КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
імені ТАРАСА ШЕВЧЕНКА
Факультет інформаційних технологій
Кафедра прикладних інформаційних систем

122 «Комп'ютерні науки»
(шифр і назва спеціальності)

«Прикладне програмування»
(назва освітньої програми)

Кваліфікаційна робота бакалавра

на тему: «Веб-застосунок для підбору фільмів із урахуванням вподобань»

Виконав _____
(Підпис)
Потреба Руслан Сергійович
(прізвище, ім'я, по батькові)

Керівник _Бойко Юлія Петрівна
(прізвище, ім'я, по батькові)

_____
(Резолюція «До захисту»)

Попередній захист:

_____
(Висновок: "До захисту в екзаменаційній комісії ")

Завідувач кафедри _____ Плескач В.Л.
(Підпис) (Прізвище, ініціали) (Дата)

Київ – 2021

Київський національний університет імені Тараса Шевченка
Кафедра прикладних інформаційних систем

Назва теми: «Веб-застосунок для підбору фільмів із урахуванням вподобань»

Освітня програма: Прикладне програмування
Спеціальність: Комп'ютерні науки

ПІБ	Підпис
Потреба Руслан Сергійович	

Назва роботи українською та англійською мовами

Веб-застосунок для підбору фільмів із урахуванням вподобань

Web application for choosing movies based on preferences

Мета бакалаврської кваліфікаційної роботи, завдання

Мета роботи: Підвищення якості підбору відеоконтенту із урахуванням вподобань користувача, шляхом розробки веб-застосунку підбору фільмів.

План роботи:

1. Проаналізувати сучасні підходи до розроблення і впровадження веб-застосунків
2. Проаналізувати архітектурні рішення та здійснити вибір програмних засобів для реалізації веб-застосунку
3. Здійснити програмну реалізацію веб-застосунку для підбору фільмів із урахуванням вподобань

ПІБ, ступінь, звання наукового керівника роботи:

_____

КАЛЕНДАРНИЙ ПЛАН ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ БАКАЛАВРА

Номер	Назва етапів кваліфікаційної роботи бакалавра	Термін виконання етапів кваліфікаційної роботи бакалавра	Відмітка про виконання
1.	Вибір теми та наукового керівника кваліфікаційної роботи бакалавра	26.10.2020	
2.	Видача завдання кваліфікаційної роботи бакалавра	23.11.2020	
3.	Настановча групова співбесіда з питань кваліфікаційної роботи бакалавра	01.12.2020	
4.	Затвердження плану кваліфікаційної роботи бакалавра	18.02.2021	
5.	Підбір та вивчення літературних та інших джерел з теми дослідження	25.02.2021	
6.	Підготовка і подання науковому керівнику першого варіанту I розділу роботи	05.03.2021	
7.	Підготовка і подання науковому керівнику першого варіанту II розділу роботи	09.04.2021	
8.	Підготовка і подання науковому керівнику першого варіанту III розділу роботи	07.05.2021	
9.	Подання роботи у першому варіанті	11.05.2021	
10.	Оформлення пояснювальної записки кваліфікаційної роботи бакалавра	12.05.2021	
11.	Подання кваліфікаційної роботи бакалавра на попередній захист	24.05.2021	
12.	Врахування зауважень керівника і подання роботи в остаточному варіанті (з відповідним висновком про допуск) на кафедру	28.05.2021	
13.	Затвердження роботи в цілому (підготовка письмового відгуку керівника, письмова рецензія на бакалаврської роботу)	11.06.2021	
14.	Захист кваліфікаційної роботи бакалавра	23.06.2021	

Здобувач вищої освіти _____
(підпис)

Керівник _____
(підпис)

ВІДОМІСТЬ ДИПЛОМНОЇ РОБОТИ

Складові частини дипломної роботи	Обсяг, арк.
Титульний аркуш	1
Завдання до дипломної роботи	1
Календарний план дипломної роботи	1
Відомість дипломної роботи	1
Анотація	1
Анотація (іноземною мовою-англійською)	1
Зміст	2
Перелік скорочень, умовних позначень, термінів	1
Вступ	2
1 розділ	17
2 розділ	20
3 розділ	8
Висновки	1
Перелік використаних джерел	3
Додатки	3

				ДП ХХХХ 00.000.00		
	ПІБ	Підп.	Дата			
Розробн.	Потреба Р.С.			Відомість дипломної роботи	Лист	Листів
Керівн.	Бойко Ю.П.					
Н/контр.	Макаренко С.А.					
Зав.каф.	Плескач В.Л.					

АНОТАЦІЯ

Дипломна робота: 63 сторінки, 18 рисунків, 2 таблиці, 30 джерел, 2 дод.

Ця дипломна робота присвячена проектуванню та розробленню веб-застосунку для підбору фільмів із урахуванням вподобань.

Метою дипломної роботи є підвищення якості підбору відеоконтенту із урахуванням вподобань користувача, шляхом розробки веб-застосунку підбору фільмів.

Для досягнення поставленої мети треба вирішити такі **завдання:**

1. проаналізувати сучасні підходи до розроблення і впровадження веб-застосунків;
2. дослідити архітектурні рішення та здійснити вибір програмних засобів для реалізації веб-застосунку;
3. здійснити програмну реалізацію веб-застосунку для підбору фільмів з урахуванням вподобань.

Об'єкт дослідження. Процес підбору фільмів на основі вподобань користувача та збереження їх для подальшого перегляду.

Предмет дослідження. Сучасні інформаційні технології розробки веб-застосунку для підбору фільмів з урахуванням вподобань користувача.

Методи дослідження. Аналіз глобального впливу «синдрому інформаційної втоми» на процес вибору фільмів для перегляду, узагальнення інформації про можливості сучасних технологій у процесі розробки веб-застосунків, моделювання веб-застосунку для підбору фільмів з урахуванням вподобань.

Ключові слова: JavaScript, React, Redux, інформаційне перенавантаження, надлишок вибору, веб-застосунок.

ABSTRACT

Thesis: 63 pages, 18 figures, 2 tables, 30 sources, 2 appendices.

This thesis is devoted to the design and development of a modern web application for choosing movies based on preferences and the study of the necessary theoretical information base.

The purpose of this thesis is to improve the quality of video content selection based on user preferences, through the development of a web application for movie choosing.

To achieve the goal, **the following tasks** are necessary to solve:

1. analyze modern approaches of the development and implementation of web applications;
2. explore architectural solutions and choose the technology stack to implement a web application;
3. develop a web application for choosing movies based on preferences.

The object of study. The process of choosing movies based on user's preferences and saving them to watch later.

The subject of study. Modern information technologies used for the development of a web application for choosing movies based on user's preferences.

Research methods. Analysis of the global impact of "information fatigue syndrome" on the process of choosing movies to watch, summarizing the information about the possibilities of modern technologies in the sphere of web applications development, modeling a web application for choosing movies based on preferences.

Keywords: JavaScript, React, Redux, information overload, overchoice, web application.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ ТА ТЕРМІНІВ

РС — рекомендаційна система

API — Applied Programming Interface

UI — User Interface

HTML — HyperText Markup Language

CSS — Cascading Style Sheets

JavaScript eXtension — JSX

SPA — Single Page Application

ЦП — центральний процесор

ОС — операційна система

ВСТУП

Актуальність теми. Упродовж останніх 10-ти років у світі виникла глобальна тенденція до діджиталізації значної частини життя середньостатистичної людини. Всесвітнє павутиння (також відоме як Всесвітня мережа або World Wide Web) дає змогу користувачам з усього світу здійснювати покупки, читати новини, навчатися та споживати терабайти різноманітного контенту. Розвиток інформаційних технологій посприяв глобальній зміні способів комунікації, розваг, отримання нової інформації тощо.

З кожним подальшим роком розвиток інформаційних технологій та їх впровадження у життя середньостатистичної людини збільшується у геометричній прогресії. Попри усі переваги, що надають нам новітні технології, цей стрімкий розвиток здатен провокувати сучасну людину на набуття так званого «синдрому інформаційної втоми». Цей симптом характеризує собою відчуття складності у здійсненні того чи іншого вибору через наявність незліченної кількості його варіантів.

На перший погляд надмірна кількість опцій для вибору може здаватися благополучною для загального розвитку особистості умовою, що надає певну свободу та безліч можливостей. Але тільки-но перед індивідом постане питання певного вибору, надлишкова кількість варіантів для вибору може призвести до виникнення феномену під назвою «надлишок вибору», котрий виникає у результаті надто великої кількості еквівалентних за значенням варіантів. Тому розробка веб-застосунку для підбору фільмів із урахуванням вподобань користувача є актуальною на сьогоднішній день.

Метою роботи є підвищення якості підбору відеоконтенту із урахуванням вподобань користувача, шляхом розробки веб-застосунку підбору фільмів.

Для того, щоб досягти поставленої мети, потрібно вирішити наступні задачі:

1. Проаналізувати сучасні підходи до розроблення і впровадження веб-застосунків;

2. Дослідити архітектурні рішення та здійснити вибір програмних засобів для реалізації веб-застосунку;
3. Здійснити програмну реалізацію веб-застосунку для підбору фільмів з урахуванням вподобань.

Об'єкт дослідження. Процес підбору фільмів на основі вподобань користувача та збереження їх для подальшого перегляду.

Предмет дослідження. Сучасні інформаційні технології розробки веб-застосунку для підбору фільмів з урахуванням вподобань користувача.

Методи дослідження. Аналіз глобального впливу «синдрому інформаційної втоми» на процес вибору фільмів для перегляду, узагальнення інформації про можливості сучасних технологій у процесі розробки веб-застосунків, моделювання веб-застосунку для підбору фільмів з урахуванням вподобань.

Новизна одержаних результатів: набув подальшого розвитку застосунок підбору фільмів для перегляду, який на відміну від існуючих, за рахунок врахування вподобань користувача знижує вплив феномену «надлишку вибору».

Практичне значення отриманих результатів. Розроблений веб-застосунок для підбору фільмів з урахуванням вподобань можливо використовувати як для індивідуального підбору контенту, так і для використання в комерційних цілях.

РОЗДІЛ 1. ОГЛЯД СУЧАСНОГО СТАНУ ПРОБЛЕМИ НАДЛИШКОВОГО ВИБОРУ В СЕГМЕНТІ ЦИФРОВИХ РОЗВАГ В ЕПОХУ ГЛОБАЛЬНОЇ ДІДЖИТАЛІЗАЦІЇ. ПОРІВНЯЛЬНА ХАРАКТЕРИСТИКА ІСНУЮЧИХ РІШЕНЬ ДЛЯ ПІДБОРУ ФІЛЬМІВ

1.1. «Інформаційний вибух» як наслідок глобальної діджиталізації

У сучасному світі економічний розвиток неможливий без умови використання інформаційних технологій, адже вони проникають у різноманітні сфери економічної діяльності та створюють передумови для соціально-економічного зростання. Глобалізація та зміни у споживчій поведінці, мобільність та доступність інформації — це актуальні тренди сьогодення. Цифровізація має суттєвий вплив на всесвітню економічну систему. Формування ефективного пласту цифрової економіки створює можливості для розвитку різномасштабного бізнесу, а також накопичення та примноження різноманітних інформаційних та фінансових світових ресурсів.

Діджиталізація є однією з найважливіших передумов для розвитку інновацій та загальносвітового економічного розвитку. Цифрова економіка базується на основі тих самих товарів та послуг, що притаманні традиційній економіці, проте надає їх комп'ютерне обладнання та різноманітні цифрові системи. Це зумовлює одну з найголовніших переваг цифрової економіки перед традиційною — ширша доступність для звичайних користувачів певних ринків, а не тільки великих компаній, а також підвищення ефективності та зниження витрат.

Глобальна діджиталізація призвела одразу до декількох наслідків, що мають суттєвий вплив на повсякденне життя пересічної людини сучасного світу.

Наслідком стрімкого розвитку інформаційних технологій та поступового їх впровадження у життя громадян на побутовому рівні став так званий ефект «інформаційного вибуху».

Феномен «інформаційного вибуху» представляє собою постійне зростання швидкості створення нової інформації та її обсягів у масштабах цілої планети. З

моменту появи мережі Інтернет до 2002-го року людством було вироблено більше інформації, ніж за всю його попередню історію. У 2012-му році глобальне поширення технологій та наявність вільного доступу до мережі Інтернет спровокували збільшення обсягу даних до 6,5 зетабайт. У 2020-му році, згідно з оновленим звітом International Data Corporation було створено, скопійовано та використано більш ніж 59 зетабайт інформації. Щоб отримати 1 зетабайт інформації, потрібно взяти 34,4 мільярди смартфонів з ємністю 32 гігабайти. Цього вистачить для того, щоб 181 раз обігнути Землю, або 11 разів обігнути Юпітер. Зростання кількості інформації, що була створена, скопійована, спожита протягом певних років можна побачити на Рисунку 1.1.

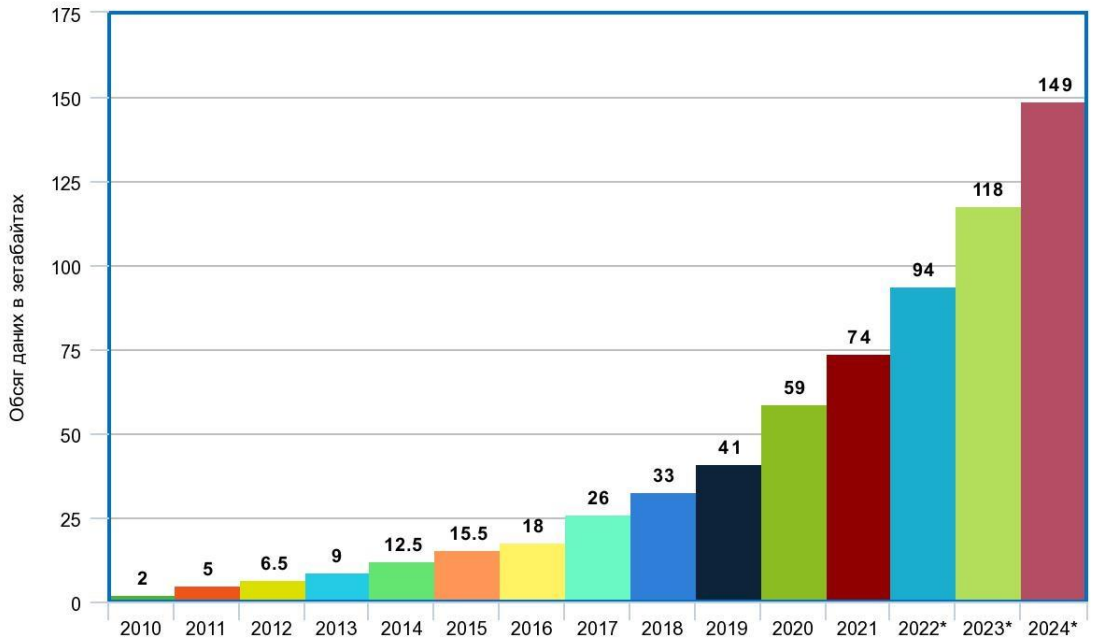

Рисунок 1. 1. Гістограма зростання кількості інформації

Таку тенденцію частково пов’язують з пандемією COVID-19, що виникла наприкінці 2019 року і спричинила перехід на віддалену роботу та відчутне зростання споживання відео-контенту. Основна її причина полягає у тому, що кожна людина, яка має доступ до Всесвітньої мережі має змогу продукувати необмежену кількість інформації: якщо раніше одна людина писала книгу і мільйон людей її читали, то тепер кожен із цього мільйона може написати власну книгу та викласти її у загальний простір. Окрім того, якщо взяти до уваги

незліченну кількість текстової та мультимедійної інформації на різних інтернет-ресурсах, то стає зрозуміло, чому обсяг інформації у світі зростає на 30% щороку. Таким чином, не дивно, що «синдром інформаційної втоми» на сьогодні є однією з найбільш актуальних проблем.

