МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ ЕКОНОМІКИ І ТЕХНОЛОГІЙ

ННІ Економіки та бізнес-освіти

Кафедра Економіки та цифрового бізнесу

Спеціальність 122 «Комп'ютерні науки»

Форма навчання Денна

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

Гнідаша Богдана Максимовича
(прізвище, ім'я, по батькові здобувача)

на тему Розробка вебзастосунку для перегляду популярних фільмів з використанням API TMDB
(повна назва теми)

за матеріалами

(повна назва бази дослідження)

науковий керівник к.т.н. _____ Селезньов М.Є.
(наук. ступінь, вчене звання) *(підпис)* *(прізвище, ініціали)*

Робота допущена до захисту в ЕК
Протокол засідання кафедри
від 12 червня 2026 р. № 15
Завідувач кафедри _____
(підпис)
к.е.н., доцент Радько В.М.
наук. ступень, вчене звання прізвище, ініціали

Кривий Ріг – 2026

ЗАТВЕРДЖЕНО
Наказ Міністерства освіти і науки, молоді та спорту України
29 березня 2012 року № 384

Форма № Н-9.01

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ ЕКОНОМІКИ І ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ЕКОНОМІКИ ТА БІЗНЕС-ОСВІТИ
(повне найменування вищого навчального закладу)

Кафедра економіки та цифрового бізнесу
Освітній ступінь бакалавр
Спеціальність 122 Комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедри _____ В.М. Радько

"30" березня 2026 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ БАКАЛАВРСЬКУ РОБОТУ ЗДОБУВАЧУ

Гнідашу Богдану Максимовичу

1. Тема роботи Розробка вебзастосунку для перегляду популярних фільмів з використанням API TMDB

науковий керівник роботи Селезньов Максим Євгенович, к.т.н.
затверджені наказом вищого навчального закладу від «23» березня 2026 р. № 193-ст

2. Строк подання здобувачем роботи 29.05.2026р.

3. Зміст кваліфікаційної бакалаврської роботи, об'єкт, предмет та мета дослідження:
Розділ 1 Теоретичні аспекти розробки вебзастосунку для перегляду популярних фільмів
Розділ 2 Проєктування та розробка вебзастосунку для перегляду популярних фільмів
Розділ 3 Функціонування та елементи інтерфейсу вебзастосунку для перегляду популярних фільмів

Об'єкт дослідження – процес отримання, обробки, збереження та відображення інформації про популярні фільми у вебзастосунку

Предмет дослідження - методи, моделі та програмні засоби розробки вебзастосунку для перегляду популярних фільмів з використанням API TMDB та локальної бази даних

Мета кваліфікаційної бакалаврської роботи – розробити вебзастосунок для перегляду популярних фільмів з використанням API TMDB, який забезпечує пошук, фільтрацію, перегляд детальної інформації про фільми, а також збереження персоналізованих даних користувача

4. Дата видачі завдання 03.04.2026р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної бакалаврської роботи	Строк виконання етапів роботи	Відмітка керівника про виконання етапів (дата, підпис)
1	Підготовка розділу 1	до 17.04.2026р.	14.04.2026р.
2	Підготовка розділу 2	до 08.05.2026р.	15.05.2026р.
3	Підготовка розділу 3	до 25.05.2026р.	26.05.2026р.
4	Реєстрація завершеної кваліфікаційної роботи	до 29.05.2026р.	28.05.2026р.
5	Отримання відгуку від наукового керівника	04.06.2026р.	04.06.2026р.
6	Отримання зовнішньої рецензії	05.06.2026р.	05.06.2026р.
7	Попередній захист кваліфікаційної роботи на кафедрі	08.06.2026р.	08.06.2026р.
8	Перевірка кваліфікаційної роботи на плагіат	12.06.2026р.	12.06.2026р.
9	Допуск кафедрою кваліфікаційної роботи до захисту	12.06.2026р.	12.06.2026р.
10	Підготовка студента до захисту в ЕК	до 19.06.2026р.	18.06.2026р.

Завдання підготував науковий керівник _____ Селезньов М. Є.

Завдання одержав здобувач _____ Гнідаш Б. М.

РЕФЕРАТ

Робота містить 68 сторінок, 28 рисунків, 32 джерела та 1 додаток.

Об’єкт дослідження - процес отримання, обробки, збереження та відображення інформації про популярні фільми у вебзастосунку.

Мета роботи - розробити вебзастосунок для перегляду популярних фільмів з використанням API TMDB, який забезпечує пошук, фільтрацію, перегляд детальної інформації про фільми, а також збереження персоналізованих даних користувача.

Методи дослідження та програмні засоби - аналіз предметної області, методи проєктування інформаційних систем, UML-моделювання, проєктування баз даних, клієнт-серверний підхід, засоби веброзробки Node.js, Express, SQLite, HTML, CSS, JavaScript, а також зовнішній програмний інтерфейс TMDB API.

У роботі проаналізовано особливості побудови вебзастосунків для перегляду фільмів, визначено функціональні вимоги до системи, спроєктовано інформаційну модель вебресурсу та локальну базу даних, розроблено серверну і клієнтську частини застосунку. Реалізовано функції перегляду популярних фільмів, пошуку за назвою, фільтрації за жанром, перегляду детальної інформації про фільм, а також персоналізовані можливості користувача, зокрема реєстрацію, вхід, роботу з обраним та історією пошуку.

Результати роботи можуть бути використані як основа для подальшого розвитку інформаційного вебзастосунку кінематографічного спрямування з можливістю розширення функціоналу, удосконалення інтерфейсу та впровадження додаткових персоналізованих сервісів.

ВЕБЗАСТОСУНОК, КЛІЄНТ-СЕРВЕРНА АРХІТЕКТУРА, ЛОКАЛЬНА БАЗА ДАНИХ, ПЕРСОНАЛІЗОВАНІ ДАНІ, ПОПУЛЯРНІ ФІЛЬМИ, ПОШУК ФІЛЬМІВ, API TMDB.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

API – Application Programming Interface, програмний інтерфейс застосунку.

TMDB – The Movie Database, сервіс для отримання інформації про фільми.

IMDb – Internet Movie Database, інформаційно-довідковий вебресурс про фільми та серіали.

UML – Unified Modeling Language, мова моделювання програмних систем.

ER – Entity-Relationship, модель «сутність-зв’язок».

CRUD – Create, Read, Update, Delete, основні операції роботи з даними.

SPA – Single Page Application, односторінковий вебзастосунок.

REST – Representational State Transfer, архітектурний підхід до організації вебвзаємодії.

HTTP – HyperText Transfer Protocol, протокол передавання даних у вебсередовищі.

JSON – JavaScript Object Notation, формат обміну структурованими даними.

HTML – HyperText Markup Language, мова розмітки вебсторінок.

CSS – Cascading Style Sheets, мова опису стилів вебсторінок.

SQL – Structured Query Language, мова запитів до бази даних.

UX – User Experience, користувацький досвід взаємодії з вебзастосунком.

ВСТУП

У сучасних умовах цифровізації суспільства вебтехнології відіграють важливу роль у забезпеченні доступу користувачів до інформаційних, комунікаційних і розважальних сервісів. Одним із найбільш поширених напрямів використання вебтехнологій є створення застосунків для роботи з медіаконтентом, зокрема для пошуку, перегляду та аналізу інформації про фільми. Розвиток інтернет-інфраструктури, зростання обсягів цифрових даних і поширення мобільних пристроїв суттєво змінили спосіб споживання аудіовізуального контенту. Якщо раніше користувач здебільшого отримував інформацію про фільми з друкованих видань, телепрограм або окремих тематичних сайтів, то нині основним джерелом відомостей стали інтерактивні вебресурси, які забезпечують швидкий доступ до структурованих даних, рейтингів, жанрових добірок, описів, акторського складу та інших характеристик кінематографічного продукту.

Актуальність теми кваліфікаційної бакалаврської роботи зумовлена тим, що сучасний користувач працює в умовах значного інформаційного навантаження. Велика кількість фільмів, серіалів, добірок, платформ і рекомендацій ускладнює процес вибору потрібного контенту. За таких умов зростає потреба у вебзастосунках, які не лише надають доступ до інформації, а й забезпечують її впорядковане подання, зручний пошук, фільтрацію та персоналізовану взаємодію. Особливого значення набуває можливість швидко отримати відомості про популярні фільми, порівняти їх за окремими характеристиками, переглянути рейтинги, ознайомитися з описом і складом акторів, а також сформувати власний список збереженого контенту.

Сфера розважальних вебсервісів постійно розвивається під впливом нових технічних рішень і змін користувацьких очікувань. Для сучасних інформаційних ресурсів важливими є не лише повнота даних, а й якість взаємодії з користувачем, швидкість роботи, адаптивність інтерфейсу, логічна структура сторінок та можливість персоналізації. Користувач очікує, що вебзастосунок

буде працювати стабільно, забезпечуватиме інтуїтивно зрозумілий доступ до основних функцій та дозволятиме виконувати типові дії без зайвих переходів і ускладнень. У випадку вебзастосунку для перегляду популярних фільмів такими діями є відкриття головної сторінки, перегляд списку актуального контенту, використання пошуку, фільтрації за жанром, перехід до сторінки фільму та робота з персональними функціями.

Ще одним важливим чинником актуальності є поширення зовнішніх програмних інтерфейсів, які дозволяють інтегрувати у вебзастосунок великі обсяги структурованих даних без необхідності створення власної повномасштабної бази контенту. У цьому контексті доцільним є використання API TMDB як джерела інформації про фільми.

Практична значущість теми також визначається тим, що розробка подібного вебзастосунку передбачає поєднання кількох важливих складових сучасної веброзробки. До них належать проєктування інформаційної моделі, побудова клієнт-серверної архітектури, організація взаємодії із зовнішнім API, використання локальної бази даних для збереження персоналізованих даних, реалізація механізмів реєстрації й авторизації, а також створення інтерфейсу, орієнтованого на реальні сценарії використання.

Об’єкт дослідження – процес отримання, обробки, збереження та відображення інформації про популярні фільми у вебзастосунку.

Предмет дослідження – методи, моделі та програмні засоби розробки вебзастосунку для перегляду популярних фільмів з використанням API TMDB та локальної бази даних.

Мета кваліфікаційної бакалаврської роботи полягає у розробці вебзастосунку для перегляду популярних фільмів з використанням API TMDB, який забезпечує пошук, фільтрацію, перегляд детальної інформації про фільми, а також збереження персоналізованих даних користувача.

Для досягнення поставленої мети в роботі було визначено такі завдання: виконати аналіз предметної області та існуючих вебресурсів для роботи з медіаконтентом, обґрунтувати вибір архітектури та технологічного стеку

вебзастосунку, спроєктувати інформаційну модель системи й локальну базу даних, визначити підходи до збору та обробки даних із зовнішнього сервісу TMDB API, реалізувати серверну частину застосунку для обробки запитів, авторизації користувачів та взаємодії з локальною базою даних, реалізувати клієнтську частину з основними сторінками та елементами інтерфейсу, забезпечити персоналізовані можливості користувача, пов’язані з обраним та історією пошуку, здійснити оцінку функціонування розробленого вебзастосунку.

У роботі використано такі методи дослідження: аналіз і узагальнення науково-технічних джерел для визначення особливостей сучасних вебресурсів у сфері медіаконтенту, методи структурного та об’єктного моделювання при побудові UML-діаграм і проєктуванні бази даних, системний підхід для формування архітектури програмного рішення, методи програмної реалізації для створення клієнтської та серверної частин застосунку, методи функціонального аналізу та тестування для перевірки працездатності системи.

Теоретичне значення роботи полягає в узагальненні підходів до побудови вебзастосунків для перегляду популярних фільмів та в систематизації рішень, пов’язаних із використанням зовнішніх інформаційних сервісів, локальних баз даних і клієнт-серверної моделі взаємодії. Практичне значення одержаних результатів полягає у створенні працездатного вебзастосунку, який забезпечує перегляд популярних фільмів, виконання пошуку, фільтрацію за жанром, перегляд детальної інформації про фільми, а також використання персоналізованих функцій зареєстрованого користувача. Розроблене рішення може бути використане як базова реалізація інформаційного вебсервісу кінематографічного спрямування та як основа для подальшого розширення функціоналу.

До основних особливостей розроблюваного вебзастосунку належить поєднання зовнішнього джерела даних та локального збереження інформації користувача. Це дозволяє одночасно працювати з актуальними кінематографічними даними та реалізовувати персоналізовані функції без

потреби створення великої локальної колекції фільмів. У межах цієї роботи зовнішній сервіс використовується для одержання даних про популярні фільми, результати пошуку, жанри, деталізовані картки фільмів та список схожих фільмів, тоді як локальна база даних забезпечує збереження облікових даних користувача, списку обраного й історії пошуку. Завдяки цьому вебзастосунок поєднує інформаційну та персоналізовану складові в межах єдиного програмного рішення.

Важливим аспектом є також навчально-прикладний характер розробки. Створення такого вебзастосунку дає змогу відпрацювати навички проєктування програмного забезпечення, організації структури проєкту, побудови модулів серверної логіки, роботи з маршрутизацією, обробки запитів і відповідей, налаштування сесій, реалізації реєстрації та входу, а також створення клієнтського інтерфейсу з динамічним відображенням даних. У сукупності це дозволяє розглядати тему роботи як таку, що має не лише теоретичне, а й вагоме практичне значення для підготовки фахівця у сфері комп’ютерних наук.

У цілому тема кваліфікаційної бакалаврської роботи відповідає сучасним тенденціям розвитку вебтехнологій, потребам користувачів у швидкому доступі до структурованої інформації та вимогам до створення функціональних інформаційних сервісів. Розробка вебзастосунку для перегляду популярних фільмів з використанням API TMDB є доцільним практичним завданням, яке дозволяє поєднати сучасні підходи до проєктування, програмної реалізації та організації взаємодії користувача з інформаційною системою.

РОЗДІЛ 1
ТЕОРЕТИЧНІ АСПЕКТИ РОЗРОБКИ ВЕБЗАСТОСУНКУ ДЛЯ ПЕРЕГЛЯДУ ПОПУЛЯРНИХ ФІЛЬМІВ

1.1 Актуальність вебресурсів для перегляду медіа-контенту та їх роль у розважальній сфері

У сучасних умовах цифрової трансформації веб-ресурси для перегляду медіа-контенту перетворилися на один із ключових інструментів організації дозвілля, доступу до розважальної інформації та вибору аудіовізуального продукту. Їх значення визначається не лише технічною можливістю відображати фільми, серіали, трейлери чи рейтинги, а й здатністю формувати для користувача зручне середовище орієнтації в надлишку контенту. Якщо раніше споживання медіа значною мірою було прив’язане до телемовлення, друкованих програм або окремих локальних каталогів, то сьогодні переважає модель цифрового доступу, в якій користувач очікує швидкого пошуку, зрозумілої структури, мобільної доступності та персоналізованих рекомендацій [1, 3, 4].

