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 стримінгового сервісу для фільмів»

Виконала: студентка 2-го курсу, групи 2КН-23м
спеціальності 122 – «Комп'ютерні науки»
(шифр і назва напряму підготовки, спеціальності)
Мельник К.І..
(прізвище та ініціали)

Керівник: к.т.н., доцент каф. КН
Озеранський В. С.
(прізвище та ініціали)
«05» 12 2024 р.

Опонент: к.т.н., доцент каф. САІТ
Варчук І. В.
(прізвище та ініціали)
«05» 12 2024 р.

Допущено до захисту
Завідувач кафедри КН д.т.н.,
проф. Яровий А. А.
«06» 12 2024 р.

Вінниця ВНТУ – 2024 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.
_____
(підпис)
"11" 10 2024 року

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТЦІ
Мельник Катерині Ігорівні

1 Тема роботи: «Інформаційна технологія стримінгового сервісу для фільмів».
Керівник роботи: Озеранський Володимир Сергійович, к.т.н., доц.
затверджені наказом вищого навчального закладу «17» 09 2024 року №310

2 Строк подання студентом роботи 11.11.

3 Вихідні дані до роботи: кількість користувачів – до 100чол; роздільна здатність файлів – до 720р; інтернет підключення – від 500 kb/s; мова програмування – об'єктно-орієнтована.

4 Зміст пояснювальної записки (перелік питань, які потрібно розробити): вступ; аналіз сучасних технологій організації стримінгового сервісу для фільмів; моделювання інформаційної технології стримінгового сервісу для фільмів; програмна реалізація технології стримінгового сервісу для фільмів; тестування та аналіз результатів роботи розробленої програми; висновки, перелік використаних джерел; додатки.

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень): схема алгоритму функціонування інформаційної технології стримінгового сервісу для фільмів; вигляд інтерфейсу користувача; вікно введення основних даних для стримінгового сервісу для фільмів; приклади роботи програм

6 Консультанти розділів проекту (роботи)

Консультанти розділів роботи в таблиці 1.

Таблиця 1 - Консультанти роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-3	Озеранський В. С., к.т.н., доц. каф. КН		
4	Адлер О.О., к.т.н., доц. каф. ЕПВМ		

7 Дата видачі завдання 02.09.2024

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз сучасних технологій організації стримінгового сервісу для фільмів	02.09.24 - 16.09.24	Вик.
2	Моделювання інформаційної технології стримінгового сервісу для фільмів	17.09.24 - 02.10.24	Вик.
3	Програмна реалізація технології стримінгового сервісу для фільмів	03.10.24 - 16.10.24	Вик.
4	Підготовка економічної частини	17.10.24 - 28.10.24	Вик.
5	Апробація результатів дослідження	29.10.24 - 03.11.24	Вик.
6	Оформлення матеріалів до захисту МКР	04.11.24 - 11.11.24	Вик.

Студентка _____ Мельник К.І.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Озеранський В.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

УДК 621.374.415

Мельник К.І. Інформаційна технологія стримінгового сервісу для фільмів. Магістерська кваліфікаційна робота зі спеціальності 122 – комп'ютерні науки, освітня програма - Системи штучного інтелекту. Вінниця: ВНТУ, 2024. 104с.

На укр. мові. Бібліогр.: 25 назв; рис.: 23; табл. 10.

У даній магістерській кваліфікаційній розроблено стримінговий сервіс для фільмів з використанням YouTube Api, Angular framework та Mongo DB. Даний сервіс знаходиться у відкритому доступі, є простим в користуванні, модульним та дозволяє здійснювати допрацювання та підтримує можливість масштабування. Сервіс використовує наступні технології та компоненти: NgRx, який застосовується для забезпечення передбачуваного стану контейнера, Angular компоненти для констроювання системи з окремих модулів, Mongo DB для збереження даних та контенту, YouTube Api для відтворення контенту, HTML, CSS, JavaScript, TypeScript для юзер інтерфейсу.

Ключові слова: Angular framework, YouTube, API, Mongo DB, Express.js, HTML, CSS, JavaScript, TypeScript, SCSS, BCrypt

ABSTRACT

Melnyk K.I. Information technology of streaming service for films. Master's qualification work in specialty 122 - computer science, educational program - Artificial intelligence systems. Vinnytsia: VNTU, 2024. 104p.

In Ukrainian. Bibliography: 25 titles; fig.: 23; tab. 10.

In this master's qualification, a streaming service for movies was developed using YouTube Api, Angular framework and Mongo DB. This service is open access, easy to use, modular and allows for refinement and supports the possibility of scaling. The service uses the following technologies and components: NgRx, which is used to ensure the predictable state of the container, Angular components for building a system from separate modules, Mongo DB for saving data and content, YouTube Api for playing content, HTML, CSS, JavaScript, TypeScript for the user interface .

Keywords: Angular framework, YouTube, API, Mongo DB, Express.js, HTML, CSS, JavaScript, TypeScript, SCSS, BCrypt

ЗМІСТ

ВСТУП

Актуальність теми. Щорічно запускаються нові стримінгові платформи, обсяг контенту на них зростає щомісяця, а кількість передплатників зростає з кожним днем. Цей процес змінює звички людей щодо перегляду фільмів і телевізійних програм. Наприклад, такі компанії, як Apple та Disney вступили на ринок стримінгових послуг починаючи з 2020 року. Найбільші світові корпорації приєдналися до конкуренції з такими гравцями, як Netflix і Amazon, а також Hulu, Facebook і YouTube. Крім того, CBS запускає свій стримінговий сервіс, а на початку наступного року до такої гонки приєднаються американські компанії HBO і NBC.

Сучасний цифровий світ переживає справжню революцію в способах доступу до розваг і розважального контенту. Однією з ключових характеристик цієї епохи є зростання популярності абонентської моделі, яка дає споживачам можливість отримувати необмежений доступ до різноманітного контенту за фіксовану щомісячну або щорічну плату.

Стримінгові сервіси, які надають доступ до великої кількості відео-, аудіо- і текстового контенту через інтернет, стали важливою частиною цього ландшафту. Вони дозволяють глядачам і слухачам переглядати фільми, серіали, веб-шоу, слухати музику, аудіокниги, та навіть читати книги і газети на своїх смартфонах, планшетах та комп'ютерах у будь-який зручний для них час. Це стало можливим завдяки швидкому розвитку інтернет-з'єднань і високоякісному відео- та аудіокодуванню. Запуск нових стримінгових платформ і послуг стає стандартом на цьому ринку. Кожна з них намагається привернути аудиторію за допомогою унікального контенту і функцій. Великі глобальні компанії, такі як Apple, Disney, Netflix, Amazon, HBO, і інші, вкладають великі кошти у розробку оригінальних фільмів і серіалів, щоб здобути конкурентну перевагу на цьому ринку [1].

У світі, який швидко переходить до цифрових форматів споживання контенту, дослідження таких тем, як стримінгові сервіси і їх вплив на звички споживачів, є надзвичайно актуальними. Вони допомагають розуміти, як споживачі вибирають і

споживають контент, як це впливає на індустрію розваг і медіа, а також які можливості і виклики виникають для бізнесу і технологій в цьому сфері.

Сьогодні все більше людей переходять до моделі передплати в різних аспектах життя. Вони беруть автомобілі "на підписку", використовують банківські послуги "по підписці", слухають музику, дивляться фільми і багато іншого. Підписка передбачає фіксовану оплату за певний період користування конкретною послугою. Об'єктом цього дослідження є стримінгові сервіси, тобто цифрові платформи, які дозволяють переглядати фільми і серіали з їхньої бібліотеки за підпискою.

Тому актуальним є розробка інформаційної технології стримінгового сервісу для фільмів, що дозволить ефективно надавати рекомендації підбору відповідного контенту відеопродукції для користувача у стримінгових відеосервісах.

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 дослідження є розширення функціональних можливостей програмного забезпечення стримінгового сервісу для перегляду фільмів.

Для досягнення поставленої мети потрібно виконати наступні **завдання**:

- здійснити аналіз сучасних технологій для організації стримінгового сервісу для фільмів;
- здійснити порівняльний аналіз програм аналогів для організації стримінгового сервісу для фільмів;
- здійснити моделювання інформаційної технології стримінгового сервісу для фільмів;
- розробити алгоритм функціонування інформаційної технології стримінгового сервісу для фільмів;
- здійснити програмну реалізацію запропонованої інформаційної технології;

- провести тестування програмного засобу та проаналізувати отримані результати.
- економічно обґрунтувати доцільність розробки інформаційної технології.

Об’єктом дослідження є процес організації стримінгового сервісу для фільмів.

Предметом дослідження є програмні засоби організації стримінгового сервісу для фільмів.

Методи дослідження. У роботі використано такі методи наукових досліджень: метод системного аналізу для аналізу структури інформаційної системи; методи математичної статистики для розробки технології надання рекомендацій.

Наукова новизна одержаних результатів: удосконалено інформаційну технологію стримінгового сервісу для фільмів, що відрізняється від існуючих наданням рекомендацій вибору відеоконтенту на основі фільтрації даних користувача, що дозволило розширити функціональні можливості програмного забезпечення стримінгового сервісу для фільмів.

Практичне значення одержаних результатів полягає в тому, що на основі проведених досліджень розроблено програмне забезпечення стримінгового сервісу для фільмів.

Запропонована інформаційна технологія сприяє підвищенню ефективності процесу надання рекомендацій для вибору фільмів в стримінгових сервісах:

- розроблено алгоритм функціонування інформаційної технології стримінгового сервісу для фільмів;
- розроблено програмне забезпечення стримінгового сервісу для фільмів.

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

Особистий внесок здобувача. Усі результати, що наведені у магістерській кваліфікаційній роботі, отримані самостійно. У працях, які написано у співавторстві, здобувачу належать: особливості організації рекомендаційної системи для стримінгового сервісу перегляду фільмів.

Апробація результатів роботи. Результати роботи були апробовані на міжнародній науково-практичній інтернет-конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2025)» (м. Вінниця, Україна, 2025 р.) [1].

Публікації. За результатами досліджень опубліковано тези доповіді на науково-практичній конференції [1].

1 АНАЛІЗ СУЧАСНИХ ТЕХНОЛОГІЙ ОРГАНІЗАЦІЇ СТРИМІНГОВОГО СЕРВІСУ ДЛЯ ФІЛЬМІВ

1.1 Обґрунтування доцільності створення стримінгового сервісу для перегляду фільмів

Потокове відтворення відео – це спосіб передачі медіафайлів, таких як відео, без потреби завантажувати їх із Інтернету. Сервіси, як от Netflix, Hulu та Megogo, використовують цей метод для надання можливості переглядати відео на різних типах пристроїв.

Потокове відтворення відео стало важливою частиною сучасного споживчого досвіду. Воно дозволяє глядачам зручно переглядати вміст в режимі реального часу, без необхідності очікування завантаження великих файлів на пристрій. Це особливо корисно в епоху швидкого інтернету та розширеної мобільності.

Процес потокового відтворення включає в себе декілька кроків. Спочатку програмний медіаплеєр на пристрої користувача встановлює з'єднання з сервером потокового відтворення. Після отримання сигналу сервер починає передавати медіапотоки, а пристрій зберігає невеликий буфер даних, щоб забезпечити безперервне відтворення.

Один з головних переваг потокового відтворення полягає в тому, що воно дозволяє глядачам споживати контент на різних пристроях, включаючи смартфони, планшети, телевізори та комп'ютери, і переривається не втратами якості відео чи аудіо. Ця універсальність робить потокове відтворення вельми популярним серед споживачів розважального контенту. Зростаюча конкуренція в галузі потокових сервісів стимулює розвиток нових технологій, які покращують якість відтворення, а також пропонують унікальний контент для привертання аудиторії. Ця галузь залишається однією з найбільш динамічних і швидкозростаючих у сфері медіа і розваг.

Ключовим аспектом технології потокового медіа є буферизація даних, які можна відтворити. Програмний відеопрогравач на вашому комп'ютері з'єднується з сервером і надсилає запит на потік. Сервер починає передавати медіапотоки і направляє їх до програвача. Потокові медіапрогравачі завантажують дані на кілька секунд вперед, щоб забезпечити неперервне відтворення відео або аудіо, навіть якщо з’єднання тимчасово втрачено. Цей процес відомий як буферизація (рис. 1.1).

Рисунок 1.1 – Загальний вигляд принципу буферизації

Буферизація дозволяє забезпечити плавне та безперервне відтворення відео, навіть у випадку невеликих затримок в потоці даних, спричинених перебоями у мережі. Зі збільшенням розміру буфера зменшується вплив перебоїв в мережі на якість трансляції [4].

Збільшення розміру буферу може бути корисним у випадках, коли ви працюєте з аудіо, відео або мережевими даними і хочете покращити ефективність чи якість відтворення або передачі даних. Розмір буферу визначає кількість даних, які можуть бути зчитані або записані перед тим, як вони будуть оброблені або відтворені.

Важливо пам'ятати, що збільшення розміру буфера може зробити вашу програму більш ефективною, але також може призвести до збільшеного використання ресурсів. Тому перед зміною розміру буфера слід ретельно обдумати, як це вплине на ваше програмне середовище і завдання.

Переваги потокового медіаконтенту включають в себе:

– миттєве відтворення – вміст починає відтворюватися майже миттєво, незалежно від розміру файлу зображення або звуку. Користувачам не потрібно чекати завантаження всього контенту;

– економія місця – не потрібно великої кількості місця на жорсткому диску для зберігання медіафайлів. Весь контент доступний в режимі онлайн, що робить займання місця незначним;

– вибір контенту – користувачі можуть вибирати, що переглядати або слухати, і не обмежуватися розкладом телепередач або радіопередач;

– мультиплатформеність – ви можете переглядати медіаконтент на різних пристроях, таких як смартфони, комп'ютери, планшети тощо;

– зменшення ризику піратства – оскільки контент не завантажується на пристрій користувача, ризик незаконного копіювання обмежений.

Проте, є деякі недоліки потокового медіаконтенту:

– потреба у якісному інтернет-з'єднанні – для користування послугами потокової передачі необхідне стабільне підключення до Інтернету через Wi-Fi або мобільні дані;

– режим реального часу – потоковий контент передається в режимі реального часу, тому вам потрібно мати доступ до Інтернету кожного разу, коли ви хочете подивитися чи послухати щось. У порівнянні з цим завантаження медіафайлу один раз дозволяє відтворювати його в будь-який момент;

– потреба у якісному з’єднанні з Інтернетом – швидкість з'єднання повинна бути достатньою для безперервного відтворення контенту.

Що стосується причин, які можуть призвести до зменшення швидкості передачі потокового контенту, то вони можуть бути зв’язані з:

– важливістю вирішення проблеми з Wi-Fi, включаючи можливість перевантаження маршрутизатора;

– необхідністю достатньої пропускної здатності для потокової передачі, зазвичай близько 5 Мбіт/с, залежно від якості контенту;

– ефективним зберіганням контенту, яке також впливає на швидкість передачі;

– якістю відтворенням, яка може бути обмежена повільною обчислювальною потужністю пристроїв, які використовуються для відтворення контенту [5].

Однією з найбільш важливих технологій, якій використовуються для стримінгових сервісів для перегляду фільмів є Digital Rights Management [6].

Управління цифровими правами (DRM) — це набір технологій і методів, які використовуються для захисту цифрового вмісту, наприклад аудіо, відео, електронних книг, програмного забезпечення тощо, від несанкціонованого доступу, копіювання та розповсюдження. Системи DRM призначені для забезпечення обмежень щодо авторських прав і контролю за використанням цифрових медіа, гарантуючи, що лише авторизовані користувачі можуть отримати доступ до вмісту та використовувати його за призначенням власників прав.

Ключові аспекти системи DRM:

− системи DRM контролюють, хто може отримати доступ до цифрового вмісту. Авторизованим користувачам надається доступ, а неавторизованим – заборонено. Зазвичай це робиться за допомогою автентифікації користувача, наприклад імені користувача та пароля або цифрових сертифікатів;

− використання методів шифрування для захисту самого вмісту. Контент зашифрований і може бути розшифрований лише авторизованими пристроями або програмним забезпеченням із відповідними ключами розшифровки. Це запобігає несанкціонованому копіюванню або зміні вмісту;

− системи DRM видають користувачам ліцензії із зазначенням умов, за яких можна використовувати вміст. Ці ліцензії можуть містити обмеження щодо кількості пристроїв, на яких можна отримати доступ до вмісту, термінів дії та можливості робити копії;

− DRM може обмежувати або контролювати можливість створення копій цифрового вмісту. Наприклад, він може обмежити кількість копіювання файлу або взагалі заборонити копіювання. Деякі системи DRM дозволяють користувачам отримувати доступ до вмісту в автономному режимі, але це часто залежить від умов, визначених у ліцензії. Офлайн-доступ зазвичай потребує періодичної онлайн-перевірки, щоб переконатися, що вміст усе ще авторизований;

− DRM може обмежувати пристрої або платформи, на яких можна отримати доступ до вмісту. Наприклад, вміст може бути обмежено певними пристроями для читання електронних книг, медіаплеєрами чи операційними системами;

− системи DRM часто включають функції моніторингу та звітності, які відстежують використання вмісту. Це допомагає правовласникам збирати дані про поведінку користувачів і забезпечити виконання умов ліцензії;

− DRM була темою суперечок. Критики стверджують, що це може обмежити права користувачів і чесне використання, тоді як прихильники стверджують, що важливо захистити інтелектуальну власність творців і видавців;

− системи DRM можуть сильно відрізнятися, і це може створювати проблеми сумісності. Наприклад, вміст, придбаний через одну систему DRM, може бути несумісним із пристроями чи програмним забезпеченням іншого постачальника;

− технологія DRM продовжує розвиватися. Новіші системи DRM можуть містити такі функції, як водяні знаки, адаптивна безпека та більша гнучкість умов ліцензування, щоб усунути деякі критичні зауваження та проблеми, пов’язані з традиційним DRM.

DRM використовується в різних галузях, включаючи розваги, видавництво, програмне забезпечення та ігри. Хоча він призначений для захисту творців контенту та видавців від піратства та несанкціонованого розповсюдження, він залишається предметом дискусій і може вплинути на досвід користувачів і цифрові права.

1.2 Аналіз відомих програм для організації стримінгових сервісів для перегляду фільмів

Існує дуже багато рішень в галузі стримінгових сервісів для перегляду фільмів фаворитами з яких є:

− Netflix є одним з найпопулярніших стримінгових сервісів у світі. Він пропонує велику кількість фільмів та серіалів, включаючи оригінальний контент, і доступний на різних пристроях [7];

− Amazon Prime Video є частиною Amazon Prime-підписки і пропонує широкий вибір фільмів та телевізійних передач, а також оригінальний контент від Amazon Studios;

− Disney+ це стримінговий сервіс від The Walt Disney Company, який має в своєму арсеналі фільми та серіали від Disney, Marvel, Star Wars і Pixar;

− Hulu спеціалізується на стрімінгу телевізійних шоу і передач, а також пропонує оригінальний контент [8];

− HBO Max дає доступ до всього контенту HBO, включаючи відомі шоу та фільми, а також оригінальний контент;

− Apple TV+ - це стримінгова платформа від Apple, яка пропонує оригінальний контент від компанії Apple;

− Peacock - це сервіс від NBCUniversal, який має різноманітний контент, включаючи класичні шоу і фільми, а також оригінальний контент;

− Paramount+ від Paramount Pictures має великий вибір фільмів і серіалів, а також оригінальний контент;

− YouTube Premium - це платна версія YouTube, яка пропонує рекламування і доступ до оригінального контенту;

− Tubi - це безкоштовна платформа для стрімінгу, яка пропонує широкий вибір фільмів і телевізійних передач з рекламою;

− Vudu - це платформа для оренди та покупки фільмів, але вона також має безкоштовний розділ з обмеженим вмістом;

− Crave - це канадська платформа стрімінгу, яка пропонує контент від HBO, Showtime, Starz та інших провайдерів.

Це лише кілька прикладів стримінгових сервісів для фільмів, і ринок продовжує розвиватися з появою нових платформ і оригінального контенту. Вибір сервісу може залежати від вашого географічного розташування та особистих вподобань у контенті.

На даний момент існує достатня кількість стримінгових сервісів і кожна людина обирає той, який їй подобається більше всього. Кожен з них намагається привернути увагу глядача всім, чим може, будь то унікальність, зручність, доступність, яскравий і приємний оку інтерфейс чи найбільше різноманіття контенту. Проаналізуємо та детальніше розглянемо декілька таких стримінгових сервісів для фільмів, аби визначитися з основними характеристиками, які будуть

впливати на моделювання нашого продукту, та виділити основні недоліки, яких треба уникнути під час вищевказаного етапу розробки.

Коли думаєш про перегляд фільмів чи серіалів, то першим на думку спадає саме Netflix. То є популярний у світі американський відеосервіс, що дає людям змогу за певну оплату отримати доступ до величезної кількості контенту різними мовами.

Ергономічний інтерфейс, що продемонстрований на рисунку 1.2, та можливість побачити цілий сезон улюбленого серіалу відразу, а не по одній серії – головна відмінна риса.

Підтримується синхронізація перегляду на усіх підключених пристроях, тобто, є можливість, наприклад, переглянути якусь частину на телевізорі, а потім продовжити перегляд на телефоні чи планшеті або навпаки. Netflix запустив власне виробництво контенту у 2013 році.

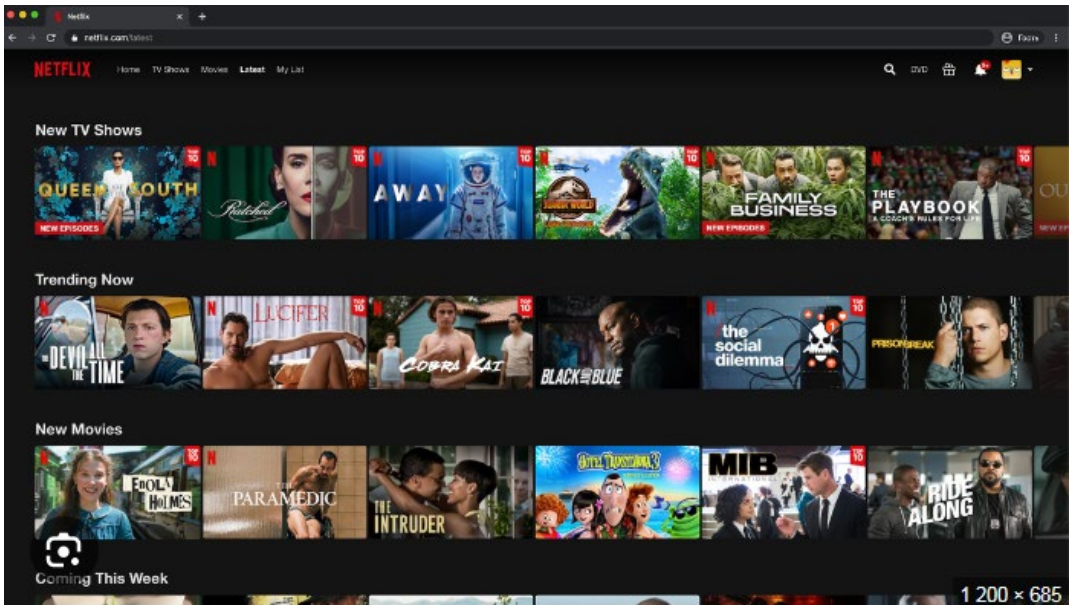

Рисунок 1.2 – Приклад інтерфейсу стримінгового сервісу «Netflix»

Використовується технологія адаптивного стримінгу, тобто сервіс підлаштовує якість медіафайлу відповідно до якості інтернету, але можна й встановити якість самостійно. Стримінговий сервіс для фільмів та серіалів Netflix використовує DASH (Dynamic Streaming over HTTP) протокол потокової передачі даних.

Наявність наступних недоліків:

– місячна плата (незважаючи на те, що Netflix пропонує різні тарифні плани, це все ще є платним сервісом, і вам потрібно буде щомісяця платити за доступ);

– обмежений новий контент (хоча Netflix виробляє багато оригінального контенту, деякі фільми і серіали можуть бути доступні на інших платформах або в кінотеатрах, перш ніж вони з'являться на Netflix);

– регіональні обмеження (деякий контент може бути доступним тільки в певних країнах через ліцензійні обмеження).

– відсутність реклами (це може бути перевагою для деяких, але також означає відсутність безкоштовного варіанта з рекламою, на відміну від деяких інших безкоштовних стримінгових платформ);

– залежність від мережі Інтернет (для користування Netflix вам потрібен стабільний і швидкий доступ до Інтернету).