«Інформаційна втома» — це стресовий стан, у якому кількість доступної інформації перевищує ліміт можливостей обробки інформації користувачем. Це призводить до можливого порушення якості процесу прийняття рішень. Численні психологічні та економічні наслідки синдрому мають безпосередній вплив на індивідуальному та суспільному рівнях. «Інформаційна втома» — це форма механізму психологічного захисту, за допомогою якого мозок людини блокує, обмежує чи ускладнює процес пошуку інформації. Це призводить до того, що інформація стає перепоною, а не перевагою.

За дослідженнями «Pew Research Center», що проводилися з 2006 по 2016 рік, майже кожен п’ятий американець страждає від «синдрому інформаційної втоми» (Рисунок 1.2).

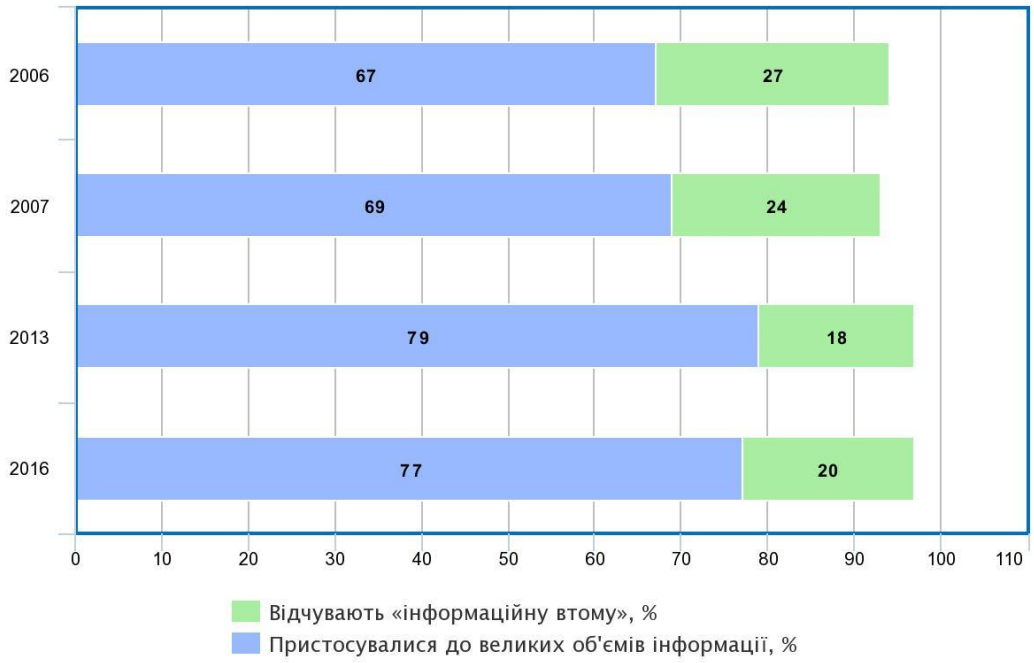

Рисунок 1. 2. Діаграма відношення людей до надлишку інформації

Дослідження у різних галузях виявили, що продуктивність індивіда зростає паралельно зі збільшенням інформації, що він отримує, доки кількість інформації

не досягне ліміту. Після того, як цей ліміт буде вичерпано, результативність індивіда знижується, тому що великий обсяг даних впливає на здатність до пріоритизації та запам'ятовування інформації.

Дослідження фірми «Basex» виявило, що через «інформаційну втому» лише тільки економіка США втрачає близько 900 млрд. доларів на рік. Стрес на робочому місці, що провокує психологічні та фізіологічні захворювання призводить до збільшення витрат на охорону здоров'я та соціальні виплати.

Інформаційний потік від незліченної множини пристроїв, технологій та організацій призводить до постійного відволікання та стресу, але люди продовжують отримувати та виробляти інформацію для себе та інших, щоб успішно співіснувати у цифровому світі. Однією з головних причин виникнення «інформаційної втоми» є не скільки технологічні винаходи, скільки неусвідомлення проблеми та низький рівень цифрової грамотності, що не дозволяє фільтрувати інформаційний потік. Окрім зареєстрованих психічних симптомів «інформаційної втоми», виникає також неможливість приймати рішення, відома як «аналітичний параліч».

Внаслідок ефекту «інформаційного вибуху» протягом повсякденного життя, кожного разу, коли потрібно зробити вибір, людина зіштовхується з незліченною кількістю його варіантів, більшість з яких неможливо навіть опрацювати, що перетворює банальний вибір фільму на нескінченний перегляд трейлерів.

Безмежна кількість варіантів вибору, з якими ми стикаємося у повсякденному житті, мають значний вплив на остаточний вибір. У дослідженні під керівництвом Шини Айенгар, професора менеджменту у бізнес-школі Колумбійського університету, одній групі людей були надані зразки 6-ти різних джемів, що можна придбати, тоді як іншій групі було надано 24 різних зразки. Друга група виявила набагато більший інтерес під час дегустації, тоді як учасники першої групи були у 10 разів більш схильні до здійснення вибору. Якщо ж мова йде про вибір в умовах реального життя з доступом до терабайтів інформації, коли варіантів вибору не 6, не 24 і навіть не 100, то ситуація ще

більше загострюється, що неодмінно призводить до виникнення феномену «надлишку вибору».

1.2. Кінематограф у розрізі цифрових розваг

Розваги є та завжди були центральною частиною людського життя. Швидкий розвиток технологій призвів до необхідності появи також і цифрових розваг, поява яких значно вплинула на те, як середньостатистична людина проводить своє дозвілля. Різноманітні форми розваг — музика, радіо, телебачення, ігри, кіно — все більше концентруються на діджитал медіа та платформах у всесвітній мережі Інтернет. Завдяки більшій доступності інноваційних технологій, таких як комп'ютери, планшети та смартфони, у наші дні фільми, книги, музика та ігри стали доступні у будь-якому місці і у будь-який час, забезпечуючи нам переважну частину нашого дозвілля. Цифрові розваги зазнали кардинальних змін протягом останніх років, Всесвітня мережа відкрила людству новий світ інтерактивних розваг за запитом. Доходи індустрії діджитал дозвілля стрімко зросли.

Серед усіх видів дозвілля особливо можна виділити кіномистецтво, що значно вплинуло на наше сучасне життя. За більш ніж столітню історію свого існування, кіно із «табу» та абсолютної заборони перетворилося у віртуальний образ життя. Кінематограф являє собою цілу платформу, що відображає зростання економіки, політики та технологічних досягнень людства. Неабияку роль відіграють фільми також і в пізнанні історії людства.

Фільми — найпопулярніша на сьогодні форма візуального мистецтва, що знаходить свого споживача у будь-якій країні світу. Захоплення кінематографом не залежить від національності, релігії чи віку, і цей факт може бути обумовлений кількома причинами.

По-перше, перегляд кіно — чудова нагода поспілкуватися з близькими людьми та друзями. У щільному графіку будніх днів та в умовах нестачі часу для родини перегляд фільму — відмінний спосіб зблизитися та спільно провести час.

По-друге, перегляд фільму допомагає відволіктися від щоденної рутини, а улюблена кінострічка піднімає настрій. Це особливо важливо в умовах напруженого та неспокійного сьогодення, у якому іноді буває складно знайти час для відпочинку.

По-третє, кіно базується на певній вигаданій чи реальній історії, яка у свою чергу має в основі моральні принципи та повчає глядача. До того ж, деякі з них засновані на біографіях реальних людей чи історичних подіях. Такі знання є корисними для інформування та розвитку світогляду, більш того, такі кінострічки можуть надихати людей. Таким чином, фільми передають суспільству важливі повідомлення, які можуть змінити людство.

По-четверте, дозвілля та розваги є чи не найважливішою причиною загальнолюдської захопленості кінематографом. Фільми розважають людей, не зважаючи на їх вік та соціальний статус. Вони також можуть принести захоплюючий досвід незалежно від жанру. На додачу, кіно — це легкодоступний та недорогий спосіб розважити себе, сім'ю та друзів, через що так багато людей їх дивляться.

1.3. Сучасний погляд користувачів на споживання кінопродукції

Під час вибору кінострічки для перегляду людина стикається з надлишковою кількістю варіантів ще на етапі вибору жанру. Стрімкий розвиток інформаційних технологій зчинив свій вплив і на кіноіндустрію, призводячи до появи десятків нових піджанрів кіномистецтва. Коли людина вимушена обирати серед такого різноманіття піджанрів, кожен з яких ще й представлений сотнями кінокартин, процес вибору стає значно складнішим.

У питанні стосовно найкращого жанру (або піджанру) не може бути одностайної відповіді, адже це залежить від особистих вподобань та навіть настрою кожного індивіду. Незважаючи на складність, дослідження уподобань серед жанрів відіграє важливу роль у процесі формування цілей розробки спеціального веб-застосунку. Спрямованість на найбільш популярні жанри

допоможе створити актуальну Базу Даних і вплине на порядок розробки застосунку.

За дослідженням «Performance Research» та «Full Circle Research Co.», що було проведене у 2020-му році у Сполучених Штатах Америки, більшість пересічних глядачів надають перевагу жанру комедії, в той час як документальні фільми приваблюють найменший відсоток аудиторії. На Гістограмі 1.3. відображена вірогідність, з якою респонденти опитування переглянули б кінострічку того чи іншого жанру.

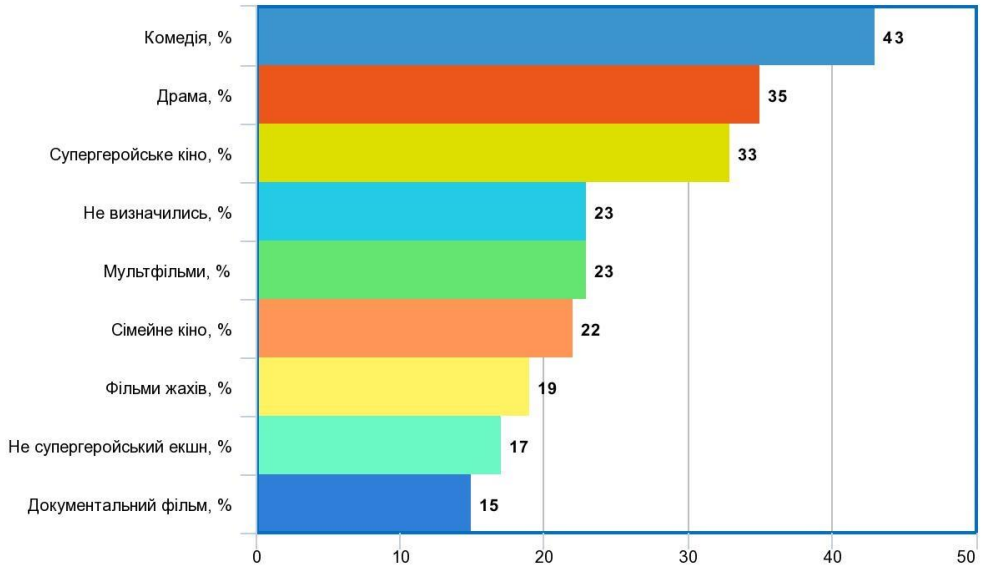

Рисунок 1. 3 Гістограма вірогідності перегляду фільму того чи іншого жанру

Через глобальну пандемію COVID-19 образ життя переважної більшості населення планети зазнав суттєвих змін. Особливої трансформації зазнав сегмент розваг. У зв'язку з розвитком потокових сервісів, навіть до початку карантину, люди надавали перевагу перегляду кіно вдома. За даними дослідження, проведеного «Performance Research» та «Full Circle Research Co.» у 2020-му році у Сполучених Штатах Америки, близько 70% респондентів надають перевагу домашньому перегляду кіно (Рисунок 1.4).

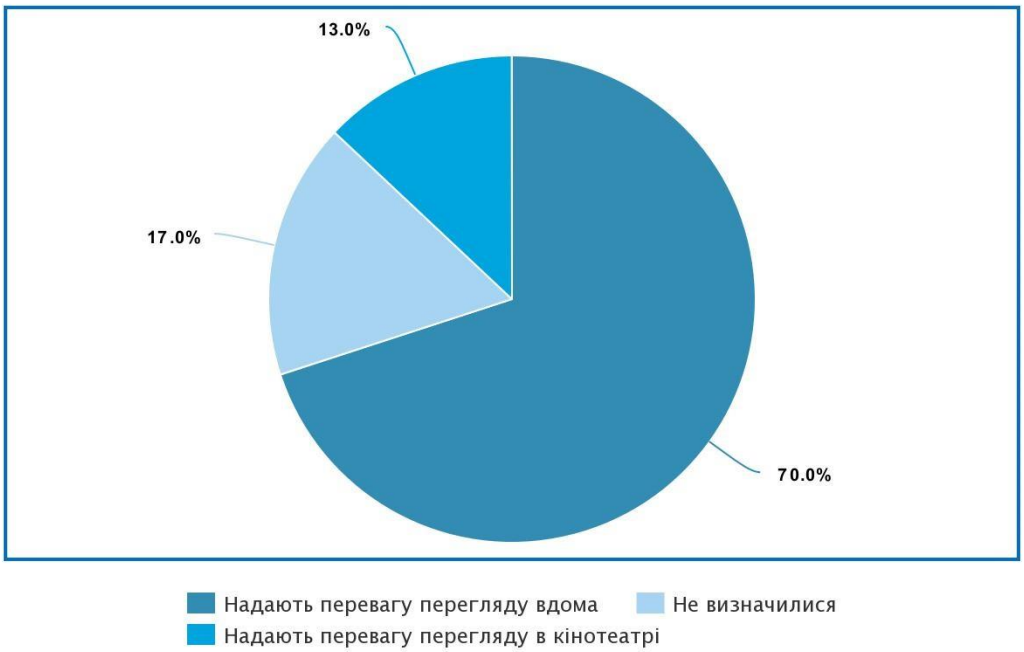

Рисунок 1. 4. Діаграма вподобань людей щодо перегляду кіно

Закриття кінотеатрів, обмеження кількості відвідувачів та побоювання за власне здоров'я закріпили звичку дивитися фільми не виходячи з дому. За рахунок доступності мережі Інтернет та великої кількості онлайн-сервісів, на яких представлена майже будь-яка кінострічка, що коли-небудь була створена, цей процес став більш приємним та простим. Тепер не потрібно обмежуватися графіком телевізійного ефіру і навіть завантажувати фільми не обов'язково.