Актуальність веб-ресурсів цього типу безпосередньо пов’язана зі зміною характеру інтернет-трафіку та зростанням ролі відеоконтенту в цифровому середовищі. За даними Ericsson Mobility Report, відео залишається найзначущішим типом трафіку, який генерують користувачі смартфонів, а його частка в структурі мобільного споживання продовжує зростати [2]. В іншому аналітичному матеріалі Ericsson зазначається, що у 2020–2024 рр. саме відеотрафік продемонстрував найвиразніше зростання серед основних категорій цифрового споживання [1]. Це свідчить про те, що розважальний відеоконтент уже давно перестав бути периферійним сегментом інтернету й фактично став одним з головних драйверів розвитку веб-сервісів, мобільних інтерфейсів і платформених моделей доступу до інформації про фільми та серіали [1, 2].

Поширення веб-ресурсів для роботи з медіаконтентом також зумовлене загальною трансформацією розважальної сфери. У сучасних дослідженнях

підкреслюється, що швидке поширення “online streaming” та “over-the-top” сервісів суттєво змінило моделі домашнього медіаспоживання, сформувавши нові поведінкові практики, зокрема інтенсивний перегляд, серійне споживання контенту та більш тісну взаємодію користувача з цифровими каталогами [4]. Водночас “video-on-demand” сервіси стали вагомим елементом цифрової економіки, а онлайн-відеоспоживання перетворилося на один із провідних способів використання інтернету [3]. За таких умов веб-ресурси для перегляду популярних фільмів виконують уже не тільки довідкову функцію, а й функцію навігації в цифровому культурному просторі, допомагаючи користувачеві знайти, оцінити та обрати релевантний контент [3, 4].

Окремим чинником актуальності є явище інформаційного перевантаження. Надмірна кількість назв, жанрів, добірок, трейлерів, платформ і рейтингів значно ускладнює процес вибору. У праці, присвяченій впливу рекомендаційної системи Netflix на choice overload, підкреслюється, що в умовах перенасичення цифрового середовища користувач стикається не лише з проблемою пошуку, а й з когнітивним навантаженням, пов’язаним із самим процесом прийняття рішення [5]. Для цього сучасні веб-ресурси використовують рейтинги, персоналізовані добірки, блоки популярного контенту, фільтри та пошукові механізми. Як показує систематичний огляд сучасних рекомендаційних систем, їх головне призначення полягає у фільтрації надлишкової інформації та скороченні часу, який користувач витрачає на пошук релевантного контенту [8]. Отже, актуальність веб-ресурсів для перегляду медіаконтенту визначається не тільки попитом на сам контент, а й потребою в ефективних інструментах його впорядкування [5, 8].

Для ресурсів, орієнтованих на фільми, надзвичайно важливим є не просто накопичення даних, а якість їх подання. Користувач очікує, що веб-ресурс дозволить швидко отримати назву стрічки, опис сюжету, дату виходу, рейтинг, акторський склад, жанр, постер, трейлер і, за можливості, схожі рекомендації. Саме тому значення набувають юзабіліті, візуальна ієрархія, логіка розміщення елементів і загальна зручність взаємодії. Систематичний огляд засвідчує, що UX

у сучасних мобільних і веб-орієнтованих застосунках розглядається як критично важливий чинник, пов’язаний із прийнятністю інтерфейсу, вимогами користувача, зручністю, контекстом використання та якістю взаємодії [6]. Це означає, що веб-ресурс для перегляду популярних фільмів повинен вирішувати не лише задачу доступу до даних, а й задачу організації досвіду користувача, тобто забезпечення інтуїтивного, швидкого та психологічно комфортного сценарію взаємодії [6, 11].

Не менш важливою є адаптивність веб-ресурсу та його доступність для різних категорій користувачів (див. рис. 1.1). Значна частина взаємодії з медіаконтентом сьогодні відбувається зі смартфонів, а отже інтерфейс має бути зручним на малих екранах, підтримувати швидке завантаження, чітку навігацію та зрозумілу структуру сторінок. Разом з тим сучасне веб-середовище дедалі більше орієнтується на інклюзивність. У дослідженні, присвяченому ментальній моделі веб-навігації для незрячих користувачів, наголошується, що сайти не завжди полегшують пошук інформації для всіх категорій користувачів, а проблеми доступності можуть призводити до фрустрації та фактичного виключення з цифрового простору [7]. У контексті веб-ресурсу для перегляду фільмів це означає, що актуальність сервісу визначається не лише його функціональністю, а й рівнем доступності, зрозумілості та універсальності інтерфейсу [6, 7]. Важливим аргументом на користь актуальності таких ресурсів є також їх здатність утримувати увагу користувача та формувати повторне використання. Згідно з дослідженням, якість системи, якість інформації та якість сервісу мають прямий позитивний вплив на задоволеність користувача, а задоволеність, у свою чергу, впливає на намір продовжувати використання платформи [9]. Для веб-ресурсів, пов’язаних із популярними фільмами, це особливо важливо, адже користувачі звертаються до них не одноразово, а регулярно. Під час вибору фільму, пошуку новинок, перегляду рейтингів або ознайомлення з деталями про стрічку. Роль такого веб-ресурсу у розважальній сфері полягає не тільки в інформуванні, а й у підтримці стійкої взаємодії між користувачем і цифровим сервісом [9].

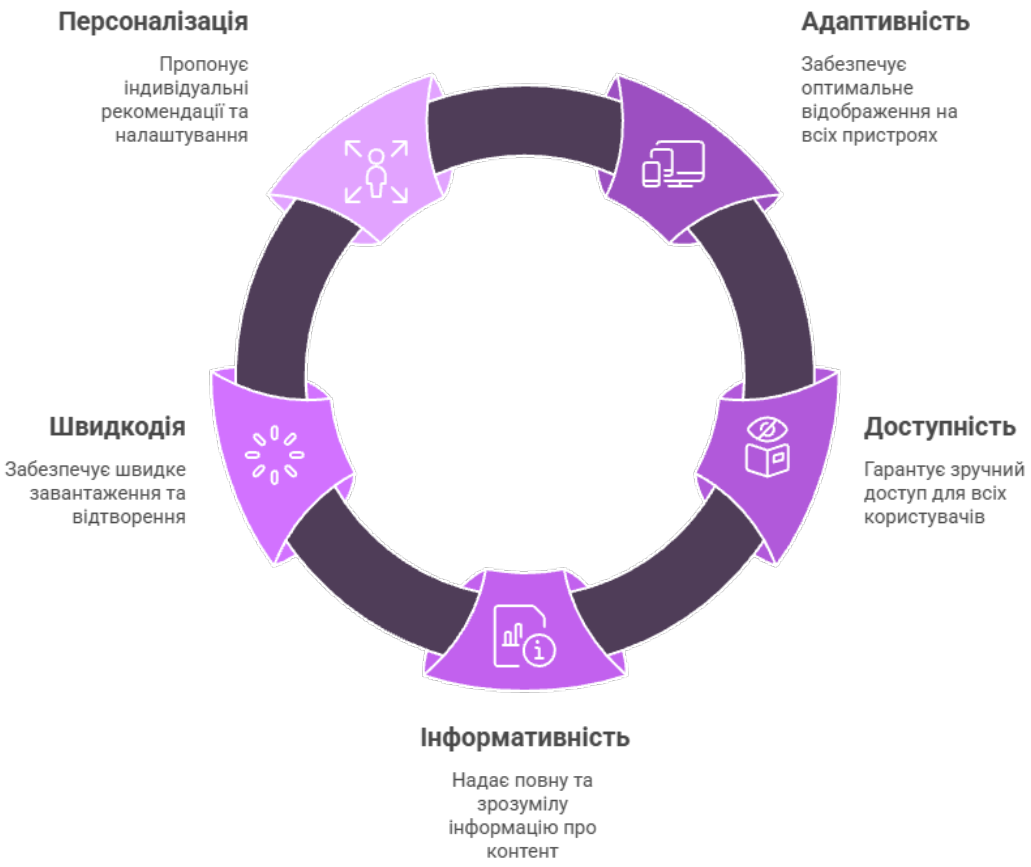

Рис. 1.1 Основні вимоги до сучасного веб-ресурсу для перегляду медіа-контенту

На сучасному етапі розвитку медіаплатформ значення персоналізації зростає ще більше. Це підтверджується не лише статтями, а й сучасними фаховими виданнями 2023–2024 років, присвяченими рекомендаційним системам. Зокрема, наголошується на тому, що рекомендаційні системи стали важливою складовою практично кожного internet-based business, а серед актуальних підходів розглядаються hybrid, deep learning- та graph-based моделі [10]. Також, підкреслюється значення сучасних сесійних підходів і глибинного навчання для формування рекомендацій у середовищах, де важливими є короткочасні поведінкові сигнали користувача [11]. Хоча такі праці мають прикладний характер, їхня тематика підтверджує загальну тенденцію. Сучасний веб-ресурс у сфері розваг дедалі частіше розглядається не просто як електронний каталог, а як інтелектуалізоване середовище добору та представлення контенту (див рис. 1.2) [8, 10, 11].

Рис. 1.2 Еволюція веб-ресурсів для перегляду фільмів: відінформаційного каталогу до персоналізованого сервісу

Слід також ураховувати, що веб-ресурси для перегляду медіаконтенту мають соціокультурне значення. Вони впливають на видимість окремих фільмів, популярність жанрів, формування глядацьких уподобань і навіть на транснаціональне поширення аудіовізуального продукту. У дослідженні Broocks і Studnicka показано, що у сфері video-on-demand важливими залишаються такі чинники, як мовна близькість, культурна спорідненість, локальні уподобання та якість контенту [3]. Це означає, що веб-ресурс для перегляду популярних фільмів працює не в нейтральному середовищі, а в системі культурних і поведінкових очікувань користувачів. Саме тому для успішного функціонування такого ресурсу необхідно поєднувати зручність навігації, якісну структуру даних, адаптивний інтерфейс та логіку представлення контенту, яка відповідає реальним сценаріям споживання [3, 4, 6].

Узагальнюючи викладене, можна стверджувати, що актуальність веб-ресурсів для перегляду медіа-контенту зумовлена сукупністю технологічних, поведінкових і соціокультурних факторів. По-перше, відеоконтент посідає провідне місце в структурі сучасного інтернет- і мобільного трафіку [1, 2]. По-друге, розважальна сфера дедалі більше орієнтується на on-demand модель доступу до інформації та контенту [3, 4]. По-третє, користувачі потребують інструментів, які зменшують інформаційне перевантаження, спрощують пошук і підвищують якість вибору [5, 8]. По-четверте, ефективність таких сервісів

безпосередньо залежить від юзабіліті, адаптивності, доступності та якості користувацького досвіду [6, 7, 9].

Розробка веб-застосунку для перегляду популярних фільмів є актуальним і практично значущим напрямом, що відповідає поточним тенденціям цифрового споживання медіа та вимогам сучасного користувача [1–11].

1.2 Огляд існуючих вебресурсів для перегляду медіа-контенту

Сучасні веб-ресурси для перегляду медіа-контенту становлять важливу складову цифрової розважальної інфраструктури, оскільки забезпечують користувачеві доступ не лише до самого контенту, а й до супровідної інформації, засобів пошуку, рейтингування, відбору та порівняння аудіовізуальних матеріалів. У сучасному цифровому середовищі такі ресурси виконують різні функції. Одні орієнтовані на безпосередній перегляд фільмів і серіалів, інші зосереджені на наданні довідкових відомостей, рецензій та рейтингових оцінок, а окремі сервіси виступають агрегаторами, які допомагають користувачеві визначити, де саме доступний той чи інший медіапродукт [1, 2, 3]. Унаслідок цього ринок веб-ресурсів для перегляду медіа-контенту є функціонально різнорідним, а його аналіз потребує врахування не лише технічних характеристик, а й особливостей користувацького сценарію взаємодії з платформою.

Для систематизації існуючих рішень доцільно виокремити декілька основних груп веб-ресурсів (див. рис. 1.3). До першої належать інформаційно-довідкові платформи, які надають розгорнуті відомості про фільми, серіали, акторів, рейтинги та відгуки користувачів. Прикладом такого ресурсу є IMDb, який поєднує інструменти пошуку, систему оцінювання, категоризацію контенту та механізми навігації за різними параметрами [1]. До другої групи можна віднести агрегатори доступності контенту, зокрема JustWatch, призначення яких полягає у визначенні сервісів, на яких користувач може переглянути певний фільм або серіал [2]. Окрему нішу займають соціально-орієнтовані ресурси

кінематографічного спрямування, наприклад Letterboxd, що поєднує елементи каталогу, рейтингової системи, списків перегляду та користувацьких рецензій [4]. Водночас Rotten Tomatoes виконує функцію агрегатора критичних і користувацьких оцінок, допомагаючи швидко оцінити загальне сприйняття фільму чи серіалу аудиторією та професійними оглядачами [5].

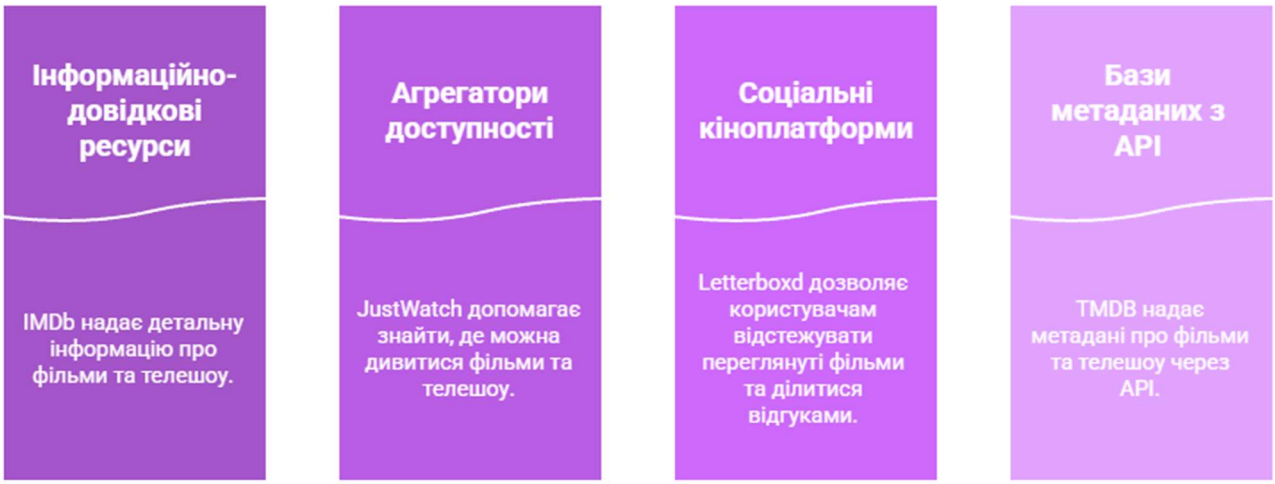

Рис. 1.3. Класифікація існуючих веб-ресурсів для перегляду медіа-контенту

Особливе значення для розробки власних веб-застосунків мають ресурси, що забезпечують структурований доступ до метаданих через програмні інтерфейси. У цьому контексті важливе місце посідає TMDB (The Movie Database), який виступає не лише базою даних про фільми, серіали, акторів і зображення, а й технологічною платформою для створення сторонніх сервісів на основі API [3]. Документація TMDB містить інструменти для пошуку, фільтрації, отримання детальної інформації про кінопродукцію та формування вибірок за різними параметрами, що робить цей ресурс зручним для побудови веб-застосунків, орієнтованих на перегляд популярних фільмів [3]. Саме тому під час огляду існуючих веб-ресурсів доцільно аналізувати їх не лише як готові платформи для кінцевого користувача, а і як приклади різних моделей організації доступу до медіа-контенту, інформації про нього та механізмів взаємодії з аудиторією.