Наступний із аналогів розглянемо стримінговий сервіс Hulu (рис. 1.3). Він заснований у 2007 році двома медіахолдингами: NBCUniversal Media та News Corporation. Що ж є цікавим це те, що для доставки контенту до користувачів, то використовується три CDN, а протягом показу всього відео застосовується лише один з них, а також те, що для під час транслювання іншого відео CDN-сервер перемикається. Hulu застосовує зашифрований протокол передачі даних – RTMP (Real Time Messaging Protocol) або ж не зашифрований, тунельований через HTTP – RTMPT (Real Time Messaging Protocol Tunnel).

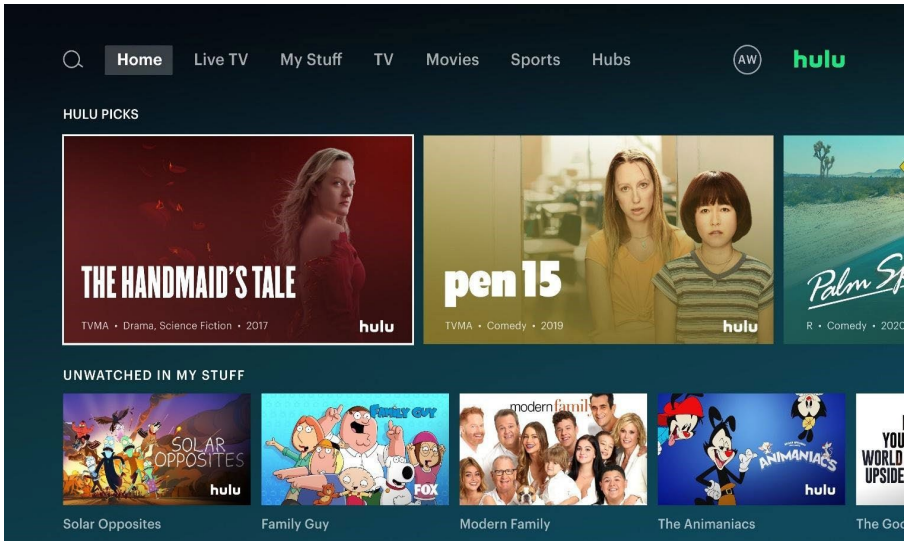

Рисунок 1.3 – Приклад інтерфейсу стримінгового сервісу «Hulu»

Окрім іноземних стримінгових сервісів є і вітчизняний – MEGOGO [9] (рис. 1.4). Він співпрацює з такими студіями, як Paramount Pictures, FOX, Warner Bros., Walt Disney та BBC.

MEGOGO був заснований у 2011 році як сервіс потокового відео. З часом платформа розширила свої можливості, включивши прямі трансляції, інтерактивні функції та аудіоформат. Сьогодні MEGOGO обслуговує мільйони користувачів і є одним із лідерів у сфері легального цифрового контенту на території Східної Європи.

Ця платформа пропонує користувачам широкий спектр розважального та освітнього контенту, включаючи фільми, серіали, телепередачі, спортивні трансляції, аудіокниги та подкасти. MEGOGO надає доступ до величезної бібліотеки фільмів і серіалів, включаючи голлівудські прем'єри, класичне кіно, європейське та азіатське кіно. Багато матеріалів доступні у високій якості, з оригінальним звуком або дубляжем, а також субтитрами.

Відмінними рисами є різноманітний контент. Серед нього є такі екземпляри, як мультфільми, фільми, серіали, аудіокниги, подкасти, інтерактивні канали, спортивний та ігровий контент. Насправді найбільшим здивуванням серед усього вищеперерахованого особисто для мене є ігровий сегмент.

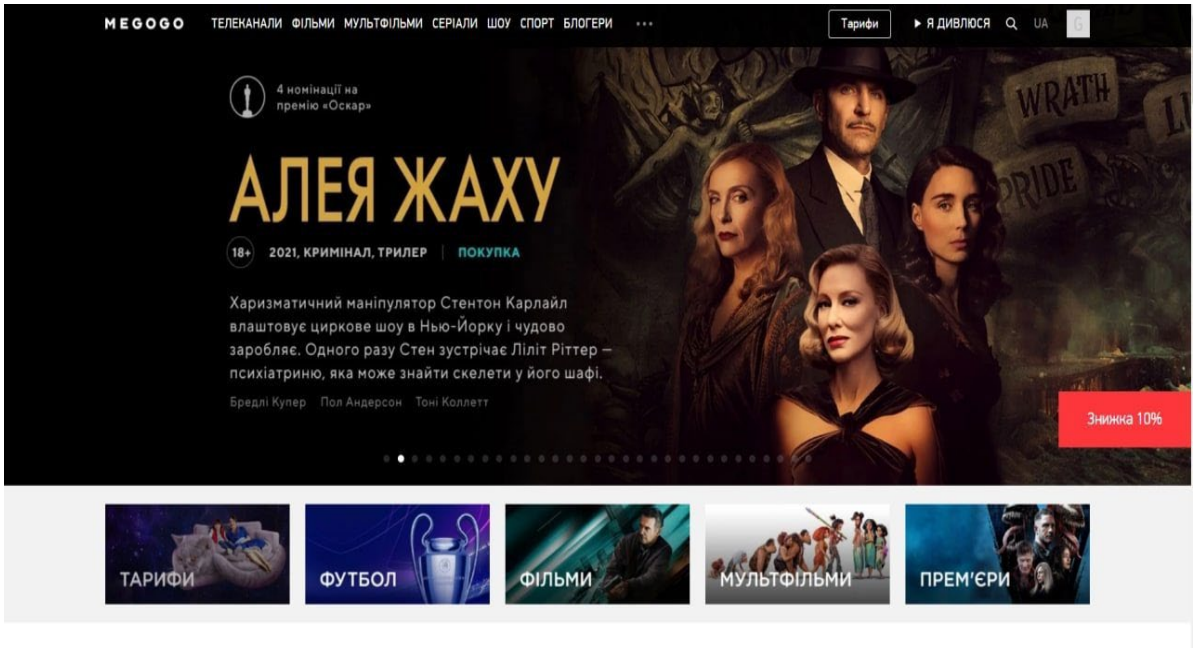

Рисунок 1.4 – Приклад інтерфейсу стримінгового сервісу «MEGOGO»

У наш час комп’ютерні ігри займають величезну нішу у житті молодих людей. З кожним роком розвиток цієї сфери досягає неймовірно великих висот, що дає велику кількість нових прихильників, а, отже, вони хочуть переглядати той контент, який транслюється на даному стримінговому сервісі. Для приваблення людей старшого віку на сервісі показують інтерактивні канали, а для найменших – мультфільми.

Контент доступний багатьма мовами, серед яких українська, англійська, естонська, грузинська, польська, турецька та багато інших.

Підсумовуючи все вищесказане, головною перевагою стримінгового сервісу MEGOGO є найрізноманітніший контент.

Як і інші системи потокової передачі медіаконтенту на вимогу, YouTube дозволяє користувачам дивитися, завантажувати, коментувати та підписуватися на інших користувачів. Переглядаючи відео в YouTube ви можете бачити його назву, опис, автора, оцінки користувачів, коментарі, дату завантаження, поділитися в інших соцмережах. Під час перегляду контенту можна обрати якість відео, субтитри та швидкість відтворення, величину та ширину вікна перегляду.

YouTube надає унікальну можливість зареєстрованим користувачам створювати, викладати та поширювати власний відеоконтент, відкриваючи двері до глобальної аудиторії. Завдяки платформі автори можуть не лише демонструвати свої таланти, ділитися знаннями або розважати глядачів, але й розвивати свої канали, формуючи стабільну спільноту підписників.

Для таких користувачів передбачено спеціальні інструменти аналітики, які дозволяють детально аналізувати показники переглядів, взаємодії з аудиторією, рівень залучення та ефективність окремих відео. Це допомагає зрозуміти, які теми та формати користуються популярністю, а також адаптувати контент під запити глядачів.

Крім того, YouTube відкриває можливості для монетизації контенту. Авторам, які досягли певного рівня популярності (наприклад, кількість підписників та годин переглядів), пропонується приєднатися до Партнерської програми YouTube. Завдяки цьому вони можуть заробляти гроші через рекламу, яку платформа інтегрує

у відео. Рекламодавці, у свою чергу, отримують доступ до цільової аудиторії, що робить цю співпрацю взаємовигідною. Таким чином, YouTube стає не лише творчим майданчиком, але й джерелом доходу для мільйонів користувачів по всьому світу (рис.1.5).

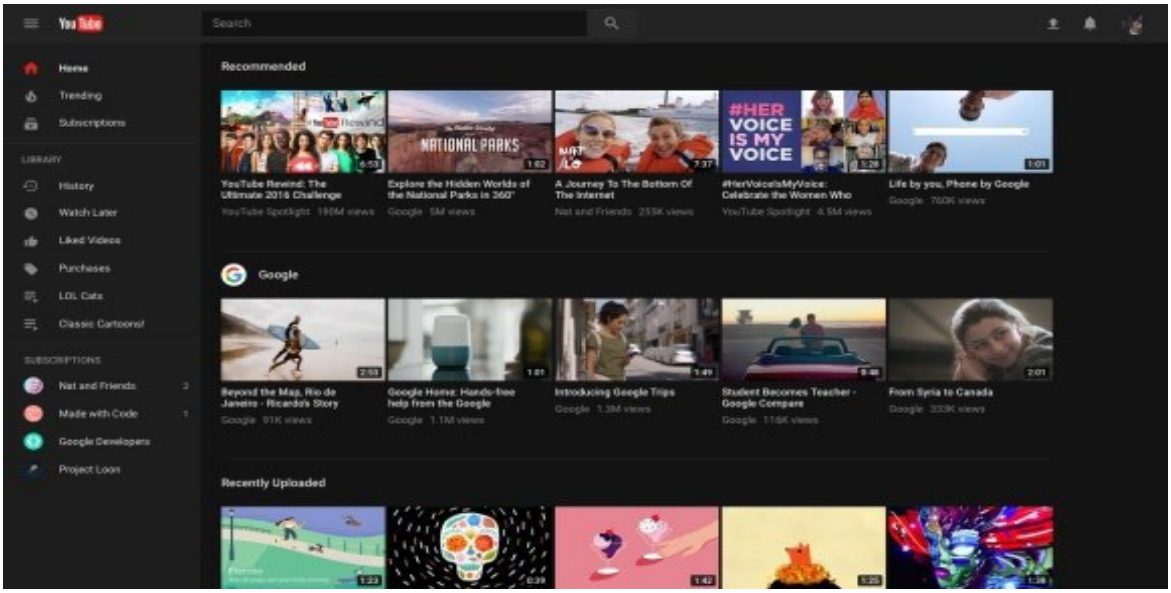

Рисунок 1.5 – Приклад інтерфейсу стримінгового сервісу «YouTube»

Після проведеного вище дослідження існуючих систем-аналогів у таблиці 1.1 наведено порівняльну характеристику усіх вищенаведних стримінгових сервісів з метою визначення усіх переваг для гарного моделювання та недоліків задля оптимізації власної розробки.

Таблиця 1.1 – Порівняльна характеристика стримінгових сервісів для фільмів

Характеристика	Netflix	Hulu	MEGOGO	YouTube
Зручний інтерфейс	+	-	+/-	+
Можливість перегляду не тільки фільмів, а й іншого контенту	-	-	+	+
Перегляд медіафайлів без авторизації	-	-	-	+
Зрозумілий інтерфейс	+	-	-	+
Можливість додання друзів	-	-	-	-
Функція рекомендацій контенту	+	-	+	+

На основі проведеної порівняльної характеристики доведено необхідність розробки стримінгового сервісу для фільмів та серіалів з компенсуванням наведених недоліків та погрішностей.

1.3 Аналіз відомих методів надання рекомендацій в експертних системах

Експертні системи - це інтелектуальні комп'ютерні системи, призначені для вирішення завдань, які зазвичай вимагають експертних знань і рішень. Вони створюються з метою відтворення знань та досвіду людського експерта в конкретній галузі або предметній області [10]. Експертні системи використовуються для автоматизації прийняття рішень, надання рекомендацій, вирішення складних проблем та підтримки прийняття рішень в різних сферах, таких як медицина, фінанси, інженерія, технічне обслуговування, наука і багато інших.

Основними складовими експертних систем значиться база знань і механізм логічних висновків. Не рідко для подання фактичних знань застосовується окремий механізм – база даних та в базі знань залишаються лише процедурні знання. Задля супроводу бази знань та поповнення її знаннями, які отримали від експертів, застосовують окремий модуль набуття знань.

Іншим не менш важливим компонентом експертної системи є інтерфейс користувача, який призначений для правильної передачі відповідей користувачу в практичній для нього формі. Цей інтерфейс потрібний і для експертів, які здійснюють операції вибору інформації з бази знань.

В експертній системі присутній модуль “логічного” аналізу – модуль порад і пояснень, який забезпечує надання обґрунтованих порад або рішень завдання, супроводжуючи їх відповідними коментарями, що пояснюють логіку пропонованих рішень.

Зазначені структурні елементи (рис.1.6) є найхарактернішими для більшості експертних систем, хоча в реальних умовах деякі з них можуть бути відсутні.

Рисунок 1.6 – Приклад структури експертної системи

Основними функціями експертних систем являються такі:

− експертні системи збирають інформацію, яка стосується конкретної галузі або області, де вони будуть застосовуватися. Ця інформація може бути представлена у вигляді бази даних, текстових документів, структурованих знань тощо [11];

− експертні системи володіють базою даних або базою правил, що містять експертні знання про конкретну галузь. Ці знання використовуються для прийняття рішень та надання рекомендацій;

− експертні системи використовують інференційні механізми для виведення нових інформаційних фактів або рекомендацій на основі доступних знань і правил. Цей процес може включати логічне розсудження, суто евристичні методи, розрахунки і т.д;

− експертні системи можуть допомагати користувачам приймати обґрунтовані рішення на основі аналізу інформації та знань. Наприклад, в медичних експертних системах вони можуть допомагати лікарям у діагностиці і лікуванні пацієнтів;

− деякі експертні системи можуть навчатися на основі нових даних і досвіду, що дозволяє їм покращувати якість прийняття рішень з часом [12];

− експертні системи можуть надавати пояснення щодо того, як вони прийшли до певного рішення або рекомендації, що допомагає користувачам розуміти логіку системи. Експертні системи є корисними інструментами для автоматизації та підтримки прийняття рішень в ситуаціях, де необхідно враховувати велику кількість даних і складних правил. Вони можуть значно покращити продуктивність і якість роботи фахівців у різних галузях.

Надання рекомендацій в інформаційних системах - це важлива складова сфери персоналізації інформації та продуктів для користувачів. Цей процес базується на аналізі даних про користувачів та об'єкти (такі як товари, статті, фільми тощо), щоб рекомендувати користувачам ті об'єкти, які ймовірно їх зацікавлять.

Існує кілька методів надання рекомендацій в інформаційних системах:

1. Фільтрування за змістом (Content-Based Filtering): Цей метод використовує інформацію про властивості об'єктів і відповідність їхніх властивостей вимогам або інтересам користувача. Наприклад, у системі рекомендацій фільмів можуть брати до уваги жанр, акторів, режисера тощо.

2. Колаборативне фільтрування (Collaborative Filtering): Цей метод базується на інформації про споживачів і їхній історії взаємодії з системою. Є два підтипи колаборативного фільтрування:

− фільтрування користувача (User-Based Filtering): Рекомендації генеруються на основі схожості між користувачами. Якщо користувач А споживав схожий контент, що і користувач Б, то система може рекомендувати користувачу А контент, який сподобався користувачу Б;

− фільтрування об'єкта (Item-Based Filtering): Рекомендації генеруються на основі схожості об'єктів (товарів, послуг). Якщо користувач споживав або виразив інтерес до певного об'єкта, то система може рекомендувати інші об'єкти, які були споживані або вподобані користувачами зі схожими інтересами.

3. Факторизація матриць (Matrix Factorization): Цей метод використовується для розкладання матриці споживачів і об'єктів у факторні

матриці. Це дозволяє знайти схожість між споживачами і об'єктами в скрижаліному просторі факторів і генерувати рекомендації на основі цього розкладання.

4. Гібридні методи (Hybrid Methods): Гібридні методи комбінують кілька методів надання рекомендацій для підвищення точності і робастості системи. Наприклад, можна поєднати фільтрування за змістом і колаборативне фільтрування для надання персоналізованих рекомендацій.

5. Ранжування (Ranking): У цьому методі система генерує список об'єктів, які потенційно можуть бути цікавими користувачу, і ранжує їх у порядку важливості або релевантності. Ранжування може використовувати різні фактори, такі як схожість, популярність, рейтинги тощо.

6. Активне навчання (Active Learning): Системи можуть використовувати методи активного навчання для збору додаткової інформації від користувача, яка допомагає покращити якість рекомендацій з часом [13].

Надання рекомендацій в експертних системах є важливою функцією, оскільки вони повинні допомагати користувачам приймати обґрунтовані рішення. Ось ще деякі методи надання рекомендацій в експертних системах:

1. Базові правила: експертні системи можуть містити базу правил, які формалізують експертні знання. Ці правила можуть бути використані для порівняння вхідних даних користувача з експертними знаннями і генерування рекомендацій на основі цього порівняння.

2. Експертні системи на основі знань. Ці системи використовують базу даних або онтологію знань, які містять структуровану інформацію про певну галузь. Рекомендації генеруються на основі запитів користувача і відповідних знань в базі.

3. Машинне навчання. Експертні системи можуть використовувати алгоритми машинного навчання для аналізу даних і навчання на основі історичних даних. Наприклад, алгоритми рекомендацій можуть використовувати колаборативний або контент-фільтрування для рекомендацій на основі користувацьких відгуків і звичок.

4. Аналіз тексту і обробка природної мови (NLP). Деякі експертні системи можуть аналізувати тексти користувачів, такі як запити або коментарі, і використовувати методи NLP для розуміння контексту і генерації рекомендацій.

5. Системи розпізнавання образів. Для деяких типів продуктів або послуг, які можна представити зображеннями, можуть використовуватися системи розпізнавання образів для рекомендацій на основі зображень.

6. Генетичні алгоритми і оптимізація. Генетичні алгоритми можуть бути використані для пошуку оптимальних рішень у просторі можливих варіантів і генерації рекомендацій на основі цього пошуку.

7. Спорідненість і семантичні відносини. Аналіз семантичних відносин між об'єктами або поняттями може використовуватися для рекомендацій на основі схожості або спорідненості.

8. Агенти інтелектуального агентства. Інтелектуальні агенти можуть надавати персоналізовані рекомендації, збираючи та аналізуючи дані про дії користувача в реальному часі.

Вибір методу надання рекомендацій залежить від конкретного завдання, типу даних і можливостей системи. Зазвичай використовують комбінацію різних методів для підвищення точності і якості рекомендацій у експертних системах [14].

Розглянемо детальніше експертні системи надання рекомендацій найвідоміших та найпопулярніших стримінгових сервісів:

− Netflix використовує систему під назвою "Cinematch" для надання рекомендацій. Ця система базується на колаборативному фільтруванні, яке враховує дії і вподобання користувачів, такі як оцінки фільмів і історія перегляду. Крім того, Netflix також використовує інші методи машинного навчання, щоб врахувати контекст і особливості кожного користувача;

− Spotify має популярну функцію "Discover Weekly", яка створює персоналізований плейлист із новою музикою для кожного користувача. Ця система використовує аналіз слухачів і споживачів музики, а також аналіз контенту пісень, щоб рекомендувати треки, які можуть сподобатися слухачам;

− рекомендаційна система YouTube аналізує історію перегляду користувача, взаємодію з відеороликами, а також спільні перегляди і вподобання для надання рекомендацій. Вона також використовує алгоритми машинного навчання для визначення схожих відео та залучення користувачів до подивитися ще;

− Amazon використовує систему рекомендацій на основі рейтингів і відгуків користувачів. Вони також враховують історію перегляду, щоб надавати персоналізовані рекомендації фільмів і телешоу;

− Pandora використовує алгоритми аналізу музики, щоб визначити музичні характеристики пісень і включити їх у свої рекомендації. Крім того, вони враховують вподобання користувачів, аналізуючи їхню історію прослуховування.

Всі ці системи намагаються надавати персоналізовані рекомендації для своїх користувачів, використовуючи різні методи аналізу даних і машинного навчання. Після обробки інформації про користувачів і контенту вони генерують списки рекомендацій, сподіваючись підвищити задоволеність та зацікавленість користувачів і покращити їхній досвід використання сервісу.

1.4 Висновки розділу 1

У першому розділі було проведено дослідження та аналіз сфери стримінгового сервісу для фільмів і об'єкта проектування. Були оглянуті існуючі аналоги систем, а також їх основні характеристики. Існуючі аналоги стримінгових сервісів для фільмів призначені для перегляду та пошуку контенту, але мають свої недоліки, такі як неможливість додавання друзів, обмежений доступ до короткої інформації та автобіографії, а також не дуже зручний інтерфейс.

З огляду на останні тенденції, все більше людей віддають перевагу перегляду контенту в Інтернеті, і, отже, розроблений стримінговий сервіс може користуватися попитом.

2 МОДЕЛЮВАННЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ СТРИМІНГОВОГО СЕРВІСУ ДЛЯ ФІЛЬМІВ

2.1 Розробка моделі інформаційної системи стримінгового сервісу для перегляду фільмів

Модель інформаційної системи (ІС) – це абстрактне відображення структури та функціоналу системи, що включає в себе всі компоненти, їх взаємозв'язки та процеси, які відбуваються всередині системи [15]. Така модель надає загальний огляд системи і є важливим інструментом для розробки, аналізу та документування інформаційних систем.

Моделюючи структуру, ми описуємо складові частини системи і відношення між ними. UML у більшості випадків застосовується в якості об’єктно-орієнтованої мови моделювання, тому не дивно, що основним видом складових частин, з яких складається система при такому підході, є класи і відношення між ними. У кожен конкретний момент функціонування системи можна вказати кінцевий набір конкретних об’єктів (екземплярів класів) і існуючих між ними зв’язків (екземплярів відношень). Проте в процесі роботи цей набір не залишається незмінним: об’єкти створюються і знищуються, зв’язки встановлюються і втрачаються.

Діаграми варіантів використання дозволяють отримати відмінне візуальне уявлення про вимоги користувачів. Діаграма Use Case визначає поведінку системи з точки зору користувача. Діаграма Use Case розглядається як головний засіб для первинного моделювання динаміки системи, використовується для з’ясування вимог замовника до системи, що розробляється, фіксації цих вимог у формі, яка дозволить проводити подальшу розробку.

До складу діаграм варіантів використання входять елементи Use Case, актори, а також відношення залежності, узагальнення і асоціації. Як і інші діаграми, діаграми Use Case можуть включати примітки і обмеження. Залежно від мети виконання процедури розрізняють такі варіанти використання:

- основні (базові) – забезпечують необхідну функціональність ПЗ, що розробляється;
- допоміжні – забезпечують виконання необхідних налаштувань системи і її обслуговування (наприклад, архівація інформації і тому подібне);
- додаткові – забезпечують додаткові зручності для користувача (як правило, реалізуються, якщо не вимагають серйозних витрат яких-небудь ресурсів ні при розробці, ні при експлуатації).

Якщо подивитися на модель використання з найзагальнішої точки зору, то неважко помітити, що в моделі є присутніми:

- внутрішня система, що моделюється, у формі набору варіантів використання, можливо пов’язаних залежностями і узагальненнями;
- зовнішнє оточення, у формі набору дійових осіб, можливо пов’язаних узагальненнями;
- зв’язок між системою, що моделюється, і зовнішнім оточенням у формі асоціацій між дійовими особами і варіантами використання.

Зазвичай абсолютно ясно, що знаходиться усередині системи, що моделюється, а що зовні. Якщо це з якоїсь причини неясно, або ж вимагається збільшити наочність діаграм, то можна скористатися спеціальною конструкцією, яка називається «Межі системи».

Модель інформаційної технології потокового сервісу для перегляду фільмів повинна включати в себе наступні компоненти:

- система створення акаунту, яка дозволяє користувачу ініціювати створення свого профілю для подальшого користування додатком;
- система логінування, що надає користувачу можливість увійти в раніше створений акаунт;
- система перегляду домашньої сторінки, котра забезпечує користувача необхідними даними пошуку контенту;
- система пошуку фільмів, яка передбачає можливість користувача шукати відео для перегляду. Пошукова система дозволяє користувачам шукати фільми з

заданими умовами. Критерії пошуку включають назву, жанр, режисера, актора тощо. Каталог фільмів зберігається в базі даних. База даних містить інформацію про всі доступні фільми, включаючи назву, жанр, режисера, актора, дату виходу, тривалість та рейтинг;

– система вивчення переваг клієнта дає можливість рекомендаційній системі визначити контент, який буде цікавий користувачу. Система рекомендацій надає користувачам ті фільми, які їм подобаються. Рекомендації можуть базуватися на історії перегляду користувачів, рейтингах фільмів чи інших факторах;

– система генерації відео на основі раніше отриманих даних про переваги і бажання клієнта генерує відеоряд, який буде запропоновано користувачу. Система зворотного зв'язку дозволяє користувачам залишати відгуки про фільм. Відгуки допоможуть іншим користувачам вибрати, які фільми варто подивитися.

