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 перегляду фільмів»

Виконала: студентка 2-го курсу, групи
2КН-22м спеціальності 122 «Комп'ютерні науки»
(шифр і назва напряму підготовки, спеціальності)
_____ Федорова В.В.
(прізвище та ініціали)

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

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

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

Вінниця ВНТУ- 2023 рік

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

ЗАТВЕРДЖУЮ
Завідувач кафедри КН
Д.т.н., проф. Яровий А.А.
28. 09. 2023 року

З А В Д А Н Н Я

НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТКИ

Федорової Вероніки Володимирівни
(прізвище, ім'я, по батькові)

1. Тема роботи Інформаційна технологія стримінгового сервісу для перегляду фільмів

керівник роботи д. т. н., проф. Іванчук Я.В.

затверджені наказом вищого навчального закладу від «18» 09 2023 року №247.

2. Строк подання студентом роботи 13. 12 2023 року.

3. Вихідні дані до роботи: швидкість відтворення відео – 0.5x, 0.75x, 1x, 1.25x, 1.5x; мова субтитрів – англійські, корейські, українські; мова озвучування відео – англійська, українська; якість відтворення відео – 144p, 240p, 360p, 480p, 720p, 1080p, 1440p.

4. Зміст текстової частини: вступ, аналіз предметної області стримінгового перегляду фільмів, проєктування інформаційної технології стримінгового сервісу для перегляду фільмів, програмна реалізація системи стримінгового сервісу для перегляду фільмів, економічна частина, висновки, список використаних джерел.

5. Перелік ілюстративного матеріалу: схема алгоритму загального функціонування стримінгового сервісу для фільмів; схема принципу дії системи рекомендацій, побудованої на основі аналізу контенту; схема клієнт-серверної архітектури.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціалита посада консультанта	Підпис, дата	
		завдання видав	виконання прийняв
1-3	Іванчук Я. В., д. т. н., проф.	29.08.23	10.11.23
4	Ратушняк О.Г., к.т.н., доц. каф. ЕПВМ	21.09.23	29.10.23

7. Дата видачі завдання 29.08. 2023 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз предметної області стримінгового перегляду фільмів. Постановка задач дослідження	1.09.23 - 07.09.23	
2	Проєктування інформаційної технології стримінгового сервісу для перегляду фільмів.	08.09.23 - 20.09.23	
3	Програмна реалізація системи стримінгового сервісу для перегляду фільмів.	21.09.23 - 20.10.23	
4	Підготовка економічної частини	21.09.23 - 29.10.23	
5	Апробація та/або впровадження результатів дослідження	30.10.23 - 05.11.23	
6	Оформлення пояснювальної записки, графічного матеріалу та презентації	06.10.23 - 10.11.23	

Студентка _____ (підпис) Федорова В.В.

Керівник роботи _____ (підпис) Іванчук Я.В.

АНОТАЦІЯ

УДК 004.8

Федорова В.В. Стримінговий сервіс для перегляду фільмів. Магістерська кваліфікаційна робота зі спеціальності 122 – Комп’ютерні науки, освітня програма - Системи штучного інтелекту. Вінниця: ВНТУ, 2023. 117 с.

На укр. мові. Бібліогр.: 25 назв; рис.: 23; табл. 9.

Дана магістерська кваліфікаційна робота присвячена розробці інформаційної технології стримінгового сервісу для перегляду фільмів. Розроблені математична модель системи підбору фільмів за вподобаннями користувача. Також розроблено систему підбору фільмів за вподобаннями користувача. Для розробки проекту використано програмне середовище Visual Studio Code, бібліотека ReactJS, розроблена на основі мови програмування JavaScript. Для автентифікації та подальшої авторизації використовувався Firebase Auth. Визначено, що використання результатів автоматизованої системи підбору фільмів дозволяє загалом значно підвищити частоту використання веб-додатку. Графічна частина складається з 4 плакатів із результатами проектування та реалізації.

Ключові слова: стримінговий сервіс, хостинг, фільми, університет, інтерфейс прикладного програмування.

ABSTRACT

Fedorova V.V. Streaming service for watching films. Master's degree in specialty 122 - Computer Science, educational program – Artificial intelligence systems. Vinnytsia: VNTU, 2023. 117 p.

In the Ukrainian language. Bibliogr .: 25 titles; fig .: 23; table 3.

This master's thesis is devoted to the development of information technology for a movie streaming service. A mathematical model of the system for selecting movies according to user preferences has been developed. A system for selecting movies according to user preferences has also been developed. The project was developed using the Visual Studio Code software environment and the ReactJS library, developed on the basis of the JavaScript programming language. Firebase Auth was used for authentication and subsequent authorization. It has been determined that the use of the results of the automated movie selection system can significantly increase the frequency of use of the web application. The graphic part consists of 4 posters with the results of design and implementation.

Keywords: streaming service, hosting, movies, university, application programming interface.

ЗМІСТ

ВСТУП

Актуальність теми. Актуальність цієї теми полягає в тому, що ринок онлайн-кінотеатру росте з кожним роком. 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. Apple і Disney вступили на ринок стримінгових послуг у 2020 році. 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 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 на основі експертної системи, 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 «Моделі, методи, технології та пристрої інтелектуальних інформаційних систем управління, економіки, навчання та комунікацій» та плану наукової та навчально-методичної роботи кафедри.

Метою дослідження є підвищення функціональних можливостей стримінгового сервісу за допомогою розробки рекомендаційної системи, що дозволить ефективно підбирати визначений тип кінопродукції.

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

- виконати аналіз предметної області та об’єкту дослідження;

− провести аналіз існуючих систем-аналогів інформаційної системи стримінгового сервісу для перегляду відео та їх характеристики;

− розробити модель інформаційної системи стримінгового сервісу для перегляду відеопродукції;

− розробити метод надання рекомендацій у стримінгових сервісів для перегляду відеопродукції;

− розробити алгоритм роботи системи підбору фільмів за вихідними даними;

− виконати програмну реалізацію стримінгового сервісу для фільмів;

− провести тестування програмного забезпечення та проаналізувати отримані результати;

− провести розрахунок економічної ефективності науково-технічної розробки за можливого її застосування.

Об’єктом дослідження є методи надання рекомендацій в інформаційних системах стримінгового сервісу для перегляду відеопродукції.

Предметом дослідження є програмний модуль рекомендаційної системи 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 методу надання рекомендацій.

Наукова новизна одержаних результатів: розроблено метод надання рекомендацій на основі фільтрації даних за контентом в інформаційній технології стримінгових сервісів для перегляду відеопродукції, що відрізняє розробку від існуючих аналогів.

Практичне значення одержаних результатів полягає у наступному:

- розроблено загальну модель функціонування інформаційної системи стримінгового сервісу для перегляду відеопродукції;

- розроблено програмний модуль для надання рекомендацій стримінгового сервісу для переглядів 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 магістранта. Усі результати, наведені у магістерській кваліфікаційній роботі, отримані самостійно.

Апробація результатів роботи. Фактичні результати роботи були апробовані на такій конференції, як «LII Науково-технічна конференція факультету інтелектуальних інформаційних технологій та автоматизації (2023)» [2].

Публікації. Результати магістерської дипломної роботи були представлені у вигляді тез на науково-технічних конференціях [2, 3].

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ СТРИМІНГОВИХ СЕРВІСІВ ДЛЯ ПЕРЕГЛЯДУ 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, і його ілюструє рисунок 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 програмних рішень в сфері 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.

Рисунок 1.2 – Приклад інтерфейсу стримінгового сервісу «Netflix»

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.

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 Інтернету (для користування Netflix вам потрібен стабільний і швидкий доступ до Інтернету).

Наступний із аналогів розглянемо стримінговий сервіс Hulu , інтерфейс якого продемонстровано на рисунку 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).

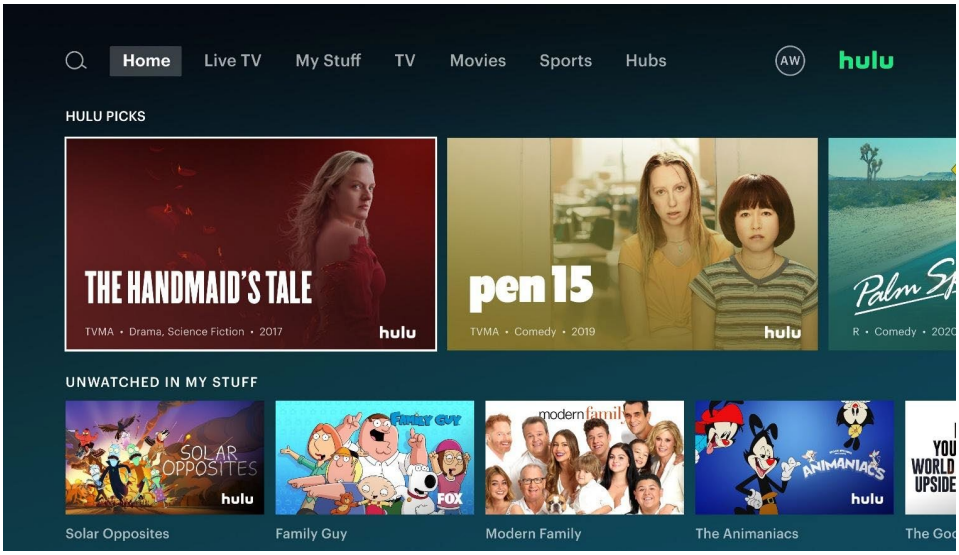

Рисунок 1.3 – Приклад інтерфейсу стримінгового сервісу «Hulu»

Окрім іноземних стримінгових сервісів є і вітчизняний – MEGOGO [9], інтерфейс якого проілюстровано на рисунку 1.4. Він співпрацює з такими студіями, як Paramount Pictures, FOX, Warner Bros., Walt Disney та BBC.

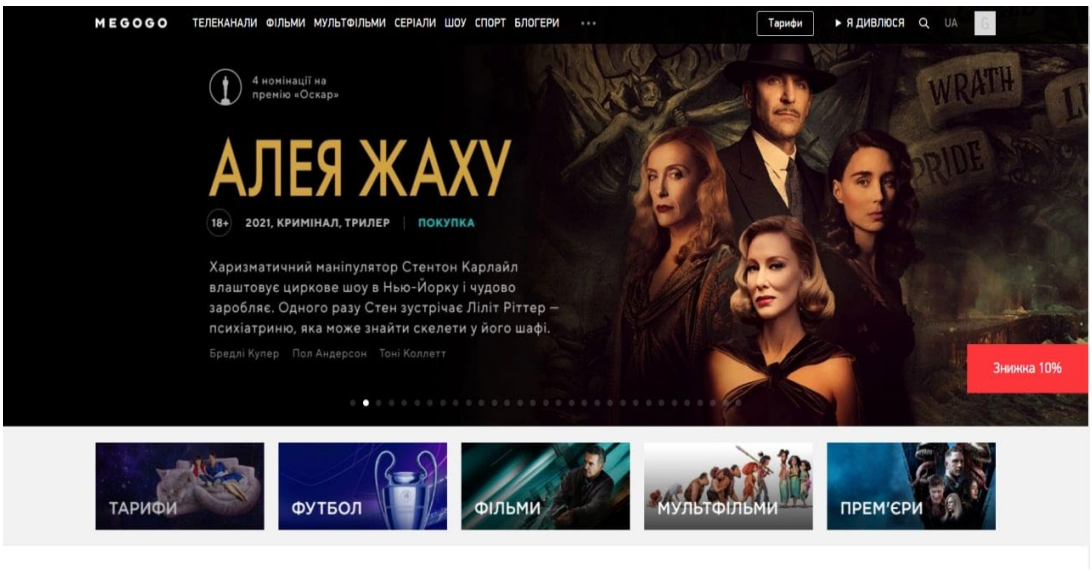

Рисунок 1.4 – Приклад інтерфейсу стримінгового сервісу «MEGOGO»

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. 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 можливість зареєстрованим користувачам викладати власний контент та примножувати свою авдиторію. Для таких користувачів існує спеціальна аналітика, за допомогою якої можна простежити, як просувається ваш контент. Багато з авторів має можливість заробляти кошти через рекламу, яку пропонує рекламодавець.

Є також платна підписка, яка надає перспективи перегляду контенту без реклами, зберігати треки та відео на пристрої, вмикати їх в офлайн-режимі, а також слухати музику з виключеним екраном. Інтерфейс YouTube продемонстровано на рисунку 1.5.

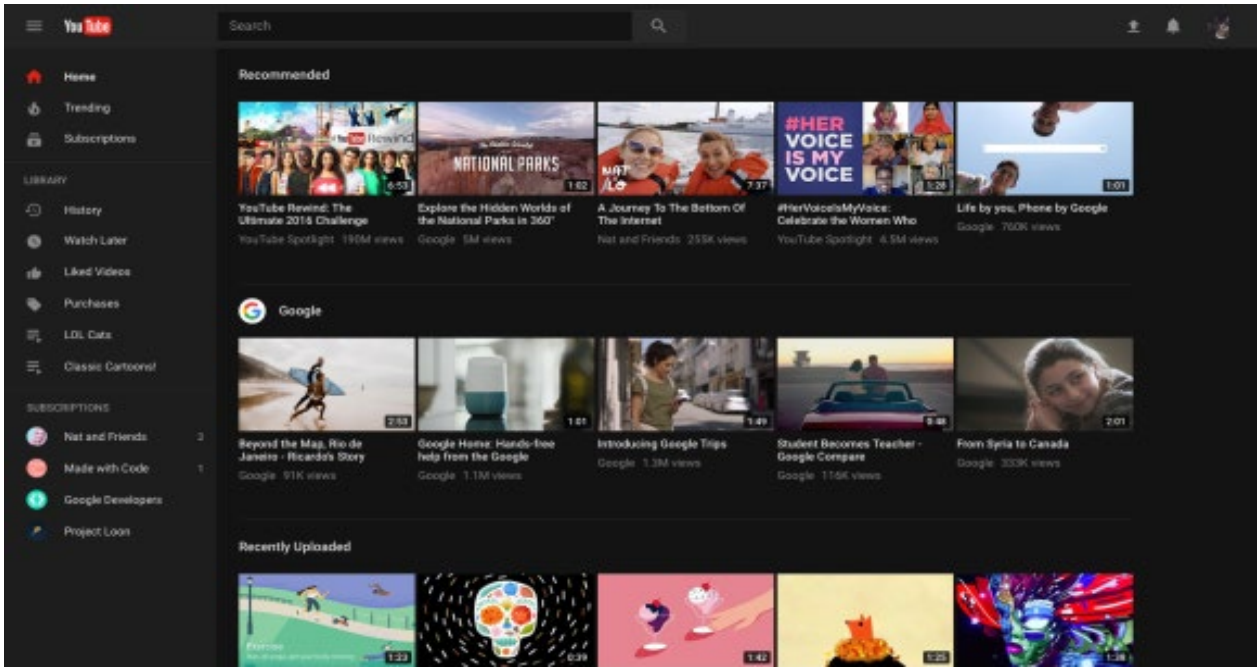

Рисунок 1.5 – Приклад інтерфейсу стримінгового сервісу «YouTube»

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

1	2	3	4	5
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 всіх недоліків та погрішностей.