1.2.1 Огляд інформаційно-довідкової платформи IMDb

Одним із найбільш відомих і функціонально насичених інформаційно-довідкових ресурсів у сфері кіноіндустрії є IMDb (Internet Movie Database). Платформа позиціонується як авторитетне джерело інформації про фільми, телесеріали та знаменитостей, а також як сервіс, що допомагає користувачам знаходити й обирати контент для перегляду [1]. Значущість IMDb підтверджується не лише її впізнаваністю, а й масштабом накопичених даних. За офіційною статистикою сервісу, станом на березень 2026 року база IMDb містила 28 120 305 назв, понад 9 117 655 користувацьких рецензій, 5 133 466 жанрових позначень і 941 195 записів у блоці parental guide. Такі показники дають підстави розглядати IMDb не просто як веб-сайт із картками фільмів, а як велику довідкову систему, що акумулює різнорідні відомості про аудіовізуальний контент.

Однією з ключових переваг IMDb є розвинена система пошуку та багатокритеріальної фільтрації. Офіційний розділ Advanced Search дозволяє здійснювати пошук не лише за назвою фільму, а й за типом контенту, датою релізу, користувацьким рейтингом, кількістю голосів, жанром, ключовими словами, мовою, країною виробництва, тривалістю, складом акторів, творчою групою, рівнем популярності та навіть наявністю нагород. Такий підхід є особливо важливим для користувачів, які не шукають конкретну назву, а прагнуть знайти фільм за певними ознаками, наприклад за жанром, роком випуску чи мінімальним рівнем рейтингу. У цьому сенсі IMDb реалізує не лише довідкову, а й навігаційну функцію, допомагаючи користувачеві звузити надто широкий інформаційний простір до релевантної вибірки [3].

Важливим елементом функціональності IMDb є структура картки окремого фільму (див. рис. 1.4). На прикладі сторінки конкретної стрічки можна побачити, що ресурс об’єднує в межах одного інтерфейсу базові метадані, візуальний контент і блоки користувацької взаємодії. Типова сторінка містить назву, рік випуску, тривалість, вікові обмеження, постер, короткий сюжетний

опис, перелік режисерів, сценаристів та основного акторського складу, рейтинг IMDb, показник популярності, відеоматеріали, фотогалерею, рецензії користувачів, критичні відгуки, метапоказники, а також розділи з деталями виробництва, країною, мовою, датами релізу, касовими зборами та технічними характеристиками.

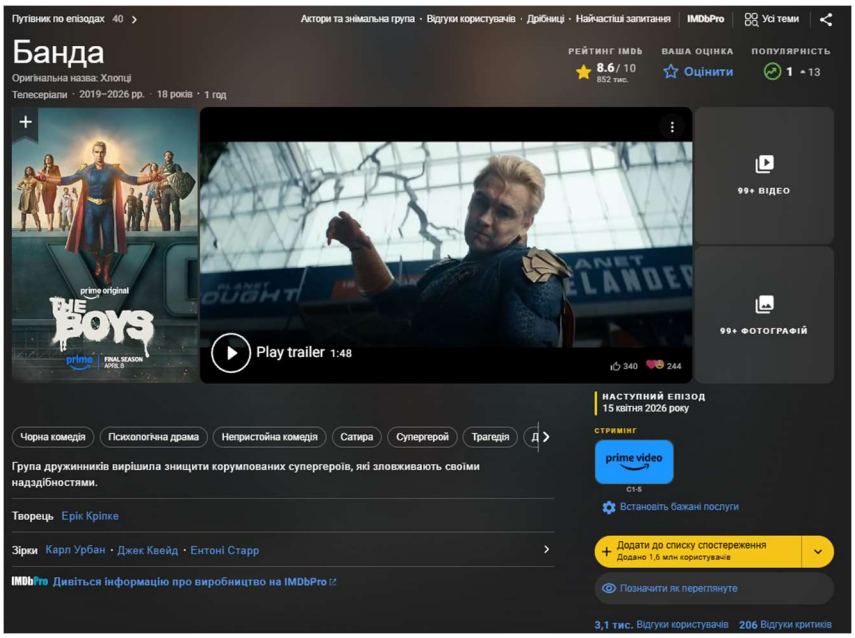

Рис. 1.4 Приклад картки фільму на платформі IMDb

Особливу роль у структурі IMDb відіграє рейтинговий механізм (див. пис. 1.5). Офіційна довідка сервісу вказує, що рейтинг на платформі базується на weighted average, тобто зваженому середньому значенні голосів користувачів. Крім того, при формуванні чартів, зокрема IMDb Top 250 Movies, враховуються лише голоси так званих regular IMDb voters, що дає змогу підвищити стабільність рейтингової системи та зменшити вплив випадкових або маніпулятивних оцінок. Для користувача це має практичне значення, оскільки рейтинг у такому випадку виступає не просто відображенням арифметичної середньої оцінки, а більш виваженим індикатором сприйняття фільму великою аудиторією. У межах веб-ресурсу для перегляду популярних фільмів саме наявність прозорої системи рейтингування є одним із ключових чинників довіри до контенту та зручності його відбору.

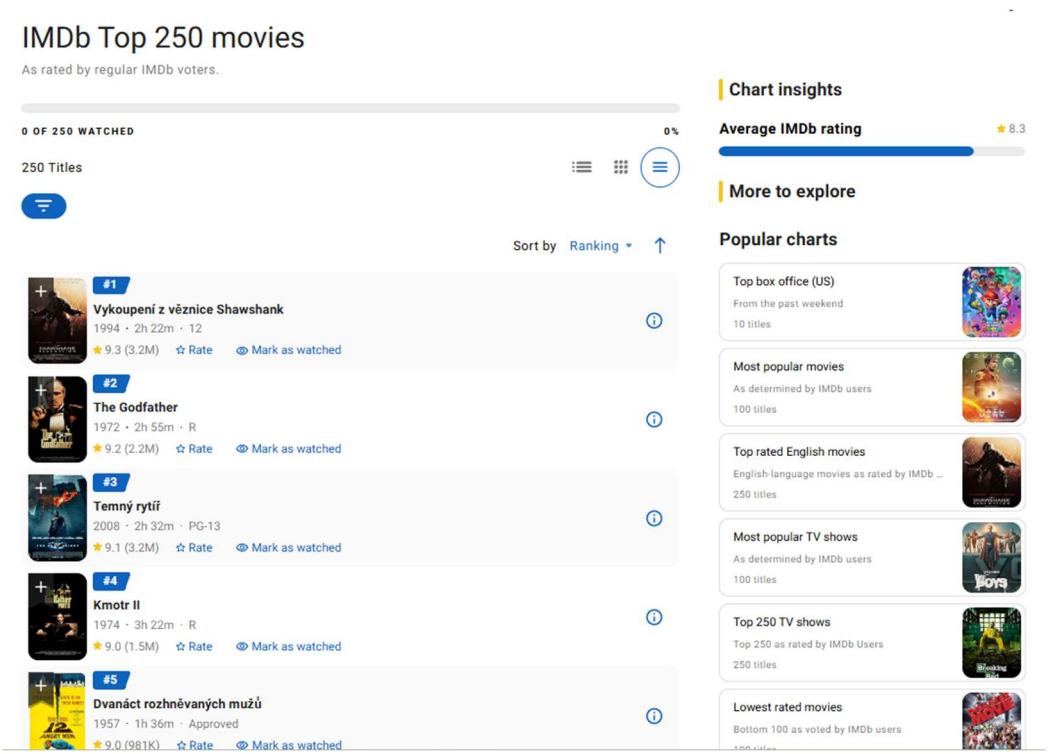

Рис. 1.5 Блок рейтингу IMDb

Не менш важливим компонентом IMDb є система користувацьких рецензій і списків. Згідно з офіційною документацією, користувацькі відгуки на сторінках фільмів відображаються за логікою Featured review order, що враховує низку параметрів і дає змогу показувати різні погляди на той самий фільм. Користувач також може змінювати порядок сортування відгуків за датою, оцінкою рецензії чи кількістю голосів. Додатково IMDb має інструмент Watchlist, призначений для збереження фільмів і серіалів, які користувач планує переглянути пізніше. Офіційно зазначено, що елементи у Watchlist можна сортувати за рейтингом IMDb, показником популярності або власним порядком.

Ще однією корисною функцією IMDb є блок What to Watch, який орієнтований на рекомендації та добірки контенту (див. рис. 1.6). У довідкових матеріалах сервісу вказано, що персоналізовані рекомендації формуються на основі оцінок користувача, його watchlist та зіставлення з поведінкою інших користувачів зі схожими вподобаннями. Це є важливим прикладом того, як інформаційно-довідковий ресурс поступово поєднує традиційну каталогізацію з елементами персоналізованого контентного відбору. Навіть якщо користувач не

використовує IMDb як стрімінгову платформу, сервіс усе одно допомагає відповісти на практичне запитання: “Що саме варто переглянути далі”.

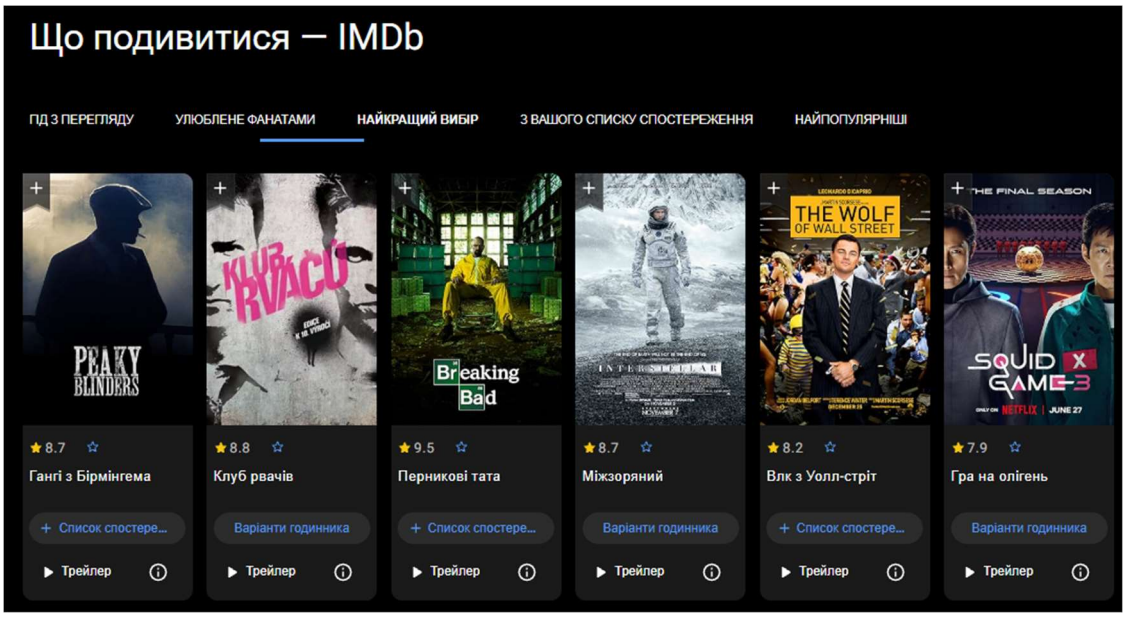

Рис. 1.6 Приклад рекомендаційного блоку What to Watch

Окремо слід звернути увагу на додаткові інформаційні модулі IMDb, зокрема Parental Guide. Цей розділ містить відомості про ступінь вираженості сцен сексуального характеру, насильства, ненормативної лексики, тем, пов’язаних із алкоголем або наркотиками, а також потенційно тривожних епізодів. Для частини аудиторії це має практичну цінність, оскільки дозволяє оцінити доречність перегляду певного фільму для сімейної чи дитячої аудиторії. Наявність такого модуля свідчить про те, що IMDb прагне охопити різні аспекти прийняття рішення щодо перегляду, а не обмежується лише загальним описом і рейтинговими показниками.

Узагальнюючи проведений аналіз, можна зробити висновок, що IMDb є потужною інформаційно-довідковою платформою, яка поєднує масштабну базу метаданих, гнучкий пошук, багаторівневу рейтингову систему, рецензії, списки та допоміжні інформаційні модулі. До її основних переваг належать великий обсяг даних, деталізовані картки фільмів, розвинуті механізми фільтрації та висока інформативність.

1.2.2 Огляд агрегатора доступності медіаконтенту JustWatch

Одним із найбільш показових прикладів агрегатора доступності медіаконтенту є JustWatch (див. рис. 1.7). На офіційній сторінці сервіс визначається як безкоштовний streaming guide, призначений для того, щоб допомагати користувачеві знаходити, де саме можна легально переглянути фільм або серіал онлайн. На відміну від інформаційно-довідкових платформ, які концентруються на метаданих, рейтингах або рецензіях, JustWatch орієнтується насамперед на вирішення практичного завдання. Швидко встановити, на якому сервісі доступний потрібний контент, у якому форматі він пропонується та на яких умовах доступу. Це дає підстави розглядати JustWatch як навігаційний інструмент у фрагментованому середовищі сучасних стрімінгових платформ.