Це підтверджується і опитуванням, що було проведене аналітичним центром «Cedos» за підтримки Українського Культурного Фонду: лише 14% людей віком 18-29 років хоча б раз на місяць відвідували кінотеатри, але при цьому майже 56% проводили вільний час за переглядом кіно. Статистичні дані демонструють нам, що експансія потокового мовлення у сегменті сучасних мас-медіа не оминула і український ринок. Це обумовлено практичністю і легкою доступністю сучасних потокових сервісів. Для онлайн-доступу до кіноконтенту користувачеві потрібні лише пристрій, що підтримує перегляд потокових трансляцій (комп'ютер, планшет або смартфон) та доступ до мережі Інтернет.

1.4. Рекомендаційні системи: витоки, призначення та загальні принципи роботи

За декілька останніх десятиліть навіть такий простий процес, як вибір фільму, зазнав суттєвих змін через глобальну експансію інформаційних технологій та пропорційне ускладнення різноманітних програмних рішень. З появою YouTube, Amazon, Netflix та багатьох подібних сервісів, рекомендаційні системи (далі — РС) знаходять застосування у все більшій кількості сфер повсякденного життя. В електронній комерції РС допомагають рекомендувати покупцям саме ті товари, що їх зацікавлять, в таргетованій рекламі РС підбирають контент відповідно до вподобань.

Перша рекомендаційна система була розроблена «Goldberg et al.» ще у 1992 році. Її покликанням було вирішення проблеми надлишку листів у електронних поштових скриньках користувачів. Цей алгоритм фільтрував інформацію на основі реакцій користувачів, що виникли у результаті взаємодії з інформацією, наданою у електронному листі. Протягом останнього часу попит на РС значно зріс. Це пояснюється тим, що такі системи дозволяють працювати зі значними за розміром обсягами даних.

Рекомендаційні системи — це алгоритми, головною метою яких є підбір релевантних елементів для користувача. Наприклад, книги для читання, музика для прослуховування, продукти для покупки чи фільми для перегляду. РС відіграють дійсно важливу роль у деяких сферах, адже можуть приносити величезний дохід, чи бути способом виділитися на фоні конкурентів у сфері.

Зазвичай, рекомендаційна система складається з двох частин: бази даних та методів модулю фільтрування. База даних призначена для зберігання інформації про предмети, їх характеристики та пов’язані з ними рейтинги. Модуль фільтрування являє собою алгоритм із двох частин: перша частина відповідає за пошук подібностей між елементами, тоді як друга частина відповідає за підбір найкращих елементів у якості рекомендацій для користувача на основі його вподобань.

У мережі Інтернет, де зберігається величезна кількість варіантів будь-чого, необхідно вміти розставляти пріоритети, фільтрувати та ефективно доставляти необхідну інформацію, щоб вирішити проблему «інформаційної втоми», яка становить потенційну загрозу для багатьох користувачів Інтернету. РС вирішують цю проблему шляхом пошуку інформації, що генерується динамічно, у величезному масиві даних, щоб надати користувачам персоналізовані послуги чи контент.

Швидке зростання об'єму доступної інформації створило потенційну проблему «інформаційного перенавантаження», що перешкоджає своєчасному доступу до необхідних елементів в Інтернеті. Відомі пошукові системи, такі як Google та Yahoo частково вирішили цю проблему, проте пріоритизація та персоналізація інформації все ще залишаються проблемою користувача. Це і збільшило попит на РС, адже вони вирішують проблему інформаційного перенавантаження шляхом фільтрації фрагментів інформації згідно з вподобаннями, інтересами чи навіть поведінкою користувача.

РС позитивно впливають як на постачальників послуг, так і на кінцевих користувачів. Вони знижують операційні витрати пошуку та вибору товарів, музики, фільмів та іншого контенту. Такі системи покращують процес прийняття рішень та підвищують рівень їх якості. У сфері електронної торгівлі РС підвищують доходи ретейлерів, адже дозволяють підібрати якомога більше релевантних продуктів для споживача. Внаслідок цього використання ефективних методів рекомендацій, що будуть підбирати актуальні та точні пропозиції для користувача неможливо переоцінити.

Такі системи є свого роду стратегіями прийняття рішень в умовах ускладненого інформаційного середовища. Більш того, РС — інструмент пошуку релевантної інформації на основі інтересів. Таким чином, подібні рішення допомагають уникнути інформаційного перенавантаження шляхом підбору персоналізованого ексклюзивного контенту.

Головною ціллю РС є підбір релевантних рекомендацій для групи користувачів за предметами чи продуктами, що відповідають їх інтересам.

Дизайн таких систем часто залежить від предметної області і конкретного набору доступних даних. Наприклад, на «Netflix» користувачі часто оцінюють фільми за шкалою від одного до п’яти, що слугує певним джерелом оцінки якості фільму для конкретного користувача. Також РС компанії «Netflix» має доступ до демографічних даних користувачів та детальних описів фільмів. Різні рекомендаційні системи можуть відрізнятися за методами аналізу джерел інформації для формування схожості між користувачами та товарами, які можна використовувати як взірець добре підібраних пар.

80% часу, що користувачі проводять за потоковим переглядом, вони переглядають фільми, підібрані рекомендаційною системою «Netflix». Компанія прагне покращити коефіцієнт утримання користувачів, що дозволить знизити витрати на приваблення нових користувачів.

Для досягнення своїх цілей «Netflix» використовує різноманітні рейтинги, наприклад *персоналізований рейтинг відео* («PVR»). Це універсальний алгоритм, що фільтрує контент за певними критеріями (наприклад, американські телешоу, комедії, історичні драми). Інколи процес фільтрації відбувається у поєднанні з додатковими функціями, такими як популярність фільму та характеристики самого користувача.

Рейтинг відео «Top-N» — ще один алгоритм компанії, що схожий на попередній, але він порівнює не інтереси користувачів з каталогу, а найпопулярніші фільми в жанрі з рештою кінострічок. Він оптимізований з використанням показників, що враховують найкращі фільми у каталозі.

«Trending Now Ranker» — цей алгоритм фіксує тимчасові тенденції, що можуть варіюватися від кількох хвилин до кількох днів. Зазвичай, такі тенденції пов’язані з:

- сезонними подіями, наприклад Хелловін спричиняє збільшення переглядів фільмів жахів;
- разовими подіями (наприклад, коронавірус на початку епідемії викликав короткостроковий інтерес до фільмів про епідемії тощо).

«Continue Watching Ranker» — це алгоритм, що фіксує фільми, які споживач почав дивитися, але не завершив перегляд. Такий контент поділяється на дві групи:

- епізодичний (наприклад, детективний серіал);
- неепізодичний (наприклад, половина фільму).

Цей алгоритм підраховує вірогідність продовження перегляду, аналізуючи контекстно-залежні сигнали (наприклад, скільки часу минуло від моменту перегляду, на якому моменті користувач перестав споживати контент тощо).

«Video-Video Similarity Ranker», або «Because you watched (BYW)» — алгоритм, схожий на метод фільтрації за основним змістом контенту. На основі елементу, який спожив користувач, алгоритм знаходить подібні елементи за допомогою матриці схожості елемент-елемент. На відміну від інших, у цьому алгоритмі не передбачена персоналізація, адже ніякі інші функції не використовуються. Тим не менш, він є персоналізованим за рахунок того, що контент, підібраний ним, є подібним до спожитого раніше.

1.5. Порівняльна характеристика існуючих рішень для підбору фільмів

Fander – сервіс для індивідуального підбору фільмів та серіалів на iOS і Android, який було розроблено за підтримки мережі кінотеатрів Multiplex. Він працює за такою схемою: на екран виводиться картка кінострічки, а користувач повинен зробити свайп: вправо – якщо вже дивився фільм і хоче його оцінити, вліво – якщо хоче пропустити і вгору – якщо хоче додати фільм в список для перегляду в майбутньому. Підбір кінострічок відбувається на основі оцінок, які виставляє користувач за допомогою пошуку стрічок подібних до тих, яким було виставлено найвищий бал.

Цей додаток має ряд переваг перед розробленим мною додатком, які роблять взаємодію з ним більш цікавою. По-перше, сервіс дає можливість застосовувати фільтри не тільки за жанром, наприклад, рік випуску. По-друге, карточки кінострічок містять більше інформації, як-от відгуки інших

користувачів, трейлер та кадри з фільму. По-третє, додаток включає багато іншого функціоналу: покупка квитків, кіно-вікторини, стрічка новин.

Утім Fander має ряд серйозних недоліків, яких вдалося уникнути у моїй розробці. По-перше, сервіс доступний лише на смартфонах чи планшетах. У той час, як мною було розроблено рішення у вигляді веб-додатку, доступ до якого можливий з будь-якого пристрою з доступом до мережі Інтернет. По-друге, хоча додаток має більше фільтрів, вони усі доступні лише у платній версії. Тобто базова версія передбачає рандомний підбір кінострічок навіть без можливості відфільтрувати їх за жанром. Таким чином, моя розробка у базові комплектації має більше можливостей для вибору. По-третє, Fander використовує рейтинг кінострічок Amazon Prime, хоча більш популярними є Rotten Tomatoes чи IMDB.

Таким чином, Fander дає можливість користуватися більшою кількістю фільтрів, але за плату, у безкоштовній версії не доступний жоден фільтр. За цим показником розроблений мною додаток має перевагу, адже безкоштовно можна використовувати фільтр за жанром. Також важливою перевагою є доступність мого рішення на усіх гаджетах з доступом до інтернет, адже Fander можна завантажити лише на iOS і Android. Проте додаток конкурентів має набагато ширший функціонал, утім це не так принципово, враховуючи поставлене перед додатком завдання.

Ще одним сервісом, що знаходиться у ТОП видачі за запитом «сервіс для підбору фільмів» у Google є Kino.today. Це сервіс для вибору фільмів та серіалів у вигляді веб-застосунку. Для підбору фільмів необхідно налаштувати фільтри та здійснити пошук, після чого сервіс надає список кінострічок, що відповідає побажання користувача.

Kino.today має ряд переваг відносно моєї розробки, серед яких велика кількість фільтрів. Сервіс пропонує до вибору 20 фільтрів, серед яких жанр, рік, країна виробництва, епоха, пора року тощо. Більш того карточка товару містить більше корисної інформації, як-от трейлери та кадри з фільму.

Проте, незважаючи на всі плюси, застосунок має ряд досить серйозних недоліків. Навіть основна перевага сервісу стосовно кількості фільтрів стає

недоліком при вирішенні поставленого завдання. Завеликий вибір фільтрів ускладнює пошук фільму для перегляду, що протирічить завданню додатку допомагати боротися з надлишком вибору. До того ж, така кількість фільтрів пішла на шкоду якості роботи додатку, адже за застосування більш, ніж 2-х фільтрів, сервіс відображає помилку і не може коректно знайти кінострічки. Інформації у карточці фільму недостатньо, немає таких важливих показників, як тривалість фільму та рейтингові бали. На додачу, у сервісі відсутня можливість додати стрічку до списку улюблених фільмів.

Загалом, Kino.today має два основних недоліки: по-перше, фільтрів забагато і вони ускладнюють пошук, по-друге, сервіс некоректно працює при використанні більше 2-х фільтрів. Також досить негативною є неможливість додати кінострічку до списку, щоб переглянути її пізніше.

Аналогом двох попередніх сервісів є Movieton – розумний пошук за фільмами і серіалами, де можна легко і швидко знайти фільми в необхідних жанрах. Головне завдання і особливість Movieton – визначення інтересів його користувачів (за допомогою міні-тестування) і формування на базі отриманих даних максимально цікавої для конкретної людини добірки фільмів. Відповіді на кілька стандартних запитань, розміщених на сайті, дозволяють розумній системі виробляти швидку фільтрацію колекцій, моментальну видачу кінокартин за суто індивідуальними інтересами користувача ресурсу.

Безпосередніми перевагами ресурсу є незвичайний механізм роботи та більша кількість фільтрів. Застосунок відрізняється принципово новим алгоритмом, який підбирає фільми не просто за фільтрами чи подібністю до інших фільмів, а бере за основу відповіді користувача на питання. До того ж, у процесі фільтрування сервіс застосовує більшу кількість критеріїв, наприклад, рік виходу чи країна виробництва.

Утім Movieton має і ряд недоліків, що погіршують враження користувача від процесу пошуку фільмів. Наприклад, для того, щоб обрати новий жанр для пошуку, необхідно заново пройти тестування, що робить підбір кінострічок довгим та нудним. З цим пов’язана і проблема неможливості використовувати

якийсь фільтр окремо, не проходячи тестування. Також у карточці кінострічки немає інформації про рейтинг фільму.

Головною перевагою та головним недоліком системи став алгоритм збору інформації про користувача. Він якісно вирізняється на фоні звичайних фільтрів, але реалізований так, що його потрібно щоразу проходити повторно за бажання змінити певний критерій. Це також ускладнює пошук фільму, що суперечить головному завданню подібних ресурсів.

Загалом, ці три ресурси демонструють різні підходи до вибору фільтрів. Fander у безкоштовній версії базує пошук на оцінках інших фільмів, але зазвичай цей пошук неточний і користувачеві підбирають нерелевантні кінострічки. Відкоригувати фільтрацію можна лише у платній версії, тому сервіс за замовчуванням не є індивідуалізованим, що суперечить основному завданню подібних додатків.

Kino.today базує пошук на фільтрах, що є оптимальним варіантом взаємодії з користувачем, що дозволяє точно виконувати поставлену задачу. Втім, кількість критеріїв занадто велика, що ускладнює як процес пошуку з боку користувача, так і роботу системи з технічної точки зору. Таким чином, ресурс не виконує поставлене перед ним завдання, адже користувачу доводиться багато часу проводити за вибором критеріїв.

Movieton має дуже незвичайний процес взаємодії, але з технічної точки зору він втілений так, що стає на заваді простому пошуку кінострічки. У користувача немає можливості використовувати фільтри окремо, а також йому доводиться перепроходити тест щоразу, коли йому потрібно змінити навіть одну умову пошуку. Це ускладнює процес підбору фільму і унеможливлює виконання завдання, поставленого перед сервісом.

Ще одним пунктом, що викликає відмінності, є наповненість карточки кінострічки. Корисною ідеєю, реалізованою в усіх аналізованих додатках, є відображення трейлерів фільмів для того, щоб полегшити користувачеві формування його власної оцінки стрічки. Кадри з фільму, втім, лише інформаційно перенавантажать юзера та відволікатимуть увагу від основної

інформації. Проте усі проаналізовані додатки не наводять інформацію про рейтинг стрічки чи використовують непопулярні рейтинги. У цьому пункті моя розробка має значну перевагу, наводячи оцінку фільму за версією IMDB, одного з сорока найпопулярніших сайтів Всесвітньої павутини.

Таким чином, розроблений мною додаток має свої переваги та враховує недоліки інших подібних сервісів для того, щоб максимально ефективно виконувати поставлене перед ним завдання, що полягає у полегшенні вибору фільму в умовах інформаційного перенасичення та подолання проблеми надлишкового вибору.

1.6. Висновок

Перехід до електронного обігу інформації призвів до надлишкового накопичення даних у вільному для людини доступі. За останні 10 років через можливість кожного створювати інформацію, її обсяг перевищив кількість інформації, створеної протягом усієї попередньої історії людства. Такі розміри доступних даних здатні викликати у певних індивідів «інформаційну втому», за якої людський мозок не встигає опрацьовувати інформацію. Більш того, така ситуація призводить до виникнення надлишкового вибору. Здавалося б, кількість альтернативних варіантів у будь-якій сфері має бути перевагою. Але у результаті цих варіантів так багато, що людина зациклюється на їх обробці та не може здійснити вибір.