1.3 Аналіз 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.

Базову структуру експертної системи проілюстровано на рис. 1.6. Зазначені структурні елементи є найхарактернішими для більшості експертних систем, хоча в реальних умовах деякі з них можуть бути відсутні.

Рисунок 1.6 – Приклад структури експертної системи

Основні функції експертних систем включають:

– 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 ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ СТРИМІНГОВОГО СЕРВІСУ ДЛЯ ПЕРЕГЛЯДУ 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 системи 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 системи стримінгового сервісу для фільмів, що проілюстровано на рисунку 2.1.

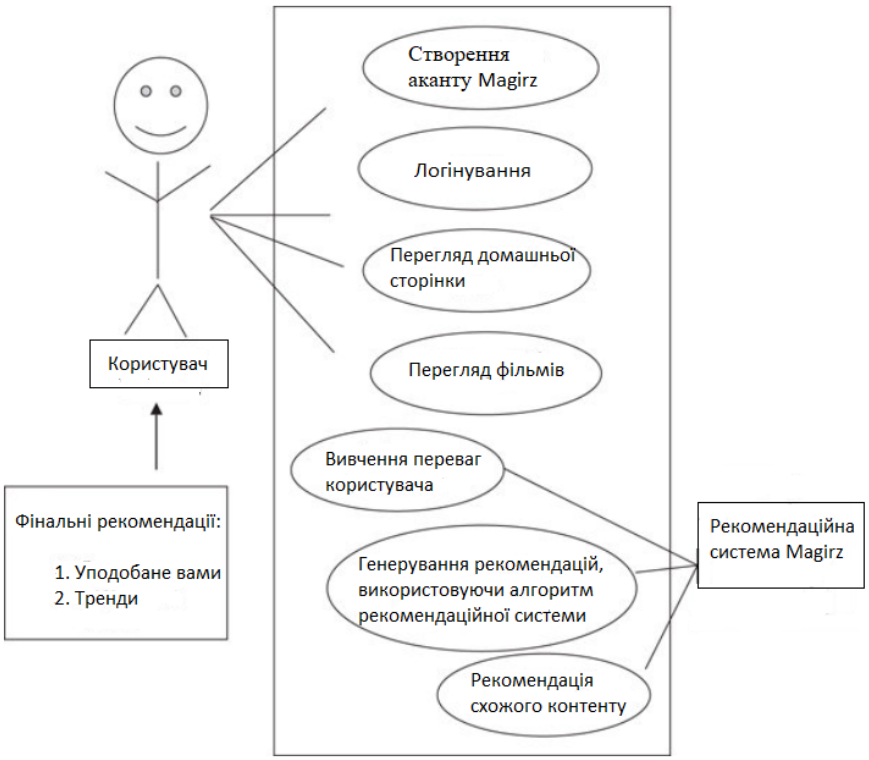

Рисунок 2.1 – Схема моделі інформаційної системи стримінгового сервісу для фільмів

На рисунку наведена функціональна модель, яка описує принципи дії інформаційної системи. При розробці моделі інформаційної системи для потокового сервісу 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 інформаційної системи стримінгового сервісу для фільмів, продемонстровану на рисунку 2.2.

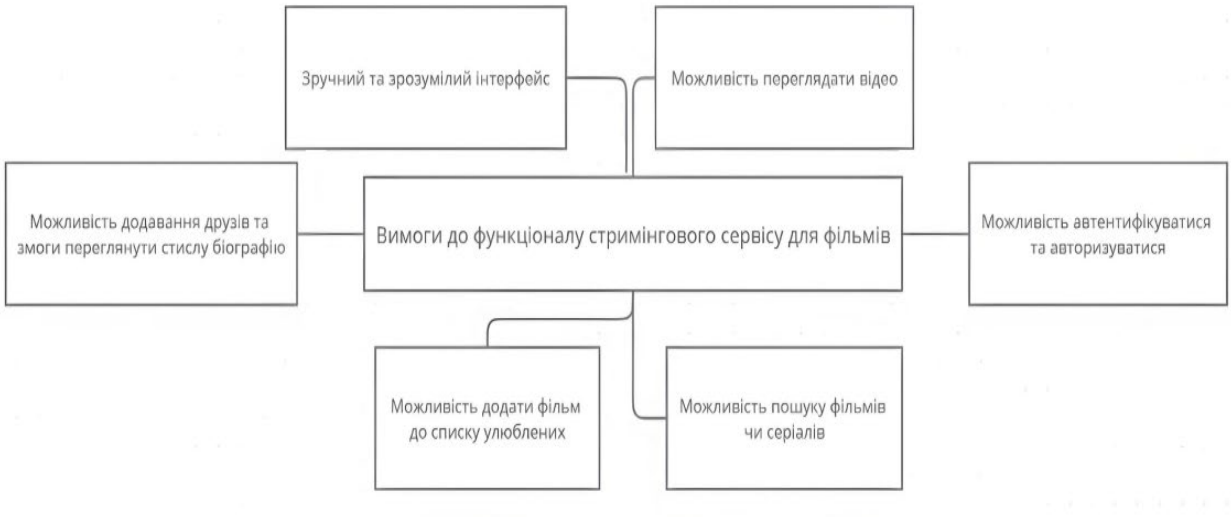

Рисунок 2.2 – Схема моделі вимог до інформаційної системи стримінгового сервісу для фільмів

Є кілька способів створити точну систему рекомендацій:

- 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.

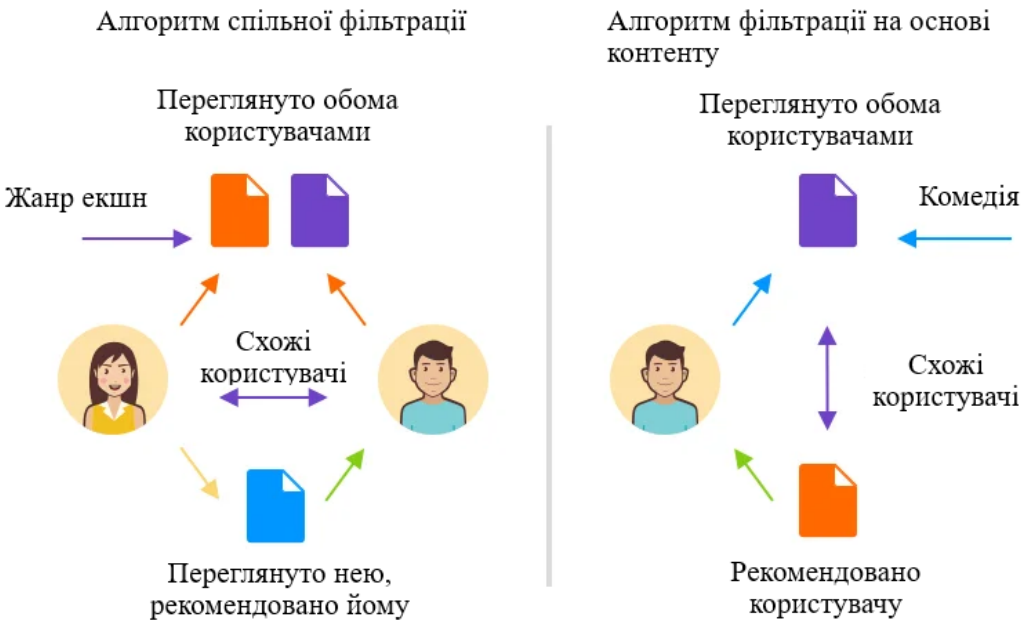

Рисунок 2.3 – Порівняльна схема найвідоміших рекомендаційних систем

Архітектура інформаційних систем для потокових служб перегляду фільмів повинна бути масштабованою та гнучкою для задоволення зростаючих потреб користувачів. Він також повинен бути надійним та ефективним, 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.

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, що продемонстрована на рисунку 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.

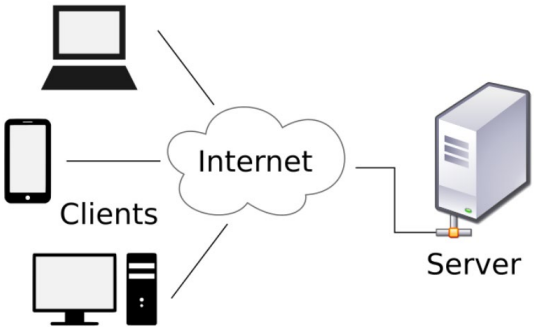

Рисунок 2.4 – Типовас схема клієнт-серверної архітектури

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 безпека (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 методу надання рекомендацій у стримінгових сервісах для перегляду фільмів на основі фільтрації за контентом

Як можна зрозуміти з назви підходу, фільтрація на основи вмісту перш за все опирається на інформацію про контент 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).

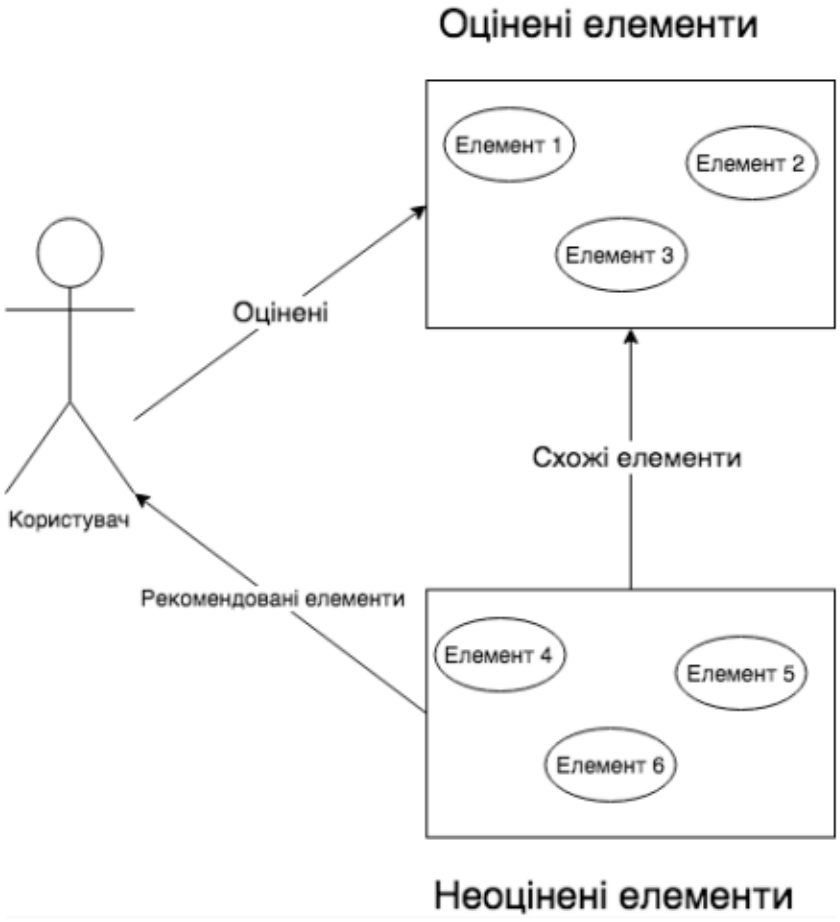

Рисунок 2.5 – Принципова схема функціонування системи рекомендацій, побудованої на основі аналізу контенту

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 A сподобався користувачу U, а елемент B схожий на елемент A, то можна зробити висновок, що елемент B також сподобається користувачу B.

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 [22].

Її можна обчислити за наступною формулою:

$$TF_{ij} = \frac{f_{ij}}{max_{zf_{zj}}}, \quad (2.1)$$

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.

$$IDF_i = log \frac{N}{n_i}, \quad (2.2)$$

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:

$$w_{ij} = TF_{ij}: * IDF_i. \quad (2.3)$$

Тоді профіль елементу d_j можна задати наступним чином:

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

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 $\overrightarrow{w_u}$:

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

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 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.

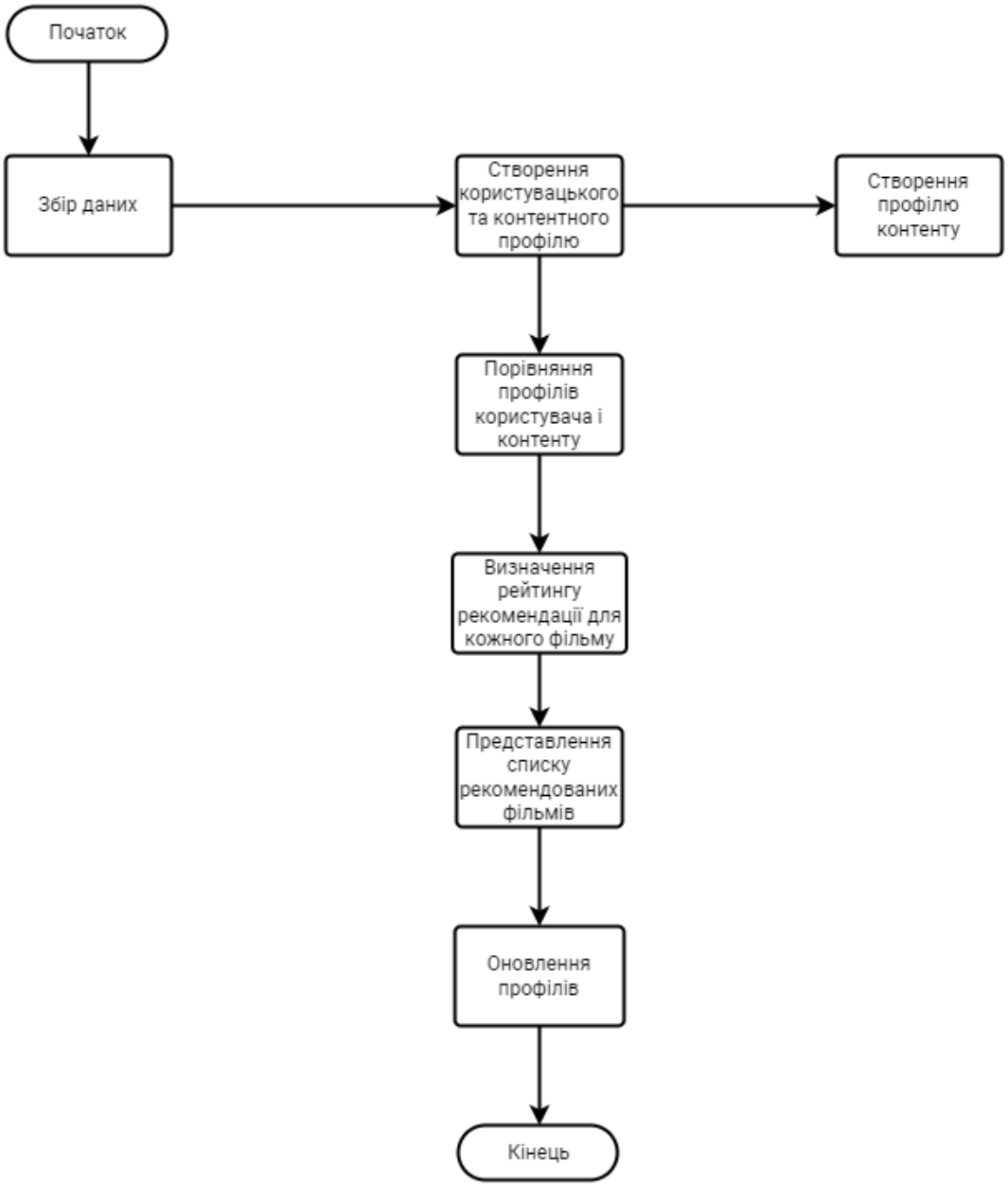

Рисунок 2.6 – Схема алгоритму загального функціонування стримінгового сервісу для фільмів

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; модуль 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 ПЕРЕГЛЯДУ ФІЛЬМІВ

3.1 Вибір та обґрунтування мов 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’єктно-орієнтованої) 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 [22].