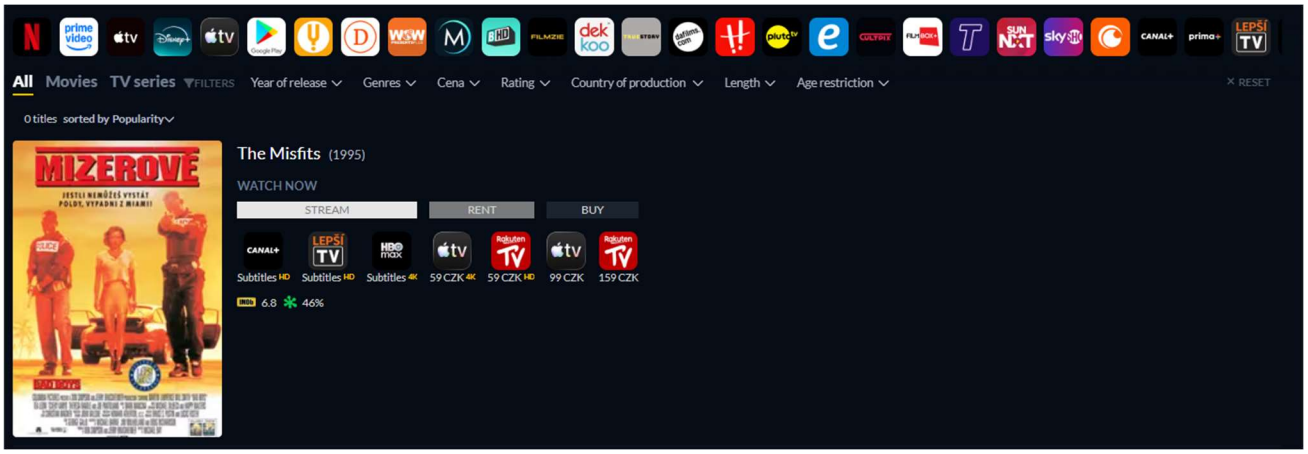

Рис. 1.7. Сторінка пошуку JustWatch

Функціональна цінність JustWatch полягає у зведенні інформації з багатьох провайдерів в одному інтерфейсі. Офіційна сторінка сервісу підкреслює, що користувач може виконувати пошук одразу по багатьох платформах, переглядати нові, популярні та майбутні релізи, а також бачити, де саме доступний контент для передплати, оренди, купівлі або безкоштовного перегляду з рекламою. Додатковою перевагою є можливість використання фільтрів за роком випуску, рейтингом IMDb, віковим обмеженням та іншими

параметрами. Для користувача це зменшує кількість зайвих переходів між платформами та спрощує прийняття рішення щодо перегляду.

Окремої уваги заслуговує система персоналізації. В офіційній довідці JustWatch зазначається, що користувач може обрати власні streaming providers, а сервіс формуватиме видачу відповідно до доступних йому платформ. Також підтримуються watchlist та custom lists, які дозволяють зберігати фільми й серіали для подальшого перегляду та фільтрувати їх за доступністю на вибраних сервісах. Згідно з глобальною прессторінкою компанії, JustWatch працює більш ніж у 140 країнах і використовує дані про інтерес користувачів для побудови streaming charts та аналітики ринку. Платформа виконує не лише прикладну навігаційну функцію, а й виступає інструментом моніторингу динаміки популярності контенту.

До основних переваг JustWatch належать централізований пошук по багатьох стрімінгових сервісах, зручна система фільтрації, актуальні рейтинги популярності та орієнтація на реальний користувацький сценарій “де подивитися”. Водночас його слабкою стороною є обмежена глибина довідкової інформації порівняно з IMDb або TMDB, оскільки ключова цінність платформи зосереджена не на повному описі фільму, а на виявленні його доступності.

1.2.3 Огляд соціально орієнтованої кіноплатформи Letterboxd

Платформа Letterboxd займає окреме місце серед веб-ресурсів для роботи з медіаконтентом, оскільки поєднує елементи каталогу фільмів, соціальної мережі та персонального щоденника переглядів (див. рис. 1.8). В офіційному описі сервіс визначається як глобальна соціальна мережа для обговорення та відкриття фільмів, де користувач може фіксувати переглянуті стрічки, оцінювати їх, писати рецензії, формувати списки, вести watchlist та підписуватися на інших учасників спільноти. Такий підхід відрізняє Letterboxd від класичних довідкових ресурсів, оскільки в центрі платформи перебуває не лише сам фільм, а й соціальна активність навколо нього.

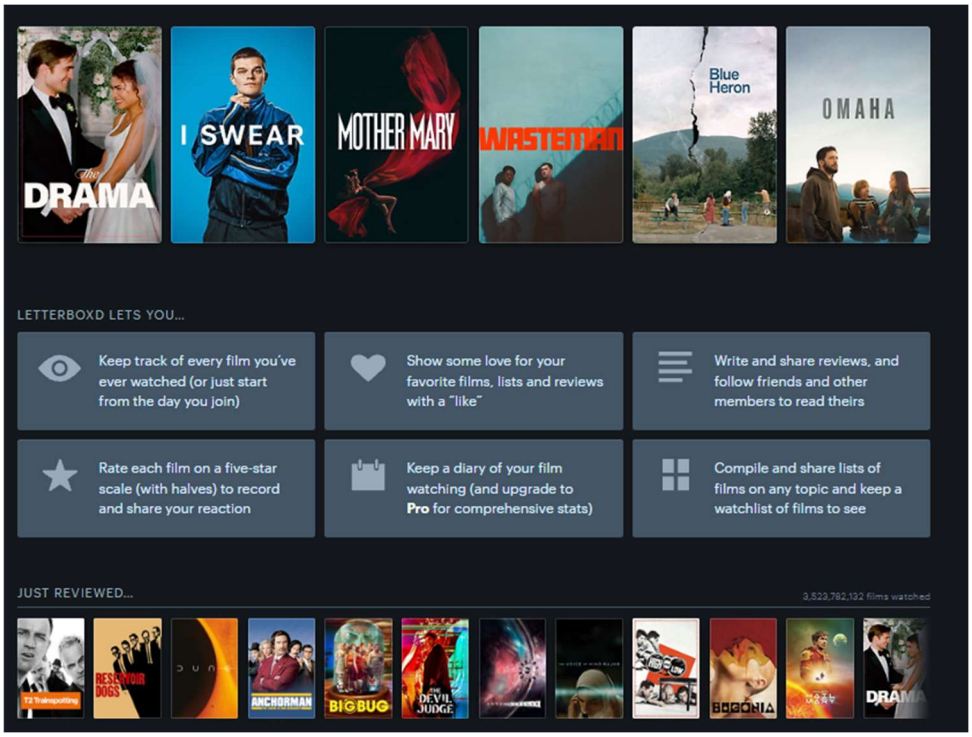

Рис. 1.8 Головна сторінка платформи Letterboxd

Однією з основних функцій Letterboxd є механізм фіксації особистої історії переглядів. На сторінках сервісу видно, що користувач може сортувати записи за датою, популярністю, датою релізу, власною оцінкою, середньою оцінкою спільноти та іншими параметрами.

Ще однією важливою особливістю платформи є її соціальна природа. Користувач може публікувати рецензії, ставити оцінки, додавати теги, відстежувати активність друзів та орієнтуватися на громадське обговорення фільмів. Це створює умови для неформального рекомендаційного середовища, де вибір стрічки часто здійснюється не лише на підставі формального рейтингу, а й через участь у спільноті. Крім того, платні підписки розширюють функціональність сервісу, додають персоналізовану статистику, фільтрацію за улюбленими стрімінговими сервісами, сповіщення про появу фільмів на платформах та інші інструменти. Letterboxd поступово інтегрує не лише соціальні, а й практичні механізми підтримки вибору контенту.

До основних переваг Letterboxd належать висока залученість користувачів, зручні інструменти формування персональних списків, можливість ведення

щоденника переглядів і сильна соціальна складова. Недоліком можна вважати те, що платформа менш орієнтована на формально структуровані метадані, ніж IMDb або TMDB, а частина розширених можливостей прив'язана до платної підписки. Letterboxd є важливим прикладом того, що веб-ресурс для перегляду популярних фільмів може бути не лише каталогом, а й простором для взаємодії, персонального обліку та залучення аудиторії через списки, оцінки й рецензії.

1.2.4 Аналіз рейтингово-рецензійної платформи Rotten Tomatoes

Сервіс Rotten Tomatoes є одним із найвідоміших веб-ресурсів, що спеціалізується на агрегуванні критичних і користувацьких оцінок фільмів та серіалів. На офіційній сторінці About зазначено, що платформа виступає одним із найбільш упізнаваних джерел рецензій і рекомендацій у сфері розваг та поєднує огляди професійних критиків із реакціями звичайних глядачів. На відміну від IMDb, де основний акцент зроблено на рейтингу користувачів та широких довідкових даних, Rotten Tomatoes фокусується на швидкій інтерпретації того, як конкретний фільм або серіал був сприйнятий критиками та аудиторією.

Ключовими елементами системи є Tomatometer і Popcornmeter. Офіційно Tomatometer визначається як відсоток позитивних рецензій професійних критиків на певний фільм або серіал, а Popcornmeter відображає частку позитивних оцінок з боку аудиторії. Якщо щонайменше 60% професійних рецензій є позитивними, твір позначається як Fresh, а за нижчого показника - як Rotten. Окремо застосовується позначка Certified Fresh, яка присвоюється високооціненим роботам за умови виконання додаткових критеріїв щодо кількості рецензій . У 2024 році сервіс також запровадив відзнаку Verified Hot, яка підкреслює найвищі показники аудиторного схвалення на основі verified ratings. У сукупності це формує систему швидкого візуального орієнтування, зручну для користувача, який хоче оперативно оцінити загальне сприйняття фільму. Платформа поєднує рейтингово-рецензійну функцію з елементами

контентної навігації. Для користувача це важливо, оскільки сервіс не лише показує, наскільки позитивно був сприйнятий фільм, а й допомагає знаходити контент, який відповідає його інтересам або очікуванням щодо якості.

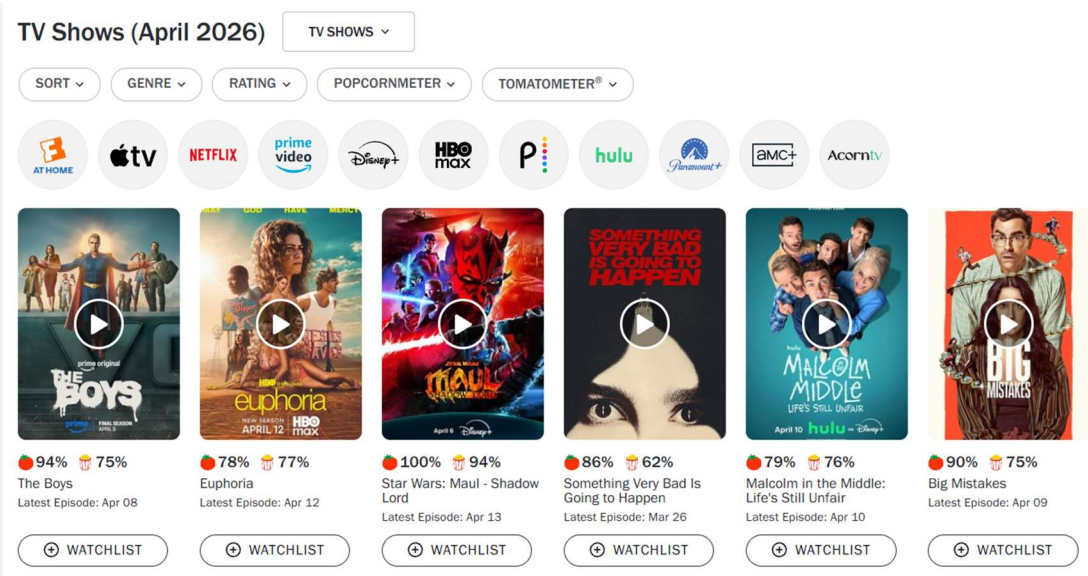

Рис. 1.9 Приклад сторінки Rotten Tomatoes із фільтрами

Перевагою Rotten Tomatoes є простота інтерпретації оцінок та виразна візуалізація результатів критичного й аудиторного сприйняття. Водночас слабкою стороною є певна редукція складної критичної інформації до одного відсоткового показника, що іноді спрощує багатовимірне враження про фільм. Крім того, сервіс не є глибокою довідковою базою в тому сенсі, в якому нею є IMDb або TMDB. Rotten Tomatoes є корисним прикладом того, як у веб-застосунку можна реалізувати зрозумілу рейтингову візуалізацію та швидкі механізми оцінювання популярності або якості контенту.

1.2.5 Аналіз бази кінематографічних метаданих TMDB

Платформа TMDB (The Movie Database) посідає окреме місце серед веб-ресурсів для роботи з медіаконтентом, оскільки поєднує функції довідкової кінобази, візуального каталогу та технологічної платформи доступу до

структурованих даних. На відміну від сервісів, основною метою яких є стрімінговий перегляд або публікація рецензій, TMDB зосереджується передусім на накопиченні, упорядкуванні та поширенні метаданих про фільми, серіали, сезони, епізоди, акторів, творчі групи та зображення. В офіційному описі сервіс характеризується як спільнотно наповнювана міжнародна база даних, що охоплює широкий спектр кіно- і телевізійного контенту та використовується як кінцевими користувачами, так і сторонніми цифровими сервісами.

У цьому аспекті TMDB відрізняється від IMDb, де акцент значною мірою зміщений у бік рейтингової та довідкової взаємодії, а також від Letterboxd і Rotten Tomatoes, де вагому роль відіграють соціальна активність, рецензії та оцінки спільноти.

Однією з основних особливостей TMDB є високий рівень структурованості інформації. У межах картки окремого фільму (див. рис. 1.10) користувач отримує доступ до базових метаданих, зокрема назви, оригінальної назви, короткого опису, жанрів, дати релізу, тривалості, постера, фонового зображення, рейтингу, популярності, виробничих компаній, країн виробництва, мов, акторського складу та додаткових пов’язаних матеріалів. Такий принцип подання даних робить платформу зручною для систематизації великого масиву контенту. На відміну від деяких інших ресурсів, де окремі елементи можуть бути винесені в різні логічні блоки або сторінки, TMDB прагне забезпечити компактне, візуально впорядковане та швидко читане представлення інформації про медіаоб’єкт.

Сильною стороною TMDB є також гнучка система пошуку та відбору контенту. У документації зазначено, що платформа підтримує кілька основних способів отримання даних: search, discover і find. Якщо пошук (search) орієнтований на знаходження об’єкта за назвою чи ключовим словом, то механізм discover дозволяє формувати складні вибірки за широким набором параметрів: жанрами, датами релізу, рейтингами, мовами, сертифікацією, популярністю та іншими ознаками. Саме це відрізняє TMDB від багатьох довідкових ресурсів, де користувачеві доступний переважно класичний текстовий пошук або базова фільтрація. У контексті аналізу веб-ресурсів для

перегляду медіаконтенту така функціональність свідчить про високу придатність TMDB до роботи з великими каталогами та масивами фільмів.

