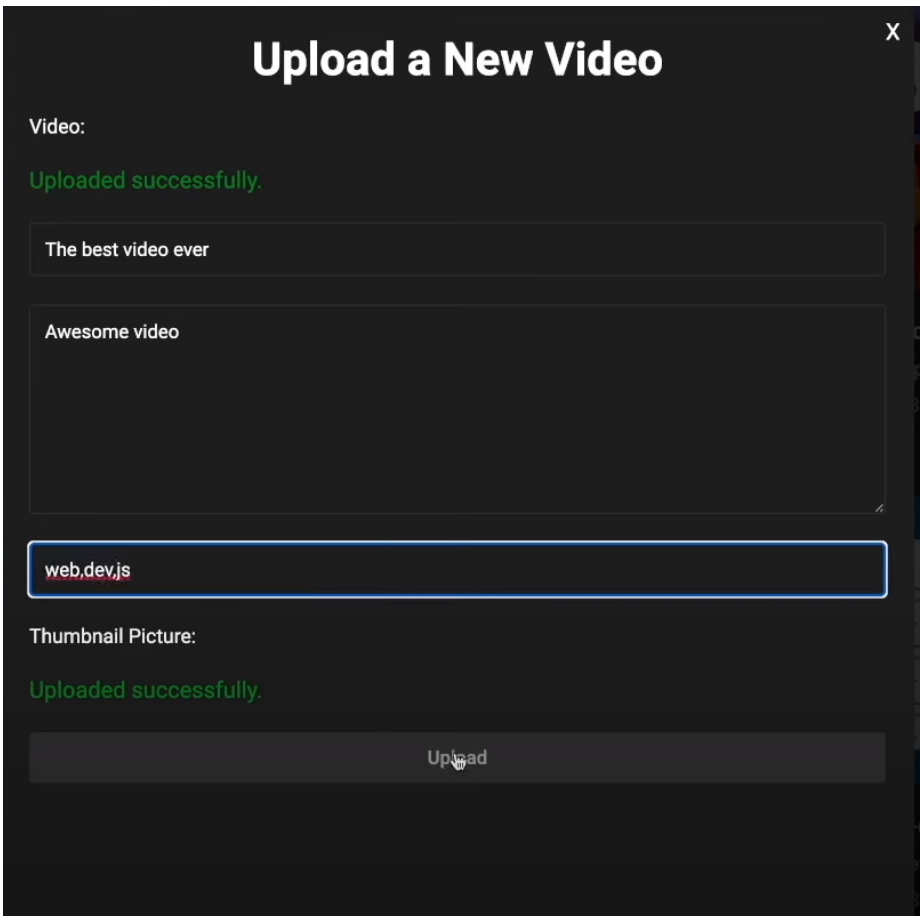

Рисунок Г.6 – Загальний вигляд заповненої форми завантаження нового відео до інформаційної технології організації відеохостингу

Наступним кроком для користувача є перехід на сторінку для отримання рекомендації, для цього потрібно ввести в поле бажану тематику для перегляду відео та обрати час завантаження відео, приклад заповненої форми для отримання рекомендації наведено на рисунку Г.7.

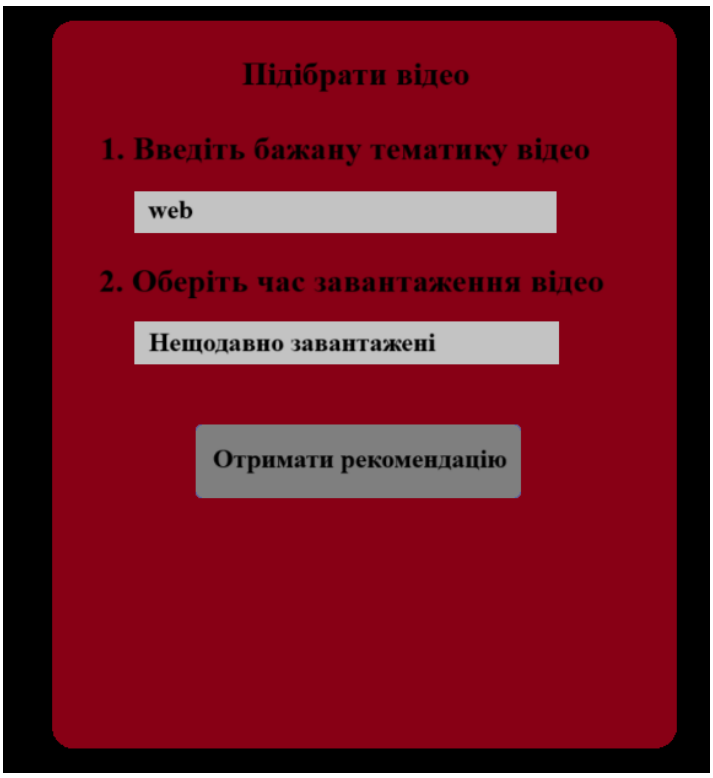

Рисунок Г.7 – Загальний вигляд заповненої форми для отримання рекомендації для перегляду відео.

Результат отримання рекомендацій щодо перегляду відео наведено на рисунку Г.8.

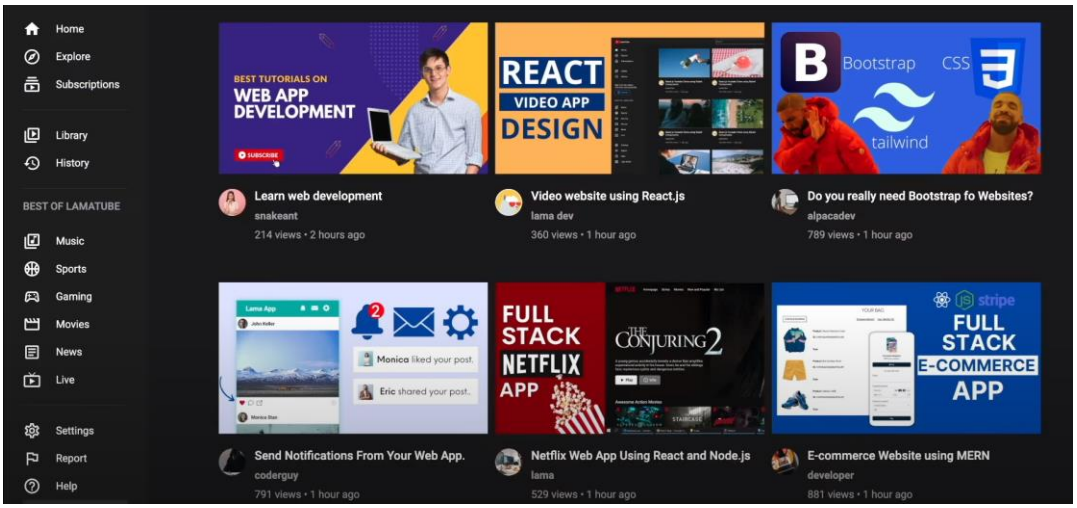

Рисунок Г.8 – Загальний вигляд результату отримання рекомендацій інформаційної технології організації відеохостингу