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 [23].

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 Програмна реалізація рекомендаційної системи стримінгового сервісу для перегляду фільмів

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

```
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 даного опитування аналізуються кодом та складаються у масив до певного користувача. Приклад опитування проведено на рисунку 3.1.

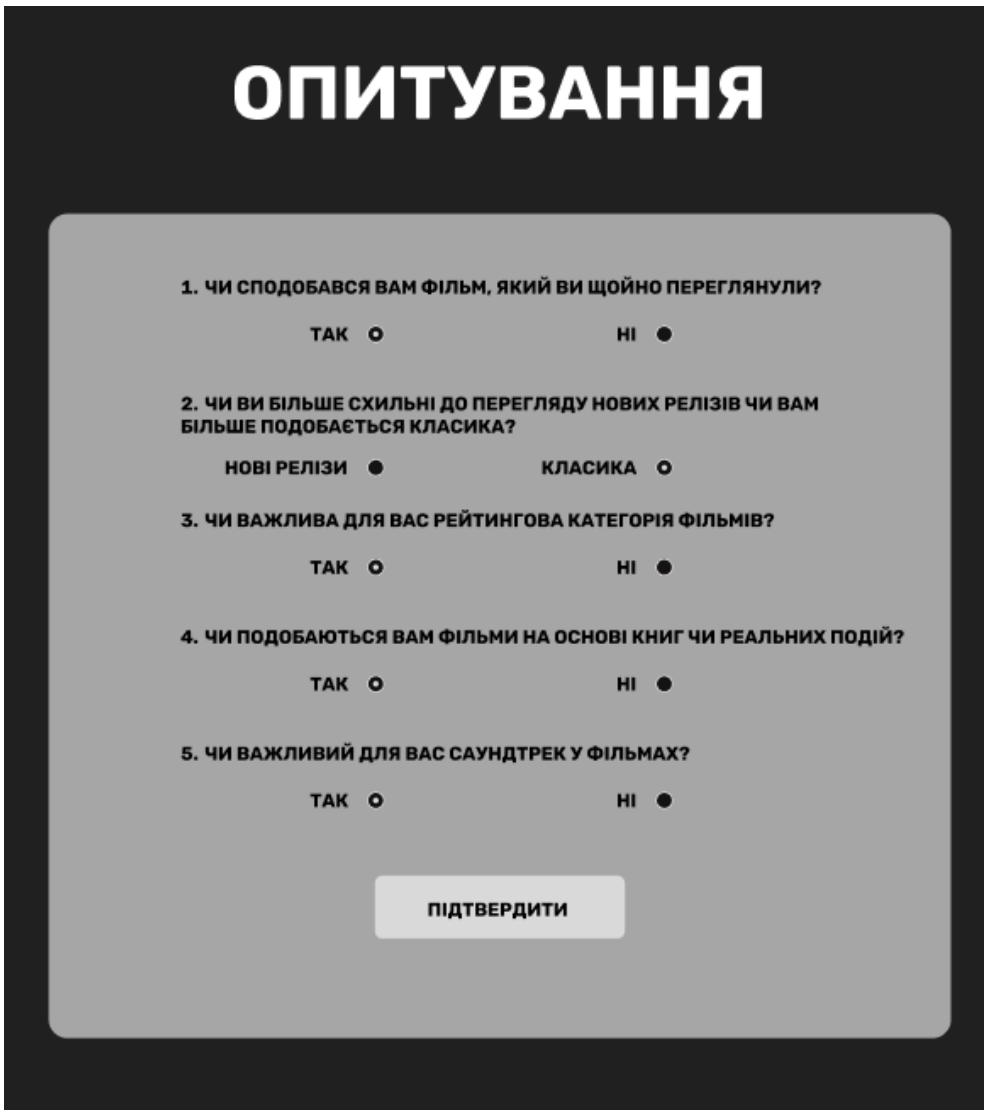

Рисунок 3.1 – Загальний вигляд інтерфейсного вікна опитування користувача

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 рекомендаційної системи стримінгового сервісу для перегляду 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 рекомендаційної системи є фільми з відповідними показниками, що показано на рисунку 3.3.

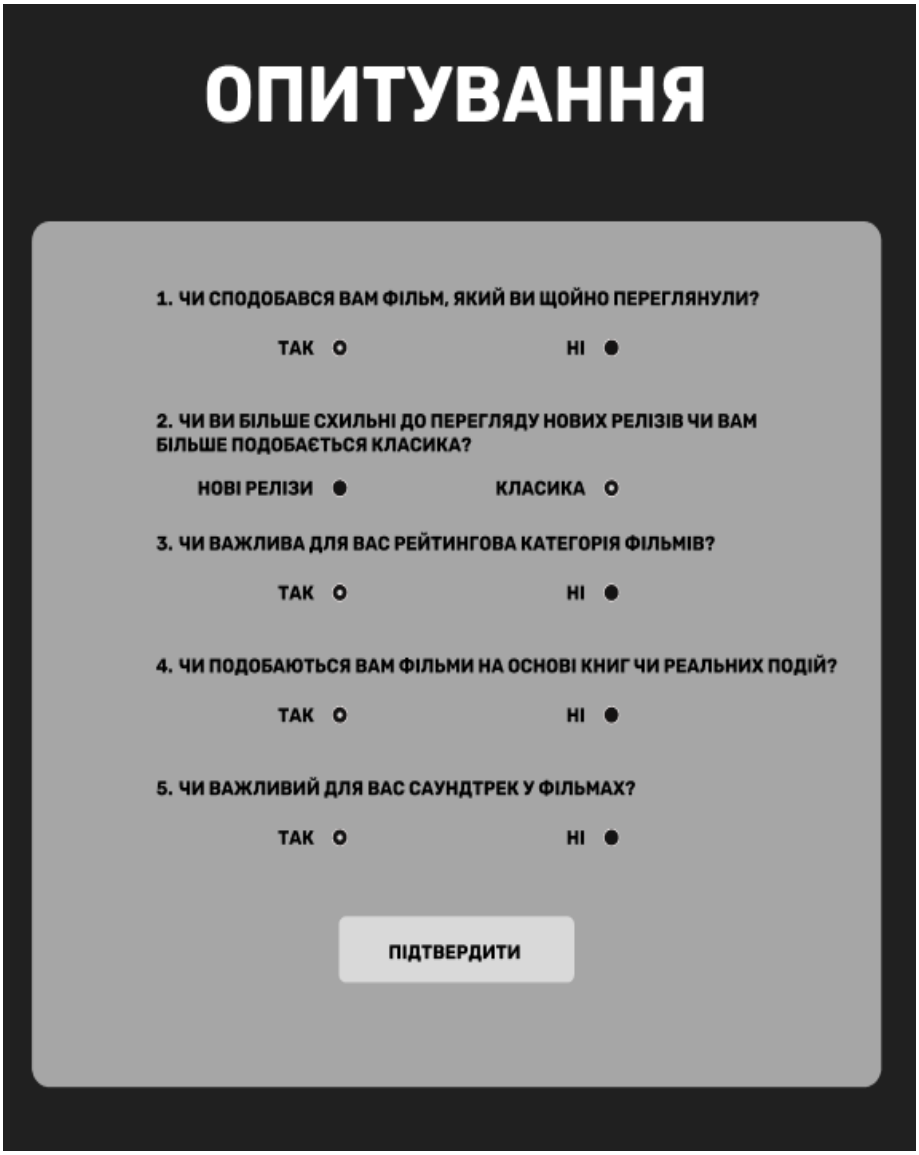

Рисунок 3.2 – Опитування користувача для першого тесту

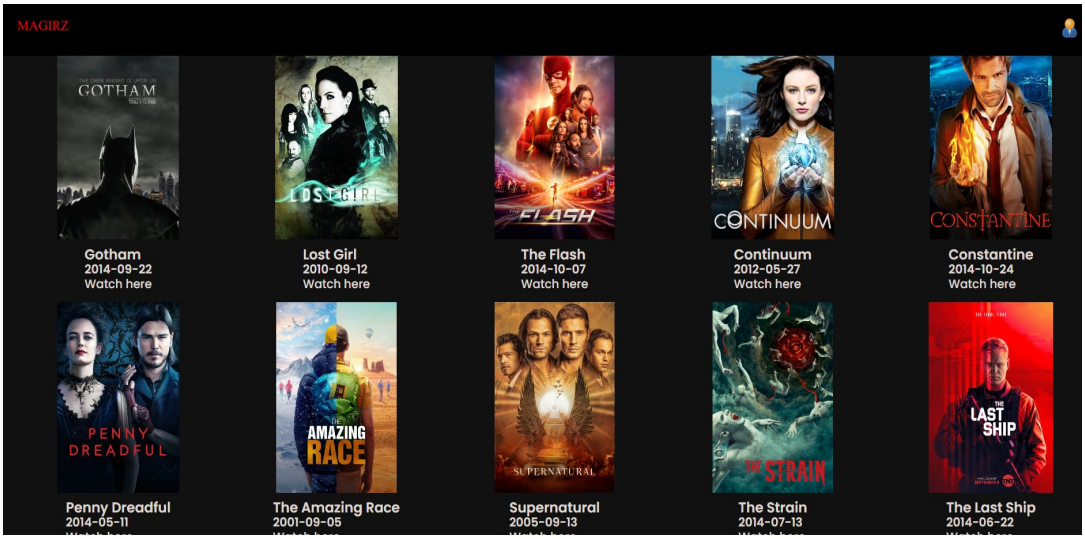

Рисунок 3.3 – Загальний вигляд інтерфейсного вікна рекомендованих фільмів, на основі фільтрації та першого тесту

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.

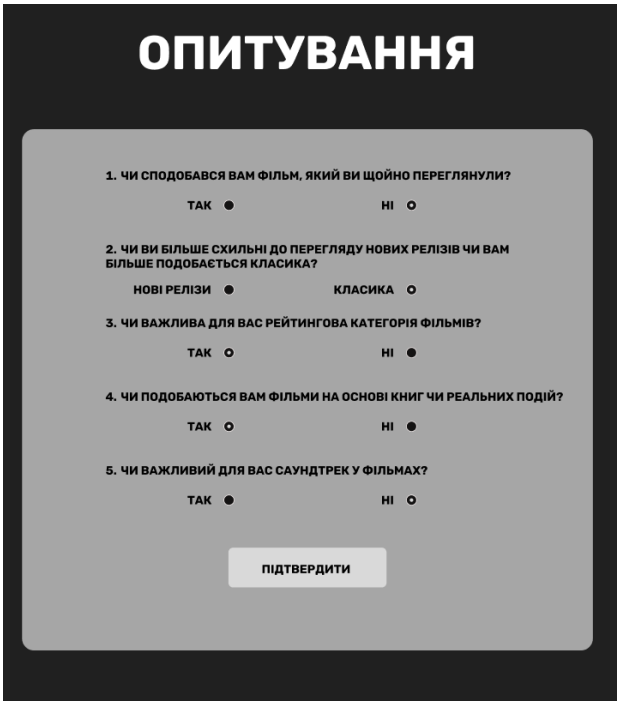

Рисунок 3.4 – Опитування користувача для другого тесту

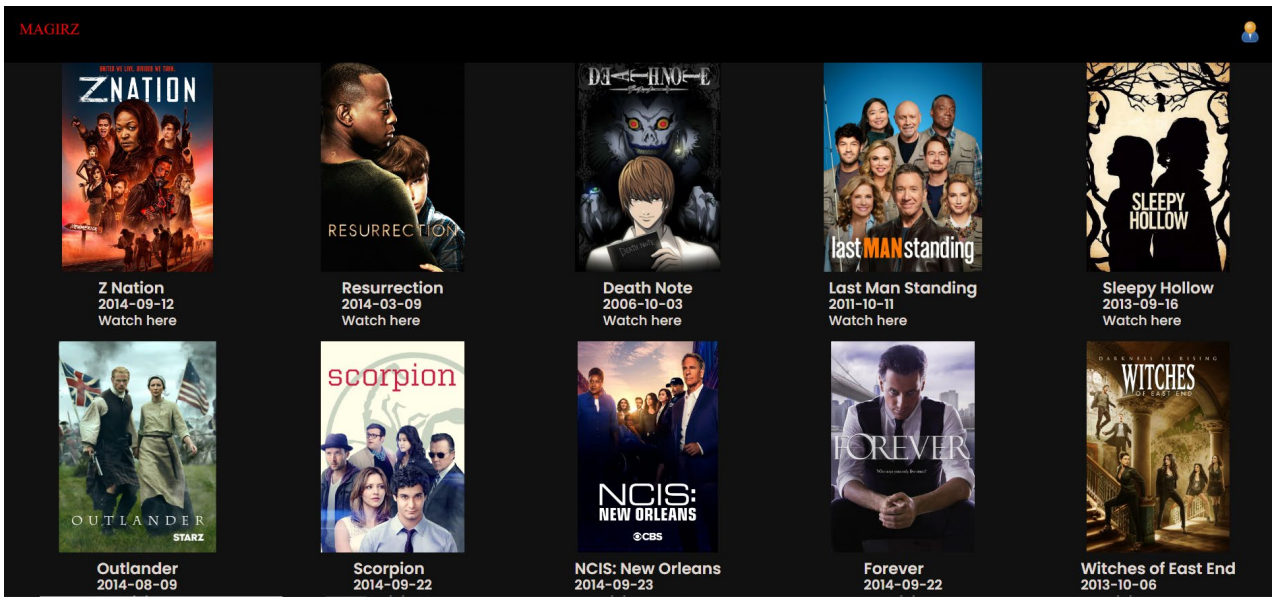

Рисунок 3.5 – Загальний вигляд інтерфейсного вікна рекомендованих фільмів, на основі фільтрації та другого тесту

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.

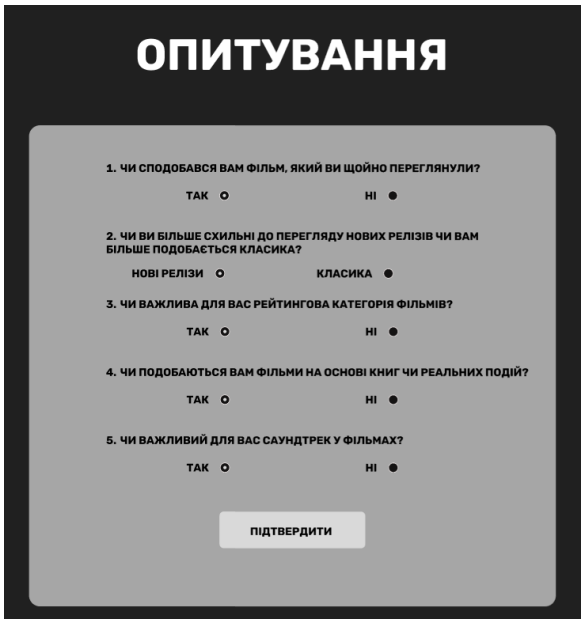

Рисунок 3.6 – Опитування користувача для третього тесту

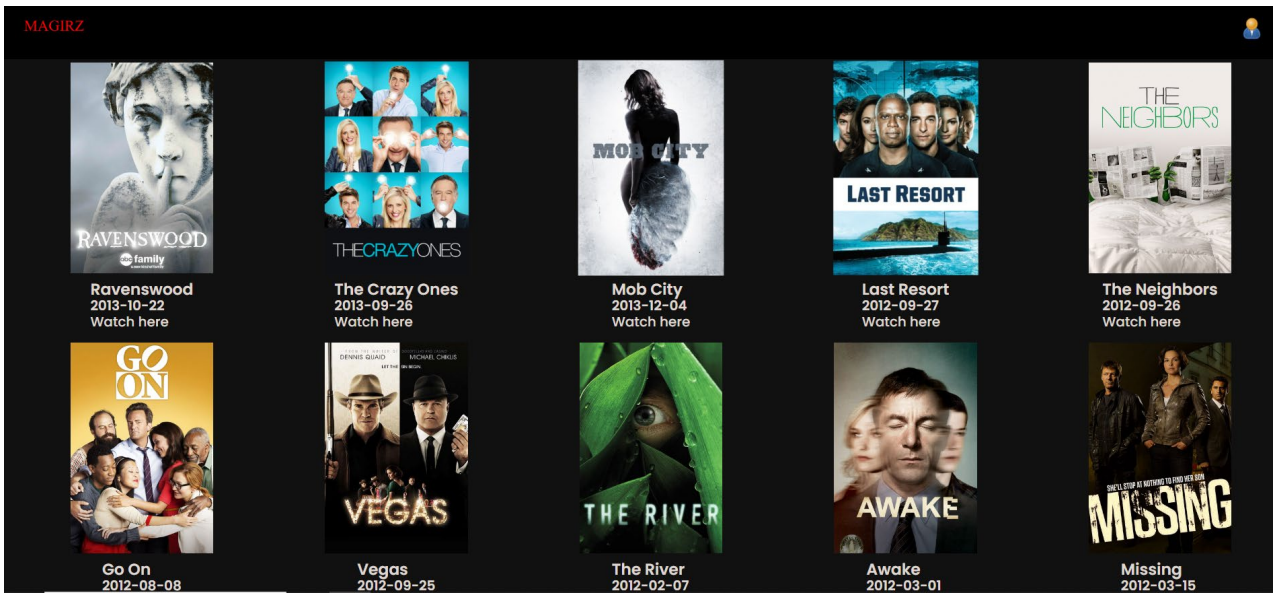

Рисунок 3.7 – Загальний вигляд інтерфейсного вікна рекомендованих фільмів, на основі фільтрації та третього тесту

Маємо можливість перейти до власного профілю, що показаний на рисунку 3.8, в якому обрати або вкладку уподобаного контенту продемонстрованою на рисунку 3.9, або вкладку з пошуком людей проілюстрованою на рисунку 3.10 для перегляду інформації про них з подальшим доданням їх у друзі.

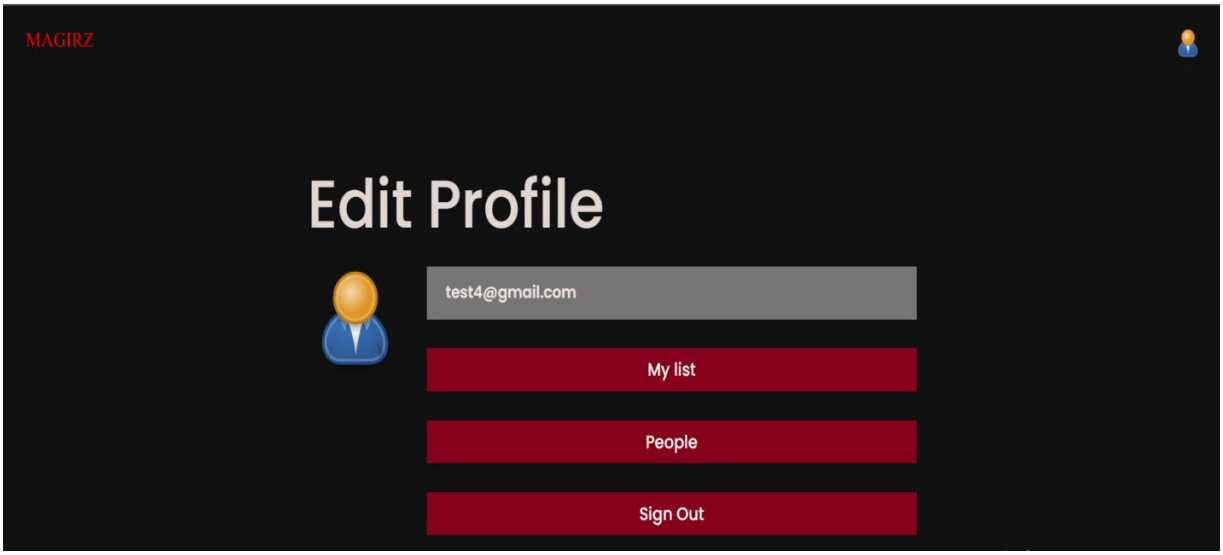

Рисунок 3.8 – Загальний вигляд інтерфейсного вікна коригування параметрів профілю користувача

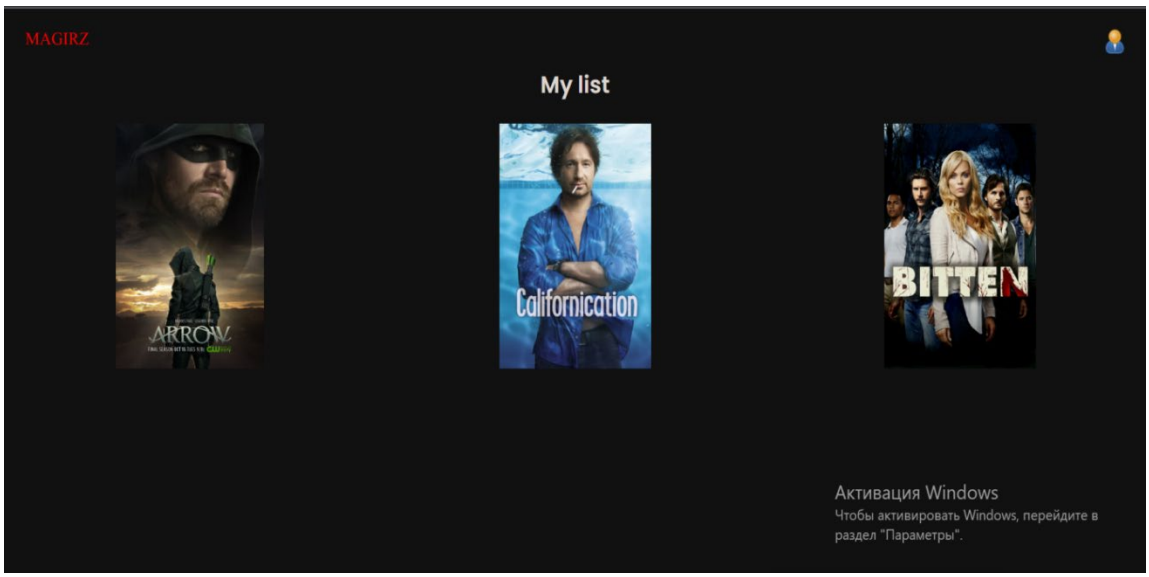

Рисунок 3.9 – Загальний вигляд інтерфейсного вікна сторінки уподобаних фільмів

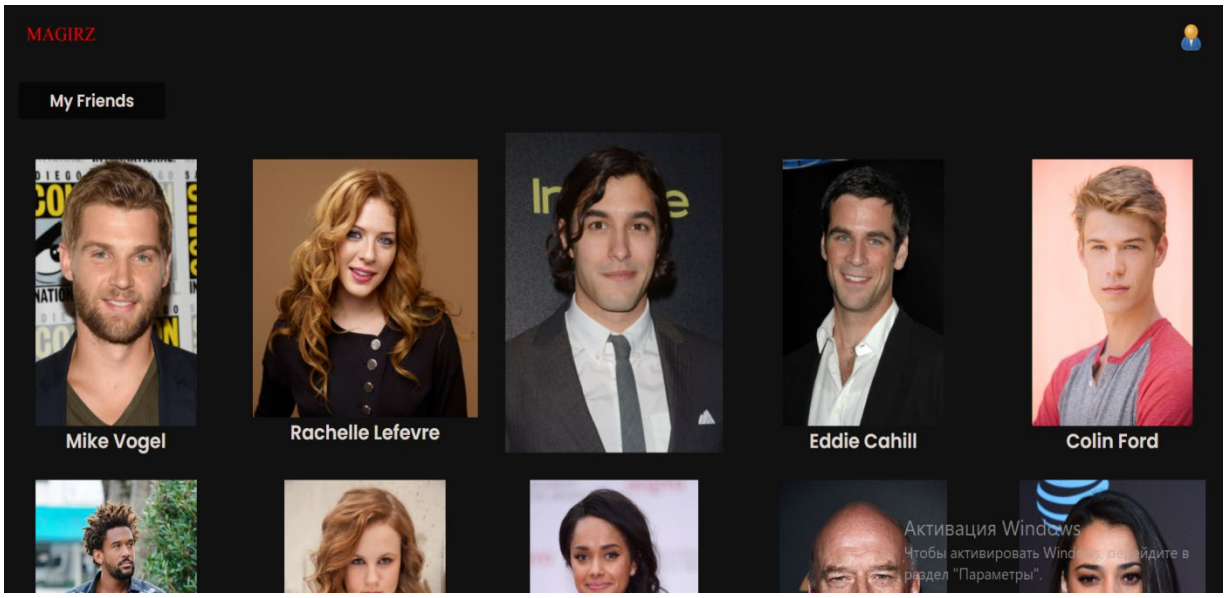

Рисунок 3.10 – Загальний вигляд інтерфейсного вікна сторінки акторів та людей

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

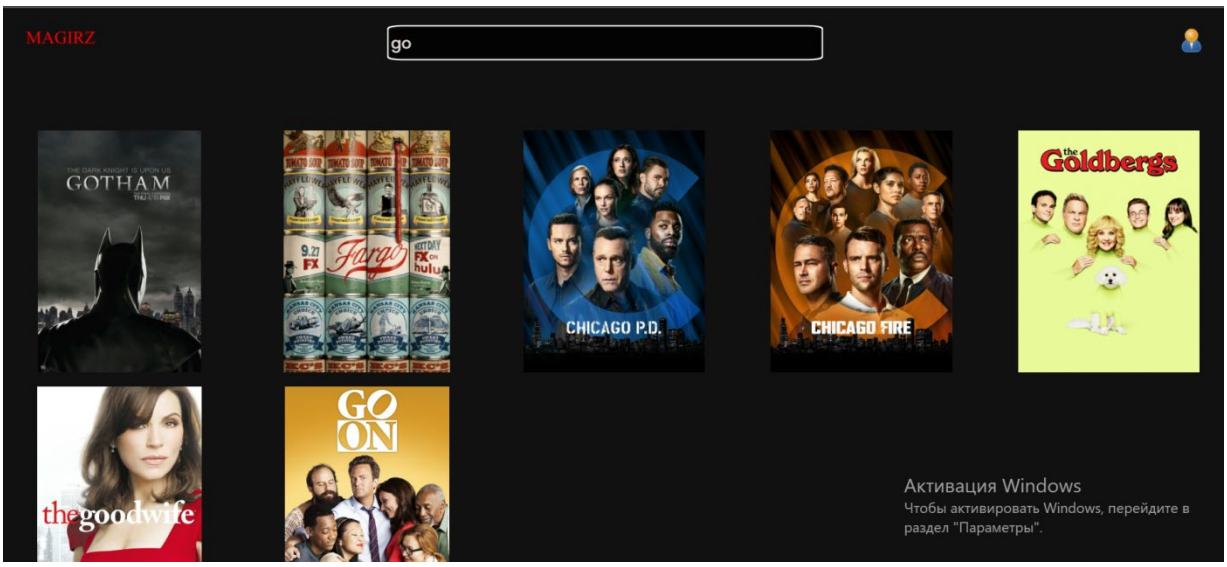

Рисунок 3.11 – Загальний вигляд інтерфейсного вікна сторінки пошуку фільмів

Протестувавши виконану роботу та порівнявши з очікуваними результатами, отримано гарні результати, адже було оптимізовано роботу сервісу та розширено функціонал, тому розроблений додаток має великий потенціал для подальшого просування та використання. Отже, мета розробки стримінгового сервісу для фільмів досягнута.

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. Стримінговий сервіс для фільмів повністю відповідає поставленим вимогам.

4 ЕКОНОМІЧНА ЧАСТИНА

4.1 Визначення комерційного потенціалу розробки

Для успішного впровадження науково-технічної розробки критично важливо, щоб вона відповідала актуальним вимогам науково-технічного прогресу та ураховувала економічні аспекти. Надання оцінки економічної ефективності результатів науково-дослідної роботи є необхідною частиною цього процесу. Дослідження, яке представлене у магістерській роботі і приурочене розробці та вивченню "Інформаційна технологія стримінгового сервісу для перегляду фільмів", належить до науково-технічних робіт, спрямованих на виведення на ринок. Рішення про комерціалізацію розробки може бути прийняте протягом проведення самої роботи, відкриваючи перспективи подальшого виведення на ринок. Цей напрямок визначається як пріоритетний, оскільки розроблені результати можуть бути корисними для різних зацікавлених сторін і приносити економічні вигоди. Однак для успішної реалізації цього процесу ключовим є привертання зацікавленого інвестора, який виявить інтерес до втілення даного проекту, і переконання його у доцільності інвестування у цю розробку [25]. Для досягнення цієї мети передбачені наступні етапи виконання робіт:

1. Проведення комерційного аудиту науково-технічної розробки, включаючи визначення науково-технічного рівня та комерційного потенціалу.
2. Розрахунок витрат на реалізацію науково-технічної розробки.
3. Проведення розрахунку економічної ефективності впровадження та комерціалізації науково-технічної розробки для потенційного інвестора, а також обґрунтування економічної доцільності комерціалізації з точки зору інвестора.

4.1 Проведення комерційного та технологічного аудиту науково-технічної розробки

Метою проведення комерційного і технологічного аудиту дослідження за темою "Інформаційна технологія стримінгового сервісу для перегляду фільмів" є розширення функціональних можливостей програмного забезпечення для захисту файлів.

Оцінювання науково-технічного рівня розробки та її комерційного потенціалу рекомендується здійснювати із застосуванням 5-ти бальної системи оцінювання за 12-ма критеріями, наведеними в табл. 4.1 [Козловський, Лесько, Кавецький].

Таблиця 4.1 – Рекомендовані критерії оцінювання науково-технічного рівня і комерційного потенціалу розробки та бальна оцінка

Бали (за 5-ти бальною шкалою)					
	0	1	2	3	4
Технічна здійсненність концепції					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено працездатність продукту в реальних
Ринкові переваги (недоліки)					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів

Продовження таблиці 4.1

Бали (за 5-ти бальною шкалою)					
	0	1	2	3	4
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною
7	Активна конкуренція великих компаній на	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-мти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років

Продовження таблиці 4.1

Бали (за 5-ти бальною шкалою)					
	0	1	2	3	4
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Результати оцінювання науково-технічного рівня та комерційного потенціалу науково-технічної розробки потрібно звести до таблиці. Для оцінки науково-технічного рівня і комерційного потенціалу розробки експертами було запрошено трьох незалежних експертів Вінницького національного технічного університету кафедри «Комп'ютерні науки»: Денисюк Валерій Олександрович, к.т.н., доцент, Козловський Андрій Володимирович, к.т.н., доцент, Колодний Володимир Володимирович, к.т.н., доцент.

Таблиця 4.2 – Результати оцінювання науково-технічного рівня і комерційного потенціалу розробки експертами

Критерії	Експерт (ПІБ, посада)		
	Денисюк В. О.	Козловський А. В.	Колодний В. В.
	Бали, виставлені експертами:		
1. Технічна здійсненність концепції	2	2	2
2. Ринкові переваги (наявність аналогів)	0	0	0
3. Ринкові переваги (ціна продукту)	4	3	4
4. Ринкові переваги (технічні властивості)	3	3	3
5. Ринкові переваги (експлуатаційні витрати)	4	4	3

Продовження таблиці 4.2

6. Ринкові перспективи (розмір ринку)	2	2	2
7. Ринкові перспективи (конкуренція)	2	2	2
8. Практична здійсненність (наявність фахівців)	2	2	3
9. Практична здійсненність (наявність фінансів)	3	3	2
10. Практична здійсненність (необхідність нових матеріалів)	4	4	3
11. Практична здійсненність (термін реалізації)	3	4	3
12. Практична здійсненність (розробка документів)	4	4	4
Сума балів	$СБ_1$=33	$СБ_2$=34	$СБ_3$=31
Середньоарифметична сума балів *СБс*	33		

За результатами розрахунків, наведених в таблиці 4.2, зробимо висновок щодо науково-технічного рівня і рівня комерційного потенціалу розробки. При цьому використаємо рекомендації, наведені в табл. 4.3 [Козловський, Лесько, Кавецький].

Таблиця 4.3 – Науково-технічні рівні та комерційні потенціали розробки

Середньоарифметична сума балів СБ, розрахована на основі висновків	Науково-технічний рівень та комерційний потенціал розробки
41…48	Високий
31…40	Вище середнього
21…30	Середній
11…20	Нижче середнього
0…10	Низький

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою "Інформаційна технологія стримінгового сервісу для перегляду фільмів" становить 33 бали, що, відповідно до таблиці 4.3 рівень комерційного потенціалу розробки вище середнього, що свідчить про комерційну важливість проведення даних досліджень.