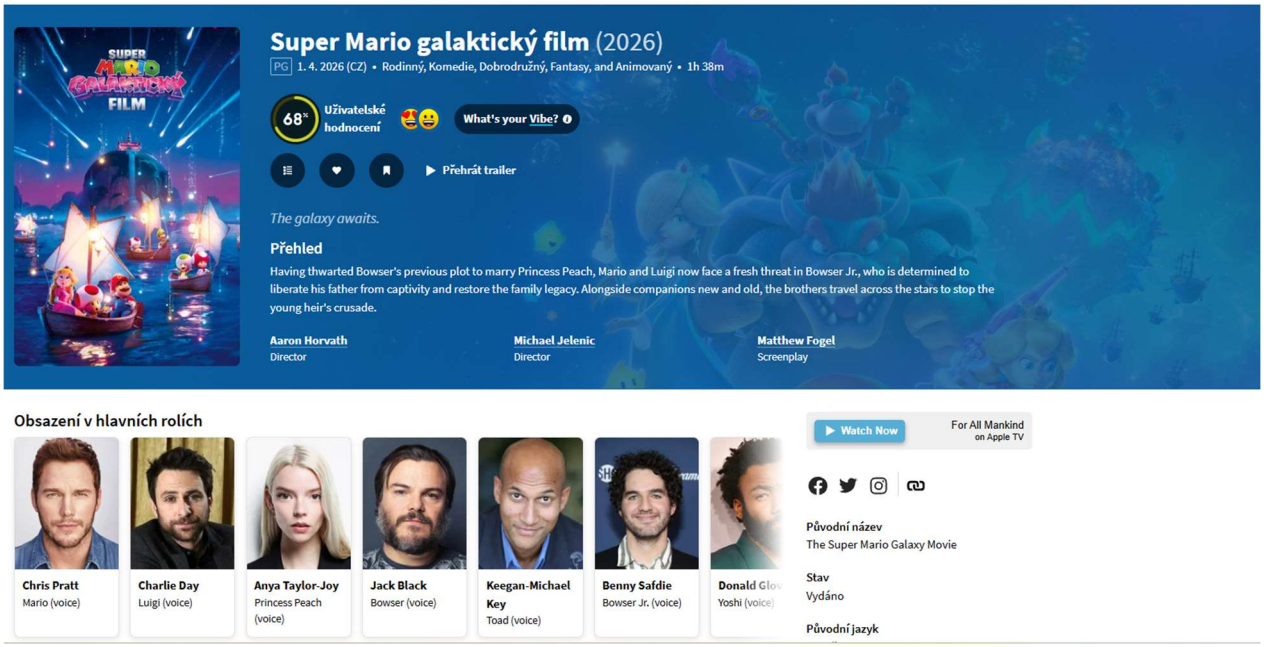

Рис. 1.10 Приклад картки фільму на платформі TMDB з основними метаданими

Ще однією суттєвою рисою платформи є орієнтація на показники популярності та актуальності контенту (див. рис. 1.11). В офіційній документації пояснюється, що popularity у TMDB формується на основі кількох факторів, серед яких активність користувачів, взаємодія з контентом та інші поведінкові сигнали. Окремо функціонує механізм trending, який дозволяє визначати контент, що привертає підвищену увагу в певний часовий проміжок. Порівняно з платформами, де акцент робиться на фіксованому рейтингу або критичних рецензіях, TMDB у цьому аспекті надає більш динамічне бачення актуальності фільмів і серіалів.

Важливою характеристикою TMDB є міжнародна спрямованість платформи. Сервіс підтримує параметри language та region, що дозволяє адаптувати представлення даних до конкретної мовної та регіональної аудиторії. Це є суттєвою перевагою порівняно з ресурсами, які мають більш виражену

орієнтацію на окремі національні ринки або обмежену локалізацію. Наявність мультимовної підтримки, міжнародних релізів, різних назв одного й того самого фільму та регіональних налаштувань розширює сферу застосування TMDB і робить платформу зручною для користувачів з різних країн.

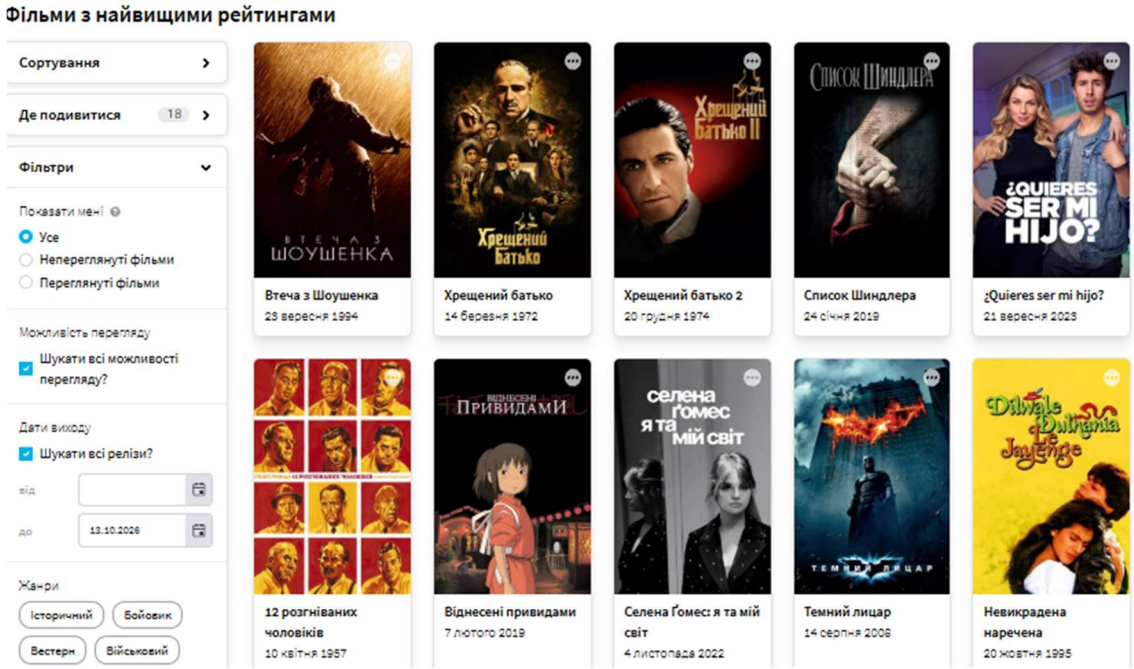

Рис. 1.11 Приклад сторінки з трендовими фільмами на TMDB

Окремо слід підкреслити технологічний характер платформи. TMDB має відкриту та добре структуровану документацію, у якій описані основні endpoint’и, параметри запитів, формат відповідей та правила інтеграції. Це суттєво відрізняє платформу від Letterboxd чи Rotten Tomatoes, які орієнтовані насамперед на кінцевого користувача, а не на зовнішнє використання даних. У випадку TMDB веб-інтерфейс і програмний інтерфейс фактично доповнюють один одного. Сайт дає змогу переглядати контент у зручному візуальному вигляді, а API - працювати з ним на рівні структурованих даних. Саме тому TMDB можна розглядати не лише як інформаційний ресурс, а і як елемент цифрової інфраструктури медіаконтенту.

Перевагами TMDB є великий обсяг структурованих метаданих, зручна візуальна організація карток контенту, підтримка пошуку й

багатокритеріального відбору, міжнародна локалізація та наявність відкритої документації. Водночас платформа має і певні обмеження. Насамперед вона не є стрімінговим сервісом у класичному розумінні та не забезпечує повноцінного перегляду фільмів безпосередньо в межах платформи. Крім того, у порівнянні з IMDb або Rotten Tomatoes тут менше уваги приділяється розвиненій системі користувацьких рецензій і публічному критичному дискурсу. Отже, TMDB сильніша як база метаданих і технічний ресурс, ніж як соціальна або рецензійна платформа.

Узагальнюючи, можна зазначити, що TMDB є одним із найбільш функціонально збалансованих веб-ресурсів у сфері кінематографічних даних. Специфіка цієї платформи полягає в поєднанні довідкової, каталогізаційної та технологічної функцій. При цьому, у порівнянні з IMDb платформа має більш виражену орієнтацію на структуровані метадані та інтеграційні можливості, а у порівнянні з JustWatch - не зосереджується на доступності контенту на зовнішніх сервісах. При цьому, у порівнянні з Letterboxd і Rotten Tomatoes - TMDB меншою мірою залежить від соціальних оцінок і рецензійної активності. Саме це дозволяє розглядати TMDB як самостійну й важливу модель веб-ресурсу для роботи з медіаконтентом, орієнтовану на повноту даних, зручність структурування та широкі можливості використання.

1.3 Вибір технологічного стеку та архітектури вебзастосунку

Для веб-застосунку, який використовує TMDB як зовнішню базу кінематографічних даних, доцільною є клієнт-серверна архітектура. У межах такого підходу клієнтська частина відповідає за відображення інтерфейсу, взаємодію з користувачем та подання результатів пошуку, тоді як серверна частина або проміжний рівень забезпечує обмін даними із зовнішнім API, обробку запитів та контроль доступу до службових параметрів (див. рис. 1.12). Такий розподіл дозволяє зробити систему більш керованою, безпечнішою та зручнішою для подальшого розвитку.

Рис. 1.12 Узагальнена схема архітектури веб-застосунку з використанням TMDB як зовнішньої бази даних

Для інтерфейсної частини обґрунтованим є використання архітектури SPA (Single Page Application). У застосунках, де користувач часто взаємодіє зі списками фільмів, картками, фільтрами та результатами пошуку, SPA забезпечує динамічне оновлення сторінки без повного перезавантаження документа. Це позитивно впливає на сприйняття швидкодії та дозволяє організувати більш цілісний користувацький сценарій. Крім того, така архітектура добре узгоджується з REST-підходом до обміну даними, який передбачає уніфіковану взаємодію між клієнтом і джерелом ресурсів.

Важливими принципами обраної архітектури є модульність і компонентний підхід. Вони дають змогу поділити інтерфейс на окремі логічні блоки, зменшити складність системи та полегшити підтримку. У поєднанні з клієнт-серверною моделлю це створює основу для гнучкої організації веб-

застосунку, який працює з зовнішньою базою даних і потребує швидкого та стабільного оновлення вмісту.

1.3.1 Вибір технологій клієнтської частини

Клієнтську частину веб-застосунку доцільно реалізовувати на основі HTML, CSS, JavaScript і React. HTML використовується для семантичної побудови структури сторінок, CSS забезпечує візуальне оформлення та адаптацію інтерфейсу до різних типів екранів, а JavaScript реалізує логіку взаємодії з користувачем і динамічне оновлення вмісту. Сукупність цих технологій становить базу сучасної веб-розробки та забезпечує сумісність із браузерним середовищем.

Використання React є доцільним завдяки його компонентній природі. React дозволяє будувати інтерфейс як дерево незалежних елементів, які можна повторно використовувати, поєднувати та оновлювати залежно від зміни стану. Для веб-застосунку, що відображає списки популярних фільмів, результати пошуку та сторінки з описом контенту, такий підхід є особливо зручним, оскільки дає можливість підтримувати узгодженість інтерфейсу та спрощує подальший супровід системи.

1.3.2 Використання зовнішнього API та організація обміну даними

Для веб-застосунку з переглядом популярних фільмів доцільно використовувати TMDB як зовнішню базу кінематографічних даних. Офіційна платформа TMDB позиціонується як велика міжнародна база метаданих про фільми, телешоу, персоналії та зображення, яка використовується розробниками та цифровими сервісами в багатьох країнах світу. Такий підхід дозволяє отримувати актуальну інформацію без потреби створювати та вручну наповнювати власну повномасштабну кінобазу на початковому етапі розробки.

Обмін даними доцільно організовувати через REST API з використанням протоколу HTTP та формату JSON. HTTP є стандартною основою мережевої взаємодії у веб-середовищі, а JSON виступає легким текстовим форматом передавання структурованої інформації, зручним для обробки в JavaScript. У випадку TMDB через API можна отримувати списки популярних фільмів, результати пошуку, жанри, рейтинги, зображення та інші метадані. Це дає змогу забезпечити динамічне наповнення інтерфейсу та роботу з актуальним контентом.

Оскільки обмін даними із зовнішньою базою відбувається в реальному часі, важливим є використання асинхронних запитів. Такий підхід дозволяє завантажувати потрібну інформацію без блокування інтерфейсу, що є особливо важливим для пошуку, фільтрації та перегляду карток фільмів. Використання TMDB API у поєднанні з REST, HTTP і JSON є логічним способом організації доступу до даних у веб-застосунку цього типу.

1.3.3 Переваги обраного стеку для реалізації вебзастосунку

Обраний технологічний стек має низку переваг для реалізації веб-застосунку, що використовує TMDB як зовнішню базу даних. Поєднання HTML, CSS, JavaScript і React забезпечує створення сучасного інтерактивного інтерфейсу, придатного для роботи з динамічним контентом. SPA-архітектура сприяє швидкому оновленню сторінки, а компонентний підхід полегшує повторне використання елементів інтерфейсу та підтримку системи.

Використання TMDB як зовнішнього джерела даних також має практичні переваги. По-перше, це забезпечує доступ до великого масиву актуальної інформації про фільми без потреби створення власної повної бази кінематографічних даних. По-друге, це зменшує трудомісткість початкового етапу розробки. По-третє, такий підхід дозволяє зосередитися на реалізації інтерфейсу, логіки пошуку й зручності користування, спираючись на вже існуючу структуровану базу даних.

Обраний стек є доцільним завдяки поєднанню адаптивності, масштабованості, швидкості розробки, зручності підтримки та ефективної роботи із зовнішньою базою кінематографічних даних. Саме ці властивості роблять його придатним для реалізації веб-застосунку, орієнтованого на перегляд популярних фільмів.

Висновки до розділу 1

Веб-платформи для перегляду популярних фільмів виконують не лише інформаційну функцію, а й забезпечують пошук, відбір, порівняння та навігацію в умовах надлишку контенту. Актуальність таких ресурсів визначається зростанням ролі відеоконтенту в цифровому середовищі та високими вимогами користувачів до швидкодії, адаптивності та зручності інтерфейсу. Сучасні платформи для перегляду медіа-контенту мають різну функціональну спрямованість. Кожен ресурс має власні переваги й обмеження, а сучасний ринок веб-ресурсів для перегляду медіа-контенту характеризується функціональною спеціалізацією, зокрема TMDB виступає масштабною базою кінематографічних метаданих із розвиненими можливостями доступу до них через API.