Цільовою аудиторією сайту є люди, котрі люблять дивитися фільми та серіали й надають перевагу переглядати їх не в кінотеатрі, а в інтернеті. Основним завданням сайту є надання необхідної інформації користувачам, а також можливість додання друзів та перегляду короткої біографії про них.

Існує величезна кількість стандартів для створення правильної й надійної архітектури, а також для розробки й інтеграції програмних систем. Застосування цих стандартів істотно збільшить шанси на успішне створення системи і її подальше безвідмовне функціонування, однак раціональність їхнього застосування повинна визначатися до моменту початку робіт, оскільки складність системи при їхній інтеграції може істотно зрости.

На основі вищенаведеної інформації побудовано модель інформаційної технології стримінгового сервісу для фільмів (рис. 2.1).

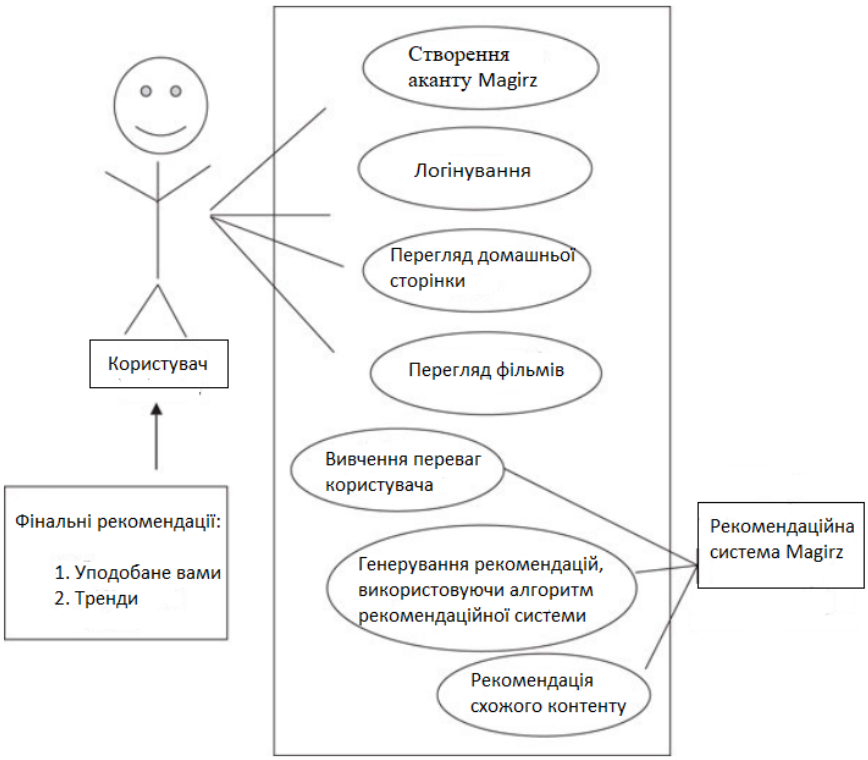

Рисунок 2.1 – Модель інформаційної технології стримінгового сервісу для фільмів

На рисунку наведена функціональна модель, яка описує принципи дії інформаційної технології стримінгового сервісу для фільмів. При розробці моделі інформаційної технології стримінгового сервісу для потокового відео слід враховувати наступні фактори [16]:

- Зручний та зрозумілий інтерфейс;
- можливість переглядати відео гарної якості;
- можливість автентифікуватися та авторизуватися;
- пошук фільмів чи серіалів;
- додавання фільмів до списку улюблених;
- додавання друзів;
- змога переглянути короткі відомості про акторів чи режисерів.

Створено модель вимог до інформаційної технології стримінгового сервісу для фільмів (рис. 2.2).

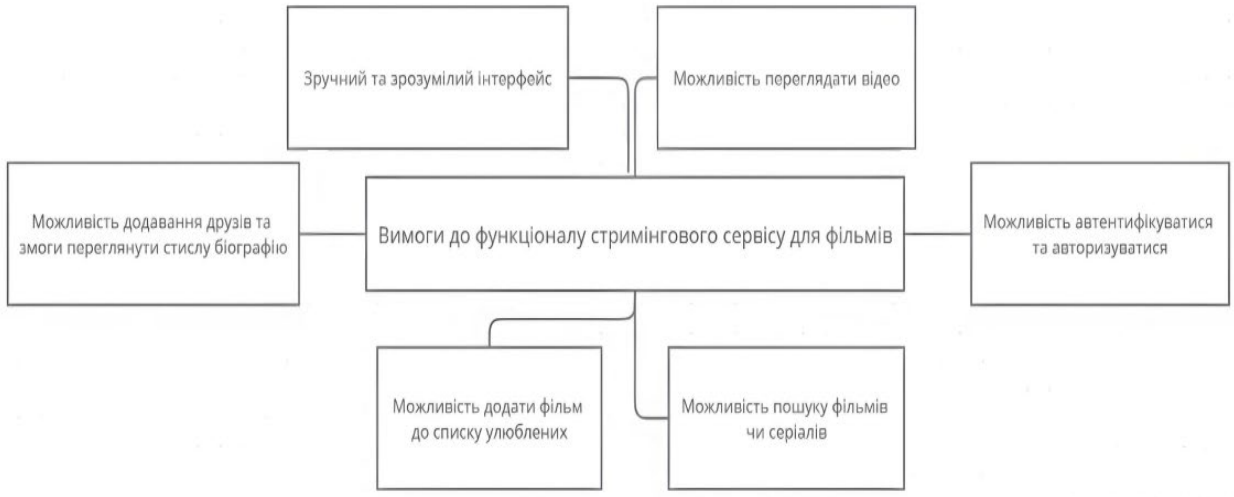

Рисунок 2.2 – Схема моделі вимог до інформаційної системи стримінгового сервісу для фільмів

Є кілька способів створити більш точну систему генерації рекомендацій (рис.2.3):

- алгоритми фільтрації на основі контенту базуються на інформації про жанри, акторів, країни тощо. Ці алгоритми аналізують, які шоу користувач переглянув і поставив лайк, а потім знаходять подібний вміст;

- алгоритми спільної фільтрації, з іншого боку, покладаються на інформацію про минулу поведінку користувачів. Вони аналізують смаки кожного користувача, а потім дають рекомендації на основі вмісту, який подобається користувачам зі схожими смаками;

- алгоритми машинного навчання можуть підвищити точність прогнозів, але потребують часу для навчання нейронних мереж.

Архітектура інформаційних систем для потокових служб перегляду фільмів повинна бути масштабованою та гнучкою для задоволення зростаючих потреб користувачів. Вона також повинна бути надійною та ефективною, щоб забезпечити безперебійну роботу служби.

Одним із підходів до розробки архітектури потокових служб є використання архітектури клієнт-сервер. У цьому випадку клієнтська частина системи відповідає за

взаємодію з користувачем, а серверна частина системи відповідає за зберігання даних і обробку запитів.

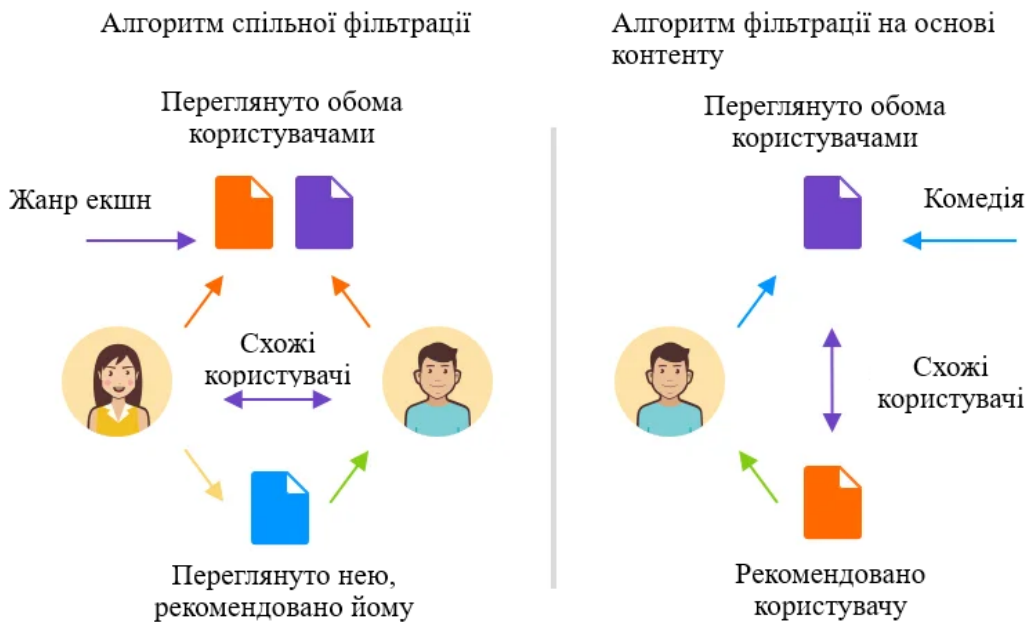

Рисунок 2.3 – Порівняльна схема найвідоміших рекомендаційних систем

Іншим підходом до розробки архітектури потокової служби є використання архітектури розподіленої системи. У цьому випадку система складається з декількох компонентів, розташованих на різних серверах. Це дозволяє додавати сервери та розширювати систему.

Для розробки стримінгового серверу будемо використовувати клієнт-серверну архітектуру. Вона отримала свою популярність, коли почав динамічно розвиватися інтернет та зосередилась значна частина інформації на серверах в базах даних.

Клієнт-серверна архітектура (рис. 2.4) означає, що головна частина ресурсів інформаційної мережі зберігається на серверах, які обслуговують клієнтів [17]. Є такі типи компонентів в даній архітектурі:

– набір серверів, що віддають інформацію або інші послуги клієнтам, які до них звертаються;

– набір клієнтів, що реалізують сервіси, які можуть надаватися серверами;

– мережа, що покриває взаємодію між серверами та клієнтами.

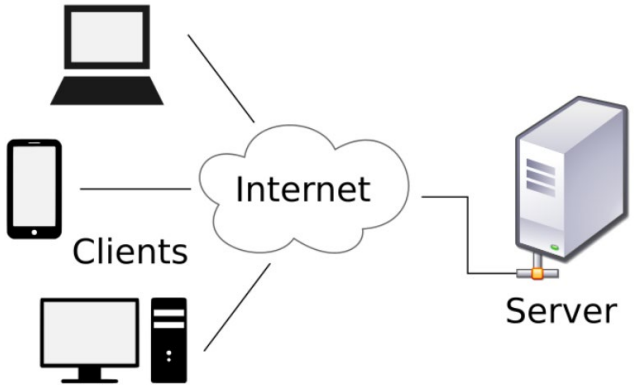

Рисунок 2.4 – Схема клієнт-серверної архітектури

Правила інтерактивності між сервером та клієнтом називається мережевим протоколом. Різноманітні протоколи лише описують різні сторони якогось типу зв'язку, а разом утворюють стек протоколів. Нові протоколи для інтернету окреслюються IETF, інші ж – ISO чи IEEE. Модель ISO, відповідно до протоколів, розрізняються на 7 рівнів за призначенням – від фізичного до прикладного. Наведу декілька найбільш використовуваних з них. Для фізичного рівня – RS-232, для канального – Ethernet, Token ring чи Fibre Channel, для мережевого – ICMP і IP, для транспортного – SPX, а для прикладного – HTTP або HTTPS. Також використовують три стеки протоколів: TCP/IP, IPX/SPX та NetBIOS/SMB.

Клієнт-серверна архітектура характеризується перш за все розподілом обов'язків між сервером і клієнтом [18]. Можна виокремити логічно три рівні операцій:

– рівень управління даними, що забезпечує доступ до даних та зберігання їх;

– рівень представлення даних, що являє собою по суті інтерфейс користувача і несе відповідність за представлення даних клієнту та введення команд від нього;

– прикладний рівень, що відповідає за головну логіку застосунку і на котрому відбувається обробка інформації.

Існує два види клієнт-серверної архітектури: дволанкова та трьохланкова.

Дволанкова клієнт-серверна архітектура визначає інтерактивність двох прграмних модулів – серверного і клієнтського. Відповідно до розподілення наведених вище функцій розрізняють:

– модель товстого клієнта. У ній сервер управляє лише даними, а інтерфейс користувача й обробка інформації концентрується на клієнтській стороні. Пристрої з обмеженою потужністю, наприклад, мобільні телефони, частенько називають товстими клієнтами;

– модель тонкого клієнта. У цій моделі логіка всього застосунку та управління даними концентрується на сервері. На клієнтській же стороні лише логіка інтерфейсу.

Триланкова клієнт-серверна архітектура взяла свій розвиток з середини 90-х років, вона завбачає розділення прикладного рівня та управління даних. Розрізняють програмний рівень, в якому концентрується прикладна логіка додатку [19]. Програми середнього рівня можуть працювати на спеціальних серверах додатків, але такі програми також можуть працювати на звичайному веб-сервері. Врешті-решт, управління даними виконується сервером даних.

Підсумовуючи вищесказане, дворівнева клієнт-серверна архітектура набагато простіша, адже запити виконуються одним сервером, проте саме через це вона й менш надійна, а, отже, висуваються більші вимоги до продуктивності сервера.

Трирівнева клієнт-серверна архітектура більш складна, але через те, що функції розподілені між серверами другого і третього рівнів, дана архітектура має:

– високий ступінь масштабованості та гнучкості;

– високу безпеку (оскільки захист можна формулювати для кожного серверу або рівня);

– високу продуктивність (оскільки завдання розділені між серверами).

Головна ідея клієнт-серверної архітектури визначається у поділі додатку на декілька компонентів, кожний з яких імплементує деякий набір сервісів. Ці компоненти можуть запускатися на різних пристроях, що підвищує надійність, рівень безпеки та продуктивність в цілому [20].

Залежно від того, яка робота надається, вирізняють наступні сервери:

– веб-сервер. Це сервер, який отримує HTTP-запити та видає їм відповідь, зазвичай разом із HTML розміткою, у якій знаходяться всі файли та медіа-потоки. Клієнти переходять на сторінку веб-серверу за URL-адресою;

– сервер застосунків. Це сервер, який імплементує прикладні програми;

– сервер баз даних. Це сервер, що використовується для обробки користувацьких запитів на SQL. При цьому СУБД знаходиться на сервері, до якого з'єднується клієнтський додаток;

– файловий сервер. Такий сервер забезпечує деякий рівень захисту від несанкціонованого доступу, він зберігає інформацію у файлах і надає клієнтам доступ до них;

– сервер друку. Даний сервер використовується для роботи з принтерами. Сервер друку надає можливість управляти принтерами, друкувати через протокол IPP або через URL принтера;

– поштовий сервер. Цей сервер використовується для поштових скриньок;

– термінальний сервер. Даний сервер надає можливість клієнтам мати обчислювальні ресурси. Технічно термінальний сервер — це потужний комп'ютер, під'єднаний до термінальних клієнтів, які зазвичай мають низькопродуктивні або застарілі робочі станції або спеціальні рішення для доступу до термінального сервера. Термінальний сервер використовується для віддаленого обслуговування користувача при наданні робочого столу;

– VPN сервер. Сервери віддаленого доступу та VPN надають віддаленим користувачам точку входу у вашу мережу. Роль віддаленого доступу/сервера VPN дозволяє реалізувати протоколи маршрутизації як для локальної, так і для глобальної мережі;

– DNS сервер. Даний сервер дає можливість перетворювати доменні імена в IP-адреси. Розподілена мережева служба, яка перетворює людські читабельні доменні імена (наприклад, www.example.com) в IP-адреси, які використовуються комп'ютерами для знаходження інших комп'ютерів у мережі. DNS є ключовим компонентом Інтернету, оскільки він дозволяє користувачам і комп'ютерам легко навігувати у Всесвітній мережі за допомогою зрозумілих доменних імен, замість

запам'ятовування числових IP-адрес;

– сервери для медіа стримінгу. Ці сервери використовуються для керування та доставки потокового аудіо та відео через інтернет.

Браузер може бути реалізацією так званих тонких клієнтів – логіка застосунку зосереджується на сервері, а функція браузера полягає переважно у відображенні інформації, завантаженої мережею з сервера, і передачі назад даних користувача. Однією з переваг такого підходу є той факт, що клієнти не залежать від конкретної операційної системи користувача, тому веб-додатки є міжплатформними сервісами. Унаслідок цієї універсальності і відносної простоти розробки веб-застосунки стали широко популярними в кінці 1990-х – початку 2000-х років. Істотною перевагою побудови веб-застосунків для підтримки стандартних функцій браузера є те, що функції повинні виконуватися незалежно від операційної системи клієнта. Замість того, щоб писати різні версії для Microsoft Windows, Mac OS X, GNU/Linux й інших операційних систем, застосунок створюється один раз для довільно обраної платформи і на ній розгортається. Проте різна реалізація HTML, CSS, DOM й інших специфікацій в браузерах може викликати проблеми при розробці веб-застосунків і подальшої підтримки. Крім того, можливість користувача налаштовувати багато параметрів браузера (наприклад, розмір шрифту, кольори, відключення підтримки сценаріїв) може перешкоджати коректній роботі застосунку.

Веб-додаток отримує запит від клієнта і виконує обчислення, після цього формує веб-сторінку і відправляє її клієнтові мережею з використанням протоколу HTTP. Саме веб-застосунок може бути клієнтом інших служб, наприклад, бази даних або стороннього веб-застосунку, розташованого на іншому сервері. Яскравим прикладом веб-застосунку є система управління вмістом статей Вікіпедії: безліч її учасників можуть брати участь у створенні мережевої енциклопедії, використовуючи для цього браузери своїх операційних систем (Microsoft Windows, GNU/Linux або будь-якої іншої операційної системи) без завантаження додаткових виконуваних модулів для роботи з базою даних.

2.2 Розробка технології надання рекомендацій у стримінгових сервісах для перегляду фільмів на основі фільтрації за контентом

Як можна зрозуміти з назви підходу, фільтрація на основі вмісту перш за все опирається на інформацію про контент в системі, а не про користувачів. Тобто рекомендації базуються на знаходженні схожих елементів, до тих, що користувач вже оцінив в минулому. Для цього необхідно створити профіль користувача та профіль елемента. Після цього, на основі параметрів елементів системи, можна зробити висновок щодо відповідності конкретного елементу конкретному користувачу. Для опису елементів системи та створення їх профілю рекомендаційні системи ставлять у відповідність кожному елементу певний набір ключових слів. Формально роботу рекомендаційної системи на основі фільтрації вмісту можна зобразити наступним чином (рис. 2.5).

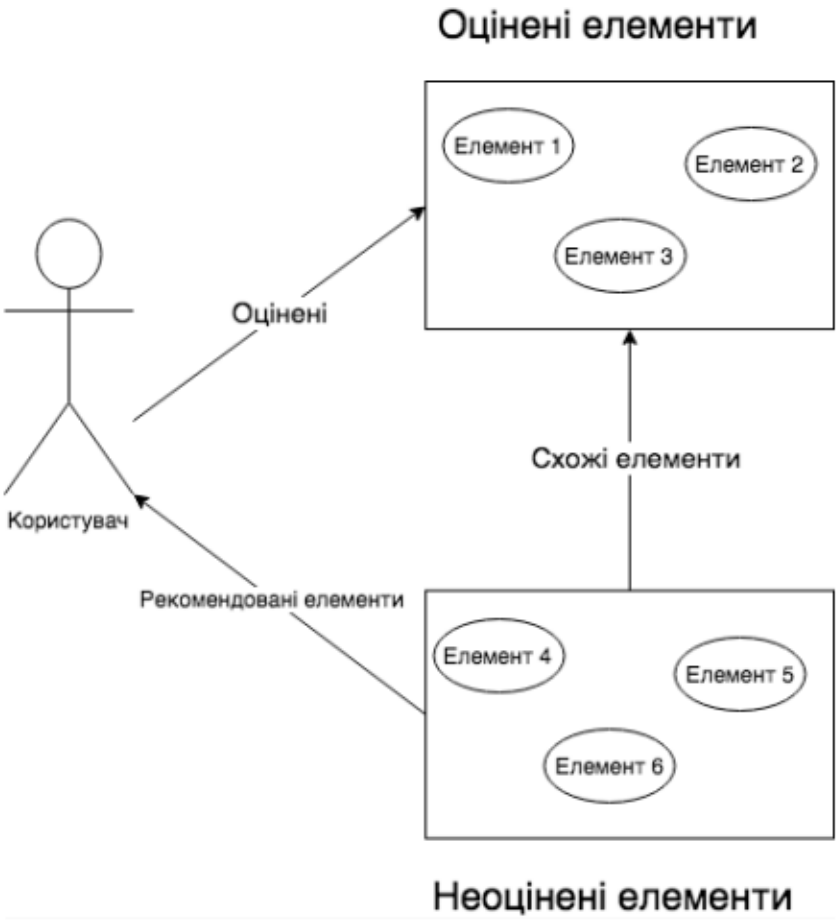

Рисунок 2.5 – Принцип дії системи рекомендацій, побудованої на основі аналізу контенту

У рекомендаційних системах з фільтрацією на основі вмісту функція зіставлення $h(u, s)$ деякого користувача u з деяким елементом s використовує інформацію про інші елементи системи $s' \in S$, які користувач u вже оцінив в минулому та схожість цих елементів з даним елементом s [21].

Тобто, в загальному випадку, для того щоб рекомендувати користувачу нові елементи, система повинна проаналізувати елементи, які користувач високо оцінив в минулому, знайти ознаки, що об'єднують ці елементи, та на основі знайдених ознак знайти нові елементи, які їм відповідають, і які не мають оцінок від користувача, тобто є новими для нього.

Формально – якщо елемент А сподобався користувачу U, а елемент В схожий на елемент А, то можна зробити висновок, що елемент В також сподобається користувачу U.

Таким чином, користувацький профіль формується у вигляді параметрів, що характеризують кожен елемент $s' \in S'$ де S' − множина оцінених елементів. У якості таких параметрів найчастіше виступають ключові слова, також вагові коефіцієнти ключових слів для кожного з елементів системи. Одним з найчастіше використовуваних способів знаходження даних вагових коефіцієнтів є TD-IDF міра [17].

Її можна обчислити за наступною формулою:

$$TF_{ij} = \frac{f_{ij}}{max_{z f_{zj}}}, \tag{2.1}$$

де f_{ij}- кількість входжень деякого ключового слова k_i в елемент d_j, при цьому ключове слово k_j зустрічається в n_i об'єктах.

Але при використанні вищезазначеної формули враховується лише частоти входження ключового слова. Це може призвести до випадку, коли максимальну вагу будуть мати найбільш розповсюджені ключові слова, що в подальшому може призвести до некоректного прогнозування вподобань конкретного користувача. Для уникнення такого випадку використовується величина IDF_i, яка є зворотною до

частоти входження ключового слова в елемент.

$$IDF_i = log\frac{N}{n_i}, \tag{2.2}$$

де N – кількість елементів, які потенційно можуть бути рекомендовані користувачу.

Виходячи з вищезазначених формул, можемо обчислити вагу w_{ij} ключового слова k_i в елементі d_j наступним чином:

$$w_{ij} = TF_{ij}: * IDF_i. \tag{2.3}$$

Тоді профіль елементу d_j можна задати наступним чином:

$$Content(d_{ij}) = (w_{1j}:, \dots, w_{kj}). \tag{2.4}$$

Рекомендаційні системи на основі фільтрації контенту при формуванні рекомендацій використовують інформацію про елементи, що вже були високо оцінені користувачем в минулому. З множини всіх елементів систему обираються лише ті, які є найбільш подібними до елементів, що вже були оцінені, таким чином будується гіпотеза, що нові елементи будуть також високо оцінені користувачем. Даний набір елементів, що вже були оцінені користувачем формують його профіль, а також вектор ваг ключових слів:

$$(w_{ui}:, \dots, w_{uk}), \tag{2.5}$$

де кожна вага w_{ui} визначає важливість ключового слова k_i для користувача u.