Магістерська кваліфікаційна робота "Інформаційна технологія стримінгового сервісу для перегляду фільмів" відноситься до науково-технічних робіт, які орієнтовані на виведення на ринок, тобто при цьому відбувається комерціалізація науково-технічної розробки. Цей напрямок є для нас пріоритетним, оскільки результатами розробки можуть користуватися не тільки самі розробники, а й інші споживачі, отримуючи при цьому суттєвий економічний ефект.

4.2 Визначення рівня конкурентоспроможності розробки

У процесі визначення економічної ефективності науково-технічної розробки також доцільно провести прогноз рівня її конкурентоспроможності за сукупністю параметрів, що підлягають оцінюванню.

У якості аналога для розробки було обрано Netflix. Основними недоліками аналога є слабка рекомендаційна система і нерелевантний інтерфейс. Також до недоліків можна віднести слабку систему взаємодії користувачів.

У розробці дана проблема вирішується завдяки покращенню алгоритму рекомендаційної системи, покращенню функціоналу та інтерфейсу. Також система випереджає аналог за такими параметрами як кількість можливих субтитрів.

Одиничний параметричний індекс розраховуємо за формулою [Козловський, Лесько, Кавецький]:

$$q_i = \frac{P_i}{P_{баз\,i}}, \tag{4.1}$$

де q_i – одиничний параметричний індекс, розрахований за i-м параметром; P_i – значення i-го параметра виробу; $P_{базі}$ – аналогічний параметр базового виробу-аналога, з яким проводиться порівняння.

Загальні технічні та економічні характеристики розробки представлено в таблиці 4.4.

Таблиця 4.4 – Основні техніко-економічні показники аналога та розробки, що проектується

Показник	Варіанти		Відносний показник якості	Коефіцієнт вагомості параметра
	Базовий (товар-конкурент)	Новий (інноваційне рішення)		
1	2	3	4	5
Мова субтитрів	1	2	2	50%
Мова відтворення фільму	1	3	3	20%
Якість відтворення фільму	3	8	2,7	30%

Нормативні параметри оцінюємо показником, який отримує одне з двох значень: 1 – пристрій відповідає нормам і стандартам; 0 – не відповідає.

Груповий показник конкурентоспроможності за нормативними параметрами розраховуємо як добуток частинних показників за кожним параметром за формулою [Козловський, Лесько, Кавецький]:

$$I_{НП} = \prod_{i=1}^{n} q_i \quad , \tag{4.2}$$

де $I_{нп}$ – загальний показник конкурентоспроможності за нормативними параметрами; q_i – одиничний (частинний) показник за i-м нормативним параметром; n – кількість нормативних параметрів, які підлягають оцінюванню.

За нормативними параметрами розроблюваний пристрій відповідає вимогам ДСТУ, тому $I_{нп} = 1$.

Значення групового параметричного індексу за технічними параметрами визначаємо з урахуванням вагомості (частки) кожного параметра [Козловський, Лесько, Кавецький]:

$$I_{ТП} = \sum_{i=1}^{n} q_i \cdot \alpha_i \quad , \tag{4.3}$$

де $I_{тп}$ – груповий параметричний індекс за технічними показниками (порівняно з виробом-аналогом); q_i – одиничний параметричний показник i-го параметра; α_i – вагомість i-го параметричного показника, $\sum_{i=1}^{n} \alpha_i = 1$; n – кількість технічних параметрів, за якими оцінюється конкурентоспроможність.

Проведемо аналіз параметрів згідно даних таблиці 4.4.

$$I_{mn} = 2 \cdot 0{,}5 + 3 \cdot 0{,}2 + 2{,}7 \cdot 0{,}3 = 2{,}41.$$

Груповий параметричний індекс за економічними параметрами розраховуємо за формулою [Козловський, Лесько, Кавецький]:

$$I_{ЕП} = \sum_{i=1}^{m} q_i \cdot \beta_i \quad , \tag{4.4}$$

де $I_{ЕП}$ – груповий параметричний індекс за економічними показниками; q_i – економічний параметр i-го виду; β_i – частка i-го економічного параметра, $\sum_{i=1}^{m} \beta_i = 1$; m – кількість економічних параметрів, за якими здійснюється оцінювання.

Проведемо аналіз параметрів згідно даних таблиці .

$$I_{ЕП}=0,75 \cdot 0,5 + 0,86 \cdot 0,5 = 0,80.$$

На основі групових параметричних індексів за нормативними, технічними та економічними показниками розрахуємо інтегральний показник конкурентоспроможності за формулою [Козловський, Лесько, Кавецький]:

$$K_{ІНТ} = I_{НП} \cdot \frac{I_{ТП}}{I_{ЕП}}, \quad (4.5)$$

$$K_{ІНТ} = 1 \cdot 2,41 / 0,80 = 3,01.$$

Інтегральний показник конкурентоспроможності $К_{ІНТ} > 1$, отже розробка переважає відомі аналоги за своїми техніко-економічними показниками.

4.3 Розрахунок витрат на проведення науково-дослідної роботи

Витрати, пов’язані з проведенням науково-дослідної роботи на тему "Інформаційна технологія стримінгового сервісу для перегляду фільмів", під час планування, обліку і калькулювання собівартості науково-дослідної роботи групуємо за відповідними статтями.

4.3.1 Витрати на оплату праці

До статті «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.

Основна заробітна плата дослідників

Витрати на основну заробітну плату дослідників ($З_о$) розраховуємо у відповідності до посадових окладів працівників, за формулою [Козловський, Лесько, Кавецький]:

$$З_о = \sum_{i=1}^{k} \frac{M_{ni} \cdot t_i}{T_p}, \quad (4.6)$$

де k – кількість посад дослідників залучених до процесу досліджень; M_{ni} – місячний посадовий оклад конкретного дослідника, грн; t_i – число днів роботи конкретного дослідника, дн.; T_p – середнє число робочих днів в місяці, T_p=21 дні.

$$З_о = 17500 \cdot 55 / 21 = 3977 \; грн.$$

Проведені розрахунки зведемо до таблиці.

Таблиця 4.5 – Витрати на заробітну плату дослідників

Найменування посади	Місячний посадовий оклад, грн	Оплата за робочий день, грн	Число днів роботи	Витрати на заробітну плату, грн
Керівник проекту	17500	795,5	5	3977
Інженер-дослідник (програміст)	16000	727,3	44	32000
Всього				35977

Основна заробітна плата робітників

Витрати на основну заробітну плату робітників ($З_р$) за відповідними найменуваннями робіт НДР на тему "Інформаційна технологія стримінгового сервісу для перегляду фільмів" розраховуємо за формулою:

$$З_p = \sum_{i=1}^{n} C_i \cdot t_i \quad , \tag{4.7}$$

де C_i – погодинна тарифна ставка робітника відповідного розряду, за виконану відповідну роботу, грн/год; t_i – час роботи робітника при виконанні визначеної роботи, год.

Погодинну тарифну ставку робітника відповідного розряду C_i можна визначити за формулою:

$$C_i = \frac{M_M \cdot K_i \cdot K_c}{T_p \cdot t_{зм}} \quad , \tag{4.8}$$

де M_M – розмір прожиткового мінімуму працездатної особи, або мінімальної місячної заробітної плати (в залежності від діючого законодавства), приймемо $M_м$=6500 грн;K_i – коефіцієнт міжкваліфікаційного співвідношення для встановлення тарифної ставки робітнику відповідного розряду (табл. Б.2, додаток Б) [Козловський, Лесько, Кавецький]; K_c – 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.

T_p – середнє число робочих днів в місяці, приблизно T_p = 21 дн; $t_{зм}$ – тривалість зміни, год. $C_1 = 6500{,}00 \cdot 1 \cdot 1{,}65 / (21 \cdot 8) = 65{,}8$ грн. $З_{р1} = 65{,}8 \cdot 75 = 4935{,}3$ грн.

Таблиця 4.6 – Величина витрат на основну заробітну плату робітників

Найменування робіт	Тривалість роботи, год	Розряд роботи	Погодинна тарифна ставка, грн	Величина оплати на робітника грн
1.Підготовчі	75	1	65,8	4935,3
2.Тестувальні	50	1	65,8	3290,2
Всього	8225,4			

Додаткова заробітна плата дослідників та робітників

Додаткову заробітну плату розраховуємо як 10 … 12% від суми основної заробітної плати дослідників та робітників за формулою:

$$З_{дод} = (З_о + З_р) \cdot \frac{Н_{дод}}{100\%}, \quad (4.9)$$

де $Н_{дод}$ – норма нарахування додаткової заробітної плати. Приймемо 11%.

$З_{дод}$ = (35977+ 8225,4) · 11 / 100% = 4862,3 грн.

4.3.2 Відрахування на соціальні заходи

Нарахування на заробітну плату дослідників та робітників розраховуємо як 22% від суми основної та додаткової заробітної плати дослідників і робітників за формулою:

$$З_н = (З_о + З_р + З_{дод}) \cdot \frac{Н_{зп}}{100\%} \quad (4.10)$$

де $Н_{зп}$ – норма нарахування на заробітну плату. Приймаємо 22%; $Зн$ = (35977+ 8225,4+4862,3) · 22 / 100% = 10794,3 грн.

4.3.3 Сировина та матеріали

До статті «Сировина та матеріали» належать витрати на сировину, основні та допоміжні матеріали, інструменти, пристрої та інші засоби і предмети праці, які придбані у сторонніх підприємств, установ і організацій та витрачені на проведення досліджень за темою "Інформаційна технологія стримінгового сервісу для перегляду фільмів".

Витрати на матеріали ($М$), у вартісному вираженні розраховуються окремо по кожному виду матеріалів за формулою:

$$M = \sum_{j=1}^{n} H_j \cdot Ц_j \cdot К_j - \sum_{j=1}^{n} B_j \cdot Ц_{в\,j}, \quad (4.11)$$

де H_j – норма витрат матеріалу j-го найменування, кг; n – кількість видів матеріалів; $Ц_j$ – вартість матеріалу j-го найменування, грн/кг; $К_j$ – коефіцієнт транспортних витрат, ($К_j$ = 1,1 … 1,15); B_j – маса відходів j-го найменування, кг; $Ц_{вj}$ – вартість відходів j-го найменування, грн/кг.