Це особливо помітно у сфері кінематографу, який є однією з найбільш популярних та діджиталізованих сфер мистецтва. Люди переглядають фільми для відпочинку та якісного проведення часу з родиною, та через надлишок відомих людям продуктів кінематографу, сімейне дозвілля зводиться до суперечок або годинних переглядів різноманітних рейтингів кіно. За статистикою, люди споживають кіноконтент саме удома, а отже процес вибору фільму лягає на плечі самих споживачів, за умови відсутності розпорядку трансляцій фільмів, як у кіно чи на телебаченні.

Одним із шляхів вирішення цієї проблеми стала поява рекомендаційних систем. Мета цих алгоритмів полягає у підборі релевантних одиниць контенту на основі уподобань та специфічних особливостей споживача. У наш час ці системи мають безліч застосувань, бо методи їхньої роботи абсолютно адаптивні. Фільтрація контенту відбувається на основі характеристик елементів та вподобань користувача. Це означає, що така система може бути адаптована як під фільми чи музику, так і під різноманітні товари.

Рекомендаційні системи перебирають на себе «тягар» вибору та підвищують його якість за рахунок того, що алгоритми здатні помічати тенденції, непомітні для людей. Вони дозволяють швидше здійснити вибір, не нехтуючи його якістю. Такі системи приносять користь і для продавців товарів чи контенту, адже люди купують більше релевантних товарів та довше переглядають цікаві для них фільми.

Таким чином, рекомендаційні системи є одним з ключових досягнень інформаційних технологій в умовах надлишкового вибору унаслідок інформаційного вибору через глобальну діджиталізацію.

РОЗДІЛ 2. ВЕБ-ЗАСТОСУНОК ЯК РІШЕННЯ ПОСТАВЛЕНОЇ ПРОБЛЕМИ. ВИБІР ЗАСОБІВ ДЛЯ РЕАЛІЗАЦІЇ ВЕБ-ЗАСТОСУНКУ ТА ОПИС ЙОГО СТРУКТУРИ.

2.1. Порівняння веб-застосунку з іншими програмними рішеннями

Веб-застосунок – це прикладне програмне забезпечення, яке запускається на веб-сервері, на відміну від комп'ютерних програм, що запускаються на локальній операційній системі пристрою. Користувач може отримати доступ до веб-застосунку через веб-браузер з активним підключенням до мережі Інтернет. Ці додатки програмуються на основі змодельованої структури клієнт-сервер (рис. 2.1) – послуги надаються користувачу через зовнішній сервер. Популярними веб-додатками є веб-пошта, роздрібна торгівля в Інтернет, онлайн-банкінг, онлайн дошки оголошень.

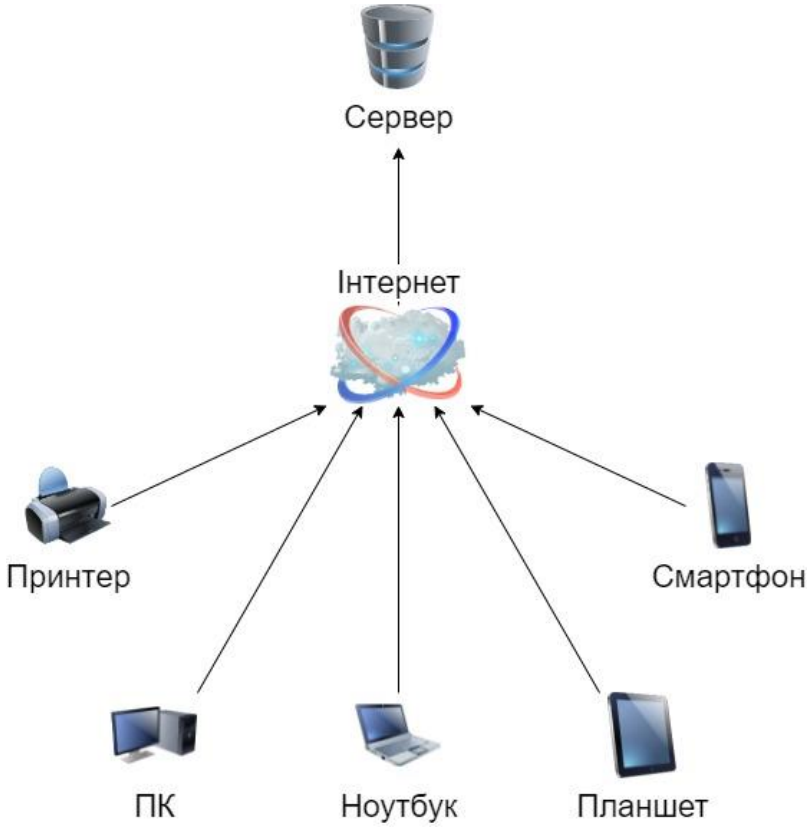

Рисунок 2. 1. Модель клієнт-сервер

Комп'ютерні чи мобільні додатки – це програмне забезпечення, призначене для виконання конкретної задачі, яка не передбачена самим

пристроєм, що зазвичай використовується кінцевим користувачем. Прикладами таких додатків є медіа-плеєри та текстові процесори.

Такі додатки розміщуються у пам'яті пристрою користувача та запускаються на локальній операційній системі. Вони доступні навіть без доступу до мережі Інтернет, якщо їхній функціонал не передбачає обміну інформацією за допомогою Всесвітньої павутини.

Якщо порівнювати між собою комп'ютерні додатки та веб-застосунки, кожен із видів буде мати свої переваги та недоліки. Утім, враховуючи поставлене перед розробленим мною застосунком завдання, можна сказати, що веб-застосунок є більш оптимальним варіантом втілення продукту з ряду причин.

На відміну від десктоп-додатків, веб-застосунки простіше підтримувати та обслуговувати, адже вони використовують один і той самий код в усьому додатку. Через це рідше можуть виникати проблеми з сумісністю.

Більш того такі додатки можна використовувати на будь-яких пристроях (телефон, ноутбук, комп'ютер, планшет тощо) та будь-яких платформах (Windows, Linux, MacOS), адже всі вони підтримуються сучасні браузери, які необхідні для відображення веб-застосунків.

Ще однією перевагою веб-додатків є те, що для доступу до них не потрібні магазини мобільних чи комп'ютерних додатків. Вони доступні одразу за посиланням чи легко віднаходяться за допомогою пошукової системи Google. Більш того користувачам не потрібно нагадувати про оновлення додатку, адже він автоматично оновлюється у всіх користувачів після завантаження нової версії на сервер.

Останнім плюсом веб-застосунків є те, що вони не потребуються завантаження чи інсталяції, адже веб-застосунки доступні через веб-браузер мобільного чи комп'ютерного пристрою цілодобово.

Усі ці причини роблять веб-застосунок найоптимальнішим варіантом для втілення продукту, що допоможе йому максимально ефективно виконувати поставлене перед ним завдання. Цьому сприяє легка доступність ресурсу на

будь-яких пристроях та платформах без необхідності попереднього завантаження та інсталяції.

Дуже подібними до веб-застосунків є веб-сайти – це групи сторінок, пов'язаних між собою, що глобально доступні під одним доменним ім'ям. Вони розміщуються на одному чи кількох серверах та доступні через мережу Інтернет чи приватну локальну мережу. Веб-сайти зазвичай використовують як простий спосіб продемонструвати свої товари та послуги. Більше того він допомагає формувати імідж та бренд для компанії, бо переважно має інформувальну функцію.

Не зважаючи на подібність цих рішень вони мають ряд досить ґрунтовних відмінностей, що впливають на їх пристосованість до виконання завдання, поставленого перед розробленим додатком.

Наприклад, на відміну від веб-сайту, що містить переважно статичний контент для всіх користувачів, веб-застосунок націлений на взаємодію з конкретним кінцевим користувачем.

Веб-сайти мають переважно інформувальну функцію, відображаючи текстовий та медіа контент для перегляду чи прослуховування без можливості користувачів впливати на його функціонування, в той час, як веб-застосунки надають можливість не тільки переглядати вміст сторінки, а й маніпулювати інформацією.

Через це завдання та можливості веб-застосунку значно ширші за завдання та можливості-веб-сайту, які фактично полягають у відображенні набору інформації у певній структурній організації.

Зважаючи на вищезгадані аргументи, веб-застосунок є найбільш оптимальним варіантом для втілення додатку для підбору фільмів з урахування вподобань.

Розташування на сторонньому сервері та доступність через веб-браузер з будь-якого пристрою та платформи робить цей варіант більш раціональним у порівнянні з комп'ютерним чи мобільним додатком. Адже повсякчасна

доступність з різних пристроїв без попередньої інсталяції відповідає меті додатку, а саме полегшує процес вибору кінострічки для перегляду.

А можливість користувача взаємодіяти з даними та ширший функціонал робить веб-застосунок більш релевантною альтернативою у порівнянні з веб-сайтом, адже дозволяє впровадити усі необхідні функції для того, щоб додаток виконував поставлені перед ним завдання.

Зважаючи на всі переваги веб-додатку перед веб-сайтом та комп'ютерним/мобільним додатком як форми втілення застосунку для підбору фільмів на основі вподобань кінцевого користувача, було вирішено розробляти продукт саме у такому форматі, щоб забезпечити якомога ефективніше виконання поставлених перед ним задач з полегшення процесу підбору кінострічки для перегляду.

2.2. Вибір технологій для реалізції веб-застосунку

Hypertext Markup Language (HTML) — мова розмітки гіпертексту. Вона дозволяє не тільки створювати, а й структурувати суцільний текст на розділи, параграфи, заголовки, посилання і блоки для веб-сторінок і застосунків.

HTML не можна вважати мовою програмування, тобто за допомогою цього інструменту неможливо створювати динамічні функції. Замість цього її використовують для організації і форматування документів у якості текстового редактору на кшталт Microsoft Word.

HTML використовує розмітку для відображення тексту, зображень та іншої інформації в веб-браузері. Розмітка включає в себе спеціальні елементи. HTML-елемент виділяється з іншого тексту за допомогою тегів, які складаються з імені елемента оточеного символами "<" і ">". Для розмітки документу використовується багато спеціальних тегів, таких як <head>, <title>, <body>, <header>, <footer>, <article>, <section>, <p>, , , <img> і багато інших.

Окрім того, у HTML присутня своя об'єктна модель — Document Object Model. Усі елементи, що знаходяться всередині тега <html>, утворюють собою спеціальне дерево документу (рис. 2.2).

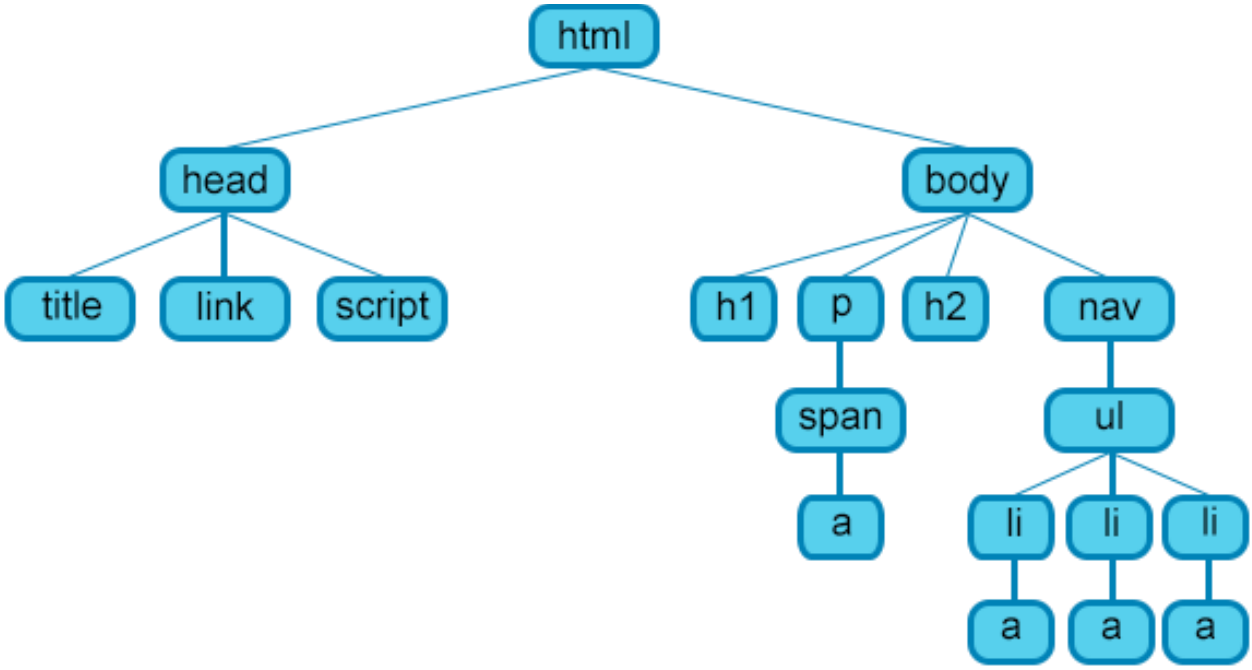

Рисунок 2. 2. Структура DOM-дерева

У HTML використовується концепція батьківських, дочірніх та сестринських елементів.

- предками називають елементи, що містять в собі інші елементи;
- нащадками називають елементи, що входядь в склад іншого елемента;
- батьківським елементом називають елемент, що містить в собі інші елементи та знаходиться вище них у DOM-ієрархії;
- дочірнім елементом називають елемент, що підпорядковується іншому (батьківському) елементу;
- сестринським елементом називають елемент, що знаходиться на одному рівні з дочірнім елементом.

HTML-документи — це файли з розширенням .html або .htm. Їх можна переглядати та навіть редагувати у будь-якому веб-браузері (наприклад, Google Chrome, Safari або Mozilla Firefox). Браузер читає і відображає вміст такого файлу, щоб користувачі інтернету могли його переглядати.

HTML було винайдено Тімом Бернерс-Лі, фізиком з дослідницького інституту ЦЕРН в Швейцарії. Він опублікував першу версію HTML в 1991 році,

тоді вона складалася з 18 тегів. З тих пір кожна нова версія доповнювалася новими тегами і атрибутів (модифікаторами тегів). Згідно із Довідником HTML Element Reference від Mozilla Developer Network, в даний час існує 140 тегів.

В HTML можна вбудувати програмний код на мові програмування JavaScript для управління поведінкою і змістом веб-сторінки. Також включення CSS в HTML описує зовнішній вигляд і макет сторінки.

Cascading Style Sheets (CSS) — формальна мова опису зовнішнього вигляду документа (веб-сторінки), написаної з використанням мови розмітки (найчастіше HTML або XHTML). Також може застосовуватися до будь-яких XML-документів, наприклад, до SVG або XUL. Це мова, яка відповідає за візуальний вигляд документів на пристрої користувача. Цей код використовується для стилізації веб-сторінок.

Як і HTML, CSS насправді не є мовою програмування. Це не мова розмітки — це мова таблиці стилів. Це означає, що вона дозволяє застосовувати стилі вибірково до елементів в документах HTML. Під документом мається на увазі набір інформації про структуру сторінки, описуваний мовою розмітки. Основна мета використання CSS — доповнити документ оформленням без програмування або складної логіки. Оформлення — це розташування окремих блоків на сторінці, кольори, шрифти, тощо.