Тоді профіль елементу та профіль користувача можна представити у вигляді TF-IDF векторів $\overrightarrow{w_s}$ та $\overrightarrow{w_u}$, при цьому функція зіставлення елемента та користувача $h(u, s)$ може бути представлена у вигляді косинусу кута між векторами $\overrightarrow{w_s}$та $\overrightarrow{w_u}$:

$$h(u,s) = \cos(\overrightarrow{w_u}, \overrightarrow{w_s}) = \frac{\overrightarrow{w_u}\overrightarrow{w_s}}{||\overrightarrow{w_u}||||\overrightarrow{w_s}||} = \frac{\sum_{i=1}^{K} w_{ui} w_{is}}{\sqrt{\sum_{i=1}^{K} w_{iu}^2}\sqrt{\sum_{i=1}^{K} w_{is}^2}}, \quad (2.6)$$

де K – загальна кількість ключових слів в системі.

Рекомендаційні системи часто базуються не лише на використанні деякої евристики та методів інформаційного пошуку, а також на використанні інших технік, таких як наївний байєсовський класифікатор, нейронні мережі, дерева рішень, кластеризація та інші техніки машинного навчання. Наприклад, рекомендаційна система, використовуючи оцінки елементів користувачем, за допомогою наївного баєсовського класифікатору може визначити елементи, які користувач ще не оцінив, та обчислити ймовірність належності певного елементу s_i до певного класу C_i (високо оцінені/низько оцінені елементи).

2.3 Розробка алгоритму функціонування інформаційної технології стримінгового сервісу для перегляду фільмів

Алгоритм - це чітко визначена послідовність кроків або інструкцій, які вказують, як виконувати певну задачу або розв'язувати конкретну проблему. Алгоритми використовуються в різних областях, включаючи математику, програмування, науку про дані, інженерію та багато інших. Основна ідея алгоритму полягає в тому, щоб визначити послідовність операцій, які, виконані вірно та в правильному порядку, призведуть до досягнення бажаного результату. Ці потужні набори інструкцій складають основу сучасної технології та керують усім: від веб-пошуку до штучного інтелекту. Ось як працюють алгоритми:

- вхід – алгоритми приймають вхідні дані, які можуть мати різні формати, наприклад числа, текст або зображення;
- обробка – алгоритм обробляє вхідні дані за допомогою серії логічних і математичних операцій, маніпулюючи та перетворюючи їх за потреби;

- вихід – після завершення обробки алгоритм створює вихідні дані, які можуть бути результатом, рішенням або іншою значущою інформацією;
- ефективність – ключовим аспектом алгоритмів є їх ефективність, спрямована на швидке виконання завдань із мінімальними ресурсами;
- оптимізація – розробники алгоритмів постійно шукають способи оптимізувати свої алгоритми, роблячи їх швидшими та надійнішими;
- реалізація – алгоритми реалізовані різними мовами програмування, що дозволяє комп'ютерам виконувати їх і отримувати бажані результати.

Немає чітко визначених стандартів написання алгоритмів. Однак це проблема, яка залежить від ресурсів. Алгоритми ніколи не пишуться з урахуванням певної мови програмування.

Як ви всі знаєте, основні конструкції коду, такі як цикли, такі як do, for, while, усі мови програмування спільно використовують керування потоком, наприклад if-else тощо. За допомогою цих загальних конструкцій можна записати алгоритм.

Алгоритми зазвичай пишуться покроково, але це не завжди так. Написання алгоритму — це процес, який відбувається після чіткого визначення предметної області. Тобто, ви повинні бути в курсі проблемної області, для якої ви розробляєте рішення.

У розробці та аналізі алгоритму другий метод зазвичай використовується для опису алгоритму. Це дозволяє аналітику аналізувати алгоритм, легко ігноруючи всі небажані визначення. Вони можуть бачити, які операції використовуються та як просувається процес. Номери кроків писати необов'язково. Щоб розв'язати задану задачу, ви створюєте алгоритм. Проблему можна вирішити різними способами. У результаті можна вивести багато алгоритмів вирішення даної задачі.

При розробці алгоритму слід враховувати наступні фактори:

- модульність – функція була ідеально розроблена для алгоритму, якщо вам дають задачу і розбивають її на маленькі-маленькі модулі або маленькі-маленькі кроки, що є базовим визначенням алгоритму;

- правильність алгоритму визначається, коли задані вхідні дані дають бажаний вихід, що вказує на те, що алгоритм розроблено правильно. Аналіз алгоритму виконано правильно;
- супроводжуваність – це означає, що алгоритм має бути розроблений у зрозумілий, структурований спосіб, щоб під час перевизначення алгоритму в нього не вносилося значних змін;
- функціональність – враховує різні логічні кроки для вирішення проблеми реального світу;
- надійність означає здатність алгоритму чітко визначати вашу проблему;
- зручність – якщо алгоритм складний для розуміння, дизайнер не пояснюватиме його програмісту;
- простота – якщо алгоритм простий, його легко зрозуміти;
- розширюваність – алгоритм має бути розширюваним, якщо інший розробник алгоритму або програміст захоче його використовувати.

Щоб почати будувати алгоритм потрібно визначити точку відліку. У даному випадку це початок роботи з програмою.

Спочатку надходять вхідні дані та аутентифікація. Користувач вводить дані для входу або реєстрації в системі. Система перевіряє ідентифікацію користувача та дозволяє доступ до особистого облікового запису.

Далі йде пошук та фільтрація фільмів. Користувач може шукати фільми за різними параметрами, такими як жанр, рік випуску, актори і т. д. Система виконує пошук та фільтрацію фільмів на основі введених критеріїв.

Наступним етапом є рекомендаційний алгоритм. Система використовує рекомендаційний алгоритм для запропонування користувачу фільмів на основі його історії перегляду та вподобань.

Після цього відтворення фільмів. Користувач може обрати фільм для перегляду. Система починає стрімінг фільму на пристрій користувача.

Користувач може управляти відтворенням (відтворення, пауза, перемотка, гучність тощо). Збереження історії перегляду: Система зберігає історію перегляду користувача для подальших рекомендацій.

Подальшим етапом є сплачена підписка та оплата. Система може обробляти оплату за підписку, яка дозволяє користувачам отримувати доступ до ексклюзивного контенту без реклами.

Також додаються коментарі та оцінки. Користувачі можуть залишати коментарі та оцінки фільмів. Синхронізація між пристроями: Система дозволяє користувачам почати перегляд фільму на одному пристрої і продовжити на іншому (рис. 2.6).

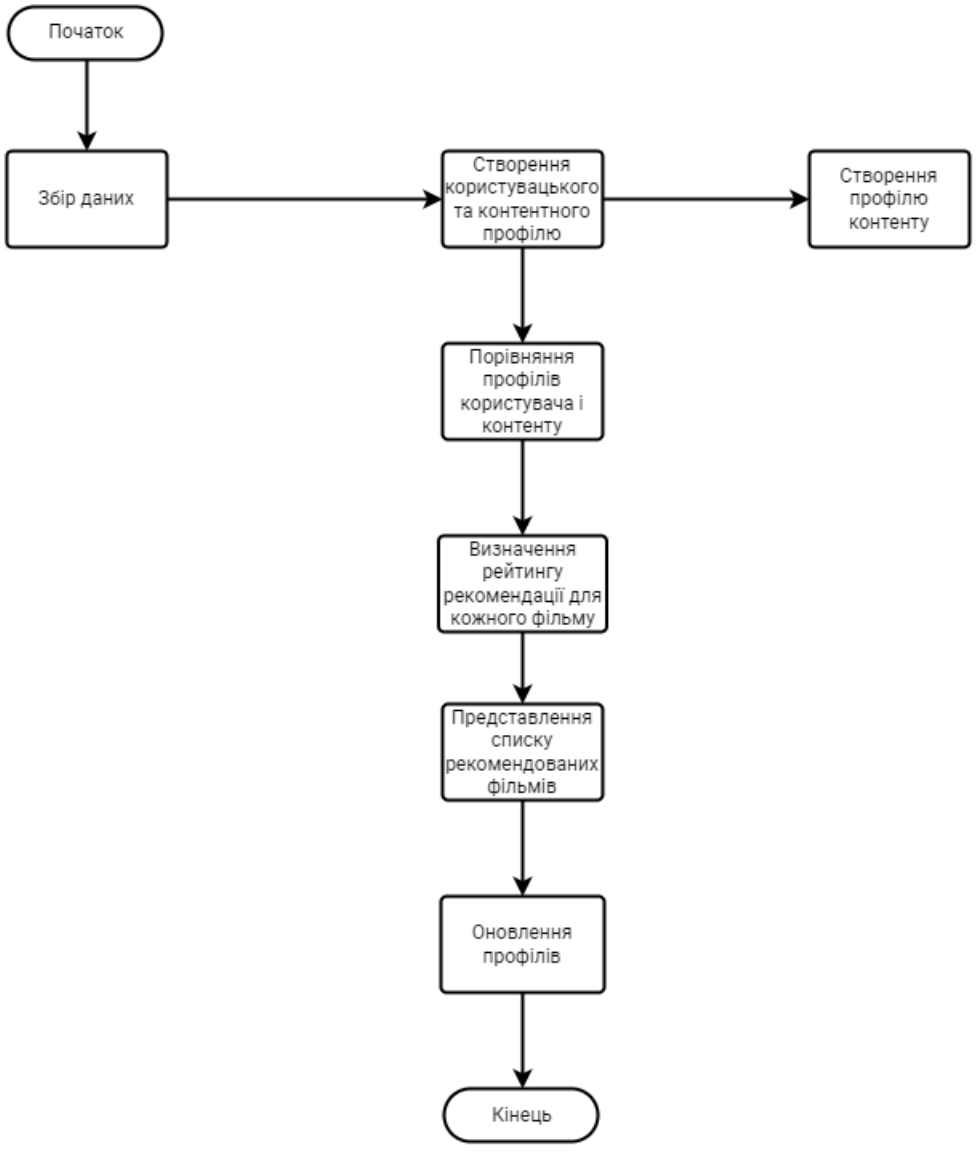

Рисунок 2.6 – Схема алгоритму загального функціонування стримінгового сервісу для фільмів

Останнім є етап забезпечення безпеки та обробка помилок. Система повинна бути захищеною від несанкціонованого доступу і здатною обробляти помилки та проблеми.

Головний функціонал реалізований у декількох нижченаведених модулях: модуль автентифікації та авторизації; модуль перегляду фільмів та головного банеру; модуль пошуку фільмів; модуль особистого кабінету; модуль уподобаних відео; модул додавання друзів.

2.4 Висновок розділу 2

У другому розділі було розроблено модель інформаційної системи стримінгового сервісу для фільмів, модель вимог до технології та узагальнений алгоритм функіонування стримінгового сервісу для фільмів. Визначено основні модулі розробленої системи. Детально описано доцільність використання технології Firebase для розробки даного сервісу. Розглянуто основні протоколи, що використовуються для стрімінгу медіаконтенту, такі як RTMP, MPEG-DASH, HLS, HDS, HTTP, HTTPS. Проаналізовано та досліджено клієнт-серверну архітектуру.

Під час моделювання та розробки стримінгового сервісу для фільмів було враховано всі вищезгадані недоліки завдяки застосуванню нових технологій.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ СТРИМІНГОВОГО СЕРВІСУ ДЛЯ ФІЛЬМІВ

3.1 Обґрунтування вибору мови та середовища програмування

Для аналізу і порівняння було обрано 3 мови програмування: JavaScript, Java та C#. Нижче наведено особливості кожної з мов програмування та порівняльну характеристику цих мов.

JavaScript - динамічна, об’єктно-орієнтована прототипна мова програмування. Реалізація стандарту ECMAScript. Найчастіше використовується для створення сценаріїв вебсторінок, що надає можливість на боці клієнта (пристрої кінцевого користувача) взаємодіяти з користувачем, керувати браузером, асинхронно обмінюватися даними з сервером, змінювати структуру та зовнішній вигляд вебсторінки.

JavaScript класифікують як прототипну (підмножина об’єктноорієнтованої) скриптову мову програмування з динамічною типізацією. Окрім прототипної, JavaScript також частково підтримує інші парадигми програмування (імперативну та частково функціональну) і деякі відповідні архітектурні властивості, зокрема: динамічна та слабка типізація, автоматичне керування пам'яттю, прототипне наслідування, функції як об’єкти першого класу. Основні архітектурні риси: динамічна типізація, слабка типізація, автоматичне керування пам’яттю, прототипне програмування, функції як об'єкти першого класу [15].

На JavaScript вплинули багато мов, при розробці була мета зробити мову схожим на Java. Мовою JavaScript не володіє будь-яка компанія або організація, що відрізняє його від ряду мов програмування, використовуваних в веб-розробці.

Синтаксис мови Java багато в чому походить від C та C++. Передусім, Java розроблялась як платформо-незалежна мова, тому вона має менше низькорівневих можливостей для роботи з апаратним забезпеченням. На відміну від C++ наявний «Garbige collector», що допомагає запобігти витоку пам'яті шляхом видалення об'єктів, які більше не використовуються додатком.

C# – об'єктно-орієнтована мова програмування з безпечною системою типізації для платформи .NET. Синтаксис C# близький до C++ і Java. Мова має строгу статичну типізацію, підтримує поліморфізм, перевантаження операторів, вказівники на функції-члени класів, атрибути, події, властивості, винятки, коментарі у форматі XML [13].

Проведемо дослідження та аналіз фреймворків JavaScript та оберемо той, який найбільш підійде для розробки. Існує декілька нових та актуальних фреймворків, серед яких є Angular, Vue.js та React [20]. Розглянемо їх по черзі.

Angular – це один із найкращих фреймворків JavaScript. Google використовує цю платформу для розробки SPA(single page application). Більше півмільйона сайтів у всьому світі використовують Angular саме через те, що він дає розробникам можливість об’єднати JavaScript з HTML і CSS.

Отже, Angular є прекрасним фреймворком, але не підходить для розробки стримінгового сервісу для фільмів через статичну типізацію TypeScript.

Vue – фреймворк JavaScript з відкритим вихідним кодом. Інтеграцію з Vue у проектах, які використовують інші бібліотеки JavaScript, спрощена, бо розроблена з ціллю адаптації. Vue досить простий у засвоєнні, для нього досить знати JavaScript і HTML. Сильною його стороною є інтерфейс командної строки, який пропонує багато плагінів, пресетів, миттєве прототипування й інтерактивний інструмент розробки проектів. Vue використовує Shadow DOM.

Проте, цей фреймворк нам також не підходить зі сторони розробки.

React є звичайно лідером на даний момент в області інфраструктури JavaScript. Він розроблений командою Facebook та зараз є також з відкритим вихідним кодом. Заснований на перевикористанні компонентів, інакше кажучи, блоки компонентів, котрі можна перевикористати, наприклад, логотип, кнопки чи поля вводу. React легко освоїти зі знаннями JavaScript, використовує JSX, тобто JavaScript всередині HTML та XML.

Проаналізувавши вищеперечислене дійдемо до висновку, що найдоцільніше використовувати React, адже, він дає найширші можливості для розробки та оптимізує її.

Visual Studio Code представляється як «легке» середовище програмування для кросплатформної розробки додатків. Visual Studio Code дає можливість для розробки додатків з графічним інтерфейсом і консольних додатків, веб-сайтів та веб-додатків.

Вбудований вкладчик, засоби рефакторинга та інструменти для співпраці з Git, автодоповнення конструкцій, навігація по коду та контекстна підказка є наявними у редакторі для кращого та комфортнішого написання коду.

Окреме слово хочу додати про додаток для роботи з Git, авторизувавшись через нього, дуже зручно працювати з сервісом та вносити свої зміни, бачити, які файли змінені, що уже є у віддаленому репозиторії, а що потрібно закомітати. Коли працюєш в команді, такого роду додатки грають величезну роль у комфортній роботі.

Підсумовуючи вищесказане наведемо особливості:

− вбудовані інструменти інтеграції з GitHub, GIT і Visual Studio Team Services для швидкого тестування, складання, пакування та розгортання різних типів додатків;

− зручна робота з проектами Unity;

− робота з Mono і Node.js за допомогою вбудованого налагоджувача;

− підтримка TypeScript і JavaScript;

− публікуйте створені програми в Microsoft Azure за допомогою служби Visual;

− послуги студійної команди;

− підтримка майже всіх мов програмування;

− написання коду для конкретного завдання з подальшою інтеграцією його в проект (з доповненням або безпосередньо);

− велика бібліотека шаблонів, готових фрагментів коду та сніппетів з можливістю додавати власні елементи;

− одночасна робота з кількома проектами (у кількох вікнах);

− інтерфейс користувача можна розділити на дві області для порівняння коду;

− функція налагодження.

Переваги :

− безліч налаштувань (як вся програма, так і інтерфейс);

− розширювана бібліотека з доповненнями та готовими рішеннями;

− багатофункціональність (редактор підтримує майже всі мови, які використовуються для створення додатків);

− простота і гнучкість.

3.2 Розробка модуля надання рекомендацій для стримінгового сервіс перегляду фільмів

Основний функціонал, який має виконувати моє програмне забезпеченні полягає у відображенні інформації про фільми та перехід для їх перегляду. Додатковим функціоналом є можливість додання фільму у вкладку улюблених, авторизація та автентифікація, перегляд людей та додання їх до кола друзів.

Для того щоб реалізувати головний функціонал використаємо API. Це сукупність інструментів та функцій в вигляді інтерфейсу для створення нових додатків, завдяки якій одна програма буде взаємодіяти з іншою. Все це дозволяє розширити функціональність продукту та зв’язувати один додаток з іншим.

Існує три способи взаємодії з API:

− дані, які програма отримує на виході після роботи з API;

− процес, що може виконувати програма за допомогою цього інтерфейсу;

− дані, що потрібно передати інтерфейсу для виконання потрібної функції.

Під час розробки функціоналу стримінгового сервісу для фільмів було використано TVMAZE API. Він надає безкоштовний, швидкий і чистий REST API, який простий у використанні, повертає JSON і відповідає принципам HATEOAS і HAL.

Серед можливих ендпойнтів використано наступні.

https://api.tvmaze.com/search.

```
const ListPage = () => {
   const [ films, setFilms ] = useState([]);
```

```
   const fetchFilms = "https://api.tvmaze.com/search/shows?q=comedy";
useEffect(() => {
     async function fetchData() {
       const request = await axios.get(fetchFilms);
setFilms(request.data);        return request;
     }       fetchData();
   }, []);
   console.log(films);        let
img;
   if (films.image && films.image !== "") {        img
= films.image.medium;
   }     return (
       <div>
       <div className="row">
       <h2 className="title__list">List</h2>
       <div className="row__posters">
{films.map((film) => {             return (
            <div className="poster" key={film.show.id}>
              <img
key={film.show.id}
className="row__poster"
src={img}
alt={film.show.name}
              />
              <h4>{film.show.name}</h4>
            </div>
          );
        })}
       </div>
     </div>
     </div>
   );
}
https://api.tvmaze.com/search/shows?q=girls,        const
requests = {
```

```
   fetchPopular: "https://api.tvmaze.com/shows",     fetchInteresting:
"https://api.tvmaze.com/shows?page=0",     fetchFriends:
"https://api.tvmaze.com/people?page=0"
}
const FriendsPage = () => {    const [friends,
setFriends] = useState([]);

  useEffect(() => {
    async function fetchData() {
      const request = await axios.get(requests.fetchFriends);
setFriends(request.data);       return request;
    }     fetchData();
}, []);   let img;
  if (friends.image && friends.image !== "") {     img =
friends.image.medium;
  }   return (
    <div>
      <div className="row">
        <h2 className="title__friend">Friends</h2>
        <div className="row__posters">
{friends.map((friend) => {             return (
            <div className="poster" key={friend.id}>
              <img
key={friend.id}
className="row__poster"
src={img}                  alt={friend.name}
              />
              <h4>{friend.name}</h4>
            </div>
          );
        })}
        </div>
      </div>
    </div>
```

```
  ); };
https://api.tvmaze.com/people.
const StartPage = () => {   return (
   <div className="startPage">
     <Banner />
     <Row title="Popular shows" fetchUrl={requests.fetchPopular} />
     <Row   title="Interesting   shows"   fetchUrl={requests.fetchInteresting}   />
</div>
  ); }.
```

Для розробки фільтрації на основі контенту потрібно сформувати базу, за допомогою якої фільми будуть пропонуватися користувачеві. В основу даної бази входять опитування користувачів та відслідковування їх вподобань, на основі того контенту, який додається до вкладки «Улюблені», та того, який переглянуто даним користувачем.

```
const RecommendationSystem = () => {    const
recommendedMovies = [
   {
     title: 'Movie 1',       genre:
'Action',
     description: 'Description for Movie 1',
   },    {
     title: 'Movie 2',       genre:
'Drama',
     description: 'Description for Movie 2',
   },
  ];
  return <MovieList movies={recommendedMovies} />;
};
const MovieList = ({ movies }) => (
  <div>
    <h1>Recommended Movies</h1>
    {movies.map((movie, index) => (
```

```
      <Movie        key={index}
title={movie.title}
genre={movie.genre}
description={movie.description}
      />
    ))}
  </div>
);
const MovieList = ({ movies }) => (
  <div>
    <h1>Recommended Movies</h1>
    {movies.map((movie, index) => (
      <Movie        key={index}
title={movie.title}
genre={movie.genre}
description={movie.description}
      />
    ))}   </div>
).
```

Опитування користувачів відбувається за заздалегідь розробленою та проаналізованою схемою. Для цього було виконано аналіз аналогів та обрано основні фактори, на які опираються користувачі під час вибору та відсіювання контенту. Серед таких питань було безліч, які майже не впливали на думку людини та вважаються безкорисними, які було відсіяно під час дослідження задля покращення та розробки функції рекомендацій фільмів. Кожне питання було ретельно проаналізовано та відібрано для оптимізації рекомендацій.

```
const SurveyForm = () => {
  const [formData, setFormData] = useState({
    question1: 'Чи сподобався вам фільм, який ви щойно переглянули? ',
question2: 'Чи ви більше схильні до перегляду нових релізів чи вам
```

```
більше подобається класика?',    question3: 'Чи важлива для вас рейтингова категорія
фільмів?',    question4: 'Чи подобаються вам фільми на основі книг чи реальних
подій?',    question5: 'Чи важливий для вас саундтрек у фільмах?',
  });
  const handleInputChange = (e) => {    const {
name, value } = e.target;    setFormData({
    ...formData,
    [name]: value,
   });
  };
  const handleSubmit = (e) => {
   e.preventDefault();
   console.log('Form submitted:', formData);
  };   return (
   <form onSubmit={handleSubmit}>
    <label>
     Question 1:       <input
type="text"          name="question1"
value={formData.question1}
onChange={handleInputChange}
      />
    </label>
    <br />      <label>
     Question 2:       <input
type="text"          name="question2"
value={formData.question2}
onChange={handleInputChange}
      />
    </label>
    <br />      <label>
     Question 3:       <input
type="text"          name="question3"
value={formData.question3}
onChange={handleInputChange}
      />
```

```
      </label>
      <br />
      <label>
        Question 4:        <input
type="text"          name="question4"
value={formData.question4}
onChange={handleInputChange}
        />
      </label>
      <br />
      <label>
        Question 5:        <input
type="text"          name="question5"
value={formData.question5}
onChange={handleInputChange}
        />
      </label>
      <br />
      <button type="submit">Підтвердити</button>
    </form>
  ); }.
```

Відповіді, отримані в рамках цього опитування, обробляються за допомогою спеціального програмного коду, який автоматично аналізує кожну відповідь. Після цього результати збираються та організовуються у вигляді масиву, що відповідає конкретному користувачеві. Це дозволяє системі ефективно зберігати й обробляти індивідуальні дані, забезпечуючи їх подальшу обробку або збереження для аналізу. Масив може містити різноманітні дані, такі як текстові відповіді, вибір варіантів або оцінки, що надаються користувачем, і може використовуватися для подальшої статистичної обробки чи створення звітів. Приклад опитування проведено на рисунку 3.1.