Проведені розрахунки зведемо до таблиці.

Таблиця 4.7 – Витрати на матеріали

Найменування матеріалу, марка, тип, сорт	Ціна за 1 кг, грн	Норма витрат, кг	Вартість витраченого матеріалу, грн
Папір канцелярський офісний (А4) BUROMAX Premium	174,9	1	174,9
Папір для заміток (А5) BUROMAX Premium Light	110	1	110
Начиння канцелярське BUROMAX Premium	190	1	190
Всього			474,9
З врахуванням коефіцієнта транспортування			522,39

4.3.4 Розрахунок витрат на комплектуючі

Витрати на комплектуючі *($К_в$)*, які використовують при проведенні НДР на тему «Інформаційна технологія стримінгового сервісу для перегляду фільмів» не потрібні.

4.3.5 Спецустаткування для наукових (експериментальних) робіт

До статті «Спецустаткування для наукових (експериментальних) робіт» належать витрати на виготовлення та придбання спецустаткування необхідного для проведення досліджень, також витрати на їх проектування, виготовлення, транспортування, монтаж та встановлення. В роботі спецустаткування не використовувалося.

4.3.6 Програмне забезпечення для наукових (експериментальних) робіт

До статті «Програмне забезпечення для наукових (експериментальних) робіт» належать витрати на розробку та придбання спеціальних програмних засобів і програмного забезпечення, (програм, алгоритмів, баз даних) необхідних для проведення досліджень, також витрати на їх проектування, формування та встановлення. Програмне забезпечення, яке було використано в роботі було безкоштовним.

4.3.7 Амортизація обладнання, програмних засобів та приміщень

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

$$\mathrm{А_{обл}} = \frac{\mathrm{Ц_б}}{\mathrm{Т_в}} \cdot \frac{t_{\mathrm{вик}}}{12}, \tag{4.12}$$

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

місяців; $T_в$ – строк корисного використання обладнання, програмних засобів, приміщень тощо, років.

$A_{обл} = (50000 \cdot 2) / (2 \cdot 12) = 4166{,}67$ *грн.*

Проведені розрахунки зведемо до таблиці 4.8.

Таблиця 4.8 – Амортизаційні відрахування по кожному виду обладнання

Найменування обладнання	Балансова вартість, грн	Строк корисного використання, років	Термін використання обладнання, місяців	Амортизаційні відрахування, грн
Персональний комп'ютер Core I3-2700	50000	2	2	4166,67
Приміщення	248700	20	2	2072,50
Всього				6239,17

4.3.8 Паливо та енергія для науково-виробничих цілей

Витрати на силову електроенергію ($B_е$) розраховуємо за формулою:

$$\text{В}_\text{е} = \sum_{i=1}^{n} \frac{W_{\text{уі}} \cdot t_i \cdot \text{Ц}_\text{е} \cdot \text{К}_{\text{впі}}}{\eta_i}, \qquad (4.13)$$

де W_{yi} – встановлена потужність обладнання на визначеному етапі розробки, кВт; t_i – тривалість роботи обладнання на етапі дослідження, год; $Ц_е$ – вартість 1 кВт-години електроенергії, грн; (вартість електроенергії визначається за даними енергопостачальної компанії), приймемо $\text{Ц}_\text{е} = 7{,}5$ грн; $К_{впі}$ – коефіцієнт, що враховує використання потужності, $К_{впі} < 1$; η_i – коефіцієнт корисної дії обладнання, $\eta_i < 1$. $\text{В}_\text{е} = 0{,}25 \cdot 280{,}0 \cdot 7{,}5 \cdot 0{,}5 / 0{,}8 = 328{,}13$ грн.

4.3.9 Службові відрядження

До статті «Службові відрядження» дослідної роботи на тему "Інформаційна технологія стримінгового сервісу для перегляду фільмів" належать витрати на відрядження штатних працівників, працівників організацій, які працюють за договорами цивільно-правового характеру, аспірантів, зайнятих розробленням досліджень, відрядження, пов'язані з проведенням випробувань машин та приладів, а також витрати на відрядження на наукові з'їзди, конференції, наради, пов'язані з виконанням конкретних досліджень.

Витрати за статтею «Службові відрядження» розраховуємо як 20…25% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{св} = (З_о + З_р) \cdot \frac{Н_{св}}{100\%}, \quad (4.14)$$

де $Н_{св}$ – норма нарахування за статтею «Службові відрядження», приймемо $Н_{св}$ = 20%.

$В_{св}$ = (35977+ 8225,4) · 20 / 100% = 8840,54 грн.

4.3.10 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» розраховуємо як 50…100% від суми основної заробітної плати дослідників та робітників за формулою:

$$I_в = (З_о + З_р) \cdot \frac{Н_{ів}}{100\%}, \quad (4.15)$$

де $Н_{ів}$ – норма нарахування за статтею «Інші витрати», приймемо $Н_{ів}$ = 50%.

$І_в$ = (35977+ 8225,4) · 50 / 100% = 22101,36 грн.

4.3.11 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» розраховуємо як 100…150% від суми основної заробітної плати дослідників та робітників за формулою:

$$В_{нзв} = (З_о + З_р) \cdot \frac{Н_{нзв}}{100\%}, \quad (4.16)$$

де $Н_{нзв}$ – норма нарахування за статтею «Накладні (загальновиробничі) витрати», приймемо $Н_{нзв}$ = 100%.

$В_{нзв}$ = (35977+ 8225,4) · 100 / 100% = 44202,72 грн.

Витрати на проведення науково-дослідної роботи на тему "Інформаційна технологія стримінгового сервісу для перегляду фільмів". розраховуємо як суму всіх попередніх статей витрат за формулою:

$$В_{заг} = З_о + З_р + З_{дод} + З_н + М + К_в + В_{спец} + В_{прг} + А_{обл} + В_е + В_{св} + В_{сп} + І_в + В_{нзв}. \quad (4.17)$$

$В_{заг}$=35977+8225,4+4862,3+10794,3+522,39+6239,17+328,14+8840,54+22101,36 +44202,72= 142093,63 грн.

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

$$ЗВ = \frac{В_{заг}}{\eta}, \quad (4.18)$$

де η - коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи, приймемо η =0,7.

ЗВ = 142093,63 / 0,7 = 202990,89 грн.

4.4 Розрахунок економічної ефективності науково-технічної розробки при її можливій 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.

Результати дослідження проведені за темою "Інформаційна технологія стримінгового сервісу для перегляду фільмів" передбачають комерціалізацію протягом 3-х років реалізації на ринку.

У цьому випадку основу майбутнього економічного ефекту будуть формувати:

ΔN – збільшення кількості споживачів яким надається відповідна інформаційна послуга у періоди часу, що аналізуються;

N – кількість споживачів яким надавалась відповідна інформаційна послуга у році до впровадження результатів нової науково-технічної розробки, приймемо 1 особа

$Ц_б$ – вартість послуги у році до впровадження інформаційної системи, приймемо 1000,00 грн;

$\pm\Delta Ц_о$ – зміна вартості послуги від впровадження результатів, приймемо зростання на 300,00 грн.

Можливе збільшення чистого прибутку у потенційного інвестора $\Delta П_i$ для кожного із 3-х років, протягом яких очікується отримання позитивних результатів від можливого впровадження та комерціалізації науково-технічної розробки, розраховуємо за формулою [Козловський, Лесько, Кавецький]:

$$\Delta П_i = (\pm \Delta Ц_o \cdot N + Ц_o \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\vartheta}{100}), \quad (4.19)$$

де λ – коефіцієнт, який враховує сплату потенційним інвестором податку на додану вартість. У 2021 році ставка податку на додану вартість складає 20%, а коефіцієнт $\lambda = 0,8333$; ρ – коефіцієнт, який враховує рентабельність інноваційного продукту). Приймемо $\rho = 40\%$; ϑ – ставка податку на прибуток, який має сплачувати потенційний інвестор, у 2023 році $\vartheta = 18\%$;

Збільшення чистого прибутку 1-го року:

$\Delta П_1 =$ (1·300+1000·1000)·0,83·0,4·(1-0,18/100%)=222125,7 грн.

Збільшення чистого прибутку 2-го року:

$\Delta П_2 =$ (1·300+1000·(1000+800))·0,83·0,4·(1-0,18/100%)=400034,01 грн.

Збільшення чистого прибутку 3-го року:

$\Delta П_3 =$ (1·300+1000·(1000+800+600))·0,83·0,4·(1-0,18/100%)=533278,68 грн.

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

$$ПП = \sum_{i=1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \quad (4.20)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному з років, протягом яких виявляються результати впровадження науково-технічної розробки, грн; T – період часу, протягом якого очікується отримання позитивних результатів від

впровадження та комерціалізації науково-технічної розробки, роки; τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, τ=18%; t – період часу (в роках) від моменту початку впровадження науково-технічної розробки до моменту отримання потенційним інвестором додаткових чистих прибутків у цьому році.

$ПП$ =222125,7 /(1+0,18)^1+400034,01 /(1+0,18)^2+533278,68 /(1+0,18)^3=

= 7721951,89 грн.

Величина початкових інвестицій PV, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки:

$$PV = k_{інв} \cdot ЗВ, \tag{4.21}$$