Представлення документа на пристрої користувача означає перетворення його в зручну форму, покращення сприйняття та доступності контенту, контроль за відображенням контенту в різних умовах. CSS у свою чергу дозволяє адаптувати контент до різних умов відображення (на екрані монітора, мобільного пристрою, на екрані телевізора тощо).

Структурно CSS складається з селекторів, що дозволяють обирати елементи для редагування, та властивостей, які приписуються цим елементам. Попереду поміщається селектор, потім ставляться фігурні дужки, між якими і знаходяться властивості CSS з присвоєними їм значеннями, назва кожної властивості відділяється від відповідного параметру двокрапкою (рис. 2.3). Всі властивості разом зі значеннями розділяються між собою крапкою з комою.

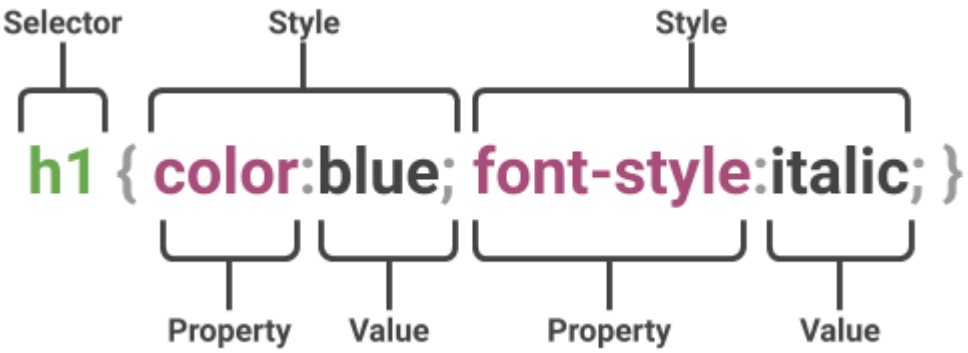

Рисунок 2. 3. Приклад CSS-селектора

На початку 1990-х різні браузери мали свої стилі для відображення веб-сторінок. HTML розвивався дуже швидко і згодом став придатним для задоволення всіх існуючих на той момент потреб з оформлення інформації, тому CSS на той час не отримав широкого визнання.

Термін «каскадні таблиці стилів» був запропонований Хоконом Лі в 1994 році. Спільно з Бертом Босом вони почали розвивати CSS. На відміну від існуючих на той момент мов стилю, CSS використовує спадкування від батька до нащадка, тому розробник може визначати стилі, грунтуючись на вже визначених раніше стилях. Проте в середині 1990-х Консорціум Всесвітньої павутини (W3C) став проявляти інтерес до CSS, і в грудні 1996 року була видана рекомендація CSS1.

JavaScript — це мова програмування, що може використовуватися як на клієнтській стороні (Front-end), так і на серверній стороні (Back-end), що може бути використаний для створення інтерактивних веб-застосунків. Найчастіше використовується для створення сценаріїв веб-сторінок, що надає можливість на стороні клієнта (пристрої кінцевого користувача) взаємодіяти з користувачем, керувати браузером, асинхронно обмінюватися даними з сервером, змінювати структуру та зовнішній вигляд сторінки.

JavaScript класифікують як прототипну (підмножина об'єктноорієнтованої), скриптову мову програмування з динамічною типізацією.

У браузері для JavaScript є все, що пов'язано з маніпулюванням веб-сторінками, взаємодією з користувачем і веб-сервером.

Наприклад, в браузері JavaScript може:

- додавати новий HTML-код на сторінку, змінювати існуючий вміст, модифікувати стилі;
- реагувати на дії користувача, клацання миші, перемістити вказівник, натискання клавіш;
- відправляти мережеві запити на віддалені сервера, завантажувати файли (технології AJAX і COMET);
- отримувати і встановлювати cookies, задавати питання відвідувачеві, показувати повідомлення;
- запам'ятовувати дані на стороні клієнта (localStorage, sessionStorage, cookies).

Наочно це виражається таким чином: можлива зміна кольору шрифту або посилання на зображення, створення або видалення блоків, відображення або приховування елементів, зміна їх текстового змісту, зчитування даних з полів форми на сторінці тощо.

Наступним пунктом у списку є бібліотека для створення користувацьких інтерфейсів під назвою React (також відома як React.js).

Технологія була розроблена компанією Facebook та була вперше представлена у 2013 році. Бібліотека є одним з найкращих рішень для створення високоефективної front-end частини веб-додатка типу Single Page Application (SPA) — веб-застосунок, що вміщується на одній сторінці HTML-документу, метою чого є забезпечення користувачеві досвіду близького до взаємодії зі звичайною настільною програмою, але за допомогою браузера.

В SPA весь необхідний HTML-, CSS- та JavaScript-код завантажується разом із самою сторінкою, або ж динамічно довантажується у фоновому режимі в залежності від потреб та дій користувача. Сторінка не оновлюється та не перенаправляє користувача на інші сторінки у процесі роботи з нею.

Основною особливістю React є концепція віртуального DOM-дерева документу, що є копією справжнього DOM-дерева сторінки, зберігається у пам'яті та синхронізується з ним. Проблема полягає у тому, що звичайний DOM (Document Object Model) не розрахований на створення високоефективних динамічних користувацьких інтерфейсів, і використання у парі з ним звичайного JavaScript або бібліотеки на кшталт jQuery не вирішує проблем з продуктивністю.

За допомогою Virtual DOM ми взаємодіємо зі звичайним DOM, але робимо це точечно, змінюючи лише те, що нам потрібно. DOM-дерево порівнюється з копією (Virtual DOM), визначається різниця між ними, а потім проводиться перемалювання тих елементів, що були змінені.

Окрім того, до головних особливостей бібліотеки React можна віднести наступні пункти:

- бібліотека є має надзвичайно широкий спектр застосування. Вона може бути використана як для розробки веб-додатків, так і для розробки мобільних додатків (React Native);
- React було створено з єдиною головною ціллю: створювати компоненти для веб-застосунків. Таким компонентом може бути що завгодно, починаючи надписом, закінчуючи цілим меню на веб-сторінці;
- бібліотеку дуже просто інтегрувати в уже створений за допомогою інших технологій продукт.

Також у React присутній свій концепт для написання розмітки документу, що має назву JavaScript eXtension, або ж JSX. Кожна компонента React має метод render, що визначає HTML-результат виконання коду. У цьому методі розробник і описує кінцеву розмітку веб-сторінки, використовуючи синтаксис JSX, що являє собою JavaScript, котрий виглядає як HTML. Хоча у минулому вважалося поганою практикою використовувати JavaScript та HTML в одному місці, та

виявилось, що вузьке поєднання цих двох технологій спрощує сприйняття процесу розробки.

Redux являє собою свого роду контейнер передбачуваних станів для JavaScript-додатків, що допомагає створювати ефективні веб-додатки, котрі ведуть себе послідовно і котрі легко тестувати.

Redux надає один єдиний об'єкт, що зберігає у собі стан усього веб-додатку, а також може включати в себе дані з зовнішнього прикладного програмного інтерфейсу (Application Programming Interface, або API) тощо. Завдяки цьому значно спрощується діагностика додатку, вирішення проблем з асинхронними запитами, керування станом натиснутої кнопки користувацького інтерфейсу, виявлення інших помилок тощо.

Основними поняттями Redux є Store, Actions, Reducers та Subscriptions, без яких використання бібліотеки є неможливим (рис. 2.4).

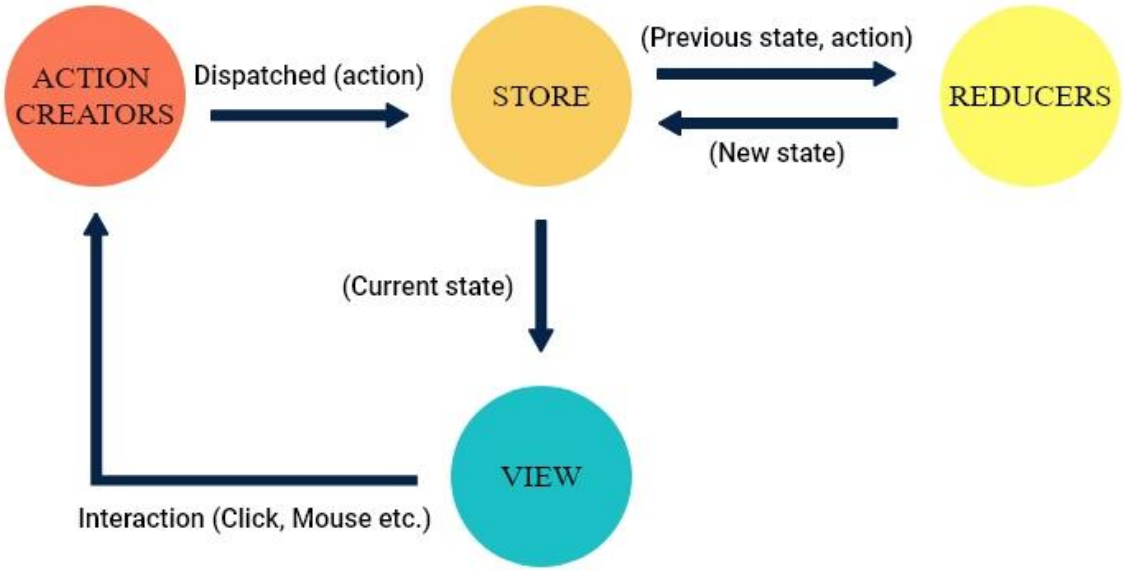

Рисунок 2. 4. Схематичне зображення Redux dataflow

Actions — це об'єкти «дії», за допомогою яких ми взаємодіємо зі станом додатку та можемо його змінювати. Такий об'єкт має містити як мінімум один ключ, який відповідає за тип дії. Процес відправки дії на обробку називається dispatch.

Reducers — це функції, які відповідають за обробку тих чи інших actions. Вони приймають у себе наступні аргументи: початковий стан додатку (initialState) та певний action.

Store являє собою загальне сховище даних з усього додатку, що формується за допомогою усіх reducers за допомогою спеціальної функції combineReducers().

Окрім того, аби окремі компоненти додатку могли реагувати на зміни у Store та виконувати ре-рендер певної частини DOM-дерева, потрібно «підписати» компоненти на зміни Store. Це можливо зробити декількома способами, але я використовував Redux Hooks — спеціальні структури бібліотеки Redux, що дозволяють компонентам реагувати на зміни сховища, а також відправляти на обробку певні actions.

Для створення зовнішнього вигляду інтерфейсу мною було обрано інтерфейсний фреймворк Material UI, що має відкритий сирцевий код. Це рішення спроектоване з використанням UI-рішень від компанії Google, що гарантує високу якість компонент під час розробки інтерфейсної графіки.

Порівняно з іншими схожими фреймворками, Material UI має ряд невід'ємних переваг, а саме:

- вичерпна документація. Цей фреймворк дуже добре задокументований, що спрощує роботу з ним та загальну навігацію фреймворком;
- єдина стилізація. Усі компоненти Material UI виконані в однаковій стилістиці, що гарантує естетичний вигляд розробленого веб-додатку.

Axios — одна з найбільш популярних JavaScript-бібліотек, що може бути використана для виконання HTTP-запитів. Може бути використана як на клієнтській частині, так і на серверній. Бібліотека підтримує усі сучасні браузери, навіть Internet Explorer.

Основою Axios є JavaScript-конструкції під назвою promise, що значно спрощує написання коду асинхронної логіки застосунку. Бібліотека має ряд переваг перед нативним методом fetch(), що надає JavaScript:

- підтримує старі версії браузерів;
- має метод для перервання HTTP-запиту;
- має метод для встановлення часу очікування відповіді;
- обладнана вбудованим захистом від CSRF;
- підтримує процес завантаження;
- автоматично приводить відповідь сервера до формату JSON.

У якості back-end частини мною було використано API під назвою IMDb API, розроблене Api Dojo. Цей інструмент являє собою веб-інтерфейс, що може витягувати інформацію про фільми та телешоу, яка є наявною на популярному веб-ресурсі imdb.com.

Зв'язок клієнтської частини з даним API здійснюється за допомогою асинхронних HTTP-запитів до конкретних кінцевих точок інтерфейсу, що надані для користування.

Кінцева точка (або endpoint) — це один з кінців каналу зв'язку. Кожна така точка являє собою певне місце, з якого API-інтерфейс може отримати доступ до ресурсів, необхідних для виконання тих чи інших функцій.

Таблиця 2. 1. Опис методів API

Назва кінцевої точки	**Опис функцій**
get-popular-movies-by-genre	Результатом виконання запиту методу є масив даних з певної кількості популярних фільмів за шуканим жанром
get-overview-details	Результатом виконання цього запиту є об'єкт з інформацією про випадковий фільм з масиву, що повертається попереднім методом

Окрім цього, було використано низку додаткових JavaScript-бібліотек, що за стандартами імплементуються у подібні проекти. До списку таких бібліотек належать:

- Redux-thunk — бібліотека, що являю собою middleware (проміжний прошарок, що за допомогою Redux реалізовувати функції, у тому числі й асинхронні HTTP-запити до сервера);
- React-redux — це офіційний пакет до бібліотеки Redux, що дозволяє компонентам React інтегруватися зі сховищем Redux, автоматично зчитуючи фрагменти стану застосунку та оновлюючи сховище;
- React-router-dom — бібліотека, що дозволяє розроблюваному застосунку переходити між різними компонентами, змінювати URL-адресу у браузері тощо;
- Styled components — це один із найновітніших способів написання CSS-коду у рамках React-застосунку. Ця бібліотека дозволяє застосовувати стилі модульним підходом, виключаючи можливість колізії імен селекторів;
- Formik — декларативна бібліотека для роботи з HTML-формами на сторінці, що спрощує процеси трекінгу значень, помилок, валідації та відправлення HTML-форми, а також стандартизує роботу з потоком взаємодії з HTML-формами;
- Particles.js — бібліотека для генерації фону для сторінки, складеного із частинок. Покращує досвід взаємодії з застосунком.

2.3. Загальна архітектура розроблюваного веб-застосунку

Фінальна версія веб-застосунку має складатися з декількох компонент, пов’язаних між собою за допомогою state-management бібліотеки, що дозволяє маніпулювати даними, дозволяючи компонентам ефективно реагувати на їх зміну.

Мною було обрано архітектуру на основі React+Redux, головною перевагою якої є чітке розподілення бізнес-логіки застосунку по модулях. У

невеликих проектах кількість таких модулів не є великою, проте у масштабних проектах їх кількість значно зростає. У разі використання такого підходу, під час розробки не виникає труднощів з визначенням, у якому місці застосунку слід імплементувати ту чи іншу логіку.

У класичному підході, ця архітектура передбачає використання двох видів компонент: контейнерних та презентаційних. Це концептуальне розподілення, що дозволяє відокремити user interface та взаємодію з даними на рівні користувацького інтерфейсу.

Контейнерна компонента — це компонента, що отримує доступ до загального сховища стану веб-застосунку (state), та слугує для взаємодії з цими даними для подальшої їх передачі до презентаційної компоненти. Кожна контейнерна компонента має свій реалізований метод render(), який відображає певну презентаційну компоненту.