При виборі технологічного стеку та архітектури веб-ресурсу доцільно зробити вибір на користь клієнт-серверної моделі, SPA-підходу, компонентної організації інтерфейсу та сучасних веб-технологій HTML, CSS, JavaScript і React. При цьому, икористання TMDB як зовнішньої бази кінематографічних даних є раціональним підходом, оскільки дає змогу працювати з актуальними структурованими даними без потреби створення власної повномасштабної бази на початковому етапі розробки.

Розробка веб-застосунку для перегляду популярних фільмів є актуальним і обґрунтованим напрямом, а використання сучасного технологічного стеку разом із зовнішньою базою даних TMDB створює належне підґрунтя для подальшого проєктування, реалізації та аналізу функціонування системи.

РОЗДІЛ 2
ПРОЄКТУВАННЯ ТА РОЗРОБКА ВЕБЗАСТОСУНКУ ДЛЯ ПЕРЕГЛЯДУ ПОПУЛЯРНИХ ФІЛЬМІВ

2.1 Проєктування інформаційної моделі та бази даних вебзастосунку

При проєктуванні вебзастосунку для перегляду популярних фільмів спочатку доцільно визначити логічну та фізичну моделі системи, а також основні сценарії взаємодії користувача з вебресурсом. Застосунок повинен поєднувати зовнішнє джерело даних про фільми та локальне збереження персоналізованих даних користувача. Відповідно до теми роботи основним джерелом інформації про фільми було обрано сервіс TMDB API, що забезпечує отримання актуальних даних без необхідності формування великої локальної бази кінематографічної інформації.

З позиції функціонального призначення вебзастосунку було визначено основні прецеденти його використання. До них належать перегляд популярних фільмів, пошук фільмів за назвою, фільтрація за жанром, перегляд детальної інформації про фільм, реєстрація, вхід до системи, перегляд профілю користувача, додавання фільму до обраного, перегляд історії пошуку та вихід із системи. Відповідні дії виконує користувач, який є основним актором системи. Діаграму прецедентів, що відображає функціональні можливості вебзастосунку, наведено на рисунку 2.1.

На логічному рівні інформаційна модель системи включає сутності `User`, `Favorite`, `SearchHistory`, `Movie` та `Genre`. Центральною прикладною сутністю є фільм, оскільки саме навколо неї будується основний функціонал застосунку. Сутність `User` використовується для представлення зареєстрованого користувача, а сутності `Favorite` і `SearchHistory` забезпечують реалізацію персоналізованих функцій. Сутність `Genre` застосовується для класифікації фільмів та реалізації жанрової фільтрації. Діаграму класів, що відображає зв’язки між основними сутностями системи, наведено на рисунку 2.2.

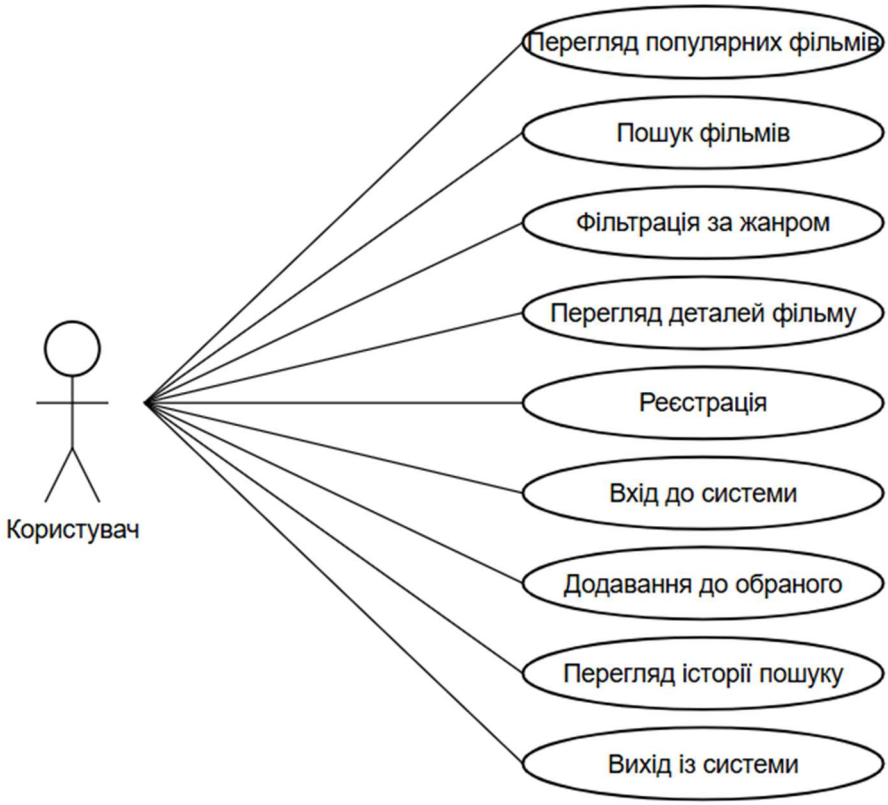

Рис. 2.1. Діаграма прецедентів вебзастосунку.

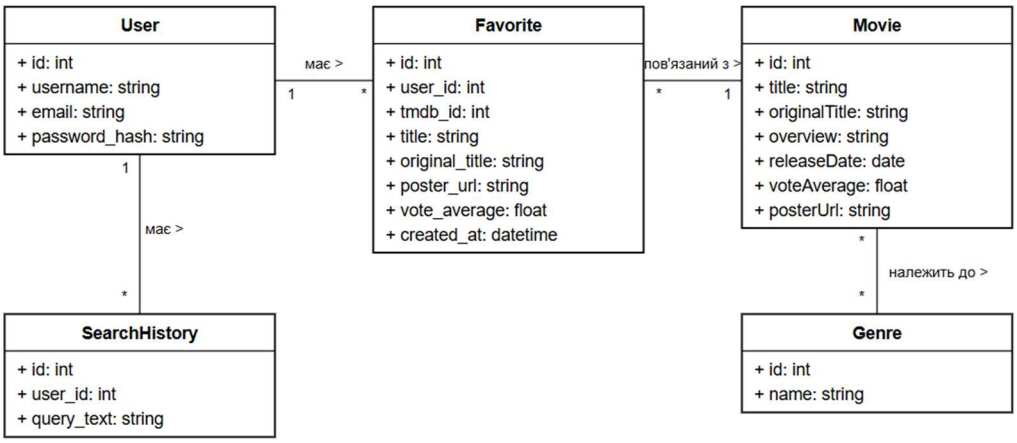

Рис. 2.2. Діаграма класів вебзастосунку

Окрему увагу було приділено моделюванню послідовності дій під час виконання окремого прецеденту. Як базовий сценарій доцільно розглядати пошук фільму або додавання фільму до обраного. У межах цього процесу користувач формує запит через клієнтську частину, після чого серверна частина обробляє отримані дані, за потреби звертається до TMDB API та повертає клієнту

підготовлений результат. Якщо користувач авторизований, додатково виконується взаємодія з локальною базою даних. Це дозволяє наочно показати розподіл функцій між інтерфейсом, сервером, зовнішнім API та локальним сховищем. Діаграму послідовності одного з базових прецедентів наведено на рисунку 2.3.

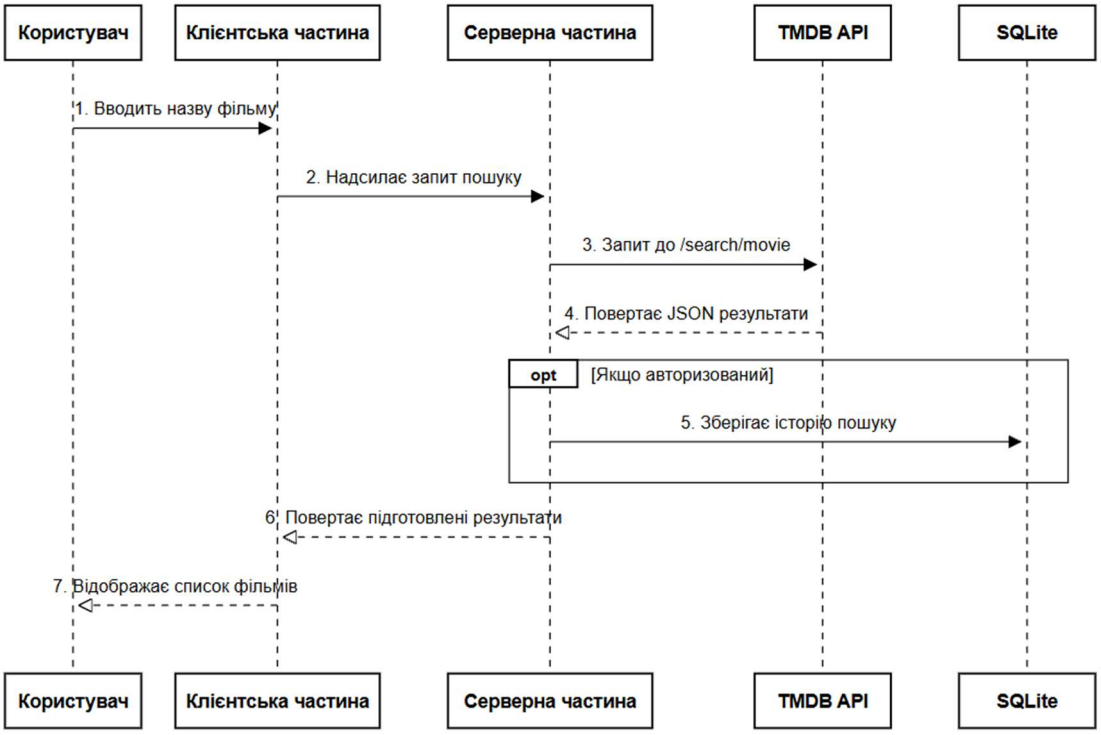

Рис. 2.3. Діаграма послідовності прецеденту

На фізичному рівні структура локальної бази даних реалізована за допомогою SQLite. Такий вибір зумовлений простотою налаштування, відсутністю потреби у виділеному сервері бази даних та достатньою функціональністю. У межах проєкту локально зберігаються не всі відомості про фільми, а лише персоналізовані дані користувача. Для цього в системі реалізовано таблиці `users`, `favorites` та `search_history`.

Таблиця `users` призначена для збереження облікових даних зареєстрованих користувачів. У ній зберігаються ідентифікатор користувача, ім’я користувача, електронна адреса, хеш пароля та дата реєстрації. Таблиця `favorites` використовується для збереження фільмів, доданих користувачем до

обраного. У ній містяться ідентифікатор користувача, ідентифікатор фільму в системі TMDB, назва, опис, дата виходу, посилання на постер, посилання на фонове зображення, рейтинг і дата додавання. Таблиця `search_history` призначена для фіксації тексту пошукових запитів користувача та часу їх виконання. ER-діаграму локальної бази даних наведено на рис. 2.4.

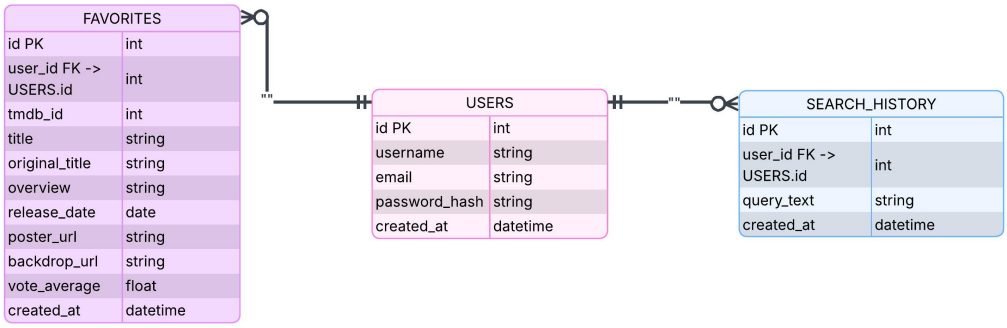

Рис. 2.4. ER-діаграма локальної бази даних з таблицями `users`, `favorites` та `search_history`

Між локальними таблицями існують зв’язки типу “один до багатьох”. Один користувач може мати багато записів у таблиці `favorites` і багато записів у таблиці `search_history`. Зовнішні сутності `Movie` та `Genre` не зберігаються в локальній базі повністю, але активно використовуються у прикладній логіці застосунку під час обробки запитів.

Для повнішого відображення технічної архітектури системи було також побудовано діаграму компонентів. У ній відображаються користувач, браузер, клієнтська частина, серверна частина, локальна база даних SQLite та зовнішній сервіс TMDB API. Діаграма компонентів дає змогу наочно показати фізичну структуру вебресурсу та взаємодію його основних складових. Відповідну діаграму наведено на рисунку 2.5.

На етапі проєктування було визначено як логічну, так і фізичну моделі вебзастосунку.

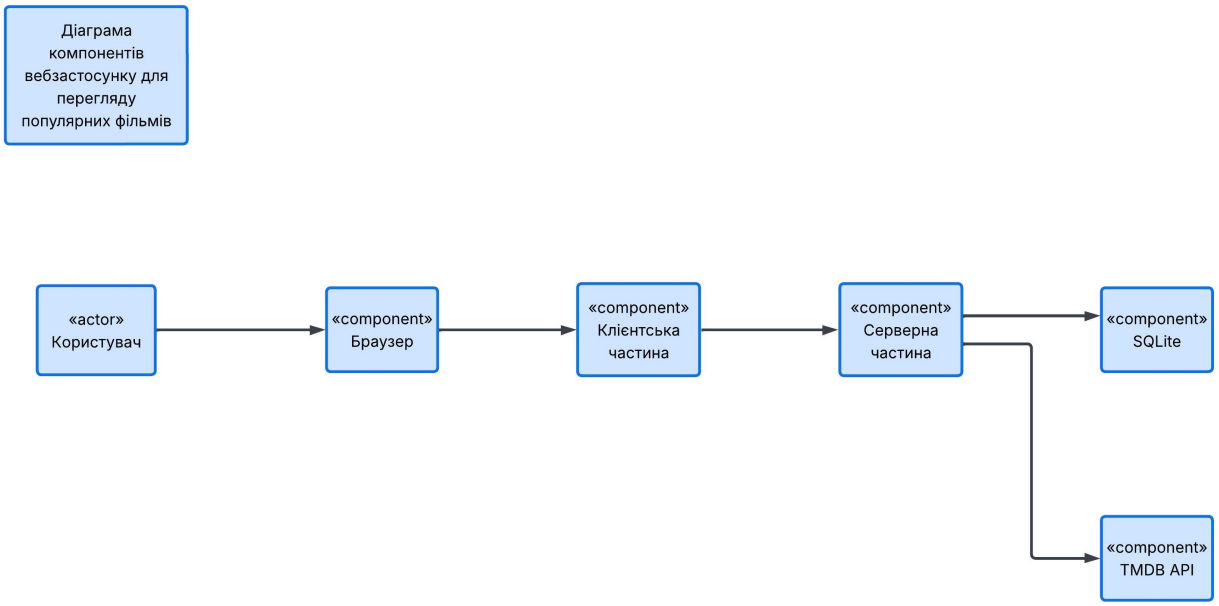

Рис. 2.5. Діаграма компонентів вебзастосунку

Використання UML-діаграм та ER-діаграми дозволило відобразити функціональні можливості системи, основні сутності, зв’язки між ними, послідовність виконання окремих прецедентів і загальну архітектуру вебресурсу.