де $k_{інв}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію, приймаємо $k_{інв}$=2; $ЗВ$ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, приймаємо 202990,89 грн.

$PV = k_{інв} \cdot ЗВ$ = 2 · 202990,89 = 405981,79 грн.

Абсолютний економічний ефект $E_{абс}$ для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме:

$$E_{абс} = ПП - PV \tag{4.22}$$

де $ПП$ – приведена вартість зростання всіх чистих прибутків від можливого впровадження та комерціалізації науково-технічної розробки, 7721951,89 грн; PV – теперішня вартість початкових інвестицій, 405981,79 грн.

$E_{абс} = ПП - PV$ = 7721951,89 - 405981,79 = 366970,09 грн.

Внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$E_в = \sqrt[Tж]{1 + \frac{E_{абс}}{PV}} - 1, \quad (4.23)$$

де $E_{абс}$ – абсолютний економічний ефект вкладених інвестицій, грн; PV – теперішня вартість початкових інвестицій, грн; $T_ж$ – життєвий цикл науково-технічної розробки, тобто час від початку її розробки до закінчення отримування позитивних результатів від її впровадження, 3 роки.

$$E_в = \sqrt[Tж]{1 + \frac{E_{абс}}{PV}} - 1 = (1+366970,09\ /405981,79)^{1/3} - 1 = 0,41.$$

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

$$\tau_{мін} = d + f, \quad (4.24)$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2023 році в Україні d =0,1; f – показник, що характеризує ризикованість вкладення інвестицій, приймемо 0,25. $\tau_{мін}$ = 0,1+0,25 = 0,35 < 0,41 свідчить про те, що внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки вища мінімальної внутрішньої дохідності. Тобто інвестувати в науково-дослідну роботу за темою «Інформаційна технологія онтологічного моделювання бази знань з організації бібліотеки» доцільно.

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

$$T_{ок} = \frac{1}{E_в}, \quad (4.25)$$

де $E_в$ – внутрішня економічна дохідність вкладених інвестицій.

$T_{ок} = 1 / 0{,}41 = 2{,}4$ р.

$T_{ок} <$ 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

4.5 Висновки до розділу 4

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою "Інформаційна технологія стримінгового сервісу для перегляду фільмів" становить 33 бали, що, свідчить про комерційну важливість проведення даних досліджень оскільки рівень комерційного потенціалу розробки вище середнього.

При оцінюванні рівня конкурентоспроможності, згідно узагальненого коефіцієнту конкурентоспроможності розробки, науково-технічна розробка переважає існуючі аналоги приблизно в 3,01 рази.

Також термін окупності становить 2,4 роки, що менше 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

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

ВИСНОВКИ

Під час виконання магістерської кваліфікаційної роботи була поставлена задача створення програмний модуль рекомендаційної системи стримінгового сервісу для фільмів.

Під час виконання поставленої задачі було виконано аналіз предметної області та об’єкту дослідження й проведено аналіз існуючих систем-аналогів інформаційної системи стримінгового сервісу для перегляду відео та їх характеристик.

Також було розроблено модель інформаційної системи стримінгового сервісу для перегляду відеопродукції, метод надання рекомендацій у стримінгових сервісів для перегляду відеопродукції та алгоритм роботи системи підбору фільмів за вихідними даними.

Виконано програмну реалізацію стримінгового сервісу для фільмів й проведено тестування програмного забезпечення та проаналізувати отримані результати.

У результаті виконання останнього розділу проведено розрахунок економічної ефективності науково-технічної розробки за можливого її застосування.

Додаток було проаналізовано згідно з функціональними характеристиками аналогів й проведено його тестування. Стримінговий сервіс для фільмів повністю відповідає поставленим цілям та вимогам.

Програмна реалізація додатку відбувалась у середовищі Visual Studio Code й за допомогою фреймворку ReactJS, із використанням Firebase й TVMAZE API.

Мету роботи досягнуто та поставлені задачі виконані у повному обсязі.

ПЕРЕЛІК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Від Netflix до «Такфлікс»: великий гід стримінговими сервісами. The Village Україна – [Електронний ресурс]. – Режим доступу: https://www.the-village.com.ua/village/culture/tv/293007-tv-netflix-amazon-disney-plus-apple-tv-plus-hbo-max-peacock-streaming.
2. Федорова В.В. Бібліотека React як ефективний інструмент розробки WEB-додатків / Я. В. Іванчук, В.В. Федорова // Матеріали конференції «LII Науково-технічна конференція факультету інтелектуальних інформаційних технологій та автоматизації (2023)»: Тез. доп. – м. Вінниця, Україна, 2023.
3. Федорова В.В. Розробка ВЕБ-додатку стримінгового сервісу для фільмів / В. С. Озеранський, В. В. Федорова // Матеріали конференції «LI Науково-технічна конференція факультету інтелектуальних інформаційних технологій та автоматизації (2022)»: Тез. доп. – м. Вінниця, Україна, 2022.
4. Файли і потоки, буферизація даних. StudFiles – [Електронний ресурс]. – Режим доступу: https://studfile.net/preview/7347055/page:2.
5. Krikke, J. «Streaming video transforms the media industry» (IEEE Computer Graphics and Applications). 2004, 537 p.
6. Speed for the streaming services – [Електронний ресурс]. – Режим доступу:https://www.allconnect.com/blog/how-much-speed-do-i-need-for-streaming.
7. Digital Rights Management – [Електронний ресурс]. – Режим доступу: https://www.britannica.com/topic/digital-rights-mana.
8. Netflix – [Електронний ресурс]. – Режим доступу: https://www.netflix.com/.
9. Hulu – [Електронний ресурс]. – Режим доступу: https://www.hulu.com/.
10. MEGOGO – [Електронний ресурс]. – Режим доступу: https://megogo.net/ua.
11. YouTube – [Електронний ресурс]. – Режим доступу: https://www.youtube.com/.

12. Клієнт-серверна архітектура та ролі серверів – [Електронний ресурс]. – Режим доступу: https://medium.com/@IvanZmerzlyi/клієнт-серверна-архітектура-та-ролі-серверів-9893d8048229.
13. Berson A. (1996). «Client/Server Architecture» (McGraw-Hill Computer Communications Series). 569 p.
14. Borrie H. «The Firebird Book: A Reference for Database Developers» (Apress) 2004, 1161p.
15. M. Morris Mano (1992). «Computer System Architecture» (Pearson), 544p.
16. Інформаційне наповнення сайту – [Електронний ресурс]. – Режим доступу: https://e-tk.lntu.edu.ua/mod/resource/ view.php?id=1200.
17. Firebase – [Електронний ресурс]. – Режим доступу: https://firebase.google.com/.
18. Firebase Authentication – [Електронний ресурс]. – Режим доступу: https://publisher.support.cleeng.com/hc/en-us/articles/4406603357074-Firebase-Authentication.
19. Хмарне сховище Firebase – [Електронний ресурс]. – Режим доступу: https://firebase.google.com/docs/storage.
20. Аутентифікація і авторизація: що це і в чому відмінність – [Електронний ресурс] – Режим доступу: https://qagroup.com.ua/publications/autentyfikatciia-i-avtoryzatciia/.
21. Haverbeke M. «Eloquent JavaScript» (No Starch Press) 2018, 472p.
22. 10 Best JavaScript Frameworks to Use in 2022 – [Електронний ресурс]. – Режим доступу: https://hackr.io/blog/best-javascript-frameworks.
23. Smith, J. «Streaming Technologies and Their Impact on the Film Industry» (International Journal of Film Studies), 2020, 635p.
24. Wang, L. «User Privacy and Data Protection in Streaming Services: Legal and Ethical Considerations» (Journal of Information Privacy), 2020, 1115p.
25. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт / Уклад. : В. О. Козловський, О. Й. Лесько, В. В. Кавецький. – Вінниця : ВНТУ, 2021. – 42 с.

ДОДАТКИ

Додаток А (обов'язковий)

Протокол перевірки кваліфікаційної роботи на наявність текстових запозичень

Назва роботи: Інформаційна технологія стримінгового сервісу для перегляду фільмів

Тип роботи: магістерська кваліфікаційна робота

(БДР, МКР)

Підрозділ кафедра комп'ютерних наук, ФІТА

(кафедра, факультет)

Показники звіту подібності Unicheck

Оригінальність 83,0% Схожість 17,0%

Аналіз звіту подібності (відмітити потрібне):

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

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

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

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

Автор роботи Федорова В.В.

Керівник роботи Іванчук Я.В.

Опис прийнятого рішення

Магістерську кваліфікаційну роботу допущено до захисту

Особа, відповідальна за перевірку Озеранський В.С.

Додаток Б (обов'язковий)

Лістинг програми