Презентаційна компонента — це компонента, яка має реалізований метод render(), та приймає вхідні параметри props, що були передані у неї за допомогою контейнерної компоненти. Такі параметри, зазвичай, містять у собі деструктуризовані дані для подальшого їх відображення у користувацькому інтерфейсі застосунку.

Окрім того, використання такої архітектури передбачає модульний підхід для створення застосунків. Такі модулі мають бути незалежними один від одного, а також мають розділену логіку обробки даних.

Структура веб-застосунку має являти собою чітко та логічно сформульовану систему розташування компонент, тобто систему навігації. Грамотна структура має прямий вплив на ранжування веб-сайту у пошукових системах, а також на сприйняття користувачів.

Невід'ємними складовими структури розроблюваного веб-застосунку є:

- клієнтський інтерфейс — те, що користувач споглядає у браузері та за допомогою чого працює з застосунком. Рівень вкладеності компонент інтерфейсу не має перевищувати чотирьох (тобто

користувачу потрібно не більше трьох кліків від головної компоненти до шуканої);

- серверна частина, що працює посередництвом Applied Programming Interface (API), до якого звертається клієнтська частина за допомогою HTTP-запитів.

Таким чином, загальна схема компонент розроблюваного веб-додатку виглядає наступним чином:

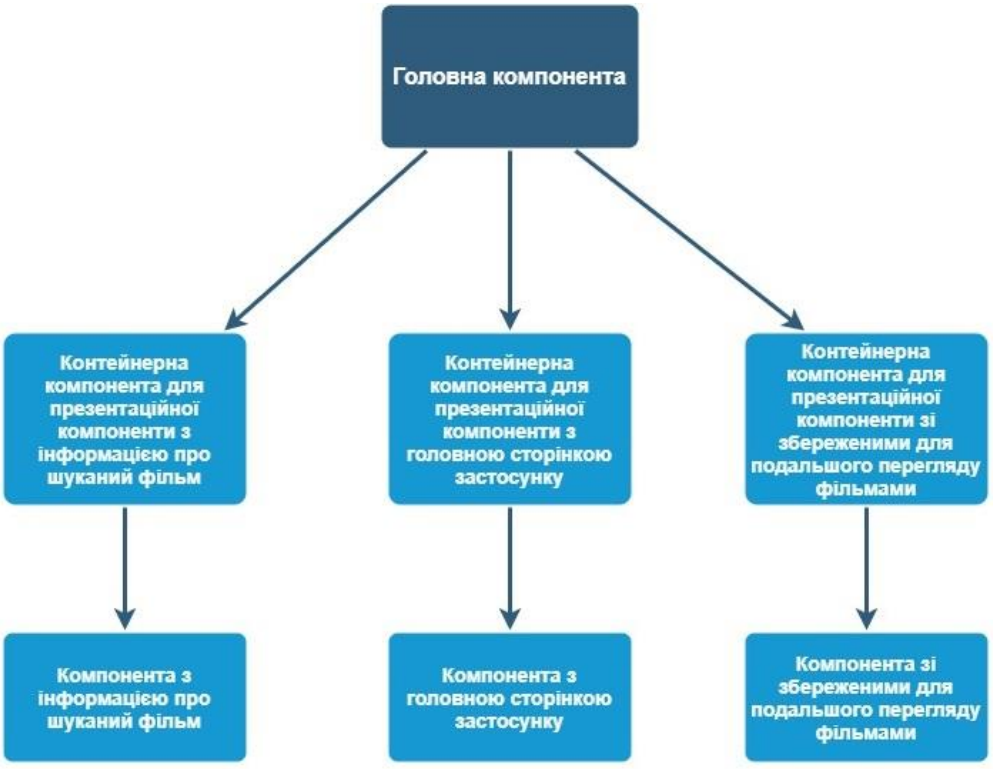

Рисунок 2. 5. Схема компонент застосунку

Таким чином, взаємодію користувача з застосунком можна описати як виклик певних функцій, що відповідають за певні дії (actions) у певний момент часу роботи із застосунком. Більш того, завдяки ефективності архітектури React+Redux, розроблюваний веб-додаток набуває властивостей Single Page Application (SPA) — це більш складна за своїми властивостями структура, аніж просто веб-сайт.

У таких додатках HTML-сторінка генерується по мірі того, як користувач взаємодіє з продуктом, завдяки динамічному оновленню за допомогою JavaScript. До головних переваг SPA відносять:

- швидкість роботи. Більшість ресурсів завантажуються за одну сесію, а у результаті дій користувача дані просто замінюються локально, що економить час;
- доступність користувацького інтерфейсу. За рахунок того, що по факту веб-сторінка всього одна, підвищується рівень доступності інтерфейсу для користувача застосунку;
- сторінка не оновлюється цілком, а лише часткового, за допомогою парадигми AJAX. Це дозволяє додатково підвищити швидкість роботи застосунку, а також збільшити його ефективність;
- SPA однаково добре працюють на будь-якій платформі, що має доступ до мережі Інтернет та встановлений браузер.

Процес взаємодії користувача можливо нагально продемонструвати за допомогою діаграми колаборації (рис. 2.6) (Collaboration Diagram), що детально демонструє усі процеси, які відбуваються під час користування веб-застосунком.

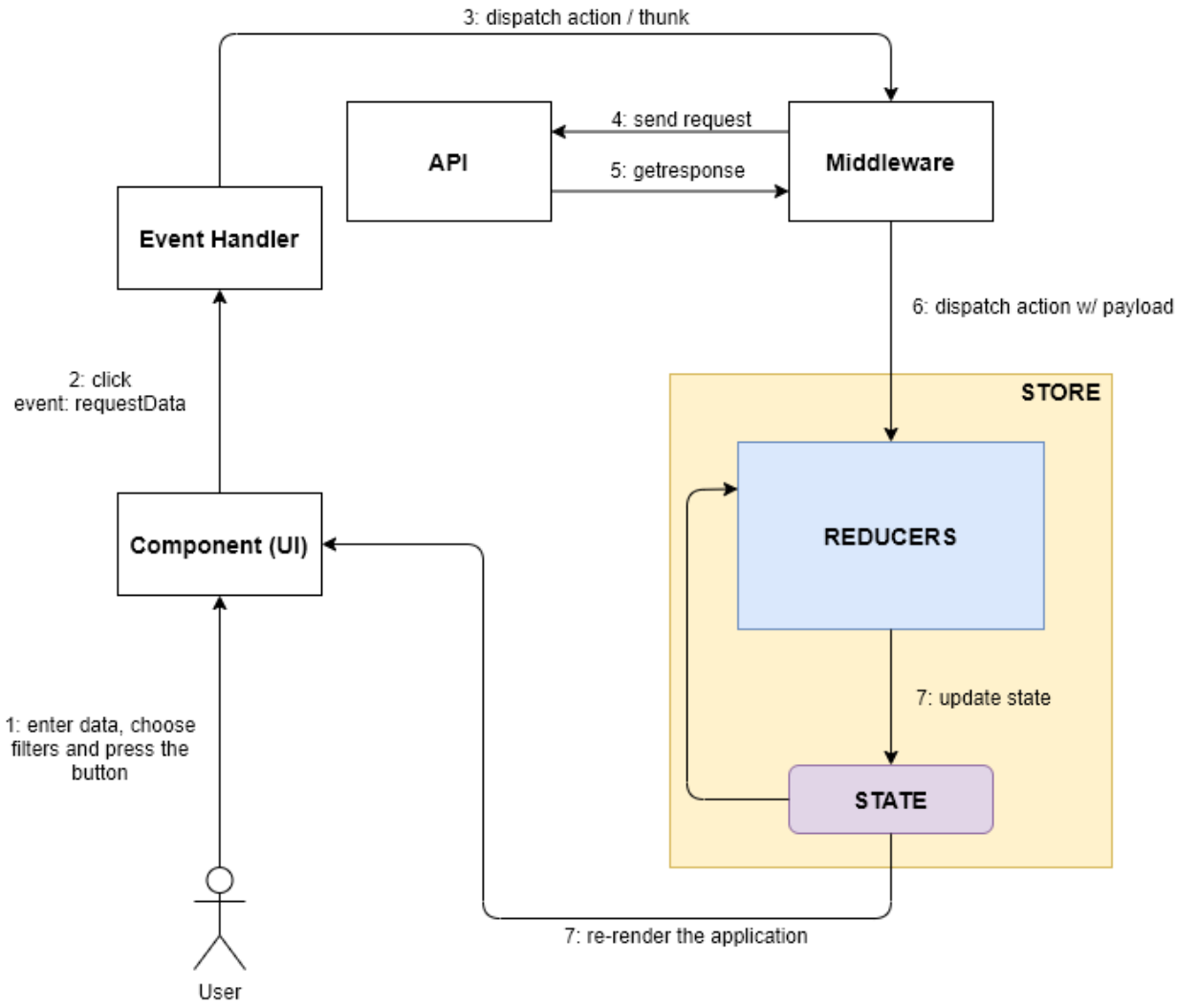

Рисунок 2. 6. Процес взаємодії користувача з веб-застосунком

2.4. Опис файлової структури проекту

Окрім того, одним з найбільш важливих аспектів під час створення того чи іншого застосунку є файлова структура проекту. Добре спроектована файлова структура здатна значно підвищити ефективність процесу розробки того чи іншого застосунку. До того, чому структура файлів та директорій відіграє ключову роль у розробці проекту, можна навести наступні аргументи:

- економія часу. За умови грамотно спланованої файлової структури, буде набагато легше знайти той чи інший файл у разі необхідності у найкоротші терміни;
- більш гнучке управління проектом. Детально продумана файлова структура дозволяє організувати робочий простір та більш системно

підтримувати проект, що дозволяє працювати над розробкою з більшою ефективністю.

Із загальною схемою файлової структури проекту можна ознайомитись нижче (рис. 2.7):

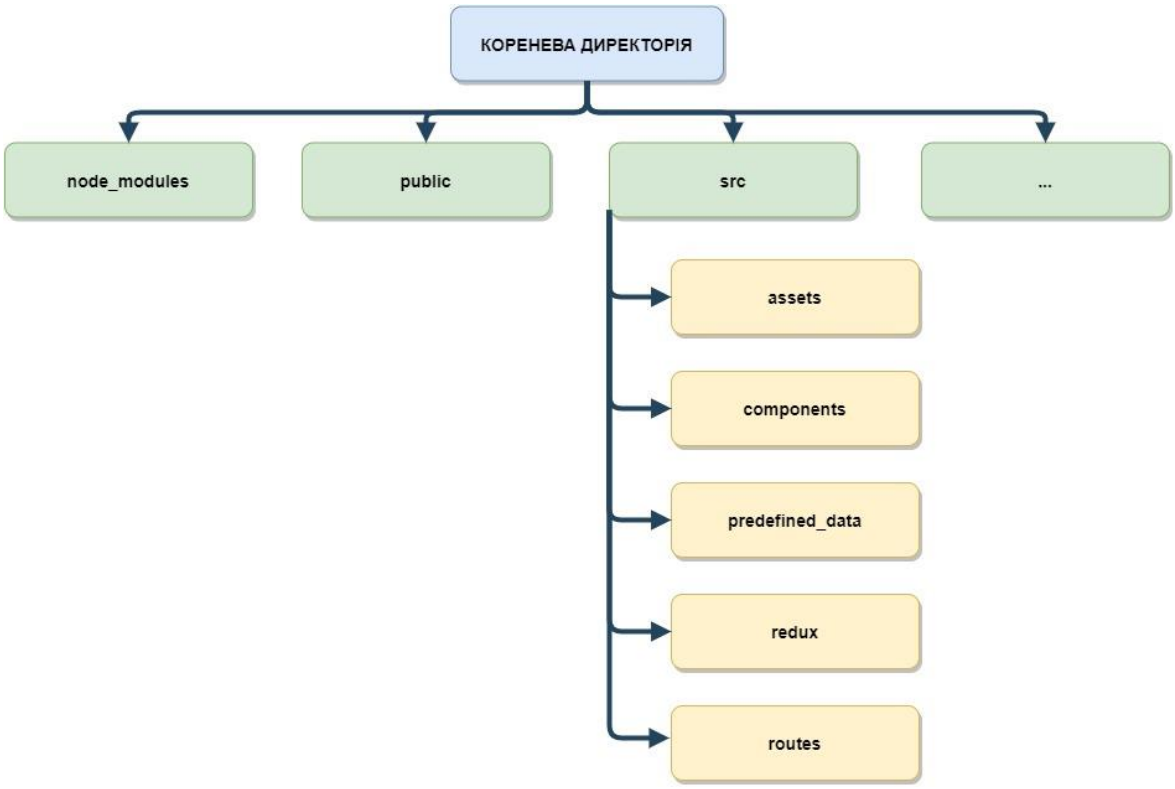

Рисунок 2. 7. Файлова структура проекту

Дивлячись на подібну структуру, можливо одразу зрозуміти, де які файли розташовані, що значно спрощує подальшу роботу над проектом та його менеджмент. Детальний опис кожної директорії наведено у таблиці 2.2.

Таблиця 2. 2. Опис директорій веб-застосунку

Назва директорії	**Опис директорії**
node_modules	Ця директорія містить в собі завантажені бібліотеки для розробки додатку, такі як Redux, Axios і так далі

public	Ця директорія містить у собі кореневий index.html файл проекту, а також ключові медіа-файли
src	Головна директорія проекту. Містить у собі усі папки та файли, що стосуються розмітки та логіки проекту
assets	Ця директорія містить у собі додаткові медіа-файли, такі як зображення головної сторінки та анімацію завантаження у форматі svg
components	Ця директорія містить у собі усі компоненти, як контейнерні, так і презентаційні, разом зі стильовими файлами застосунку
predefined_data	Ця директорія містить у собі дані про кіножанри, що потрібні для коректних запитів до API
redux	Ця директорія містить в собі декілька файлів, що утворюють повноцінну структуру сховища Redux
routes	Ця директорія містить в собі файл з JavaScript-об’єктом, що зберігає у собі шляхи для переходу до відповідних компонент

Продовження таблиці 2. 2. Опис директорій веб-застосунку

2.5. Висновок

У другому розділі кваліфікаційної роботи мною було проаналізовано переваги веб-застосунку як рішення поставленої проблеми над іншими програмними рішеннями. Веб-додаток є оптимальним рішенням, так як його легше підтримувати та обслуговувати за рахунок використання одного коду для усіх платформ, таких як комп'ютери, планшети, смартфони тощо. Такий застосунок буде без проблем функціонувати на будь-якому пристрої, що має доступ до мережі Інтернет та встановлений браузер.

На додаток, мною було обрано та проаналізовано одні з найпопулярніших та найоптимальніших на сьогодні технологій веб-розробки, а саме: HTML, CSS, JavaScript, React, Redux та бібліотеку Axios, за допомогою яких можна створити високоефективний додаток (SPA).

В результаті цього аналізу мною було створено повноцінний веб-застосунок, а також описано його архітектуру та файлову структуру, що є невід'ємною частиною ефективного проекту.