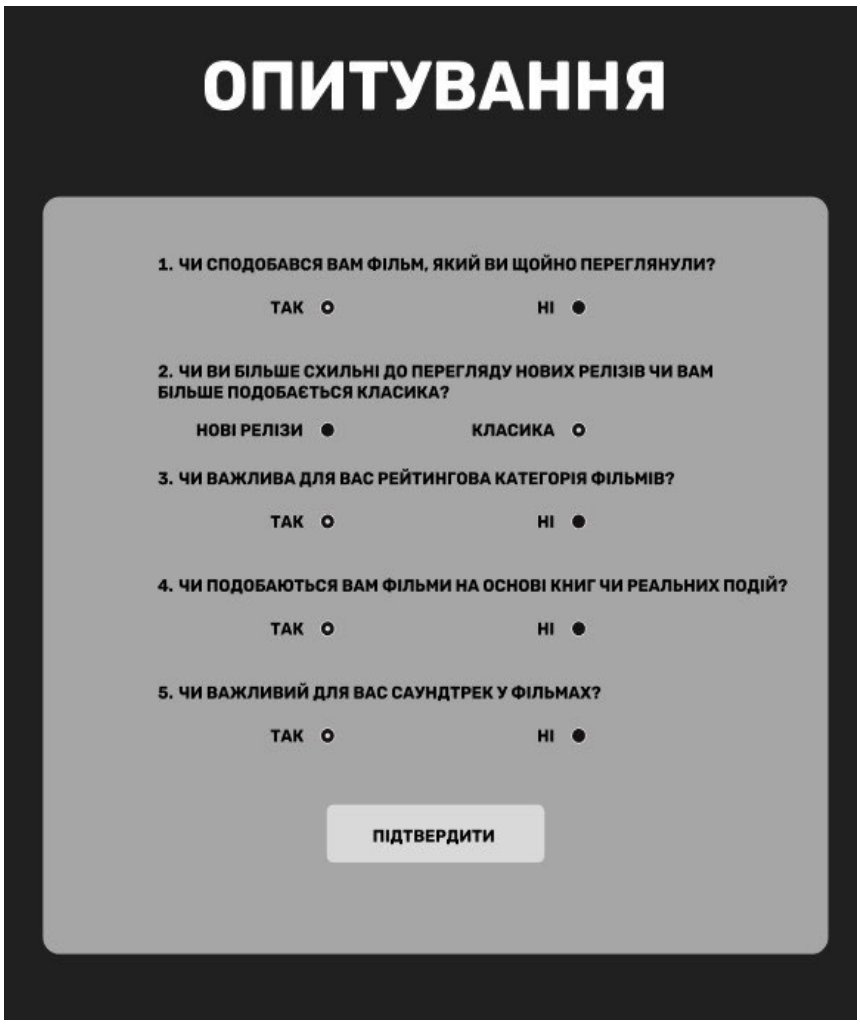

Рисунок 3.1 – Загальний вигляд інтерфейсного вікна опитування користувача

Фільми, які переглядаються, збираються додатком в окремий масив, створений для кожного конкретного користувача. За допомогою коду аналізується відео-контент за допомогою ключових факторів, які зберігаються у масивах жанру, акторів та опису.

Вкладка «Улюблені» використовується за схожим принципом. Фільми, додані у дану вкладку, зберігаються у окремий масив. Далі за допомогою коду виконується аналіз для подальшого використання цих даних з метою подальшого пропонування контенту.

3.3 Тестування програмного забезпечення стримінгового сервісу для фільмів

Для початку тестування відкрито форму, за допомогою якої, надаються рекомендації. Для першого тесту обрано наступні відповіді, які продемонстровано на рисунку 3.2.

Результатом роботи програмного забезпечення стримінгового сервісу для фільмів є відеоконтент з відповідними показниками, які показані на рисунку 3.3.

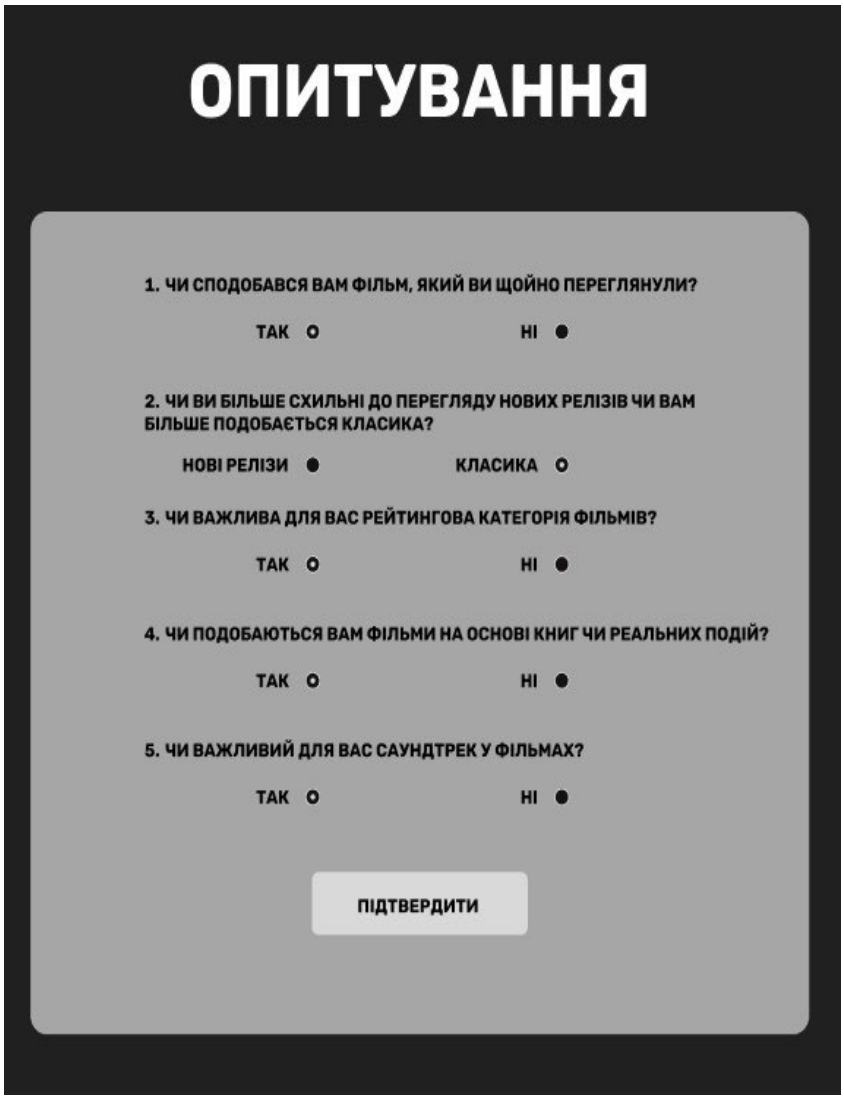

Рисунок 3.2 – Опитування користувача для першого тесту

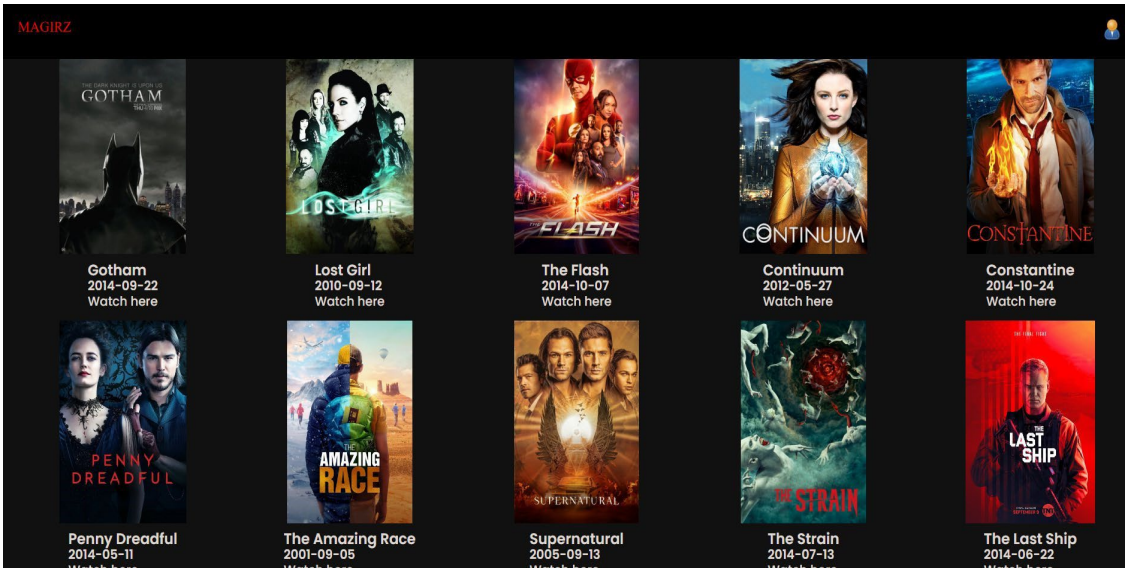

Рисунок 3.3 – Загальний вигляд інтерфейсного вікна рекомендованих фільмів, на основі фільтрації при першому тесті

Для другого тесту обрано наступні відповіді, які продемонстровано на рисунку 3.4.

Результатом роботи рекомендаційної системи є фільми з відповідними показниками, що показано на рисунку 3.5.

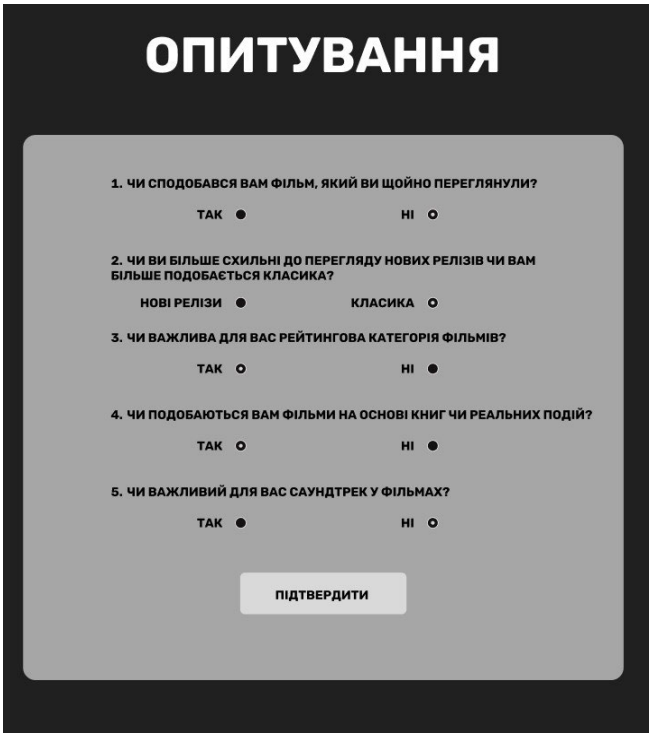

Рисунок 3.4 – Опитування користувача для другого тесту

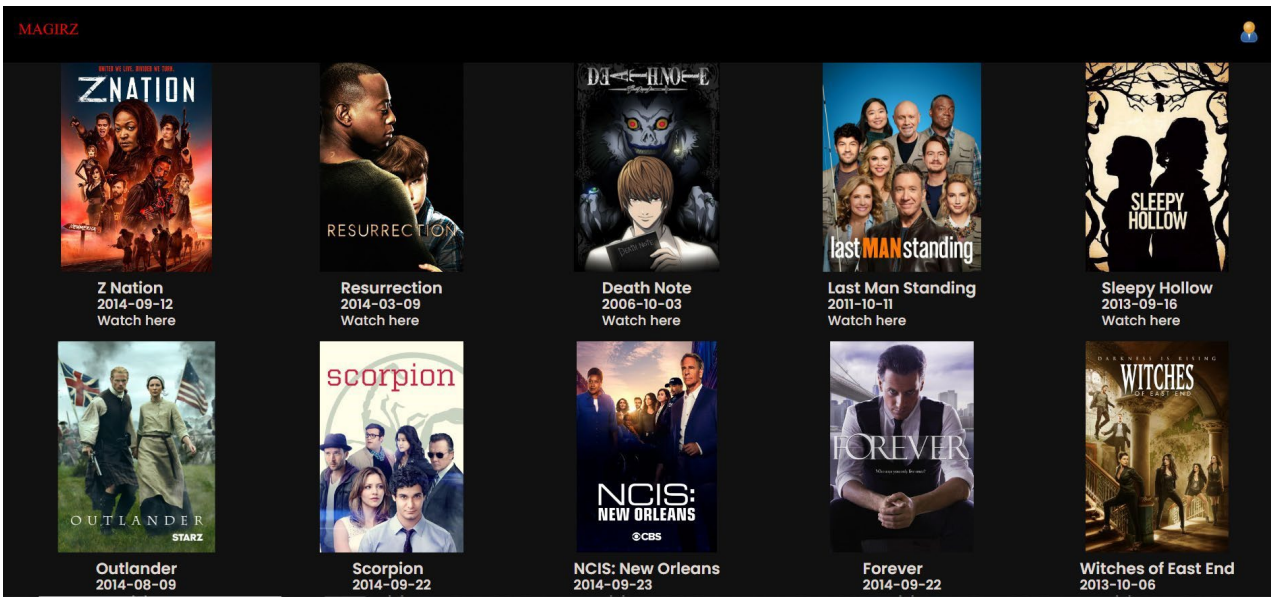

Рисунок 3.5 – Загальний вигляд інтерфейсного вікна рекомендованих фільмів, на основі фільтрації та другого тесту

Для третього тесту обрано наступні відповіді, які продемонстровано на рисунку 3.6.

Результатом роботи рекомендаційної системи є фільми з відповідними показниками, що показано на рисунку 3.7.

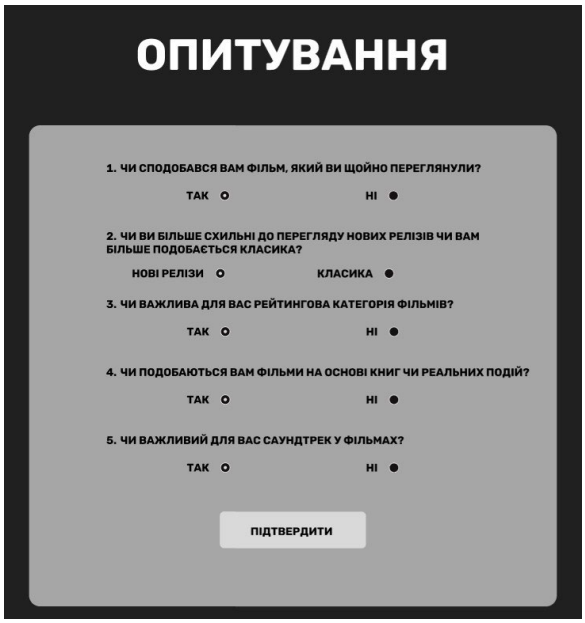

Рисунок 3.6 – Опитування користувача для третього тесту

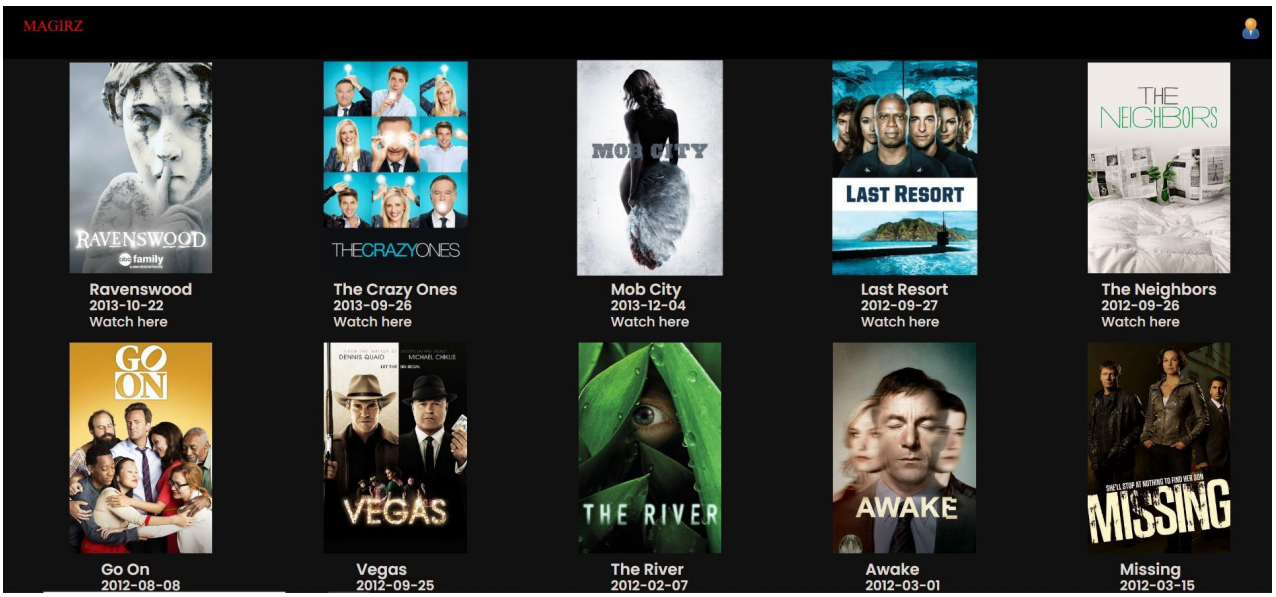

Рисунок 3.7 – Загальний вигляд інтерфейсного вікна рекомендованих фільмів, на основі фільтрації та третього тесту

Далі ми маємо можливість перейти до власного профілю (рис. 3.8), в якому обрати або вкладку уподобаного контенту (рис. 3.9), або вкладку з пошуком людей (рис. 3.10) для перегляду інформації про них з подальшим доданням їх у коло друзів.

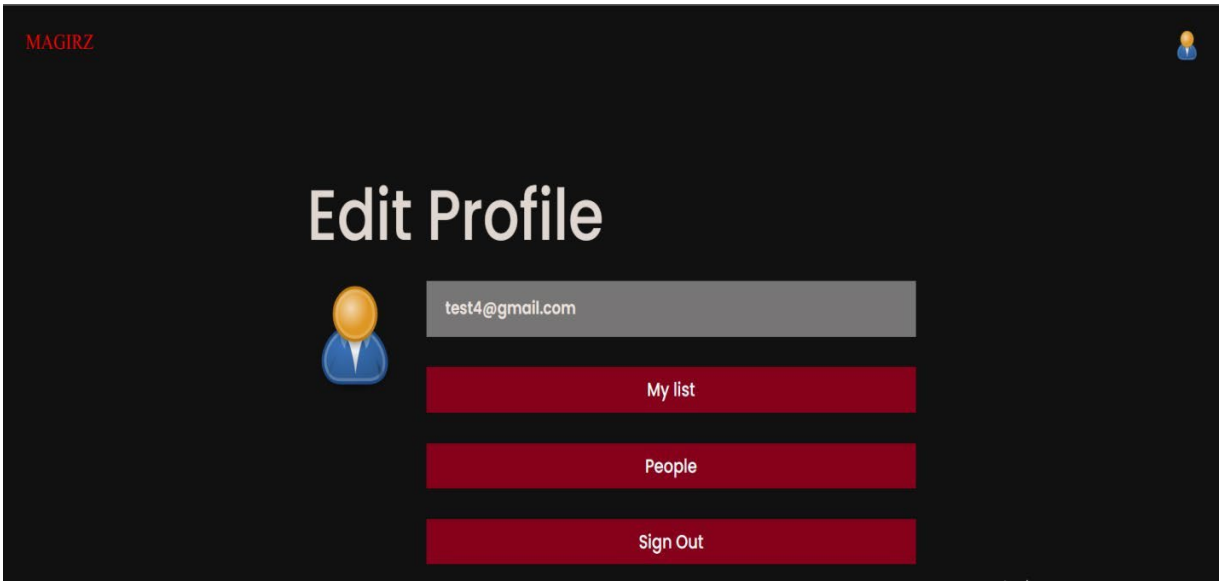

Рисунок 3.8 – Загальний вигляд інтерфейсного вікна коригування параметрів профілю користувача

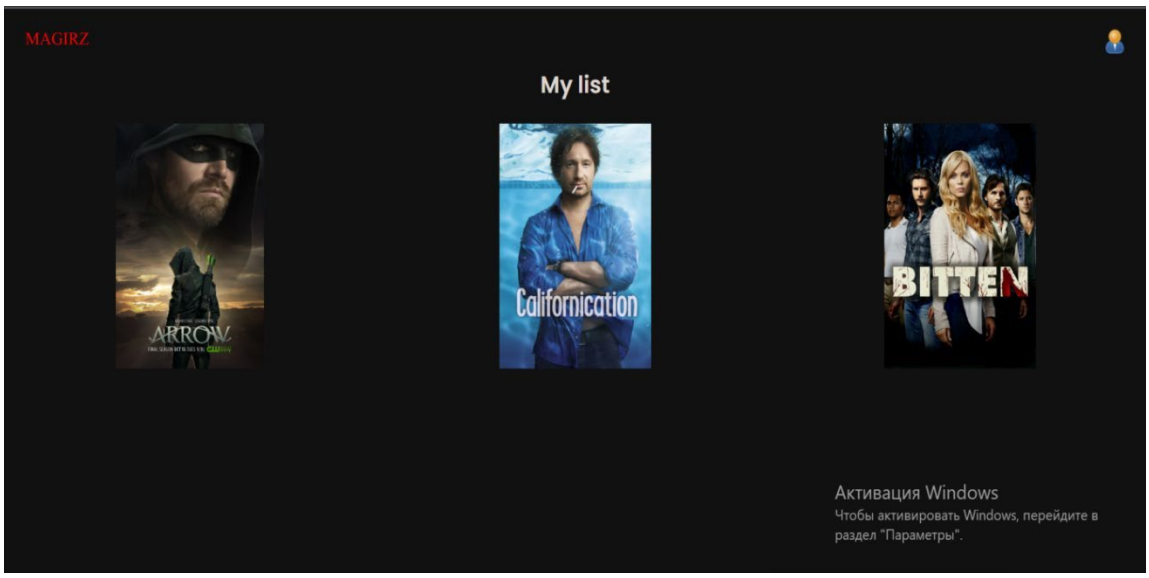

Рисунок 3.9 – Загальний вигляд інтерфейсного вікна сторінки уподобаних фільмів

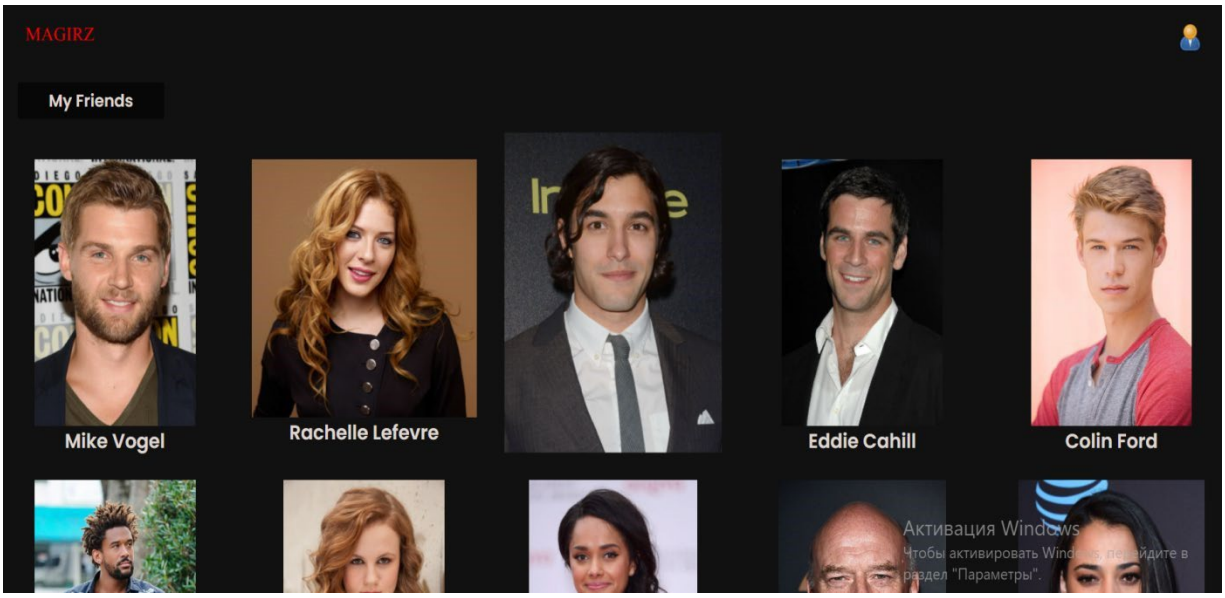

Рисунок 3.10 – Загальний вигляд інтерфейсного вікна сторінки акторів та людей

Для швидкого пошуку існує сторінка, на якій можна ввести назву або її частину та одразу знайти потрібний контент (рис. 3.11).