```
<!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>
    </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 } from "react";
import "./Row.css";
import { useHistory } from "react-router-dom";
import axios from "../../config/axios/axios";

const Row = ({ title, fetchUrl, value }) => {
  const [movies, setMovies] = useState([]);
  const history = useHistory();

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

  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 ПЕРЕГЛЯДУ ФІЛЬМІВ

Виконала: студентка 2-го курсу, групи 2КН-22м спеціальності 122 «Комп'ютерні науки»

_____ Федорова В.В.
(прізвище та ініціали)

Керівник: д. т. н., проф.

_____ Іванчук Я.В.
(прізвище та ініціали)

«07» 12 2023 р.

Вінниця ВНТУ - 2023 рік

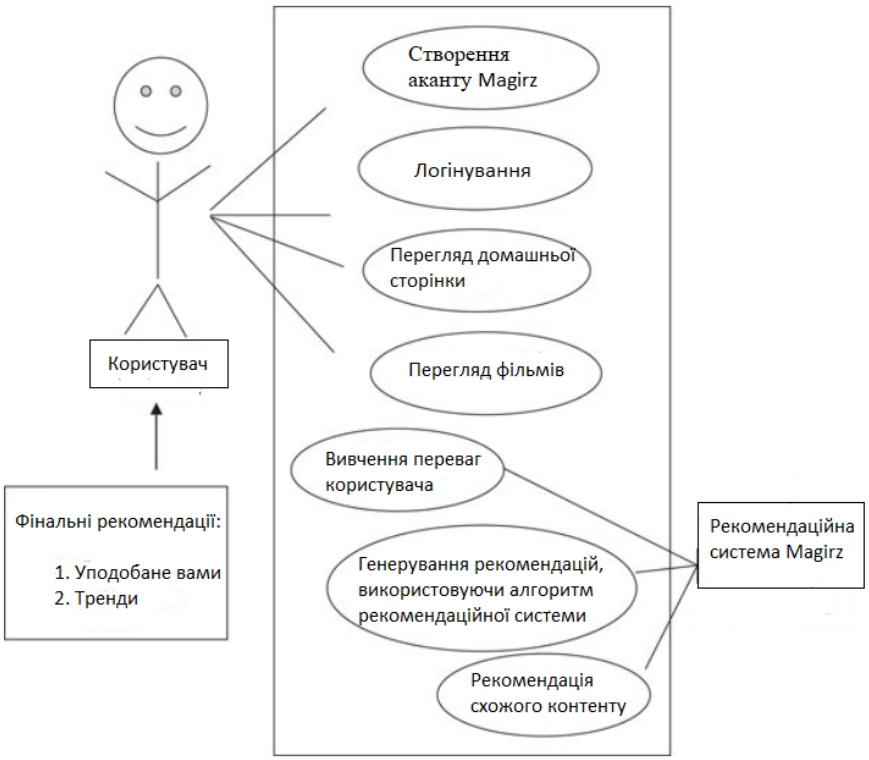

Рисунок В.1 – Схема моделі інформаційної системи стримінгового сервісу для фільмів

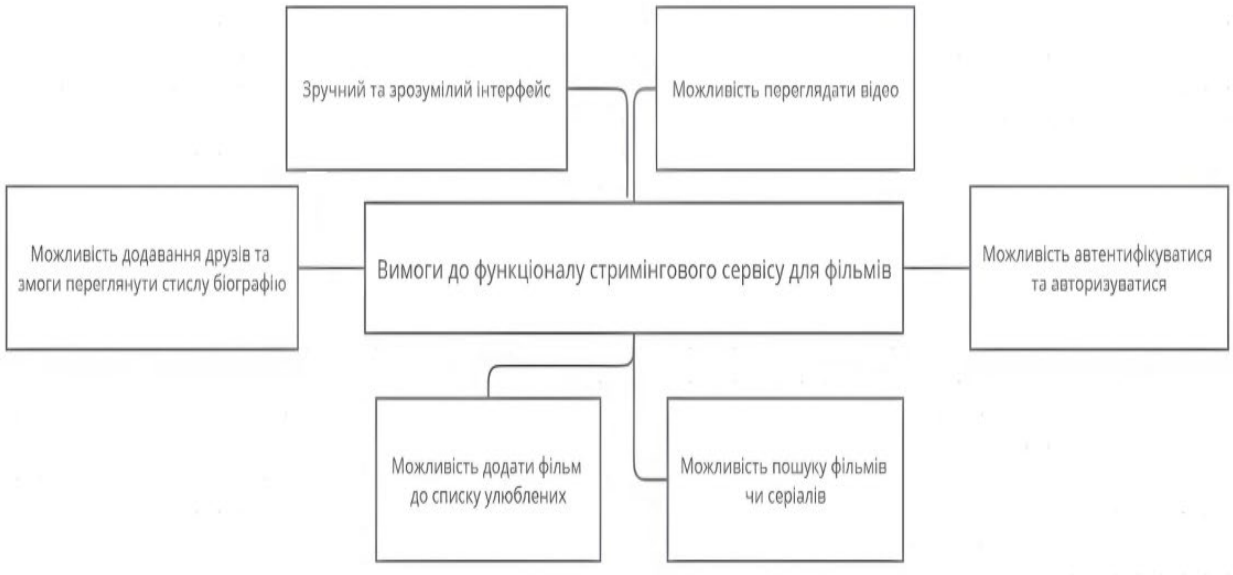

Рисунок В.2 – Схема моделі вимог до інформаційної системи стримінгового сервісу для фільмів

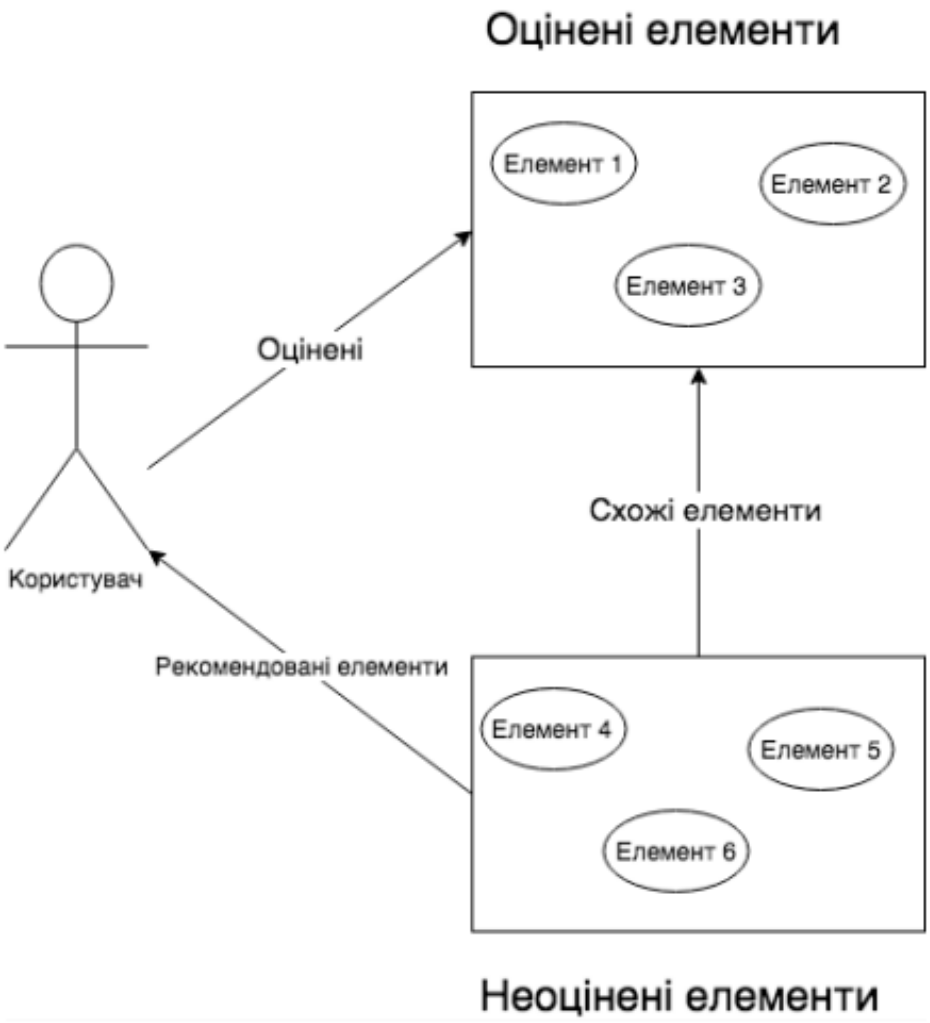

Рисунок В.3 – Принципова схема надання рекомендацій, побудованої на основі аналізу контенту

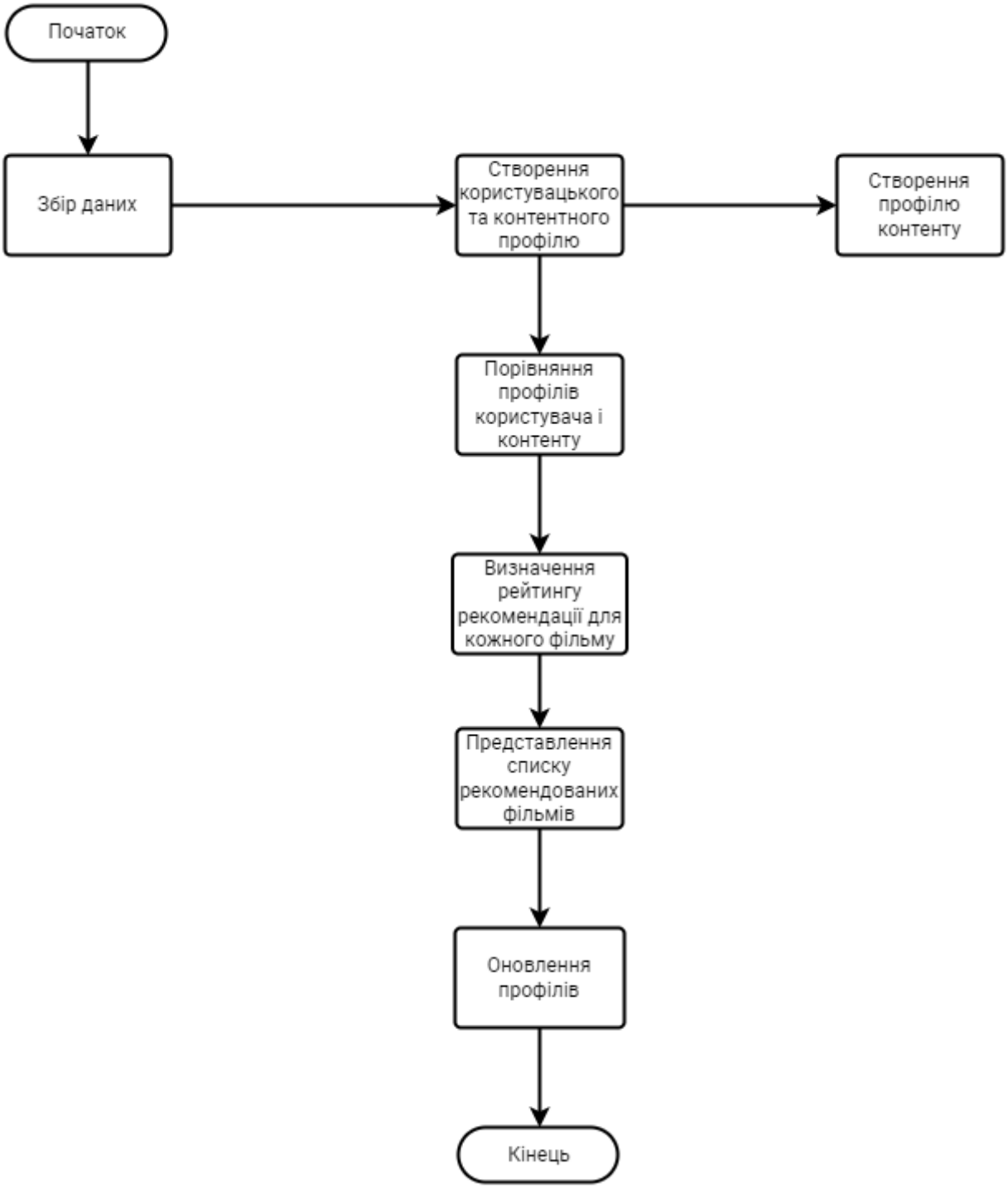

Рисунок В.4 – Схема алгоритму загального функціонування стримінгового сервісу для фільмів