РОЗДІЛ 3. СПЕЦИФІКАЦІЯ СИСТЕМНИХ ВИМОГ. АНАЛІЗ ФУНКЦІОНАЛУ РОЗРОБЛЕНОГО ДОДАТКУ ТА ІНСТРУКЦІЯ КОРИСТУВАЧА. ТЕСТУВАННЯ ЗАСТОСУНКУ

У результаті розробки було створено ефективний веб-додаток, що покликаний усунути проблему надлишкового вибору фільму для перегляду. Зважаючи на обраний перелік програмних рішень, вдалося створити рішення, яке буде доступне на будь-якому пристрої, що має доступ до мережі Інтернет та встановлений браузер.

Додаток не потребує жодних спеціальних системних компонентів, що робить його універсальним рішенням.

3.1. Системні вимоги для функціонування веб-застосунку

Мінімальними системними вимогами для коректного функціонування додатку є центральний процесор (ЦП), що підтримує тактову частоту від 1.9 ГГц, а також архітектуру x86 або x64, 2 гігабайти оперативної пам’яті та дисплей типу Super VGA, що підтримує розподільну здатність 1024x768. Комп’ютер, побудований на базі таких комплектуючих буде спроможний коректно відображати застосунок у окремій вкладинці інтернет-браузеру.

Окрім того, розроблений мною застосунок потребує встановлений браузер Google Chrome, що має версію 85.0.4183.121 та нижче. Цей пункт стосується як комп’ютерів під управлінням операційних систем Windows, Linux та MacOS, так і мобільних пристроїв під управлінням ОС Android та iOS.

Завдяки тому, що перелік системних вимог для коректної роботи веб-додатку є мінімальним, будь-яка сучасна система зможе гарантувати роботу застосунку.

Найбільш сприятливими умовами щодо мережевого зв’язку будуть пропускна здатність більше, ніж 50 КБ/с і затримка менше, ніж 150 мс.

3.2. Аналіз функціоналу розробленого веб-застосунку

Створений веб-застосунок може бути розглянутий у якості сучасного прикладного інструменту, що, подібно до усіх сучасних веб-додатків, реалізує наступні функції: можливість вибору бажаного жанру кінострічки, пошук фільму за відповідно обраним жанром, перегляд інформації про підібраний фільм та можливість збереження певних кінострічок у локальній пам’яті пристрою для подальшого перегляду.

Зважаючи на це, можна виділити такі основні функції веб-застосунку:

- вибір бажаного жанру кінострічки;
- пошук кінострічки відповідно до обраного жанру;
- перегляд загальної інформацій про підібраний фільм (назва, рейтинг, жанри, сюжет тощо.);
- збереження певних кінострічок у локальній пам’яті пристрою для подальшого перегляду.

У результаті підбору рекомендації, додаток надає користувачеві інформацію про один випадковий фільм з можливістю почати пошук знову, якщо поточна кінострічка його не зацікавила.

3.3. Інструкція користувача

Після запуску веб-застосунку, користувач потрапляє на головну сторінку, яка являє собою першочергову відправну точку взаємодії користувача з додатком. На головній сторінці (рис. 3.1) користувач може споглядати наступні елементи:

- текстове привітання користувача у додатку;
- випадаючий список з можливістю вибору жанру;
- кнопку, що виконує функцію пошуку фільму за обраним жанром;
- кнопку, що веде користувача до секції збережених фільмів.

Завдяки цим елементам управління застосунком, користувач зможе в повній мірі керувати продуктом.

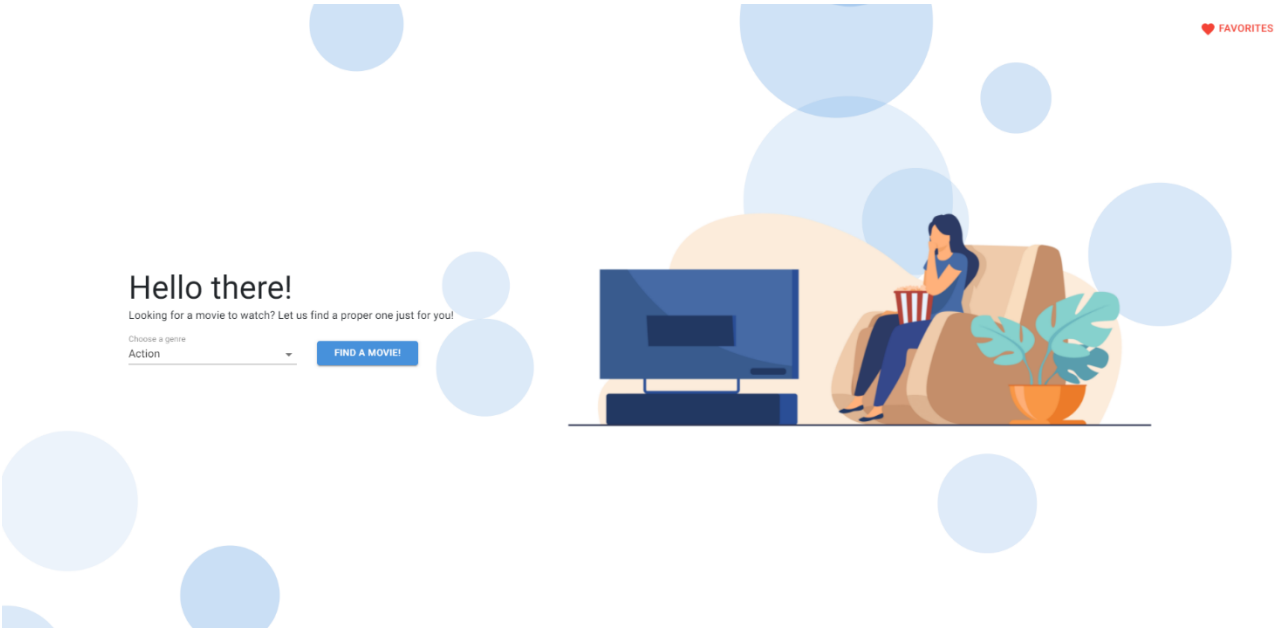

Рисунок 3. 1. Домашня сторінка веб-застосунку

Для того, щоб обрати бажаний жанр кіно для перегляду, користувачеві необхідно натиснути на стрілку (або сам випадаючий список), у результаті чого відкриється відповідне меню (рис. 3.2).

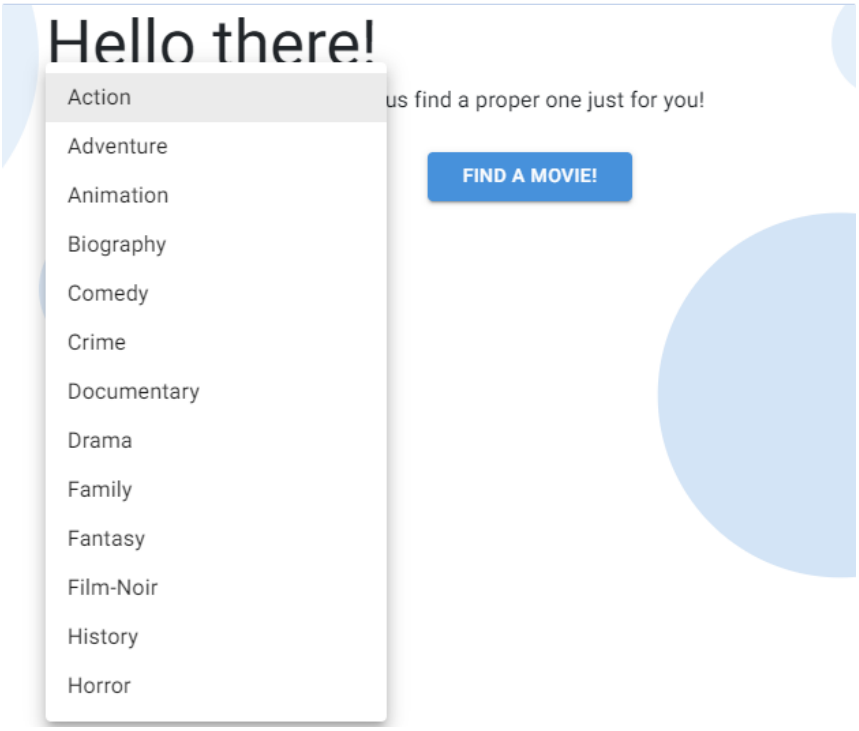

Рисунок 3. 2. Випадаючий список із вибором бажаного жанру

Після того, як бажаний жанр було обрано, користувач має натиснути кнопку «FIND A MOVIE!» (рис. 3.3), для того, щоб розпочати процес пошуку рекомендації.

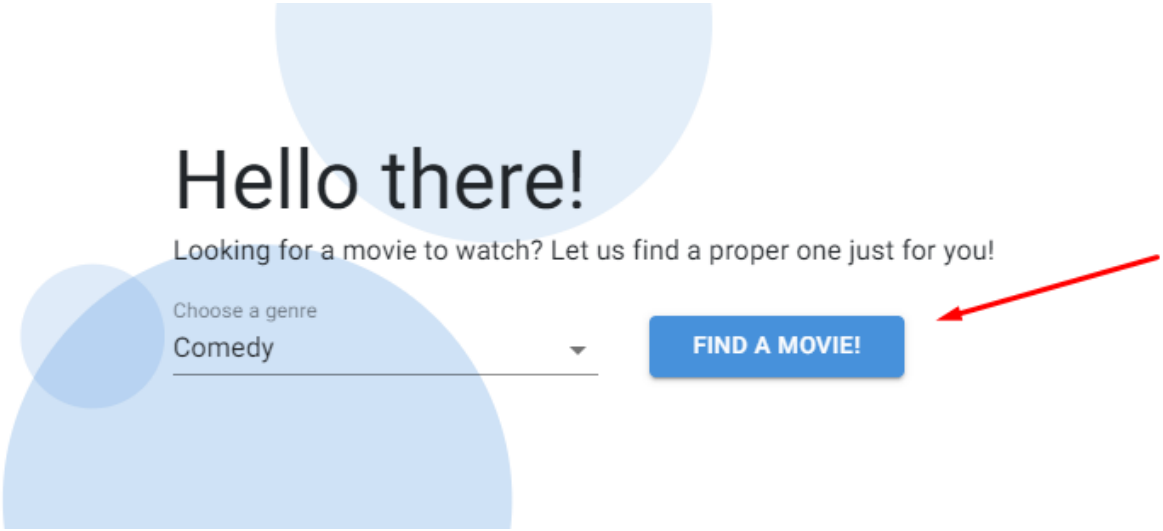

Рисунок 3. 3. Кнопка пошуку фільму за обраним жанром

У результаті цих дій, користувача буде перенаправлено на наступну сторінку застосунку, на котрій він зможе отримати інформацію про підібраний фільм та ознайомитись з нею більш детально (рис. 3.4).

Рисунок 3. 4. Компонента з інформацією про підібраний фільм

Для того, щоб уникнути інформаційного перенавантаження, користувачеві надається інформація лише про один випадковий фільм, який він може пропустити, або зберегти у відповідну секцію для подальшого перегляду.

На цій сторінці користувач має наступні елементи, що дозволяють керувати застосунком:

- кнопка «SEARCH AGAIN», що відповідає за можливість повторного пошуку фільму за тим самим жанром, у разі, якщо підібраний фільм не зацікавив користувача;
- кнопка «SAVE TO FAVORITES», що відповідає за можливість зберегти інформацію про фільм у локальній пам'яті пристрою для подальшого перегляду;
- кнопка «FAVORITES», що відповідає за перехід до сторінки Favorites, де відображається інформація про збережені фільми.

Окрім елементів управління, користувач також може споглядати ключову інформацію про кінострічку, що була надана сервером, а саме:

- плакат кінострічки;
- назву кінострічки та рік її випуску;
- жанрові теги кінострічки (їх може бути декілька, так як в рамках одного фільму декілька жанрів можуть перетинатися);
- рейтинг фільму за 10-бальною шкалою за версією IMDB;
- тривалість фільму у хвилинах;
- загальний сюжет фільму.

У разі натиснення кнопки «SAVE TO FAVORITES», спрацює функція, що відповідає за додавання обраної кінострічки до переліку обраних для подальшого перегляду. Окрім цього, користувач отримає відповідне сповіщення про те, що відповідний фільм було додано до секції Favorites (рис. 3.5).

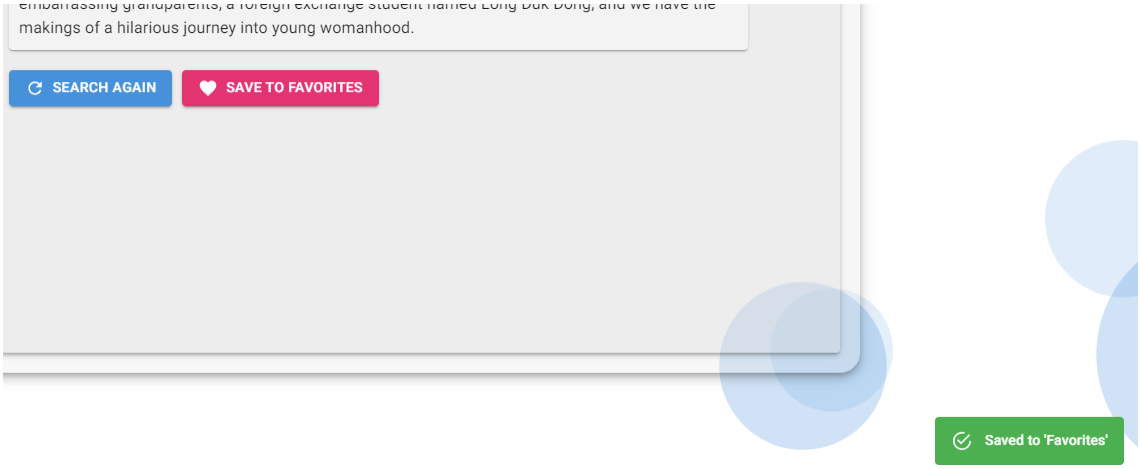

Рисунок 3. 5. Приклад сповіщення про успішне додання до списку улюблених фільмів

Для того, щоб переглянути сторінку зі збереженими фільмами, необхідно натиснути на кнопку «FAVORITES», що знаходиться у правому верхньому куті застосунку. На сторінці Favorites користувач може переглянути список збережених раніше кінострічок, а саме інформацію про назву, сюжет та постер відповідної кінострічки. (рис. 3.6).