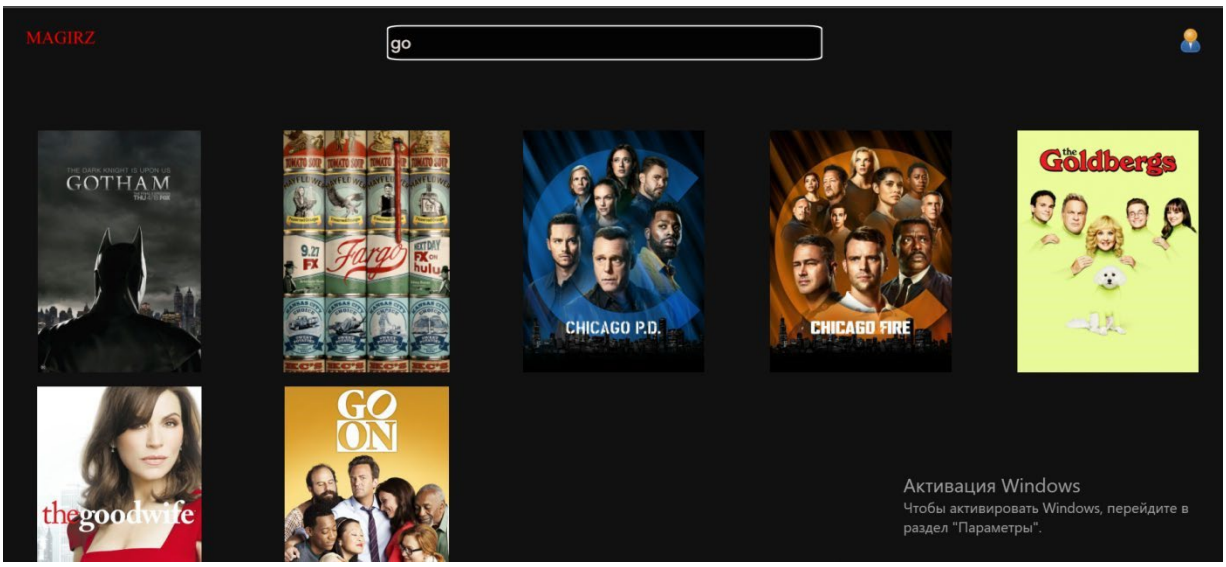

Рисунок 3.11 – Загальний вигляд інтерфейсного вікна сторінки пошуку фільмів

Проведемо порівняння функціональних можливостей програм стримінгового сервісу для фільмів (табл. 3.1)

Таблиця 3.1 – Аналіз функціональних можливостей програм стримінгового сервісу для фільмів

	Авторизація клієнта	Створення плейлисту по уподобанням	Додавання фільмів в колекцію	Опитування користувача	Додавання користувачів у «коло друзів»
NetFlix	+	+	+	-	-
YouTube	+	+	+	-	-
MEGOGO	+	+	+	-	-
Розроблене програмне забезпечення	+	+	+	+	+

З таблиці 3.1 видно, що розроблений програмний продукт володіє новими фунціональними можливостями: Опитування користувачів стримінгового сервісу; Додавання користувачів у «коло друзів».

3.4 Висновок до розділу 3

У третьому розділі проведено вибір мови програмування та середовища розробки, обґрунтовано доцільність їх використання, вибір фреймворків та компонентів, що є актуальним при розробці стримінгового сервісу для фільмів. Додаток проаналізовано згідно з функціональними характеристиками аналогів та проведено тестування його роботи, а також тестування нової функції надавання рекомендацій на основі фільтрації за контентом.

Було проведено тестування розробленого програмного забезпечення, що підтвердило розширення функціоналу додаванням двох функцій: Опитування користувачів стримінгового сервісу; Додавання користувачів у «коло друзів».

4 ЕКОНОМІЧНА ЧАСТИНА

4.1 Проведення комерційного та технологічного аудиту інформаційної технології стримінгового сервісу для фільмів

Метою проведення комерційного і технологічного аудиту є оцінювання науково-технічного рівня та рівня комерційного потенціалу розробки, створеної в результаті науково-технічної діяльності, тобто під час виконання магістерської кваліфікаційної роботи.

Для проведення комерційного та технологічного аудиту залучаємо 3-х незалежних експертів, якими є провідні викладачі кафедри КН.

Оцінювання науково-технічного рівня розробки та її комерційного потенціалу здійснюємо із застосуванням п’ятибальної системи оцінювання за 12-ма критеріями, а результати зводимо до таблиці 4.1.

Таблиця 4.1 - Результати оцінювання науково-технічного рівня і комерційного потенціалу інформаційної технології стримінгового сервісу для фільмів

Критерії	Експерти		
	Озеранський В.С.	Сілагін О.В.	Арсенюк І.Р.
	Бали, виставлені експертами		
1	2	3	4
Технічна здійсненність концепції	4	3	4
Ринкові переваги (наявність аналогів)	4	3	3
Ринкові переваги (ціна продукту)	4	4	3
Ринкові переваги (технічні властивості)	3	3	4
Ринкові переваги (експлуатаційні витрати)	3	4	3
Ринкові перспективи (розмір ринку)	3	4	4
Ринкові перспективи (конкуренція)	3	4	3
Практична здійсненність (наявність фахівців)	4	3	3
Практична здійсненність (наявність фінансів)	3	4	4
Практична здійсненність (необхідність нових матеріалів)	3	3	4

Продовження таблиці 4.1

1	2	3	4
Практична здійсненність (термін реалізації)	3	3	3
Практична здійсненність (розробка документів)	4	4	4
Сума балів	41	41	41
Середньоарифметична сума балів, СБ	41		

За результатами розрахунків, наведених в таблиці 1 робимо висновок про те, що науково-технічний рівень та комерційний потенціал інформаційної технології стримінгового сервісу для фільмів – високий.

4.2 Розрахунок витрат на здійснення науково-дослідної роботи розробки інформаційної технології стримінгового сервісу для фільмів

Витрати на оплату праці. Належать витрати на виплату основної та додаткової заробітної плати керівникам відділів, лабораторій, секторів і груп, науковим, інженерно-технічним працівникам, конструкторам, технологам, креслярам, копіювальникам, лаборантам, робітникам, студентам, аспірантам та іншим працівникам, безпосередньо зайнятим виконанням конкретної теми, обчисленої за посадовими окладами, відрядними розцінками, тарифними ставками згідно з чинними в організаціях системами оплати праці, також будь-які види грошових і матеріальних доплат, які належать до елемента «Витрати на оплату праці».

Основна заробітна плата дослідників. Витрати на основну заробітну плату дослідників ($З_о$) розраховують відповідно до посадових окладів працівників, за формулою:

$$З_о = \sum_{i=1}^{к} \frac{М_{ni} \cdot t_i}{Т_р}, \quad (4.1)$$

де к – кількість посад дослідників, залучених до процесу дослідження; $М_{ni}$ – місячний посадовий оклад конкретного розробника (інженера, дослідника, науковця

тощо), грн.; $Т_р$ – число робочих днів в місяці; приблизно $Т_р$ = (21…23) дні; $t_і$ – число робочих днів роботи розробника (дослідника).

Зроблені розрахунки зводимо до таблиці 4.2.

Таблиця 4.2 – Витрати на заробітну плату дослідників

Посада	Місячний посадовий оклад, грн.	Оплата за робочий день, грн.	Число днів роботи	Витрати на заробітну плату, грн.
Керівник	15000	714	24	17136
Розробник	13000	619	20	12380
Консультанти	14000	667	10	6670
Всього:	36186			

Основна заробітна плата робітників. Витрати на основну заробітну плату робітників ($З_р$) за відповідними найменуваннями робіт розраховують за формулою:

$$З_р = \sum_{і=1}^{n} C_і \cdot t_і, \quad (4.2)$$

де $С_і$ – погодинна тарифна ставка робітника відповідного розряду, за виконану відповідну роботу, грн/год; $t_і$ – час роботи робітника на виконання певної роботи, год.

Погодинну тарифну ставку робітника відповідного розряду С і можна визначити за формулою:

$$С_і = \frac{М_м \cdot К_і \cdot К_с}{Т_р \cdot t_{зм}}, \quad (4.3)$$

де $М_м$ – розмір прожиткового мінімуму працездатної особи або мінімальної місячної заробітної плати (залежно від діючого законодавства), у 2024 році $М_м$=8000 грн; $К_і$ – коефіцієнт міжкваліфікаційного співвідношення для встановлення тарифної ставки робітнику відповідного розряду; $К_с$ – мінімальний коефіцієнт співвідношень місячних тарифних ставок робітників першого розряду з нормальними умовами праці виробничих об’єднань і підприємств до законодавчо встановленого розміру

мінімальної заробітної плати; $Т_р$ – середня кількість робочих днів в місяці, приблизно $Т_р = 21…23$ дні; $t_{зм}$ – тривалість зміни, год.

Зроблені розрахунки зводимо до таблиці 4.3.

Таблиця 4.3 – Витрати на заробітну плату робітників

Найменування робіт	Трудомісткість, н-год.	Розряд роботи	Погодинна тарифна ставка	Тариф. коеф.	Величина, грн.
Програмування	120	10	86,7	1,82	10404
Налагодження	80	8	78,1	1,64	6248
Тестування	100	5	64,8	1,36	6480
Всього					23132

Додаткова заробітна плата. Додаткова заробітна плата $З_д$ всіх розробників та робітників, які брали участь у виконанні даного етапу роботи, розраховується як (10…12)% від суми основної заробітної плати всіх розробників та робітників, тобто:

$$З_д = 0{,}1 \cdot (З_о + З_р) = 0{,}1 \cdot (36186 + 23132) = 5931{,}8 \text{ грн.}$$

Відрахування на соціальні заходи. Нарахування на заробітну плату $Н_{зп}$ розробників та робітників, які брали участь у виконанні даного етапу роботи, розраховуються за формулою:

$$Н_{зп} = \beta \cdot (З_о + З_р + З_д) =$$
$$= 0{,}22 \cdot (1216545 + 68144 + 128469) = 14355 \text{ грн.}$$

де $З_о$ – основна заробітна плата розробників, грн.; $З_р$ – основна заробітна плата робітників, грн.; $З_д$ – додаткова заробітна плата всіх розробників та робітників, грн.; β – ставка єдиного внеску на загальнообов'язкове державне соціальне страхування, % (приймаємо для 1-го класу професійності ризику 22%).

Розрахунок витрат на матеріали. Витрати на матеріали М, що були використані під час виконання даного етапу роботи, розраховуються за формулою:

$$M = \sum_{1}^{n} Н_i \cdot Ц_i \cdot К_i, \quad (4.4)$$

де $Н_i$ – кількість матеріалів і-го виду, шт.; $Ц_i$ – ціна матеріалів і-го виду, грн.; $К_i$ – коефіцієнт транспортних витрат, $К_i$ = (1,1…1,15); n – кількість видів матеріалів.

Зроблені розрахунки зводимо до таблиці 4.4.

Таблиця 4.4 – Матеріали, що використані на розробку

Найменування комплектуючих	Ціна за одиницю, грн.	Витрачено	Вартість витрачених комплектуючих, грн.
Папір канцелярський офісний (А4)	250	1	250
FLASH-пам'ять (64 ГБ)	200	1	200
Всього, з врахуванням коефіцієнта транспортних витрат			495

Спецустаткування для наукових (експериментальних) робіт. Вартість спецустаткування визначається за прейскурантом гуртових цін або за даними базових підприємств за відпускними і договірними цінами.

$$В_{спец} = \sum_{1}^{к} Ц_i \cdot С_{пр.i} \cdot К_i, \quad (4.5)$$

де $Ц_i$ – ціна придбання спецустаткування і-го виду, грн.; $С_{пр.i}$ – кількість одиниць спецустаткування відповідного виду, шт.; $К_i$ – коефіцієнт транспортних витрат, $К_i$ = (1,1…1,15); n – кількість видів спецустаткування.

Зроблені розрахунки зводимо до таблиці 4.5.

Таблиця 4.5 – Витрати на придбання спецустаткування

Найменування спецустаткування	Ціна за одиницю, грн.	Витрачено	Вартість спецустаткування, грн.
Лазерна проекційна система Multi-GRAF1000	46000	1	46000
Всього, з врахуванням коефіцієнта транспортних витрат			50600

Програмне забезпечення. До балансової вартості програмного забезпечення входять витрати на його інсталяцію, тому ці витрати беруться додатково в розмірі 10…12% від вартості програмного забезпечення. Балансову вартість програмного забезпечення розраховують за формулою:

$$В_{прг} = \sum_{1}^{к} Ц_{іпрг} \cdot С_{прг.і} \cdot К_{і}, \quad (4.6)$$

де $Ц_{іпрг}$ – ціна придбання програмного забезпечення і-го виду, грн.; $С_{прг.і}$ – кількість одиниць програмного забезпечення відповідного виду, шт.; $К_{і}$ – коефіцієнт, що враховує інсталяцію, налагодження програмного забезпечення, $К_{і}$ = (1,1…1,12); к – кількість видів програмного забезпечення.

Зроблені розрахунки зводимо до таблиці 4.6.

Таблиця 4.6 – Витрати на придбання програмного забезпечення

Найменування програмного забезпечення	Ціна за одиницю, грн.	Витрачено	Вартість програмного забезпечення, грн.
Прикладний пакет Microsoft Office	7800	1	7800
Програмний засіб IDE - Visual Studio Code	10000	1	10000
Всього, з врахуванням коефіцієнта інсталяції та налагодження			19580

Амортизація обладнання. Амортизація обладнання, комп'ютерів та приміщень, які використовувались під час (чи для) виконання даного етапу роботи.

У спрощеному вигляді амортизаційні відрахування А в цілому бути розраховані за формулою:

$$А = \frac{Ц_б}{Т_в} \cdot \frac{t}{12}, \quad (4.7)$$

де $Ц_б$ – загальна балансова вартість всього обладнання, комп'ютерів, приміщень тощо, що використовувались для виконання даного етапу роботи, грн.; t – термін використання основного фонду, місяці; $Т_в$ – термін корисного використання основного фонду, роки.

Зроблені розрахунки зводимо до таблиці 4.7.

Таблиця 4.7 - Амортизаційні відрахування за видами основних фондів

Найменування	Балансова вартість, грн.	Строк корисного використання, років	Термін використання, місяців	Сума амортизації, грн.
Лазерна проекційна система Multi-GRAF1000	46000	5	2	1533
Програмно-аналітичний комплекс	20000	2	2	1667
Всього	3200			

Витрати на електроенергію для науково-виробничих цілей. Витрати на силову електроенергію $В_е$, якщо ця стаття має суттєве значення для виконання даного етапу роботи, розраховуються за формулою:

$$\text{Ве} = \sum \frac{W_i \cdot t_i \cdot \text{Ц}_\text{е} \cdot \text{К}_\text{впі}}{\text{ККД}} = \frac{0{,}35 \cdot 20 \cdot 7{,}5 \cdot 0{,}95}{0{,}98} + \frac{0{,}38 \cdot 160 \cdot 7{,}5 \cdot 0{,}95}{0{,}98} = 493 \text{ грн.},$$

W_i – встановлена потужність обладнання, кВт; t_i – тривалість роботи обладнання на етапі дослідження, год.; Це – вартість 1 кВт електроенергії, грн.; $К_{впі}$ – коефіцієнт використання потужності; ККД – коефіцієнт корисної дії обладнання.

Зроблені розрахунки зводимо до таблиці 4.8.

Таблиця 4.8 – Витрати на електроенергію

Найменування обладнання	Потужність, кВт	Тривалість годин роботи
Лазерна проекційна система Multi-GRAF1000	0.35	20
Програмно-аналітичний комплекс	0.38	160

Інші витрати. До статті «Інші витрати» належать витрати, які не знайшли відображення у зазначених статтях витрат і можуть бути віднесені безпосередньо на собівартість досліджень за прямими ознаками.

Витрати за статтею «Інші витрати» розраховуються як 50…100% від суми основної заробітної плати дослідників та робітників за формулою:

$$\text{Ів} = \left(З_\text{о} + З_\text{р}\right) \cdot \frac{\text{Н}_\text{ів}}{100\%} = (36186 + 23132) \cdot \frac{80}{100} = 47454 \text{ грн.},$$

де $Н_{ів}$ – норма нарахування за статтею «Інші витрати».

Накладні (загальновиробничі) витрати. До статті «Накладні (загальновиробничі) витрати» 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 за статтею «Накладні (загальновиробничі) витрати» розраховуються як 100…150% від суми основної заробітної плати дослідників та робітників за формулою:

$$\text{Внзв} = \left(З_о + З_р\right) \cdot \frac{Н_{нзв}}{100\%} = (36186 + 23132) \cdot \frac{140}{100} = 83045 \text{ грн.}, \qquad (4.8)$$

де $Н_{нзв}$ – норма нарахування за статтею «Накладні (загальновиробничі) витрати».

Витрати на проведення науково-дослідної роботи. Витрати на проведення науково-дослідної роботи розраховуються як сума всіх попередніх статей витрат за формулою:

$$В_{заг} = З_о + З_р + З_{дод} + З_н + К_в + В_{спец} + В_{прг} + А_{обл} + В_е + \qquad (4.9)$$
$$+ I_в + В_{нзв} =$$
$$= 36186 + 23132 + 5931{,}8 + 14355 + 495 + 50600 + 19580 + 3200 + 493 +$$
$$47454 + 83045 = 284472 \text{ грн.}$$

Загальні витрати. Загальні витрати ЗВ на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховуються за формулою:

$$ЗВ = \frac{В_{заг}}{\eta} = \frac{284472}{0{,}9} = 316080 \text{ грн.},$$

де η – коефіцієнт, що характеризує етап виконання науково-дослідної роботи. Оскільки, якщо науково-технічна розробка знаходиться на стадії технічного проектування, то η=0,9.

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

В ринкових умовах узагальнюючим 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.

В даному випадку відбувається розробка засобу, тому основу майбутнього економічного ефекту буде формувати: ΔN – збільшення кількості споживачів, яким надається відповідна інформаційна послуга в аналізовані періоди часу; N – кількість споживачів, яким надавалась відповідна інформаційна послуга у році до впровадження результатів нової науково-технічної розробки; $Ц_б$ – вартість послуги у році до впровадження інформаційної системи; $\pm\Delta Ц_о$ – зміна вартості послуги (зростання чи зниження) від впровадження результатів науково-технічної розробки в аналізовані періоди часу.

Можливе збільшення чистого прибутку у потенційного інвестора ΔПі для кожного із років, протягом яких очікується отримання позитивних результатів від можливого впровадження та комерціалізації науково-технічної розробки, розраховується за формулою:

$$\Delta П = (\pm\Delta Ц_о \cdot N + Ц_о \cdot \Delta N_i)_i \cdot \lambda \cdot \rho \cdot \left(1 - \frac{\vartheta}{100}\right), \qquad (4.10)$$

де ±ΔЦ – зміна основного якісного показника від впровадження результатів науково-технічної розробки в аналізованому році. Зазвичай, таким показником може бути зміна ціни реалізації одиниці нової розробки в аналізованому році (відносно

року до впровадження цієї розробки); $\pm\Delta Ц_о$ може мати як додатне, так і від'ємне значення (від'ємне – при зниженні ціни відносно року до впровадження цієї розробки, додатне – при зростанні ціни); N – основний кількісний показник, який визначає величину попиту на аналогічні чи подібні розробки у році до впровадження результатів нової науково-технічної розробки; $Ц_о$ – основний якісний показник, який визначає ціну реалізації нової науково-технічної розробки в аналізованому році; $Ц_б$ – основний якісний показник, який визначає ціну реалізації існуючої (базової) науково-технічної розробки у році до впровадження результатів; ΔN – зміна основного кількісного показника від впровадження результатів науково-технічної розробки в аналізованому році. Зазвичай таким показником може бути зростання попиту на науково-технічну розробку в аналізованому році (відносно року до впровадження цієї розробки); λ – коефіцієнт, який враховує сплату потенційним інвестором податку на додану вартість. У 2024 році ставка податку на додану вартість становить 20%, а коефіцієнт λ =0,8333; ρ – коефіцієнт, який враховує рентабельність інноваційного продукту (послуги). Рекомендується брати ρ =0,2...0,5; ϑ – ставка податку на прибуток, який має сплачувати потенційний інвестор, у 2024 році ϑ =18%.

Очікуваний термін життєвого циклу розробки 1 рік, тому:

$$\Delta П_1 = ((15000\text{-}500) \cdot 100 + 5000 \cdot 100) \cdot 0{,}83 \cdot 0{,}2 \cdot (1 - 0{,}18) = 204180 \text{ грн.}$$

Збільшення чистого прибутку 2-го року:

$$\Delta П_2 = ((15000\text{-}500) \cdot 100 + 5000 \cdot (100 + 70)) \cdot 0{,}83 \cdot 0{,}2 \cdot (1 - 0{,}18) =$$

$$=251822 \text{ грн.}$$

Збільшення чистого прибутку 3-го року:

$$\Delta П_3 = ((15000\text{-}500) \cdot 100 + 5000 \cdot (100 + 70 + 50)) \cdot 0{,}83 \cdot 0{,}2 \cdot (1 - 0{,}18) = =285852$$

грн.

Далі розраховують приведену вартість збільшення всіх чистих прибутків ПП, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-технічної розробки:

$$ПП = \sum_{i=1}^{T} \frac{\Delta П_i}{(1+\tau)^t} = \frac{204180}{(1+0{,}18)^1} + \frac{251822}{(1+0{,}18)^2} + \frac{285852}{(1+0{,}18)^3} = 527867 \text{ грн.},$$

де ΔП – збільшення чистого прибутку у кожному з років, протягом яких виявляються результати впровадження науково-технічної розробки, грн.; Т – період часу, протягом якого очікується отримання позитивних результатів від впровадження та комерціалізації науково-технічної розробки, роки (приймаємо Т=1 рік); τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, τ =0,05…0,15; t – період часу (в роках) від моменту початку впровадження науково-технічної розробки до моменту отримання потенційним інвестором додаткових чистих прибутків у цьому році.

Далі розраховують величину початкових інвестицій PV , які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки. Для цього можна використати формулу:

$$PV = k_{\text{інв}} \cdot ЗВ = 1 \cdot 316080 = 316080 \text{ грн.}$$

де $k_{\text{інв}}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію. Це можуть бути витрати на підготовку приміщень, розробку технологій, навчання персоналу, маркетингові заходи тощо; зазвичай кінв=1...5, але може бути і більшим; ЗВ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, грн.

Тоді абсолютний економічний ефект $Е_{абс}$ або чистий приведений дохід для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме:

$$Е_{абс} = ПП - PV = 527867 - 316080 = 211787 \text{ грн.},$$

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

Оскільки $Е_{абс}>0$, то можемо припустити про потенційну зацікавленість інвесторів у розробці.

Для остаточного прийняття рішення з цього питання необхідно розрахувати внутрішню економічну дохідність Ев або показник внутрішньої норми дохідності вкладених інвестицій та порівняти її з так званою бар'єрною ставкою дисконтування, яка визначає ту мінімальну внутрішню економічну дохідність, нижче якої інвестиції в будь-яку науково-технічну розробку вкладати буде економічно недоцільно.

Внутрішня економічна дохідність інвестицій Ев, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки інформаційної технології стримінгового сервісу для фільмів, розраховується за формулою:

$$Е_в = \sqrt[Т_ж]{1 + \frac{Еабс}{PV}} - 1 = \sqrt[3]{1 + \frac{211787}{316080}} - 1 = 0{,}29,$$

де $Т_ж$ – життєвий цикл розробки, роки.

Визначимо бар'єрну ставку дисконтування $\tau_{мін}$, тобто мінімальну внутрішню економічну дохідність інвестицій, нижче якої кошти у впровадження науково-технічної розробки та її комерціалізацію вкладатися не будуть.

Мінімальна внутрішня економічна дохідність вкладених інвестицій $\tau_{мін}$ визначається за формулою:

$$\tau_{мін} = d + f = 0{,}12 + 0{,}05 = 0{,}17,$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2024 році в Україні d =0,9...0,12; f – показник, що характеризує ризикованість вкладення інвестицій; зазвичай величина f =0,05...0,5, але може бути і значно вищою.

Оскільки $E_в$=0,29>$\tau_{мін}$=0,17, то потенційний інвестор може бути зацікавлений у фінансуванні впровадження науково-технічної розробки та виведенні її на ринок, тобто в її комерціалізації.

Далі розраховуємо період окупності інвестицій $T_о$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки інформаційної технології стримінгового сервісу для фільмів:

$$T_о = \frac{1}{E_в} = \frac{1}{0,29} = 3,4 \text{ року.}$$

Оскільки $T_о$<1…4-х років, то це свідчить про комерційну привабливість науково-технічної розробки інформаційної технології стримінгового сервісу для фільмів і може спонукати потенційного інвестора профінансувати впровадження цієї розробки та виведення її на ринок.

4.4 Висновки до розділу 4

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Інформаційна технологія стримінгового сервісу для фільмів» становить 41 бал, що, свідчить про комерційну важливість проведення даних досліджень.

Також термін окупності становить 3,4 роки, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

Отже можна зробити висновок про доцільність проведення науково-дослідної роботи за темою «Інформаційна технологія стримінгового сервісу для фільмів».

ВИСНОВКИ

У ході виконання магістерської кваліфікаційної роботи розроблено інформаційну технологію стримінгового сервісу для фільмів.