2.2 Опис використовуваних підходів до збору та обробки даних

У розробленому вебзастосунку основним джерелом інформації про фільми є сервіс TMDB API. За його допомогою отримуються дані про популярні фільми, результати пошуку, жанри, детальна інформація про окремий фільм, а також перелік схожих фільмів. Використання зовнішнього API забезпечує актуальність відомостей та усуває потребу у самостійному наповненні локальної бази великим обсягом кінематографічної інформації.

У застосунку реалізовано клієнт-серверний підхід до обробки даних. Користувач взаємодіє з інтерфейсом у браузері, після чого клієнтська частина надсилає запити до серверної частини застосунку. Сервер звертається до TMDB API, отримує відповідь у форматі JSON, виконує необхідну обробку та повертає

клієнту підготовлені результати. У результаті це дозволяє приховати ключ доступу до API, централізувати логіку обробки даних та забезпечити єдину структуру відповідей для клієнтської частини.

Для формування списку популярних фільмів використовується метод /movie/popular. Якщо користувач обирає фільтрацію за жанром, застосовується метод /discover/movie, який дозволяє отримати перелік фільмів відповідно до заданого жанру. Для пошуку фільмів за назвою використовується метод /search/movie. Перелік жанрів завантажується через метод /genre/movie/list. Для сторінки детальної інформації застосовується метод /movie/{movie_id}, а для блоку рекомендацій на сторінці фільму додатково використовується метод /movie/{movie_id}/similar. Основні методи TMDB API, використані у вебзастосунку, наведено в таблиці 2.1.

Таблиця 2.1

Основні методи TMDB API, використані у вебзастосунку

Метод TMDB API	Призначення	Основні параметри
/movie/popular	Отримання списку популярних фільмів	language, page, region
/discover/movie	Отримання фільмів за жанровим фільтром	language, page, with_genres, sort_by
/search/movie	Пошук фільмів за назвою	query, language, page, include_adult
/genre/movie/list	Отримання списку жанрів	language
/movie/{movie_id}	Отримання детальної інформації про фільм	movie_id, language
/movie/{movie_id}/similar	Отримання списку схожих фільмів	movie_id, language, page

Під час виконання запитів до TMDB API використовуються додаткові параметри. Зокрема, параметр language=uk-UA застосовується для отримання локалізованих назв і описів, параметр page використовується для організації пагінації, параметр include_adult=false виключає небажаний контент із результатів пошуку, а параметр with_genres використовується для фільтрації фільмів за жанром. Для сторінки детальної інформації також використовується параметр append_to_response=credits,videos, що дозволяє в одному запиті отримати додаткові дані про акторський склад та відеоматеріали.

Окрему увагу приділено попередній обробці отриманих даних. У відповідях TMDB API міститься значна кількість полів, однак для роботи клієнтської частини використовуються лише ті з них, що необхідні для відображення в інтерфейсі. Для короткого представлення фільму використовуються ідентифікатор, назва, оригінальна назва, короткий опис, дата виходу, рейтинг, постер, фонове зображення та ідентифікатори жанрів. Для сторінки детальної інформації додатково використовуються жанри, тривалість, режисер, акторський склад, трейлер, офіційний сайт фільму та список схожих фільмів.

Важливою частиною обробки є формування повних URL-адрес зображень. TMDB API повертає не повні посилання на постери та фонові зображення, а лише відносні шляхи до файлів. У серверній частині застосунку здійснюється побудова повних URL-адрес шляхом поєднання базової адреси зображень, відповідного розміру та отриманого шляху. Отже, клієнтська частина отримує готові посилання, які можуть безпосередньо використовуватись для відображення зображень.

На серверному рівні також реалізовано локальну обробку персоналізованих даних. Якщо авторизований користувач виконує пошук, текст пошукового запиту зберігається до таблиці search_history. Якщо користувач додає фільм до обраного, мінімально необхідний набір відомостей про фільм зберігається у таблиці favorites.

У застосунку передбачено базову валідацію вхідних даних та обробку помилок. Якщо користувач не вводить назву фільму під час пошуку, сервер повертає повідомлення про необхідність введення пошукового запиту. Якщо у запиті передано некоректний ідентифікатор фільму, формується відповідне повідомлення про помилку. Для доступу до функцій обраного та історії пошуку передбачена перевірка авторизації. У разі відсутності авторизації сервер повертає код 401. Схему взаємодії клієнтської частини, серверної частини, бази даних та TMDB API наведено на рисунку 2.6.

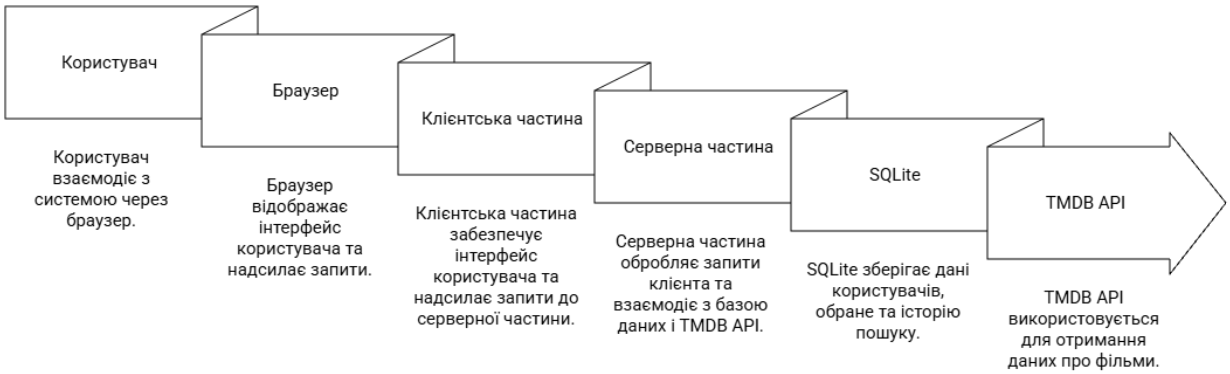

Рис. 2.6. Схема взаємодії користувача, клієнтської частини, серверної частини, локальної бази даних та TMDB API

У вебзастосунку використовується комбінований підхід до збору та обробки даних. Основні кінематографічні дані надходять із TMDB API, а персональні дані користувача зберігаються локально в базі даних SQLite. Серверна частина виконує роль проміжної ланки між зовнішнім API та клієнтською частиною, а також забезпечує валідацію, перетворення та локальне збереження даних.

2.3 Реалізація серверної та клієнтської частини вебзастосунку

Реалізація вебзастосунку здійснюється шляхом поділу системи на серверну та клієнтську частини. Серверна частина відповідає за роботу із

зовнішнім API, локальною базою даних, сесіями користувачів та серверними маршрутами, а клієнтська частина забезпечує відображення інтерфейсу та взаємодію користувача із системою.

Серверна частина реалізована на платформі Node.js з використанням фреймворку Express. Запуск застосунку здійснюється через файл `server.js`, у якому виконується ініціалізація бази даних та запуск HTTP-сервера. Основне налаштування застосунку реалізовано у файлі `src/app.js`, де здійснюється підключення JSON-обробки, налаштування сесій, реєстрація маршрутів для сторінок та API, а також обробка помилок.

Для підтримки авторизації використовується пакет `express-session`. Після успішного входу до системи ідентифікатор користувача зберігається у сесії, що дає можливість отримувати доступ до персоналізованих функцій без повторної авторизації при кожному запиті. Для збереження паролів використовується бібліотека `bcryptjs`, за допомогою якої здійснюється хешування паролів перед записом до бази даних.

У серверній частині реалізовано модуль роботи з локальною базою даних SQLite. У файлі `database.js` визначено створення таблиць `users`, `favorites` та `search_history`, а також функції для створення користувача, збереження фільмів в обраному, отримання списку обраного та ведення історії пошуку.

Окрему роль відіграє реалізація авторизації користувача. У файлі `auth.service.js` здійснюється перевірка коректності введених даних, перевірка унікальності email, хешування пароля та перевірка правильності пароля під час входу. У файлі `auth.controller.js` після успішної реєстрації або входу ідентифікатор користувача записується до сесії. Фрагмент коду, що відповідає за реєстрацію та вхід користувача, наведено на рисунку 2.7.

Для взаємодії із зовнішнім сервісом TMDB API реалізовано окремий модуль `tmdb.service.js`. У ньому забезпечується формування запитів до API, передавання параметрів, обробка відповідей та перетворення отриманих даних у формат, зручний для клієнтської частини. Саме в цьому модулі реалізовано отримання списку популярних фільмів, результатів пошуку, жанрів, детальної

інформації про фільм та списку схожих фільмів. Фрагмент коду, що відповідає за взаємодію з TMDB API, наведено на рисунку 2.8.

```
async function registerUser({ username, email, password }) {
  const normalizedUsername = String(username || "").trim();
  const normalizedEmail = String(email || "").trim().toLowerCase();
  const normalizedPassword = String(password || "");

  if (!normalizedUsername || !normalizedEmail || !normalizedPassword) {
    const error = new Error("Потрібно заповнити ім'я користувача, email та пароль.");
    error.status = 400;
    throw error;
  }

  if (normalizedUsername.length < 2) {
    const error = new Error("Ім'я користувача має містити щонайменше 2 символи.");
    error.status = 400;
    throw error;
  }

  if (!isValidEmail(normalizedEmail)) {
    const error = new Error("Введено некоректний email.");
    error.status = 400;
    throw error;
  }

  if (normalizedPassword.length < 6) {
    const error = new Error("Пароль має містити щонайменше 6 символів.");
    error.status = 400;
    throw error;
  }

  const existingUser = getUserByEmail(normalizedEmail);
  if (existingUser) {
    const error = new Error("Користувач із таким email уже існує.");
    error.status = 409;
    throw error;
  }
const passwordHash = await bcrypt.hash(normalizedPassword, 10);
const user = createUser({
  username: normalizedUsername,
  email: normalizedEmail,
  passwordHash
});

return toPublicUser(user);
```

Рис. 2.7. Фрагмент коду, що відповідає за реєстрацію користувача

```
async function getPopularMovies({ page = 1, genreId = "" }) {
  const normalizedPage = Math.min(Math.max(Number(page) || 1, 1), 500);
  const hasGenreFilter = Boolean(genreId);
  const endpoint = hasGenreFilter ? "/discover/movie" : "/movie/popular";

  const data = await tmdbRequest(endpoint, {
    language: "uk-UA",
    page: normalizedPage,
    region: "UA",
    include_adult: "false",
    sort_by: hasGenreFilter ? "popularity.desc" : undefined,
    with_genres: hasGenreFilter ? genreId : undefined
  });

  return {
    page: data.page,
    totalPages: Math.min(data.total_pages || 1, 500),
    results: Array.isArray(data.results) ? data.results.map(mapMovieSummary) : []
  };
}

async function searchMovies({ query, page = 1, genreId = "" }) {
  const normalizedQuery = String(query || "").trim();
  const normalizedPage = Math.min(Math.max(Number(page) || 1, 1), 500);

  if (!normalizedQuery) {
    const error = new Error("Потрібно ввести назву фільму для пошуку.");
    error.status = 400;
    throw error;
  }
const data = await tmdbRequest("/search/movie", {
  language: "uk-UA",
  page: normalizedPage,
  query: normalizedQuery,
  include_adult: "false"
});

const filteredResults = Array.isArray(data.results)
  ? data.results.filter((movie) => {
      if (!genreId) {
        return true;
      }

      return Array.isArray(movie.genre_ids) && movie.genre_ids.includes(Number(genreId));
    })
  : [];

return {
  page: data.page,
  totalPages: Math.min(data.total_pages || 1, 500),
  results: filteredResults.map(mapMovieSummary)
};


sync function getMovieDetails(movieId) {
const [movie, similar] = await Promise.all([
  tmdbRequest(`/movie/${movieId}`, {
    language: "uk-UA",
    append_to_response: "credits,videos"
  }),
  tmdbRequest(`/movie/${movieId}/similar`, {
    language: "uk-UA",
    page: 1
    })
  ]);

  const payload = mapMovieDetail(movie);
  payload.similar = Array.isArray(similar.results)
    ? similar.results
        .map(mapMovieSummary)
```

Рис. 2.8 Фрагмент коду, що реалізує отримання списку фільмів і детальної інформації з TMDB API

У системі реалізовано окрему групу операцій для роботи з персоналізованими даними користувача. До них належать додавання фільму до обраного, видалення фільму з обраного, перегляд історії пошуку та її очищення. Ця логіка реалізована у файлах `library.controller.js`, `library.service.js` та `database.js`. Фрагмент коду, що ілюструє CRUD-операції для списку обраного та історії пошуку, наведено на рисунках 2.9 та 2.10.

```
} = require("../db/database");

function listFavorites(userId) {
  return getFavorites(userId);
}

function addFavorite(userId, movie) {
  saveFavorite(userId, movie);
}

function deleteFavorite(userId, tmdbId) {
  return removeFavorite(userId, tmdbId);
}

function listSearchHistory(userId, limit) {
  return getSearchHistory(userId, limit);
}

function addSearchQuery(userId, query) {
  const normalizedQuery = String(query || "").trim();

  if (!normalizedQuery || !userId) {
    return;
  }

  saveSearchQuery(userId, normalizedQuery);
}

function deleteSearchHistory(userId) {
  clearSearchHistory(userId);
}

module.exports = {
  addFavorite,
  addSearchQuery,
```

Рис. 2.9. Фрагмент коду, що реалізує обробку операцій з обраним та історією пошуку на рівні контролера

```
function saveFavorite(userId, movie) {
  const statement = database.prepare(`
    INSERT INTO favorites (
      user_id,
      tmdb_id,
      title,
      original_title,
      overview,
      release_date,
      poster_url,
      backdrop_url,
      vote_average,
      created_at
    )
    VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, CURRENT_TIMESTAMP)
    ON CONFLICT(user_id, tmdb_id) DO UPDATE SET
      title = excluded.title,
      original_title = excluded.original_title,
      overview = excluded.overview,
      release_date = excluded.release_date,
      poster_url = excluded.poster_url,
      backdrop_url = excluded.backdrop_url,
      vote_average = excluded.vote_average,
      created_at = CURRENT_TIMESTAMP
  `);

  statement.run(
    userId,
    movie.tmdbId || movie.id,
    movie.title,
    movie.originalTitle || "",
    movie.overview || "",
    movie.releaseDate || "",
    movie.posterUrl || "",
    movie.backdropUrl || "",
    movie.voteAverage || null
  );
}

function removeFavorite(userId, tmdbId) {
  const statement = database.prepare("DELETE FROM favorites WHERE user_id = ? AND tmdb_id = ?");
  const result = statement.run(userId, tmdbId);

  return result.changes > 0;
}

function getSearchHistory(userId, limit = 8) {
  const statement = database.prepare(`
    SELECT
      id,
      query_text AS query,
      created_at AS createdAt
    FROM search_history
    WHERE user_id = ?
    ORDER BY datetime(created_at) DESC
    LIMIT ?
  `);

  return statement.all(userId, limit);
}

function saveSearchQuery(userId, query) {
  const statement = database.prepare(`
    INSERT INTO search_history (user_id, query_text, created_at)
    VALUES (?, ?, CURRENT_TIMESTAMP)
    ON CONFLICT(user_id, query_text) DO UPDATE SET
      created_at = CURRENT_TIMESTAMP
  `);

  statement.run(userId, query);
}

function clearSearchHistory(userId) {
  database.prepare("DELETE FROM search_history WHERE user_id = ?").run(userId);
}
```

Рис. 2.10. Фрагмент коду, що реалізує збереження, видалення та отримання даних обраного й історії пошуку в локальній базі даних

Клієнтська частина реалізована засобами HTML, CSS та JavaScript. У проєкті створено три основні сторінки: головну сторінку `index.html`, сторінку детальної інформації `movie.html` та сторінку профілю користувача `profile.html`. Логіка клієнтської частини винесена в окремі модулі `home.js`, `movie.js`, `profile.js`, `api.js` та `ui.js`.