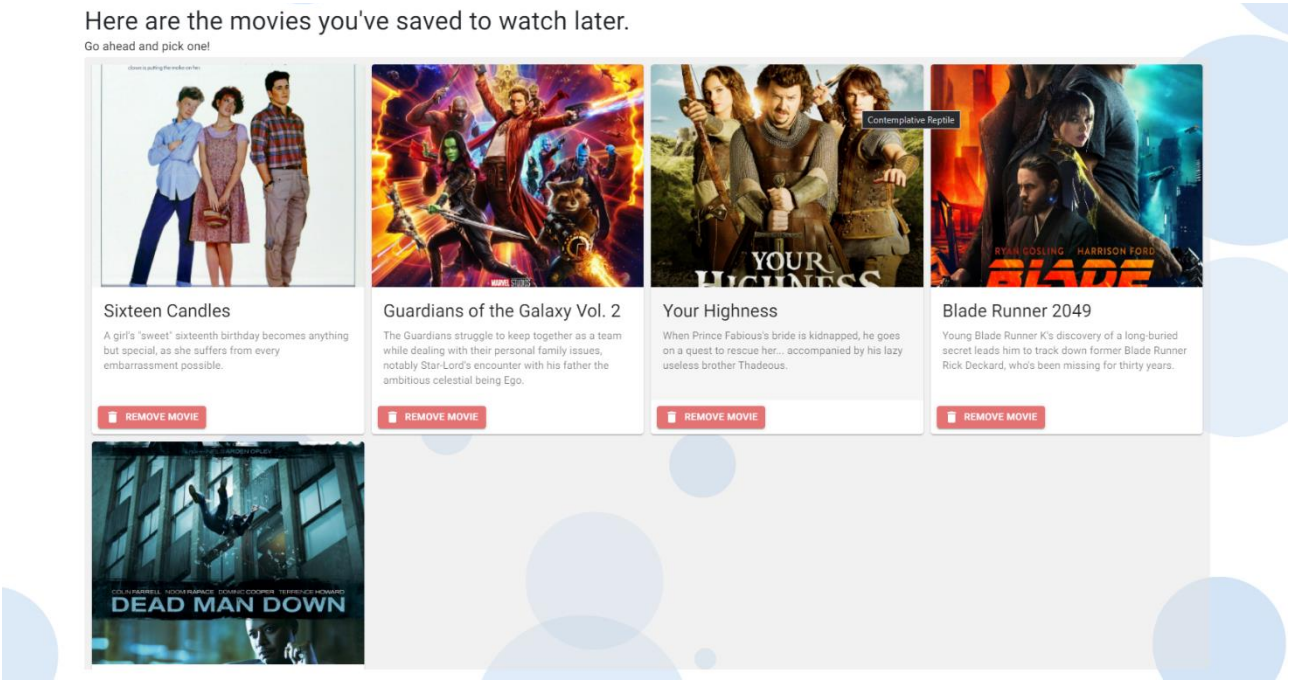

Рисунок 3. 6. Компонента зі списком улюблених фільмів

Окрім цього, на даній сторінці наявний ще один елемент управління додатком: кнопка «REMOVE MOVIE», яка дозволяє видалити той чи інший фільм зі списку збережених.

3.4. Тестування застосунку

Для тестування веб-додатку на предмет відповідності сучасним вимогам до швидкості завантаження, доступності, використання найкращих практик у розробці та SEO-показників, було обрано «Google Lighthouse», що є вбудованим доповненням інтернет-браузера Google Chrome та являє собою автоматизований інструмент з відкритим сирцевим кодом, що слугує для покращення якості веб-сторінок. Він в автоматичному режимі проводить тестування веб-ресурсу за такими показниками:

- Performance (відповідає за швидкість завантаження ресурсу);
- Accessibility (доступність ресурсу);
- Best Practices (аналіз ресурсу на предмет використання сучасних практик у розробці);
- SEO (аналіз того, наскільки добре ресурс буде ранжуватися у пошукових системах, використання семантично-правильних HTML-тегів тощо).

Як видно із результатів тестування (рис. 3.7), створений додаток задовольняє норми та загальні правила ефективного функціонування веб-застосунку, що робить його цілком придатним до використання.

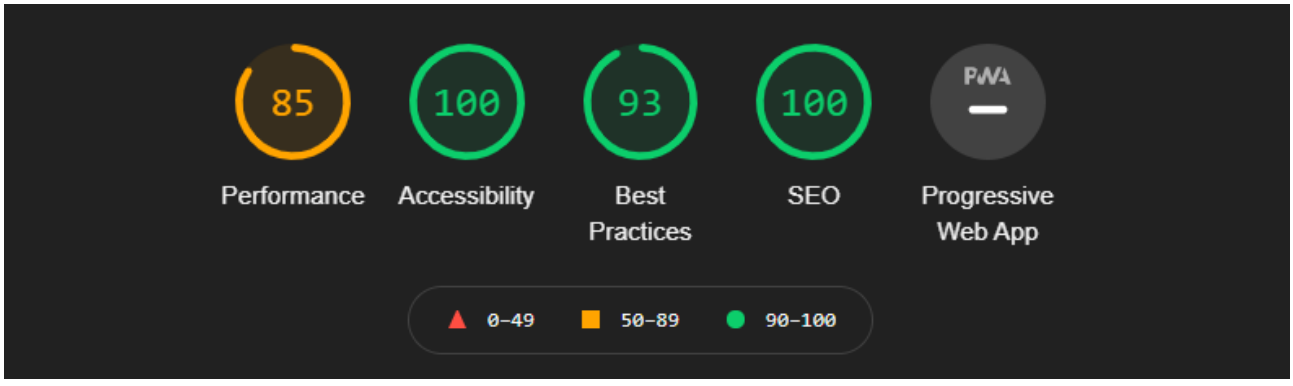

Рисунок 3. 7. Результати тестування веб-застосунку за допомогою Google Lighthouse

3.5. Висновок

У цьому розділі кваліфікаційної роботи мною було продемонстровано загальний алгоритм роботи із розробленим веб-застосунком, а саме такі дії, як: пошук фільму за бажаним жанром, ознайомлення з інформацією про кінострічку, можливість отримання наступної рекомендації у разі, якщо поточний фільм не зацікавив користувача, а також збереження інформації про кінострічку для подальшого перегляду.

Також було виведено мінімальні системні вимоги для коректної роботи застосунку. Завдяки тому, що перелік цих системних вимог на сьогодні можна вважати мінімальним, веб-застосунок зможе без перешкод функціонувати на базі будь-якої сучасної системи.

Окрім цього, веб-застосунок було проаналізовано на предмет відповідності сучасним стандартам розробки за допомогою вбудованого у інтернет-браузер Google Chrome інструменту під назвою Google Lighthouse. Аналіз проведено за наступними пунктами:

- Performance (відповідає за швидкість завантаження ресурсу);
- Accessibility (доступність ресурсу);
- Best Practices (аналіз ресурсу на предмет використання сучасних практик у розробці);
- SEO (аналіз того, наскільки добре ресурс буде ранжуватися у пошукових системах, використання семантично-правильних HTML-тегів тощо).

ВИСНОВКИ

У даній кваліфікаційній роботі мною було розглянуто проблему надлишкового вибору в сегменті цифрових розваг в епоху глобальної діджиталізації, що спричинена набуттям людством «синдрому інформаційної втоми», який характеризує собою складнощі у здійсненні вибору через надлишкову кількість доступних варіантів в умовах глобальної діджиталізації світу.

У відповідності до теми дипломної роботи мною було проведено аналіз існуючих рішень для підбору фільмів, виявлено їх недоліки та переваги, що дозволило мені створити оптимальне рішення проблеми.

У результаті проведених досліджень, мною створено веб-застосунок для підбору фільмів із урахуванням вподобань у якості рішення поставленої проблеми. Для цього було обрано та проаналізовано наступні технології веб-розробки: Hypertext Markup Language (HTML) стандарту 5.0, Cascading Style Sheets (CSS) версії 3.0, JavaScript, а також JavaScript-бібліотек React, що використовується для побудови користувацьких інтерфейсів, Redux, що використовується для загального менеджменту стану веб-застосунку та MaterialUI, а також додаткової бібліотеки Axios, що слугує для скоєння HTTP-запитів до сервера.

Було створено сучасну архітектуру веб-застосунку на базі React та Redux, а також впроваджено найкращі практики у процес розробки.

Після розробки застосунку було проведено аналіз мінімальних системних вимог для його коректної роботи. На додаток, застосунок було протестовано за допомогою Google Lighthouse на предмет відповідності сучасним стандартам веб-розробки.

Результатом виконання кваліфікаційної роботи став повноцінний сучасний веб-застосунок, що здатний зменшити вплив інформаційного перенавантаження на людину, а також спростити процес вибору кінострічки для перегляду, не перенасичуючи користувача додатковою нерелевантною інформацією.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. David Flanagan «JavaScript: The Definitive Guide 7th Edition», O'Reilly Media; August 11, 2020, 600 ст.
2. Douglas Crockford «JavaScript: The Good Parts», O'Reilly Media; May 1, 2008, 176 ст.
3. Why we can’t just pick a movie?, GCFGlobal: веб-сайт. URL: https://edu.gcfglobal.org/en/ (дата звернення: 19.03.2021)
4. Overchoice, Wikipedia.org: веб-сайт. URL: https://en.wikipedia.org/ (дата звернення: 19.03.2021)
5. Information Overload, Pewresearch.org: веб-сайт. URL: https://www.pewresearch.org/ (дата звернення: 19.03.2021)
6. Barry Schwartz «The Paradox Of Choice», Brilliance Audio; April 22, 2014, 304 ст.
7. Study Shows 70% of Consumers Would Rather Watch New Movies at Home Variety.com: веб-сайт. URL: https://variety.com/ (дата звернення: 27.03.2021)
8. React: A JavaScript library for building user interfaces, Reactjs.org: веб-сайт. URL: https://reactjs.org/ (дата звернення: 03.04.2021)
9. JavaScript.info, JavaScript.info: веб-сайт. URL: https://javascript.info/ (дата звернення: 07.04.2021)
10. Redux: A Predictable State Container for JS Apps, Redux.js.org: веб-сайт. URL: https://redux.js.org/ (дата звернення: 13.04.2021)
11. MATERIAL-UI: React components for faster and easier web development, Material-ui.com: веб-сайт. URL: https://material-ui.com/ (дата звернення: 17.04.2021)
12. Styled Components: Visual primitives for the component age, Styled-components.com: веб-сайт. URL: https://styled-components.com/ (дата звернення: 19.04.2021)

13. Alex Banks «Learning React: Functional Web Development with React and Redux», O'Reilly Media; May 18, 2017, 488 ст.
14. David DuRocher «HTML & CSS QuickStart Guide: The Simplified Beginners Guide to Developing a Strong Coding Foundation, Building Responsive Websites, and Mastering the Fundamentals of Modern Web Design», ClydeBank Media LLC; January 21, 2021, 468 ст.
15. Anthony Accomazzo «Fullstack React: The Complete Guide to ReactJS and Friends», Fullstack.io; September 12, 2017, 836 ст.
16. Chris Minnick «Coding with JavaScript For Dummies», For Dummies; May 26, 2015, 368 ст.
17. MDN Web Docs: Resources for developers, by developers, Developer.mozilla.org: веб-сайт. URL: https://developer.mozilla.org (дата звернення: 26.04.2021)
18. particles.js, Vincentgarreau.com: веб-сайт. URL: https://vincentgarreau.com/particles.js/ (дата звернення: 03.05.2021)
19. Adding CSS Animations with Styled Components, medium.com: веб-сайт. URL: https://medium.com/ (дата звернення: 04.05.2021)
20. IMDb API, Rapidapi.com: веб-сайт. URL: https://rapidapi.com/ (дата звернення: 05.05.2021)
21. API Endpoints - What Are They? Why Do They Matter?, Smartbear.com: веб-сайт. URL: https://smartbear.com/ (дата звернення: 06.05.2021)
22. Panos Louridas «Algorithms (The MIT Press Essential Knowledge series)», The MIT Press; August 18, 2020, 312 ст.
23. Alex Banks, Eve Porcello «Learning React: Functional Web Development with React and Redux», O'Reilly Media; May 18, 2017, 350 ст.
24. Alex Banks, Eve Porcello «Learning React: Modern Patterns for Developing React Apps», O'Reilly Media; July 7, 2020, 310 ст.
25. Paul McFedries «Web Design Playground: HTML & CSS the Interactive Way», Manning Publications; May 19, 2019, 440 ст.

26. Ben Frain «Responsive Web Design with HTML5 and CSS: Develop future-proof responsive websites using the latest HTML5 and CSS techniques, 3rd Edition», Packt Publishing; April 30, 2020, 408 ст.
27. React Router, Reactrouter.com: веб-сайт. URL: https://reactrouter.com/ (дата звернення: 07.05.2021)
28. Axios: Promise based HTTP client for the browser and node.js, Axios-http.com: веб-сайт. URL: https://axios-http.com/ (дата звернення: 08.05.2021)
29. 10 Steps To Conquering Information Overload, Forbes.com: веб-сайт. URL: https://www.forbes.com/ (дата звернення: 19.03.2021)
30. Stackoverflow: Every has a tab open to Stack Overflow, Stackoverflow.com: веб-сайт. URL: https://stackoverflow.com/ (дата звернення: 14.05.2021)

ДОДАТКИ

Додаток А. Код асинхронних функцій запитів до IMDb API

```
export function fetchMovieList(query) {
  return axios.get(
    'https://imdb8.p.rapidapi.com/title/get-popular-movies-by-genre',
    {
      params: { genre: query },
      headers: {
        'x-rapidapi-key': '0107db1c18msh764556930ecf372p1f9d45jsx4406c16ca7af',
        'x-rapidapi-host': 'imdb8.p.rapidapi.com',
      },
    },
  )
}

export function fetchRandomMovieFromList(movieId) {
  return axios.get('https://imdb8.p.rapidapi.com/title/get-overview-details', {
    params: {
      tconst: movieId,
      currentCountry: 'US',
    },
    headers: {
      'x-rapidapi-key': '0107db1c18msh764556930ecf372p1f9d45jsx4406c16ca7af',
      'x-rapidapi-host': 'imdb8.p.rapidapi.com',
    },
  })
}

export function fetchMovieData(query) {
  return async (dispatch) => {
    dispatch(toggleInfoUpdate(true))
    try {
      const movieList = await fetchMovieList(query)
      const movieId =
        movieList.data[Math.floor(Math.random() * movieList.data.length)]
      const cutMovieId = movieId.substring(7, movieId.length - 1)
      const response = await fetchRandomMovieFromList(cutMovieId)
      dispatch({
        type: STORE_RANDOM_MOVIE,
```

```
        payload: response.data,
      })
      dispatch(toggleInfoUpdate(false))
    } catch (error) {
      if (error.response) {
        console.log(error.response.data)
        console.log(error.response.status)
        console.log(error.response.headers)
        // dispatch(showAlert(error.response))
      } else if (error.request) {
        console.log(error.request)
      } else {
        console.log('Error', error.message)
      }
      // console.log(error.config)
    }
  }
}
```

Додаток Б. Код головної компоненти проекту

```
import React from 'react'
import './App.css'
import { Switch, Route } from 'react-router-dom'
import { StylesProvider } from '@material-ui/core/styles'
import { HomeContainer } from './components/Home/HomeContainer'
import { routes } from './routes/routes'
import { MovieSuggestionContainer } from
'./components/MovieSuggestion/MovieSuggestionContainer'
import { ParticlesBackground } from
'./components/ParticlesBackground/ParticlesBackground'
import { SuccessAlert } from './components/Alerts/SuccessAlert'
import { FavoriteMoviesContainer } from
'./components/FavoriteMovies/FavoriteMoviesContainer'

function App() {
  return (
    <div className="App">
      <StylesProvider injectFirst>
        <Switch>
          <Route exact path={routes.homepage} component={HomeContainer} />
          <Route
            exact
            path={routes.movie_suggestion}
            component={MovieSuggestionContainer}
          />
          <Route
            exact
            path={routes.favorites}
            component={FavoriteMoviesContainer}
          />
        </Switch>
        <SuccessAlert />
      </StylesProvider>
      <ParticlesBackground />
    </div>
  )
}

export default App
```