В роботі було виконано дослідження та аналіз сфери стримінгового сервісу для фільмів і об'єкта проектування. Були оглянуті існуючі аналоги систем, а також їх основні характеристики. Існуючі аналоги стримінгових сервісів для фільмів призначені для перегляду та пошуку контенту, але мають свої недоліки, такі як неможливість додавання друзів, обмежений доступ до короткої інформації та автобіографії, а також не дуже зручний інтерфейс. Розроблено модель інформаційної технології стримінгового сервісу для фільмів, модель вимог до технології та узагальнений алгоритм функіонування інформаційної технології стримінгового сервісу для фільмів. Визначено основні модулі розробленої системи. Детально описано доцільність використання технології Firebase для розробки даного сервісу. Розглянуто основні протоколи, що використовуються для стрімінгу медіаконтенту, такі як RTMP, MPEG-DASH, HLS, HDS, HTTP, HTTPS. Проаналізовано та досліджено клієнт-серверну архітектуру. Було проведено вибір мови програмування та середовища розробки, вибір фреймворків та компонентів, що є актуальним при розробці стримінгового сервісу для фільмів.

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Інформаційна технологія стримінгового сервісу для фільмів» становить 41 бал, що, свідчить про комерційну важливість проведення даних досліджень. Термін окупності становить 3,4 роки, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

Було проведено тестування розробленого програмного забезпечення, що підтвердило розширення функціоналу додаванням двох функцій на базі надавання рекомендацій на основі фільтрації за контентом: Опитування користувачів стримінгового сервісу; Додавання користувачів у «коло друзів».

Отже всі завданні виконано у повному обсязі, мету робото досягнуто.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Озеранський В.С., Сімончук С.В., Мельник К.І., Ярова О.А. Рекомендаційна система для стримі для стримінгового сервісу перегляду фільмів. Матеріали Міжнародної науково-практичної інтернет-конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2025)» − [Електронний ресурс]. – https://conferences.vntu.edu.ua/index.php/mn/mn2025/paper/viewFile/22905/18878
2. Від Netflix до «Такфлікс»: великий гід стримінговими сервісами. The Village Україна – [Електронний ресурс]. – Режим доступу: https://www.thevillage.com.ua/village/culture/tv/293007-tv-netflix-amazon-disney-plus-appletv-plus-hbo-max-peacock-streaming.
3. Файли і потоки, буферизація даних. StudFiles – [Електронний ресурс]. – Режим доступу: https://studfile.net/preview/7347055/page:2.
4. Krikke, J. «Streaming video transforms the media industry» (IEEE Computer Graphics and Applications). 2004, 537 p.
5. Speed for the streaming services – [Електронний ресурс]. – Режим доступу:https://www.allconnect.com/blog/how-much-speed-do-i-need-forstreaming.
6. Digital Rights Management – [Електронний ресурс]. – Режим доступу: https://www.britannica.com/topic/digital-rights-mana.
7. Netflix – [Електронний ресурс]. – Режим доступу: https://www.netflix.com/.
8. Hulu – [Електронний ресурс]. – Режим доступу: https://www.hulu.com/.
9. MEGOGO – [Електронний ресурс]. – Режим доступу: https://megogo.net/ua.
10. YouTube – [Електронний ресурс]. – Режим доступу: https://www.youtube.com/.
11. Клієнт-серверна архітектура та ролі серверів – [Електронний ресурс]. – Режим доступу: https://medium.com/@IvanZmerzlyi/клієнт-сервернаархітектура-та-ролі-серверів-9893d8048229.
12. Berson A. (1996). «Client/Server Architecture» (McGraw-Hill Computer Communications Series). 569 p.

13. Borrie H. «The Firebird Book: A Reference for Database Developers» (Apress) 2004, 1161p.
14. M. Morris Mano (1992). «Computer System Architecture» (Pearson), 544p.
15. Інформаційне наповнення сайту – [Електронний ресурс]. – Режим доступу: https://e-tk.lntu.edu.ua/mod/resource/ view.php?id=1200.
16. Firebase – [Електронний ресурс]. – Режим доступу: https://firebase.google.com/.
17. Firebase Authentication – [Електронний ресурс]. – Режим доступу: https://publisher.support.cleeng.com/hc/en-us/articles/4406603357074-FirebaseAuthentication.
18. Хмарне сховище Firebase – [Електронний ресурс]. – Режим доступу: https://firebase.google.com/docs/storage.
19. Аутентифікація і авторизація: що це і в чому відмінність – [Електронний ресурс] – Режим доступу: https://qagroup.com.ua/publications/autentyfikatciia-i-avtoryzatciia/.
20. Haverbeke M. «Eloquent JavaScript» (No Starch Press) 2018, 472p.
21. 10 Best JavaScript Frameworks to Use in 2022 – [Електронний ресурс]. – Режим доступу: https://hackr.io/blog/best-javascript-frameworks.
22. Smith, J. «Streaming Technologies and Their Impact on the Film Industry» (International Journal of Film Studies), 2020, 635p.
23. Wang, L. «User Privacy and Data Protection in Streaming Services: Legal and Ethical Considerations» (Journal of Information Privacy), 2020, 1115p.
24. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт / Уклад. : В. О. Козловський, О. Й. Лесько, В. В. Кавецький. – Вінниця : ВНТУ, 2021. – 42 с.
25. Методичні вказівки до виконання магістерських кваліфікаційних робіт для студентів спеціальності 122 «Комп’ютерні науки» [Електронний ресурс] / уклад.: А. А. Яровий, О. К. Колесницький. – Вінниця : ВНТУ, 2023. – (58 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 стримінгового сервісу для фільмів

Тип роботи: магістерська кваліфікаційна робота
(БДР, МКР)

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

Показники звіту подібності Turnitin

Оригінальність 81,00% Схожість 19,00%

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 Turnitin щодо роботи.

Автор роботи _____ Мельник К.І.

Керівник роботи _____ Озеранський В.С.

Опис прийнятого рішення

Магістерську кваліфікаційну роботу допущено до захисту

Особа, відповідальна за перевірку _____ Озеранський В.С.

Додаток Б (обов'язковий)

Фрагмент лістингу програмного коду

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1"
/>
    <meta name="theme-color" content="#000000" />
    <link                                                    rel="stylesheet"
href="https://use.fontawesome.com/releases/v5.15.4/css/all.css"
integrity="sha384-
DyZ88mC6Up2uqS4h/KRgHuoeGwBcD4Ng9SiP4dIRy0EXTlnuz47vAw
meGwVChigm" crossorigin="anonymous">
    <link rel="preconnect" href="https://fonts.googleapis.com">
    <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
    <link
href="https://fonts.googleapis.com/css2?family=Poppins:wght@100;300;
500;600&display=swap" rel="stylesheet">
    <title>Netflix clone</title>
  </head>
  <body>
    <noscript>You need to enable JavaScript to run this app.</noscript>
    <div id="root"></div>
  </body> </html>
import React, { useState } from "react"; import
"./Loginpage.css"; import SignUpPage from
"../SignUpPage/SignUpPage";

const Loginpage = () => {   const [signIn,
setSignIn] = useState(false);   return (
    <div className="loginPage">
      <div className="loginPage__background">
        <img
          className="loginPage__logo"
          src="logo.png"
alt=""         />
        <button className="loginPage__btn" onClick={() => setSignIn(true)}>
          Sign in
        </button>
        <div className="loginPage__gradient" />
      </div>
      <div className="loginPage__body">
        {signIn ? (
          <SignUpPage />
```

```
      ) : (
        <>
          <h1>Unlimited films, TV programmes and more.</h1>
          <h2>Watch anywhere. Cancel at any time.</h2>
          <h3>
            Ready to watch? Enter your email to create or restart your
membership.
          </h3>

          <div className="loginPage__input">
            <form>
              <input type="email" placeholder="Email Address" />
              <button
                className="loginPage__getStarted"
                onClick={() => setSignIn(true)}
              >
                GET STARTED
              </button>
            </form>
          </div>
        </>
      )}
    </div>
  </div>
 );
};

export default Loginpage; import React from "react"; import
"normalize.css"; import "./Homepage.css"; import HomeNav from
"../../components/HomeNav/HomeNav"; import Banner from
"../../components/Banner/Banner"; import Row from
"../../components/Row/Row";
import requests from "../../config/requests/requests";

const Homepage = () => {


 return (
  <div className="homePage">
    <HomeNav />

    <Banner />
    <Row fetchUrl={requests.fetchTrending} />
  </div>
 );
};
```

```
export default Homepage; import React from 'react';
import { useSelector } from "react-redux"; import {
useHistory } from "react-router-dom"; import
"./ListPage.css";
import Nav from "../../components/Nav/Nav";

const ListPage = () => {     const user = useSelector((state)
=> state.user.user);     const history = useHistory();     const
movies = user?.movie;
   const showMovies = (movies) => {
let shows = [];         for(let el in movies) {
shows.push(movies[el].movie);
       }
       console.log(shows);
       return shows;
   };

   return (
      <div className="listPage">
         <Nav />
         <h2>My list</h2>
         <div className="liked">
            {showMovies(movies).map(movie => (
               <img
               onClick={() => history.push("/movie/" + movie.id)}
               className="list__poster"
key={movie.id}
               src={
                 movie?.image?.original
                   ? movie?.image?.original
                   : movie?.show?.image?.medium
               }
               alt={movie.name}
              />
))}
         </div>
      </div>
   );
}

export default ListPage;
import React, { useState, useEffect } from "react"; import
"./MoviePage.css"; import axios from
"../../config/axios/axios"; import Nav from
"../../components/Nav/Nav"; import { useDispatch } from
"react-redux"; import requests from
"../../config/requests/requests";
import { liked, removeMovie } from "../../features/userSlice";
```

```
const MoviePage = () => {   const url =
window.location.href;   const ID =
url.substr(url.lastIndexOf("/") + 1);   const [movies,
setMovies] = useState([]);   const dispatch =
useDispatch();
  const add = document.querySelector(".add");
  const move = document.querySelector(".move");

  useEffect(() => {     async function fetchData() {       const
request = await axios.get(requests.fetchTrending);
setMovies(request.data);       return request;
    }     fetchData();
  }, []);

  const exactMovie = [movies.filter((movie) => movie?.id.toString() ===
ID)];

  let description = exactMovie[0][0]?.summary;
  if (description) {     description = description.replace(/<\/?[a-zA-
Z]+>/gi, "");
  }

  function truncate(description, n) {     return description?.length > n ?
description.substr(0, n - 1) + "..." :
description;
  }

  function handleAdd(event){     event.preventDefault();
    if(add){         add.setAttribute("hidden",
"hidden");
       move.removeAttribute("hidden", "hidden");
    }
  }

  function handleRemove(event){
event.preventDefault();     if(move){
move.setAttribute("hidden", "hidden");
       add.removeAttribute("hidden", "hidden")
    }
  }

  return (
    <div className="moviePage">
      <Nav />
      <div className="movie__container">
        <div className="movie__img">
```

```
        <img src={exactMovie[0][0]?.image?.original} alt="" />
      </div>
      <div className="movie__info">
        <h1 className="movie__name">{exactMovie[0][0]?.name}</h1>
        <h2 className="movie__genres">
          {exactMovie[0][0]?.genres.join(" ")}
        </h2>
        <div            className="movie__summary">{truncate(description,
500)}</div>
        <button className="movie__button">
          <a href={exactMovie[0][0]?.url}>
            Play
          </a>
        </button>
        <button
          onMouseDown={() => {
            dispatch(
              liked({                movie:
exactMovie[0][0]
              })
            )
}}
          onClick={handleAdd}
className="movie__button add">My list</button>
          <button
            hidden
            onMouseDown={() => {
              dispatch(
removeMovie({                  moie:
exactMovie[0][0],
                })
              );
            }}
            onClick={handleRemove}
            className="movie__button move"
          >
            Remove
          </button>
      </div>
    </div>
  </div>
 );
};

export default MoviePage; import React from
'react'; import "./MyFriendsPage.css"; import Nav
from "../../components/Nav/Nav"; import {
useSelector } from "react-redux";
```

```
import { useHistory } from "react-router-dom";

const MyFriendsPage = () => {     const user =
useSelector((state) => state.user.user);     const history =
useHistory();     const noAvatar =

"https://upload.wikimedia.org/wikipedia/commons/9/9a/No_avatar.png";     const
friends = user?.person;
   const showFriends = (friends) => {
      let people = [];         for(let
el in friends) {
         people.push(friends[el].person);
      }
      console.log(people);         return
people;
   };

   return (
      <div className="myFriendsPage">
         <Nav />
         <h2 className="myFriendsPage__header">My friends</h2>
         <div className="followed">
            {showFriends(friends).map(person => (
               <img
               onClick={() => history.push("/person/" + person.id)}
className="followed__poster"
               key={person.id}
               src={
                 person?.image?.original
                   ? person?.image?.original
                   : noAvatar
               }
               alt={person.name}
              />
))}
         </div>
      </div>
   );
}

export default MyFriendsPage; import React, { useState,
useEffect } from "react"; import "./PeoplePage.css";
import Nav from "../../components/Nav/Nav"; import {
useHistory } from "react-router-dom"; import axios from
"../../config/axios/axios";
import requests from "../../config/requests/requests";
```

```
const PeoplePage = () => {   const history =
useHistory();   const [people, setPeople] =
useState([]);   const noAvatar =

"https://upload.wikimedia.org/wikipedia/commons/9/9a/No_avatar.png";

  useEffect(() => {     async function fetchData() {       const
request = await axios.get(requests.fetchPeople);
setPeople(request.data);       console.log(request);
     return request;
   }     fetchData();
  }, []);

  return (
   <div className="peoplePage">
    <Nav />

    <div className="btn">
     <button
      className="people__button"
      onClick={() => history.push("/myFriends")}
     >
      My Friends
     </button>
    </div>
    <div className="people__posters">
     {people.map((people) => (
      <div className="people__container">
       <img
        onClick={() => history.push("/person/" + people.id)}
className="search__poster"
         key={people.id}              src={people?.image?.original     ?
people?.image?.original         :
noAvatar}
         alt={people.name}
        />
        <div className="people__info">
          <h3 className="people__name">
            {people?.name}
          </h3>
        </div>
       </div>
     ))}
    </div>
   </div>
  );
};
```

```
export default PeoplePage; import React, { useState,
useEffect } from "react"; import "./PersonPage.css";
import axios from "../../config/axios/axios"; import Nav
from "../../components/Nav/Nav"; import { useDispatch }
from "react-redux"; import requests from
"../../config/requests/requests";
import { followed, remove } from "../../features/userSlice";

const PersonPage = () => {   const url =
window.location.href;   const ID =
url.substr(url.lastIndexOf("/") + 1);   const [people,
setPeople] = useState([]);   const dispatch =
useDispatch();   const add =
document.querySelector(".add");
  const move = document.querySelector(".move");

  useEffect(() => {     async function fetchData() {       const
request = await axios.get(requests.fetchPeople);
setPeople(request.data);
     return request;
    }
    fetchData();
  }, []);

  const exactPerson = [people.filter((person) => person?.id.toString() === ID)];

  function handleAdd(event){     event.preventDefault();
    if(add){         add.setAttribute("hidden",
"hidden");
       move.removeAttribute("hidden", "hidden");
    }
  }

  function handleRemove(event){
event.preventDefault();     if(move){
move.setAttribute("hidden", "hidden");
       add.removeAttribute("hidden", "hidden")
    }
  }

          const SurveyForm = () => {
            const [formData, setFormData] = useState({
              question1: 'Чи сподобався вам фільм, який ви щойно переглянули? ',
              question2: 'Чи ви більше схильні до перегляду нових релізів чи вам більше
подобається класика?',     question3: 'Чи важлива для вас рейтингова категорія фільмів?',
          question4: 'Чи подобаються вам фільми на основі книг чи реальних подій?',
          question5: 'Чи важливий для вас саундтрек у фільмах?',
            });
```

```
  const handleInputChange = (e) => {     const {
name, value } = e.target;
    setFormData({
      ...formData,
      [name]: value,
    });
  };
  const handleSubmit = (e) => {     e.preventDefault();
    console.log('Form submitted:', formData);
  };
  return (
    <form onSubmit={handleSubmit}>
      <label>         Question 1:         <input
type="text"           name="question1"
value={formData.question1}
          onChange={handleInputChange}
        />
      </label>
      <br />       <label>         Question 2:
<input           type="text"
name="question2"
value={formData.question2}
          onChange={handleInputChange}
        />
      </label>
      <br />       <label>         Question 3:
<input           type="text"
name="question3"
value={formData.question3}
          onChange={handleInputChange}
        />
      </label>
      <br />       <label>         Question 4:
<input           type="text"
name="question4"
value={formData.question4}
          onChange={handleInputChange}
        />
      </label>
      <br />       <label>         Question 5:
<input           type="text"
name="question5"
value={formData.question5}
          onChange={handleInputChange}
        />
      </label>
      <br />
      <button type="submit">Підтвердити</button>
    </form>
```

```
          );
        };   console.log(exactPerson);
  return (
    <div className="personPage">
      <Nav />
      <div className="person__container">
        <div className="person__img">
          <img src={exactPerson[0][0]?.image?.original} alt="" />
        </div>
        <div className="person__info">
          <h1                                  className="person__name">Name:
{exactPerson[0][0]?.name}</h1>
          <h2 className="person__birthday">
            Birthday: {exactPerson[0][0]?.birthday}
          </h2>
          <h2 className="person__country">
            Country: {exactPerson[0][0]?.country?.name}
          </h2>
          <h3
className="person__gender">{exactPerson[0][0]?.gender}</h3>
          <button className="person__button">
            <a href={exactPerson[0][0]?.url}>Look</a>
          </button>
            <button
              onMouseDown={() => {
                dispatch(                 followed({
person: exactPerson[0][0],
                  })
                );
              }}
              onClick={handleAdd}
className="person__button add"
            >
              Add friend
            </button>
<button
              hidden
              onMouseDown={() => {
                dispatch(                 remove({
person: exactPerson[0][0],
                  })
                );
              }}
              onClick={handleRemove}
              className="person__button move"
            >
              Remove
            </button>
```

```
      </div>
     </div>
    </div>
  ); };
export default PersonPage; import React
from "react";
import { useSelector } from "react-redux"; // import {
selectUser } from "../../features/userSlice"; import { auth }
from "../../config/firebase/firebase"; import { useHistory }
from "react-router-dom"; import Nav from
"../../components/Nav/Nav";
import "./ProfilePage.css"; const ProfilePage = () => {
const user = useSelector((state) => state.user.user);   const
history = useHistory();   return (
   <div className="profilePage">
     <Nav />
     <div className="profilePage__body">
       <h1>Edit Profile</h1>
       <div className="profilePage__info">
         <img
src="https://pbs.twimg.com/profile_images/1240119990411550720/hBEe
3tdn_400x400.png"
           alt=""
         />
         <div className="profilePage__details">
           <h2>{user.email}</h2>
           <div className="profilePage__inners">
           <button
              onClick={() => history.push("/myList")}
              className="profilePage__myList profilePage__button">
              My list
            </button>
<button
              onClick={() => history.push("/people")}
              className="profilePage__friends profilePage__button">
              People
            </button>
<button
              onClick={() => auth.signOut()}
className="profilePage__signout profilePage__button">
              Sign Out
            </button>
          </div>
        </div>
      </div>
    </div>
  ); };
export default ProfilePage;
```

```
import React, { useState, useEffect } from "react"; import
"./SearchPage.css"; import axios from
"../../config/axios/axios"; import requests from
"../../config/requests/requests"; import { useHistory } from
"react-router-dom"; const SearchPage = () => {   const
[movies, setMovies] = useState([]);   const [show,
handleShow] = useState(false);   const history =
useHistory();   const [value, setValue] = useState("");
const transitionNavbar = () => {     if (window.scrollY >
100) {       handleShow(true);
   } else {
     handleShow(false);
   }
 };
 useEffect(() => {
   window.addEventListener("scroll", transitionNavbar);
   return () => window.removeEventListener("scroll", transitionNavbar);
 }, []);
 const filteredMovies = movies.filter((movie) => {
   return movie.name.toLowerCase().includes(value.toLowerCase());
 });   useEffect(() => {     async function fetchData() {       const
request = await axios.get(requests.fetchTrending);
setMovies(request.data);       console.log(request);       return
request;
   }     fetchData();   },
[]);
 return (
   <div className="searchPage">
     <div className={`nav ${show && `nav__black`}`}>
       <div className="nav__contents">
         <img            onClick={() =>
history.push("/")}
           className="nav__logo"

src="https://assets.stickpng.com/images/580b57fcd9996e24bc43c529.png
"
           alt="logo"
         />
         <input
           onChange={(event) => setValue(event.target.value)}
className="search__input"            placeholder="Search..."
           type="text"
         />
         <img
           onClick={() => history.push("/profile")}
className="nav__avatar"

src="https://pbs.twimg.com/profile_images/1240119990411550720/hBEe
3tdn_400x400.png"
```

```
          alt="user logo"
        />
      </div>
    </div>
    <div className="search__posters">
      {filteredMovies.map((movie) => (
        <img
          onClick={() => history.push("/movie/" + movie.id)}
className="search__poster"
          key={movie.id}
          src={
            movie?.image?.original
              ? movie?.image?.original
              : movie?.show?.image?.medium
          }
          alt={movie.name}
        />
      ))}
    </div>
  </div>
 );
};

export default SearchPage; import React, { useRef }
from "react"; import { auth } from
'../../config/firebase/firebase'
import "./SignUpPage.css"; const SignUpPage = ()
=> {     const emailRef = useRef(null);     const
passwordRef = useRef(null);     const register = (e)
=> {         e.preventDefault();
auth.createUserWithEmailAndPassword(
          emailRef.current.value,
passwordRef.current.value
).then((authUser) => {
console.log(authUser);         }).catch(err => {
          alert(err.message);
       });
   };

   const signIn = (e) => {         e.preventDefault();
auth.signInWithEmailAndPassword(
emailRef.current.value,
passwordRef.current.value         ).then((authUser)
=> {             console.log(authUser);
}).catch(err => {
          alert(err.message);
       });     };
 return (
   <div className="signUpPage">
```

```
    <form>
      <h1>Sign In</h1>
      <input type="email" ref={emailRef} placeholder="Email" />
      <input type="password" ref={passwordRef} placeholder="Password" />
      <button type="submit" onClick={signIn}>
        Sign in
      </button>
      <h4>
        <span className="signUpPage__gray">New to Netflix? </span>
        <span className="signUpPage__link" onClick={register}>
Sign Up now.
        </span>
      </h4>
    </form>
  </div>
 );
};

export default SignUpPage; import React, { useState, useEffect }
from "react"; import "./Banner.css"; import axios from
"../../config/axios/axios"; import requests from
"../../config/requests/requests"; import { useDispatch } from "react-
redux"; import {  liked, removeMovie } from
"../../features/userSlice";

const Banner = () => {   const [movie, setMovie] =
useState([]);   const dispatch = useDispatch();   const
add = document.querySelector(".add");
 const move = document.querySelector(".move");

 useEffect(() => {     async function fetchData() {       const request = await
axios.get(requests.fetchTrending);       setMovie(
request.data[Math.floor(Math.random() * request.data.length - 1)]
    );
    return request;
  }

  fetchData();
 }, []);

 let description = movie?.summary;
 if (description) {     description = description.replace(/<\/?[a-zA-
Z]+>/gi, "");
 }

 function truncate(string, n) {     return string?.length > n ?
string.substr(0, n - 1) + "..." : string;   }
```

```
  function handleAdd(event){     event.preventDefault();
    if(add){         add.setAttribute("hidden",
"hidden");
       move.removeAttribute("hidden", "hidden");
    }
  }

  function handleRemove(event){
event.preventDefault();     if(move){
move.setAttribute("hidden", "hidden");
       add.removeAttribute("hidden", "hidden")
    }
  }

  return (     <header
className="banner"
     style={{         backgroundImage:
`url(${movie?.image?.original})`,         backgroundSize: "90%",
       backgroundPosition: "center center",
     }}
   >
     <div className="banner__contents">
       <h1 className="banner__title">{movie?.name}</h1>
       <div className="banner__buttons">
         <button className="banner__button">
           <a href={movie?.url}>
             Play
           </a>
         </button>
         <button
           onMouseDown={() => {
             dispatch(
liked({
                 movie: movie,
               })
             )

           }}
           onClick={handleAdd}
className="banner__button add">My list</button>
           <button
             hidden
             onMouseDown={() => {
               dispatch(
removeMovie({
                   moie: movie,
                 })
               );
```