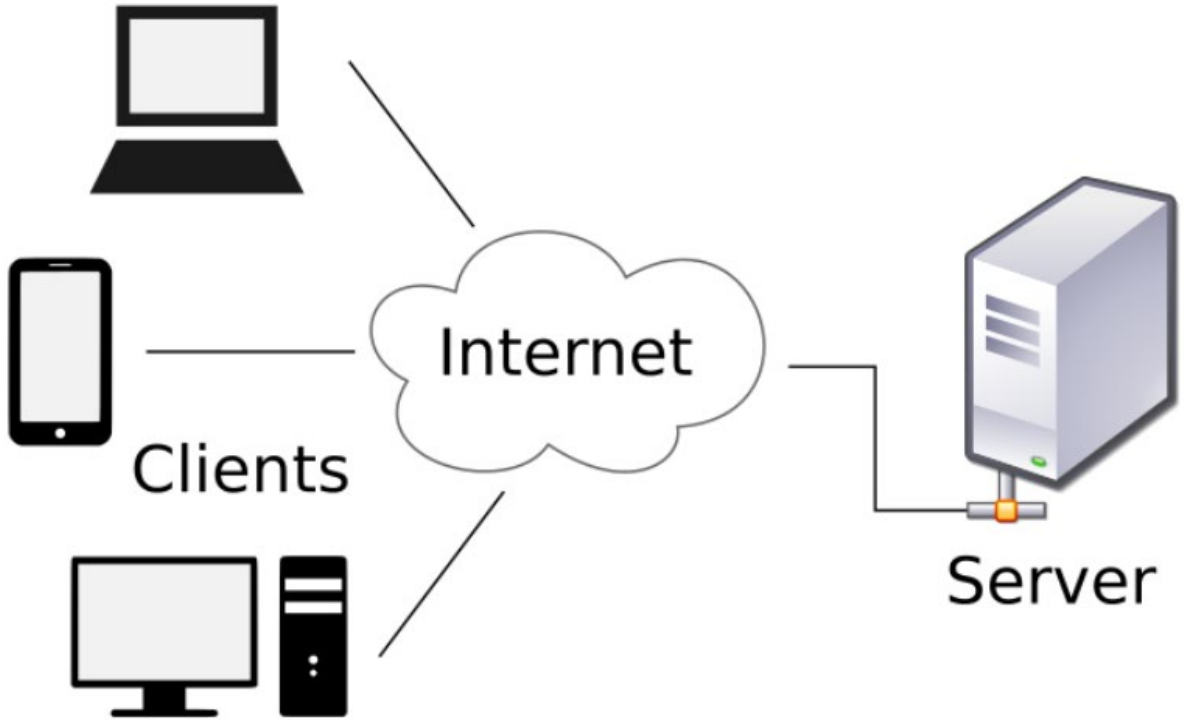

Рисунок В.5 – Схема клієнт-серверної архітектури

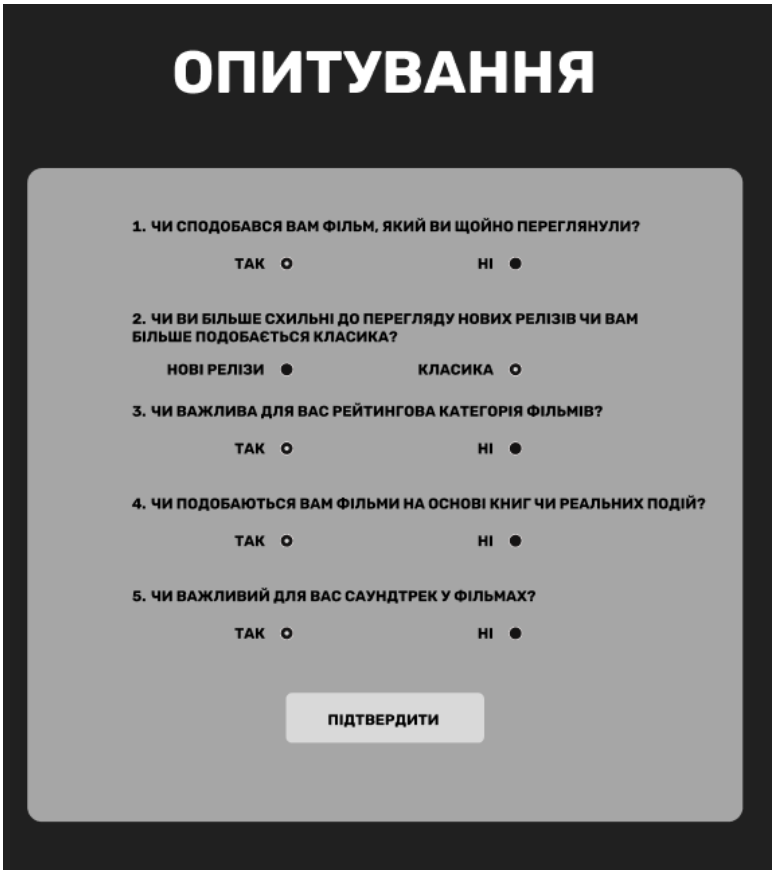

Рисунок В.6 – Опитування користувача для першого тесту

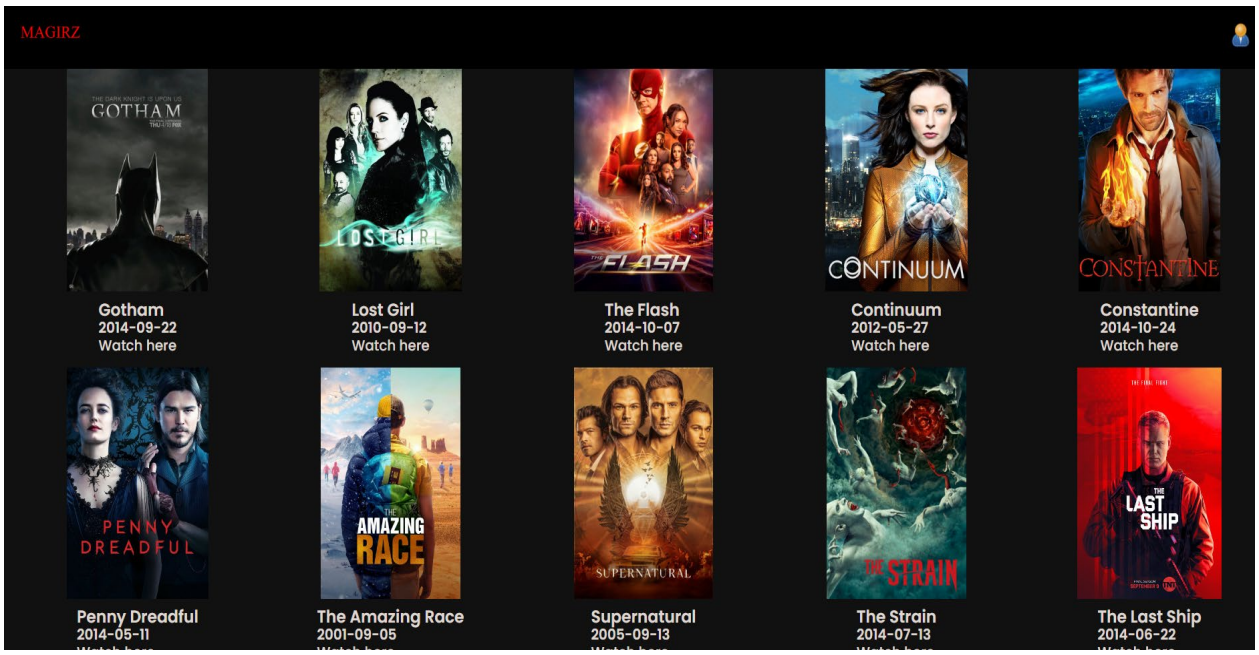

Рисунок В,7 – Загальний вигляд інтерфейсного вікна рекомендованих фільмів, на основі фільтрації після першого тесту

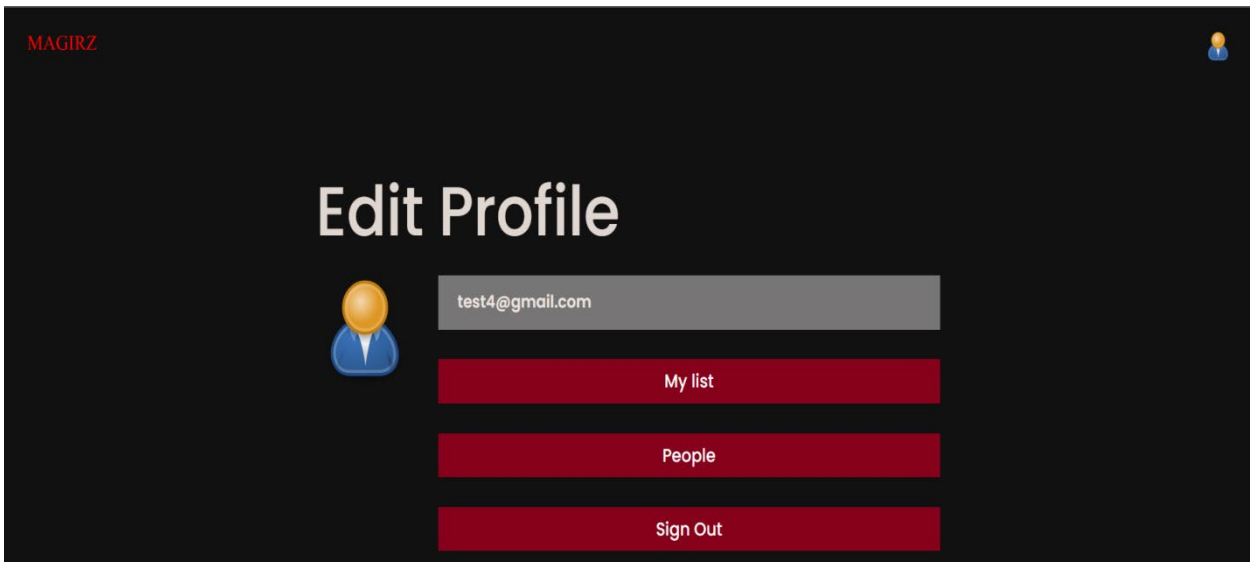

Рисунок В.8 – Загальний вигляд інтерфейсного вікна сторінки власного профілю

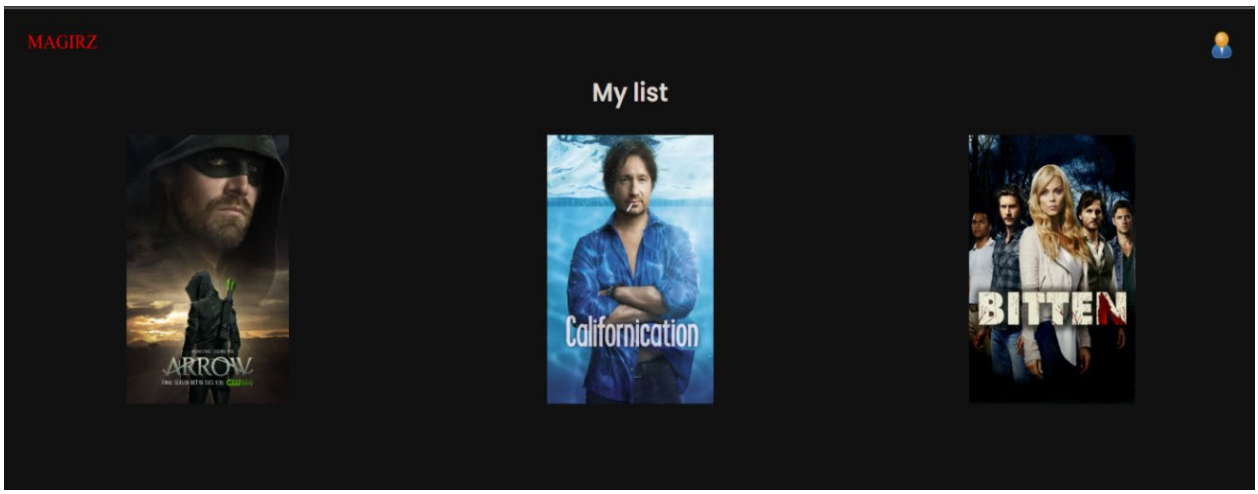

Рисунок В.9 – Загальний вигляд інтерфейсного вікна сторінки уподобаних фільмів

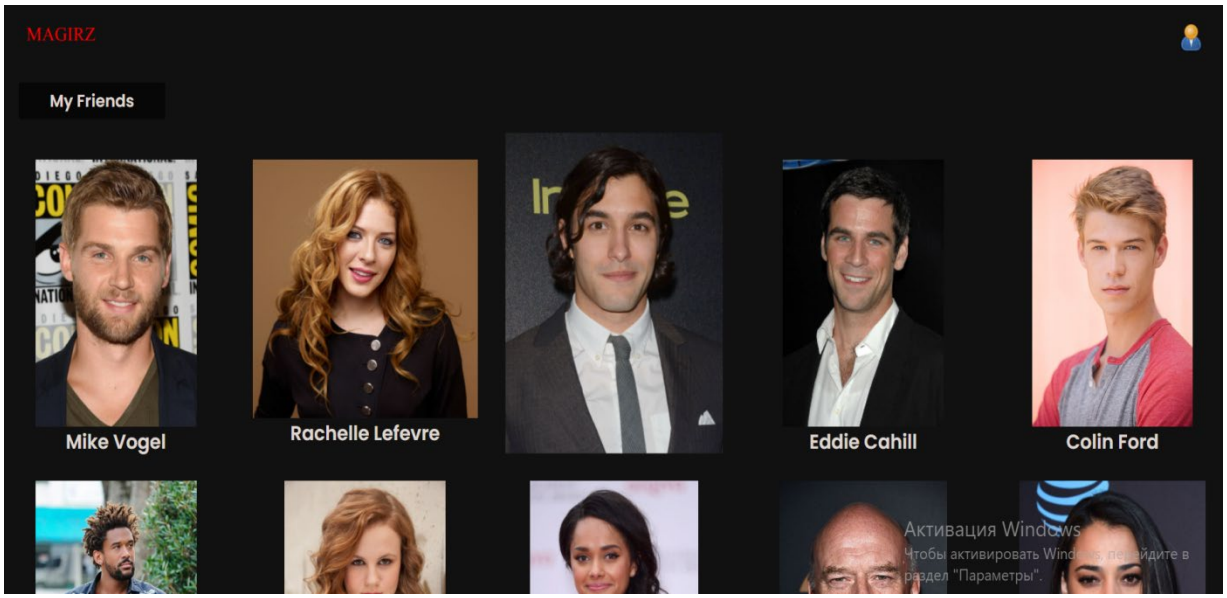

Рисунок В.10 – Загальний вигляд інтерфейсного вікна сторінки акторів та людей

Додаток Г (довідниковий)
Інструкція користувача

Для початку роботи зі стримінговим сервісом для фільмів необхідно мати підключення до інтернету з встановленим браузером.

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 ви зможете побачити сторінку власного профілю.

Після цього ви зможете переглянути сторінку з фільмами й відкрити сторінку пошуку.

Обравши уподобаний фільм, ви можете перейти на його сторінку й переглянути інформацію або додати до списку улюблених на профілі, що ви бачили вище.

Після перегляду певної кількості фільмів, яка визначається алгоритмом, вам 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 контенту. Серед таких питань було безліч, що майже не впливали на думку людини й вважаються безкорисними, що було відсіяно під час дослідження для покращення й розробки функції рекомендацій фільмів.

Також є можливість шукати людей, переглядати інформацію про них та додавати їх до друзів.