```
            }}
            onClick={handleRemove}
className="banner__button move"
          >
            Remove
          </button>        </div>
      <h1            className="banner__description">{truncate(description,
200)}</h1>
    </div>
    <div className="banner__fadeBottom" />     </header>
  );
};

export default Banner;
const Button = (props) => {

   const {          value,
func
   } = props;

   return (
      <button className="btn" onClick={() => func}>
         {value}
      </button>
   );
}

export default Button; import React, { useState,
useEffect } from "react"; import { useHistory } from
"react-router-dom"; import "./HomeNav.css";

const HomeNav = ({updateData}) => {   const
[show, handleShow] = useState(false);   const
history = useHistory();

  const transitionNavbar = () => {     if
(window.scrollY > 100) {
handleShow(true);
   } else {
     handleShow(false);
   }
  };
  useEffect(() => {
    window.addEventListener("scroll", transitionNavbar);
    return () => window.removeEventListener("scroll", transitionNavbar);
  }, []);   return (
    <div className={`nav ${show && `nav__black`}`}>
      <div className="nav__contents">
```

```
      <img          onClick={() =>
history.push("/")}
        className="nav__logo"
        src="../"          alt="logo"
      />
      <ul>
        <li>
          <i
            onClick={() => history.push("/search")}
            className="fas fa-search"></i>
        </li>
      </ul>        <img
        onClick={() => history.push("/profile")}          className="nav__avatar"

src="https://pbs.twimg.com/profile_images/1240119990411550720/hBEe
3tdn_400x400.png"
        alt="user logo"
      />
    </div>
  </div>
 );
};

export default HomeNav; import React, { useState,
useEffect } from "react"; import { useHistory } from
"react-router-dom"; import "./Nav.css";

const Nav = () => {   const [show, handleShow] =
useState(false);   const history = useHistory();

 const transitionNavbar = () => {     if
(window.scrollY > 100) {
handleShow(true);
   } else {       handleShow(false);
}
 };
 useEffect(() => {
   window.addEventListener("scroll", transitionNavbar);
   return () => window.removeEventListener("scroll", transitionNavbar);
 }, []);   return (
   <div className={`nav ${show && `nav__black`}`}>
     <div className="nav__contents">
       <img          onClick={() =>
history.push("/")}
         className="nav__logo"

src="https://assets.stickpng.com/images/580b57fcd9996e24bc43c529.png
"
```

```
        alt="logo"
       />
       <img
        onClick={() => history.push("/profile")}          className="nav__avatar"
src="https://pbs.twimg.com/profile_images/1240119990411550720/hBEe
3tdn_400x400.png"
        alt="user logo"
       />
     </div>
   </div>
  );
};

export default Nav; import React, { useState, useEffect }
  return (
   <div className="row">
    <div className="row__posters">


      {movies.map((movie) => (
       <div      className="card__container"          onClick={()    =>
history.push(`/movie/${movie?.id ? movie?.id : movie?.show?.id}`)}>
         <img
         className="row__poster"           key={movie?.id ? movie?.id :
movie?.show?.id}           src={movie?.image?.original       ?
movie?.image?.original          :
movie?.show?.image?.medium}
         alt={movie.name}
        />
         <div className="card__info">
          <h3 className="card__name">{movie?.name ? movie?.name :
movie?.show?.name}</h3>
           <h4    className="card__date">{movie?.premiered          ?
movie?.premiered : movie?.show?.premiered}</h4>
           <a      href={movie?.url       ?         movie?.url      :
movie?.show?.url} className="card__site">Watch here</a>
         </div>
       </div>
      ))}
    </div>
   </div>
  );
};

export default Row;
```

Додаток В (обов'язковий)

ІЛЮСТРАТИВНА ЧАСТИНА

«ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ СТРИМІНГОВОГО СЕРВІСУ ДЛЯ ФІЛЬМІВ»

Виконала: студентка 2-го курсу, групи 2КН-23м
спеціальності 122 – «Комп'ютерні науки»
(шифр і назва напряму підготовки, спеціальності)

Мельник К.І.
(прізвище та ініціали)

Керівник: к.т.н., доцент каф. КН

Озеранський В. С.
(прізвище та ініціали)

«05» 12 2024 р.

Вінниця ВНТУ - 2024 рік

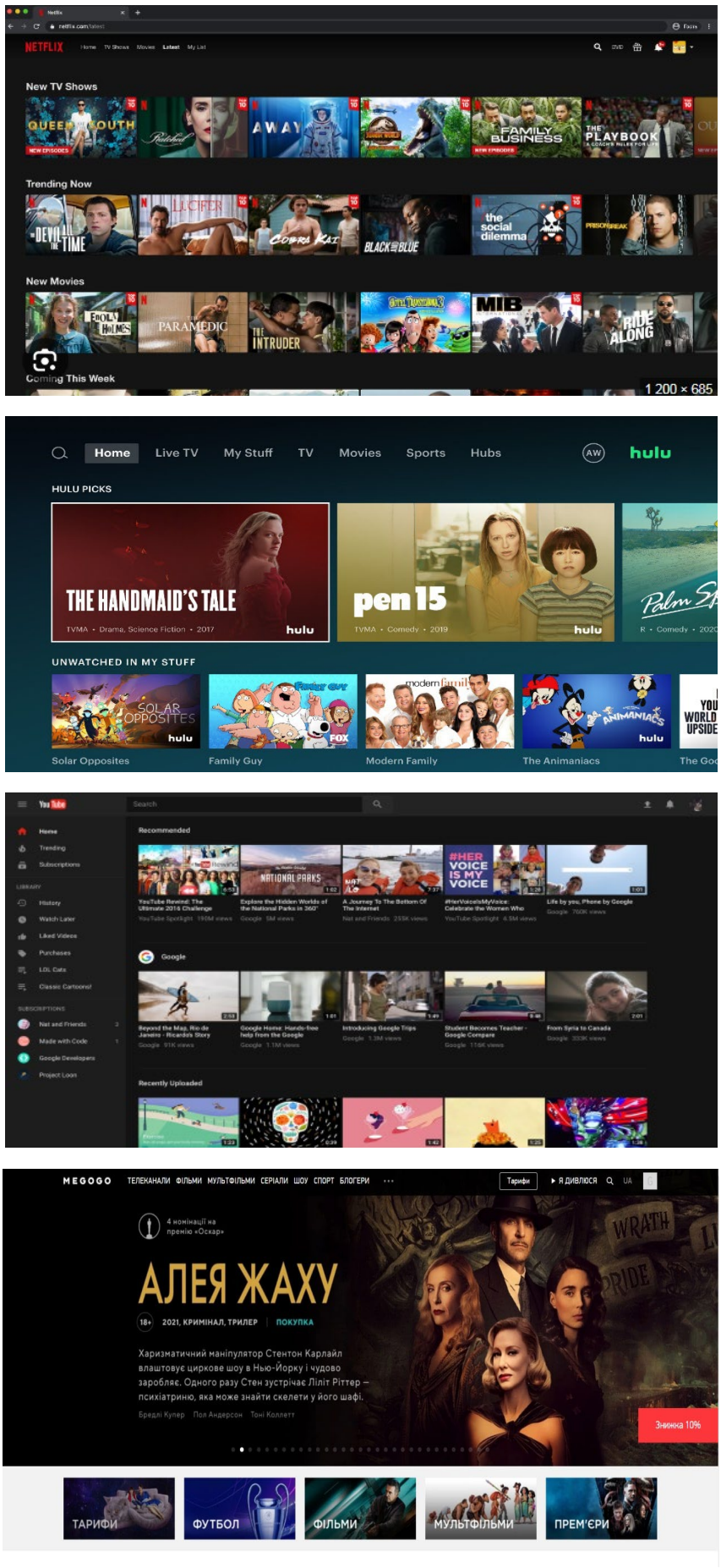

Рисунок В.1 – Порівняння систем-аналогів стримінгових сервісів для фільмів

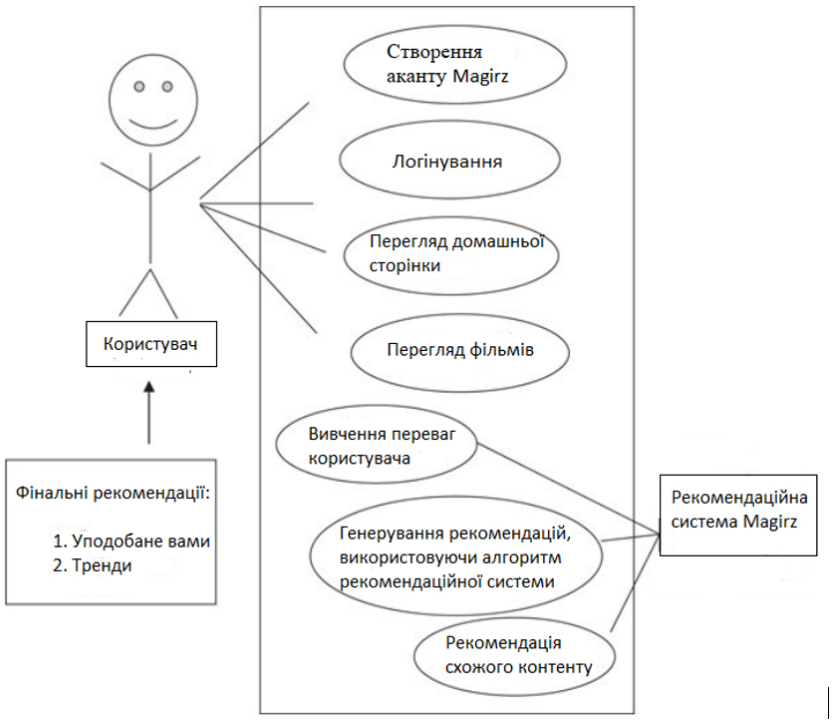

Рисунок В.2 – Модель інформаційної технології стримінгового сервісу для фільмів

Рисунок В.3 – Вимоги до стримінгового сервісу для фільмів

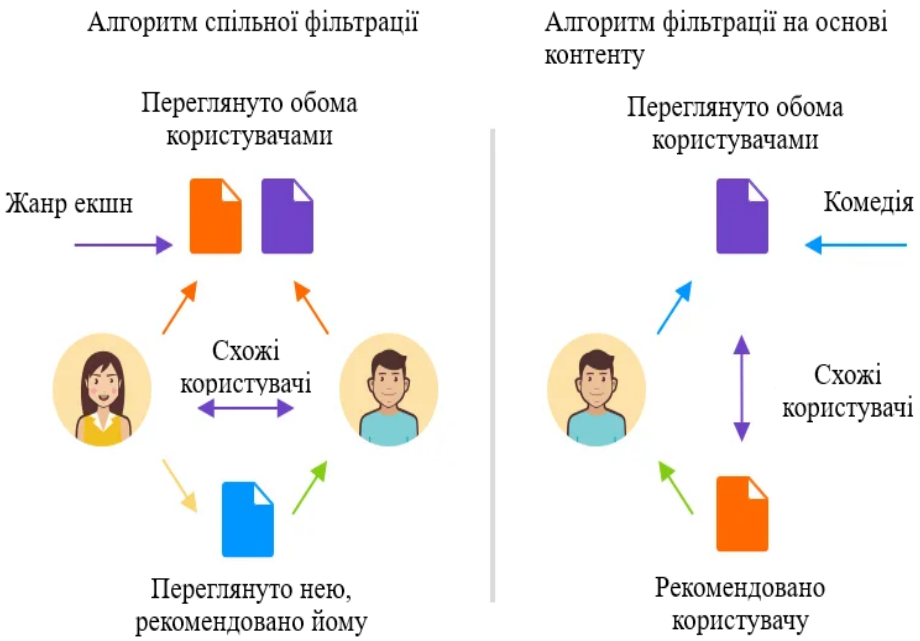

Рисунок В.4 – Порівняння відомих рекомендаційних систем

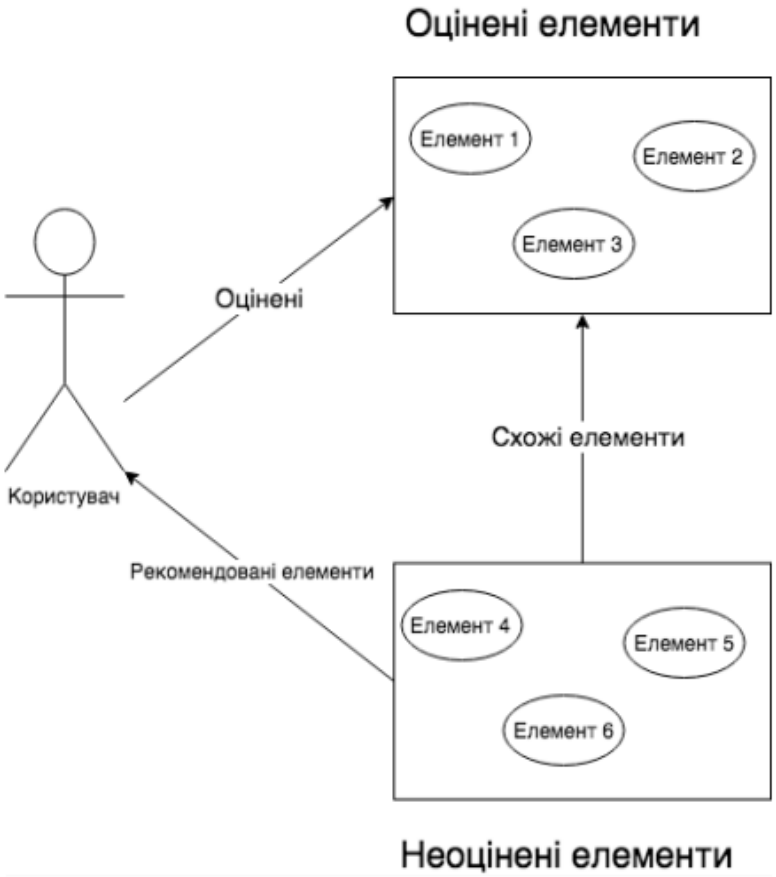

Рисунок В.5 – Принцип дії системи рекомендацій на основі аналізу контенту

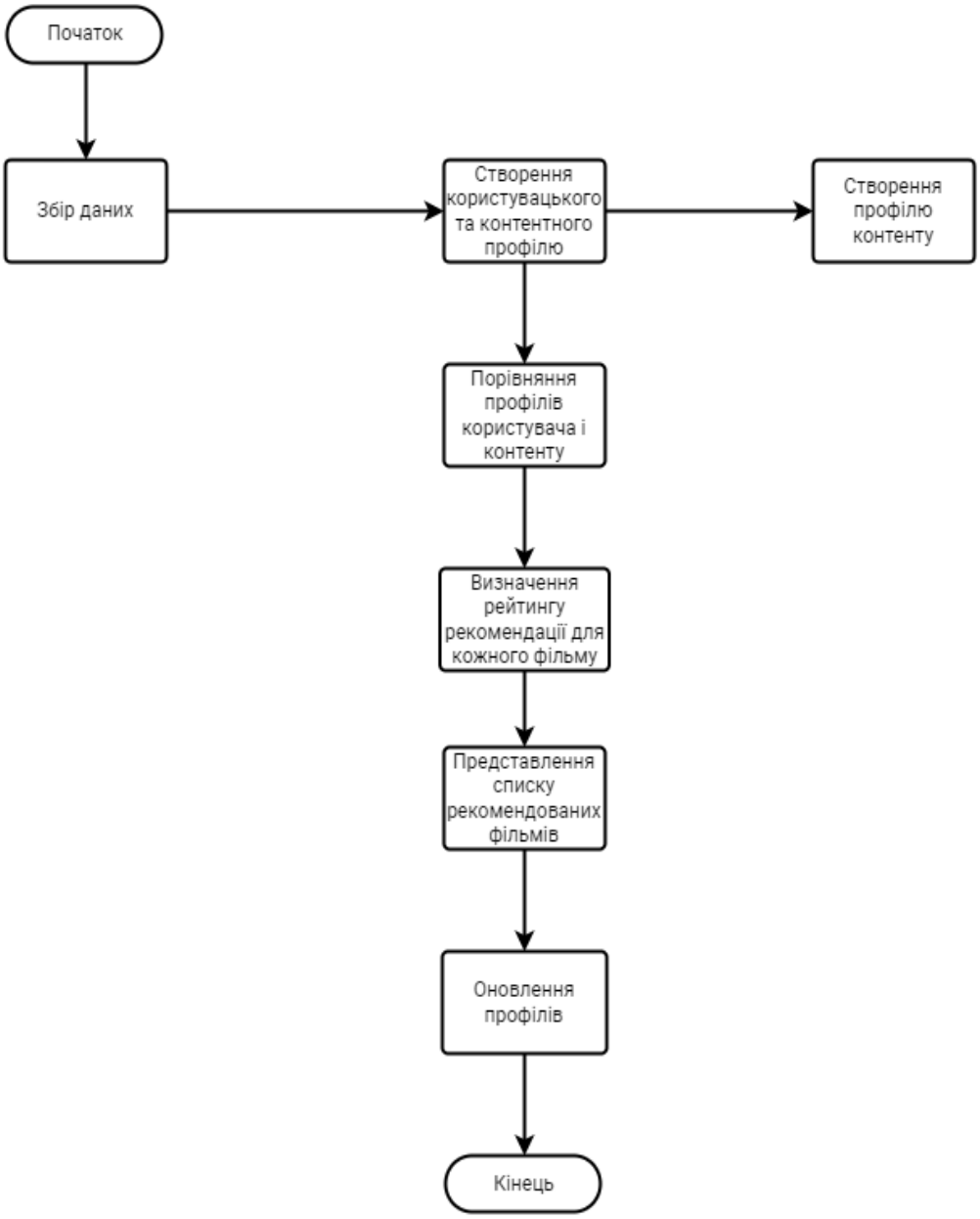

Рисунок В.6 – Алгоритм функціонування інформаційної технології стримінгового сервісу для фільмів

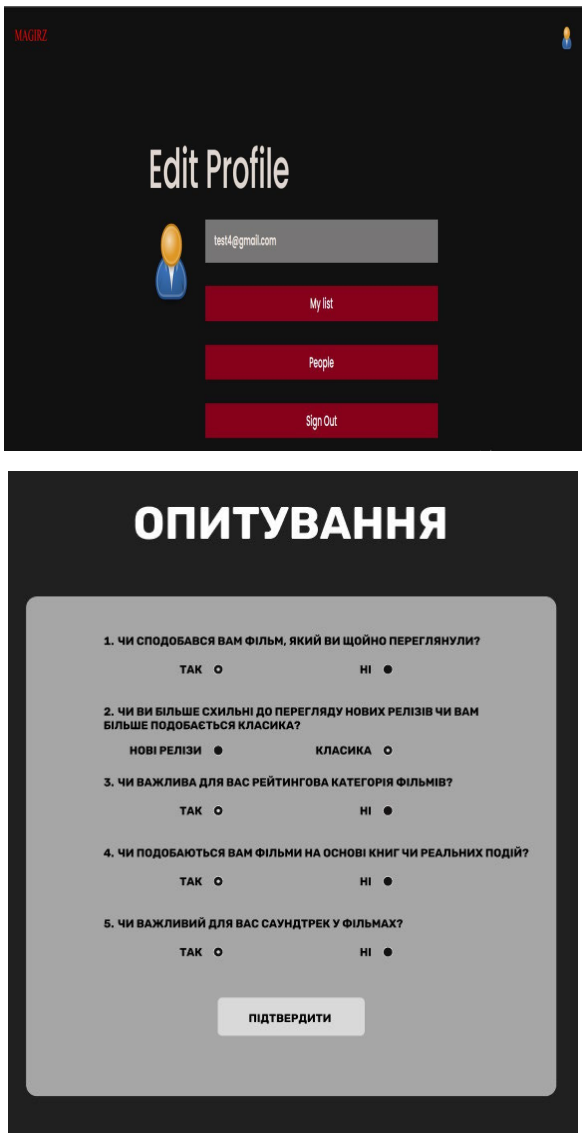

Рисунок В.7 – Графічний інтерфейс розробленого стримінгового сервісу для фільмів

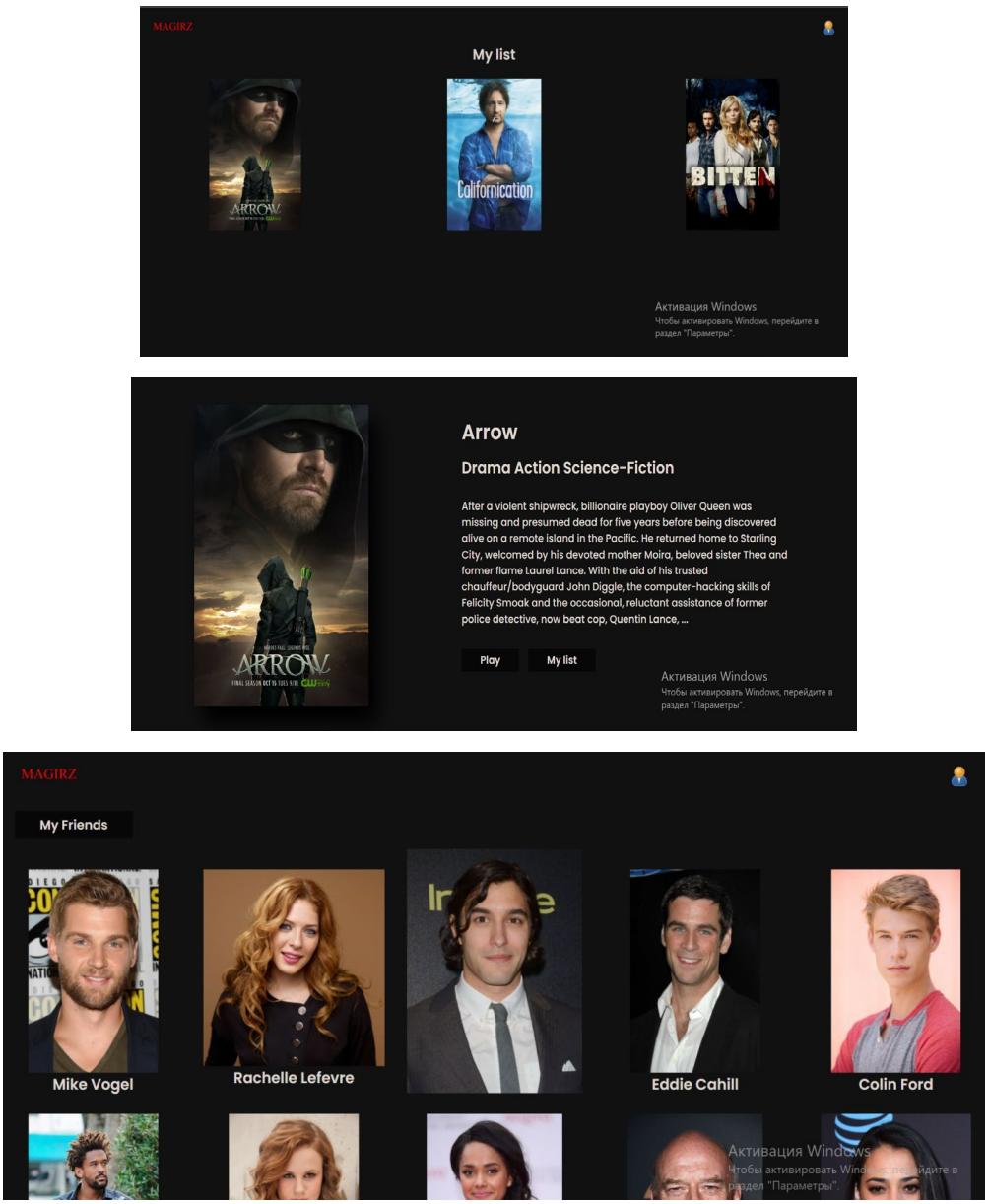

Рисунок В.8 – Графічний інтерфейс розробленого стримінгового сервісу для фільмів

Додаток Г (довідниковий)
Інструкція користувача

Щоб розпочати роботу із розробленим стримінговим сервісом для фільмів, необхідно мати підключення до мережі інтернет та встановити будь-який браузер.

Після цього можна авторизуватися або пройти автентифікацію. Для того щоб пройти автентифікацію, необхідно ввести адресу у полі Email Address, для авторизації потрібно натиснути кнопку Sign In.

При успішній авторизації користувач зможете побачити сторінку власного профілю.

Після цього можна переглянути сторінку з фільмами й відкрити сторінку пошуку.

Обравши уподобаний фільм, користувач може перейти на його сторінку й переглянути інформацію або додати до списку улюблених у своєму профілі, який було описано вище.

Після перегляду певної кількості фільмів, яка визначається алгоритмом, користувачу буде запропоновано пройти опитування. Опитування користувачів продовжується за заздалегідь розробленою і проаналізованою схемою. Для того було виконано аналіз аналогів й обрано основні фактори, на які опираються користувачі під час вибору і відсіювання медіаконтенту. Серед таких питань було чимало тих, що майже не впливали на думку людини й вважаються безкорисними, що було відсіяно під час дослідження для покращення й розробки функції рекомендацій фільмів.

Також в розробленому програмному забезпеченні є можливість пошуку людей за спільними інтересами, переглядання інформації про них та додавання їх до кола своїх друзів.