У файлі `home.js` реалізовано завантаження популярних фільмів, обробку пошуку, зміну жанрового фільтра, пагінацію, а також додавання фільмів до обраного. Крім того, реалізовано синхронізацію параметрів сторінки з адресним рядком браузера. Фрагмент коду, що відповідає за завантаження та відображення списку фільмів на головній сторінці, наведено на рисунку 2.11.

```
  statusText.textContent = "Завантаження фільмів...";
  if (state.mode === "search" && state.query) {
    sectionTitle.textContent = `Результати пошуку: ${state.query}`;
  } else if (state.genre) {
    sectionTitle.textContent = `Фільми за жанром: ${getSelectedGenreLabel()}`;
  } else {
    sectionTitle.textContent = "Популярні фільми";
  }

  try {
    const payload =
      state.mode === "search" && state.query
        ? await moviesApi.searchMovies({
            query: state.query,
            page: state.page,
            genre: state.genre
          })
        : await moviesApi.getPopularMovies({
            page: state.page,
            genre: state.genre
          });

    state.totalPages = payload.totalPages || 1;
    state.moviesById = new Map(payload.results.map((movie) => [movie.id, movie]));

    renderMovies(payload.results);
    updatePagination();
    statusText.textContent =
      state.mode === "search" && state.query
        ? "Показано результати пошуку."
        : "Показано список популярних фільмів.";

    if (state.mode === "search" && state.query) {
      await loadHistory();
    }
  } catch (error) {
    moviesGrid.innerHTML = createEmptyStateMarkup("Помилка завантаження", error.message);
    paginationInfo.textContent = "";
    statusText.textContent = "Сталася помилка під час звернення до сервера.";
  }
}
```

Рис. 2.11 Фрагмент коду, що реалізує завантаження та відображення списку фільмів на головній сторінці

У файлі `profile.js` реалізовано форми входу та реєстрації, перевірку полів, завантаження профільних даних, видалення фільмів з обраного та очищення історії пошуку. У файлі `movie.js` реалізовано логіку сторінки деталей фільму, завантаження повної інформації про фільм, відображення схожих фільмів та додавання або видалення фільму з обраного.

Також, передбачено обробку типових проблемних ситуацій. Якщо у відповіді відсутній постер або опис, в інтерфейсі відображається резервний текст. Якщо під час виконання запиту виникає помилка, користувач отримує відповідне повідомлення. Для персоналізованих функцій застосовується перевірка авторизації, що обмежує доступ до них для неавторизованих користувачів.

У межах цього етапу було реалізовано серверну та клієнтську частини вебзастосунку для перегляду популярних фільмів з використанням TMDB API. Серверна частина забезпечує роботу із зовнішнім API, локальною базою даних та сесіями користувачів, а клієнтська частина реалізує інтерфейс для пошуку, перегляду фільмів, авторизації та використання персоналізованих функцій.

Висновки до розділу 2

Розглянуто проєктування та реалізацію вебзастосунку для перегляду популярних фільмів з використанням TMDB API. Було визначено інформаційну модель системи, у межах якої поєднуються зовнішні дані про фільми та локальні дані користувачів.

Спроєктовано локальну базу даних SQLite, до складу якої входять таблиці користувачів, обраних фільмів та історії пошуку. Також описано підходи до збору й обробки даних, які ґрунтуються на використанні серверної частини як проміжної ланки між клієнтською частиною та TMDB API.

У результаті реалізовано серверну та клієнтську частини вебзастосунку. Розроблений застосунок забезпечує перегляд популярних фільмів, пошук за назвою, фільтрацію за жанром, перегляд детальної інформації про фільм,

реєстрацію та авторизацію користувачів, роботу з обраним та історією пошуку. Отриманий результат є достатньою основою для подальшого опису функціонування застосунку.

РОЗДІЛ 3
ФУНКЦІОНУВАННЯ ТА ЕЛЕМЕНТИ ІНТЕРФЕЙСУ ВЕБЗАСТОСУНКУ ДЛЯ ПЕРЕГЛЯДУ ПОПУЛЯРНИХ ФІЛЬМІВ

3.1 Огляд функціоналу розробленого вебзастосунку

Розроблений вебзастосунок призначений для перегляду популярних фільмів, отримання детальної інформації про них та використання персоналізованих функцій після авторизації користувача. Його функціональне призначення полягає в поєднанні зовнішнього інформаційного ресурсу TMDB API з локальними можливостями збереження персоналізованих даних користувача.

Базовий функціонал застосунку охоплює перегляд популярних фільмів, пошук за назвою, фільтрацію за жанром, перегляд детальної інформації про обраний фільм, реєстрацію та авторизацію користувача, роботу зі списком обраних фільмів та перегляд історії пошуку. Кожна з цих функцій реалізована як окремий елемент загальної системи та доступна через вебінтерфейс браузера. Сукупність зазначених можливостей формує завершений користувацький сценарій, який відповідає поставленим завданням дипломної роботи.

Після відкриття головної сторінки (див. рис. 3.1) користувачеві відображається перелік популярних фільмів, отриманий із TMDB API через серверну частину застосунку. У візуальній структурі головної сторінки центральне місце займає блок карток фільмів, який дає змогу швидко ознайомитися з основною інформацією про доступний контент. Кожна картка містить назву фільму, постер, рейтинг, дату виходу та короткий опис.

Важливим елементом головної сторінки є форма пошуку, яка дозволяє користувачу знаходити фільми за назвою. Пошук виконується через серверну частину, яка звертається до TMDB API та повертає сформований список результатів. Пошуковий механізм працює у зв’язці з жанровим фільтром, що дозволяє уточнювати результати. Наявність поєднання текстового пошуку і

тематичного фільтра створює більш гнучкий механізм навігації в контенті та підвищує практичну цінність застосунку.

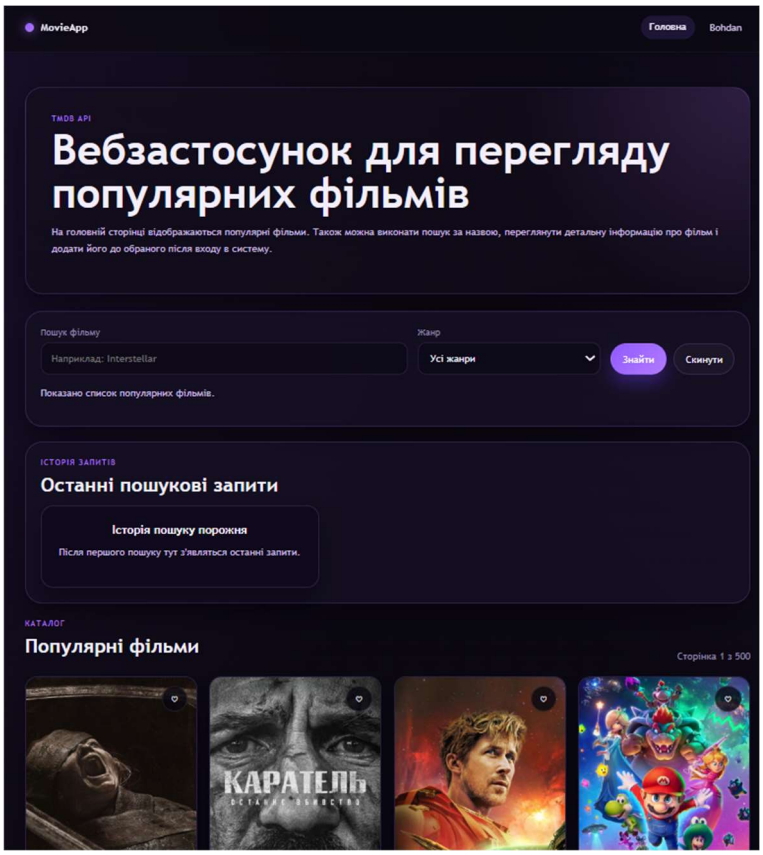

Рис. 3.1. Головна сторінка вебзастосунку

Окрім пошуку, на головній сторінці реалізовано фільтрацію за жанром. Жанри завантажуються із TMDB API та відображаються у випадаючому списку. Після вибору конкретного жанру користувач отримує відфільтрований перелік фільмів, який формується на серверному рівні з використанням відповідних параметрів запиту до TMDB API. Така можливість є важливою для зручності використання вебзастосунку, оскільки дозволяє скоротити кількість нерелевантних результатів і зосередити увагу на потрібній категорії контенту.

Для роботи з великим набором результатів у застосунку реалізовано пагінацію. Вона дає змогу переходити між окремими сторінками списку популярних фільмів або пошукових результатів. Пагінація підтримує впорядковану структуру навігації та запобігає перевантаженню інтерфейсу

надмірною кількістю карток на одній сторінці. У результаті користувач отримує більш стабільний і зрозумілий спосіб взаємодії з інформацією.

Суттєвою функціональною частиною системи є сторінка детальної інформації про фільм (див. рис. 3.2). Перехід на цю сторінку здійснюється після вибору відповідної картки на головній сторінці або в результатах пошуку. У межах цієї сторінки користувач отримує розширений набір відомостей: назву, оригінальну назву, дату виходу, рейтинг, тривалість, режисера, жанри, опис, акторський склад, посилання на трейлер, посилання на офіційний сайт фільму та перелік схожих фільмів. Це дозволяє використовувати застосунок не лише як інструмент пошуку, а і як засіб детального ознайомлення з кінематографічним контентом.

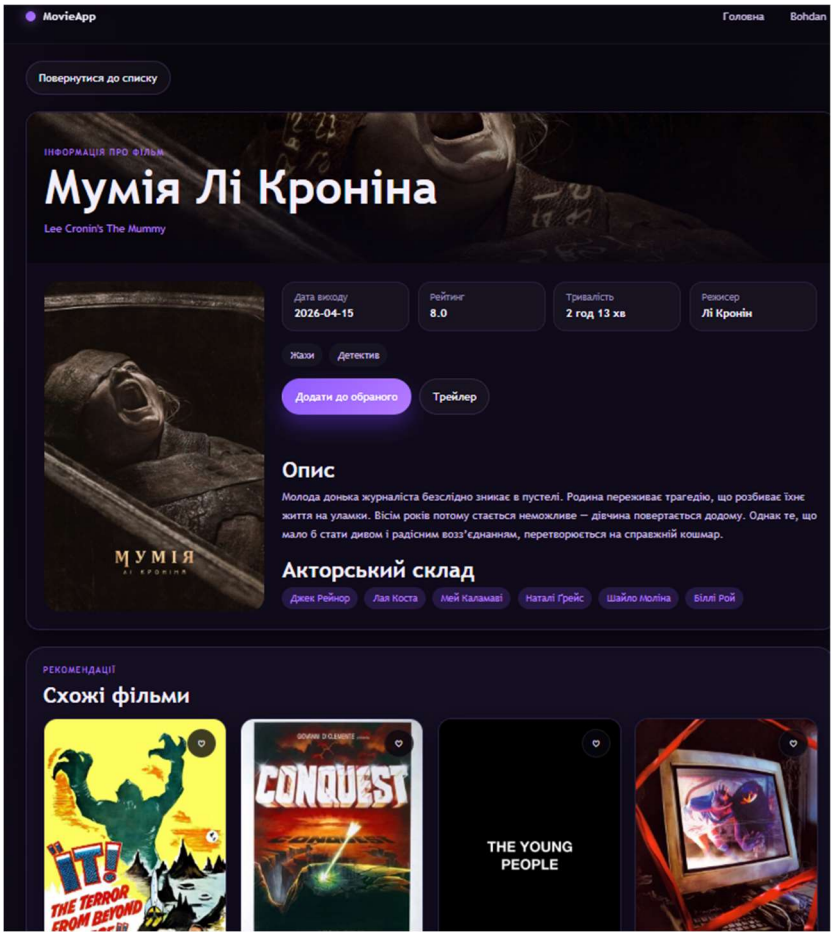

Рис. 3.2. Сторінка детальної інформації про фільм

Наявність блоку схожих фільмів є окремою перевагою реалізованого функціоналу. Цей блок розширює навігаційні можливості користувача та сприяє перегляду пов’язаного контенту без повернення до головної сторінки. У функціональному сенсі це формує елемент рекомендаційного механізму, що підвищує зручність користування застосунком і наближає його до реальних інформаційно-розважальних сервісів.

Окреме місце у функціоналі займає підсистема авторизації користувача. Для неавторизованого користувача на сторінці профілю відображаються форми реєстрації та входу (див. рис. 3.3). Після успішного створення облікового запису або входу до системи користувач отримує доступ до персоналізованих можливостей. Таким чином, застосунок поєднує відкритий інформаційний функціонал, доступний без авторизації, і додаткові персональні функції, що стають доступними після входу до системи.

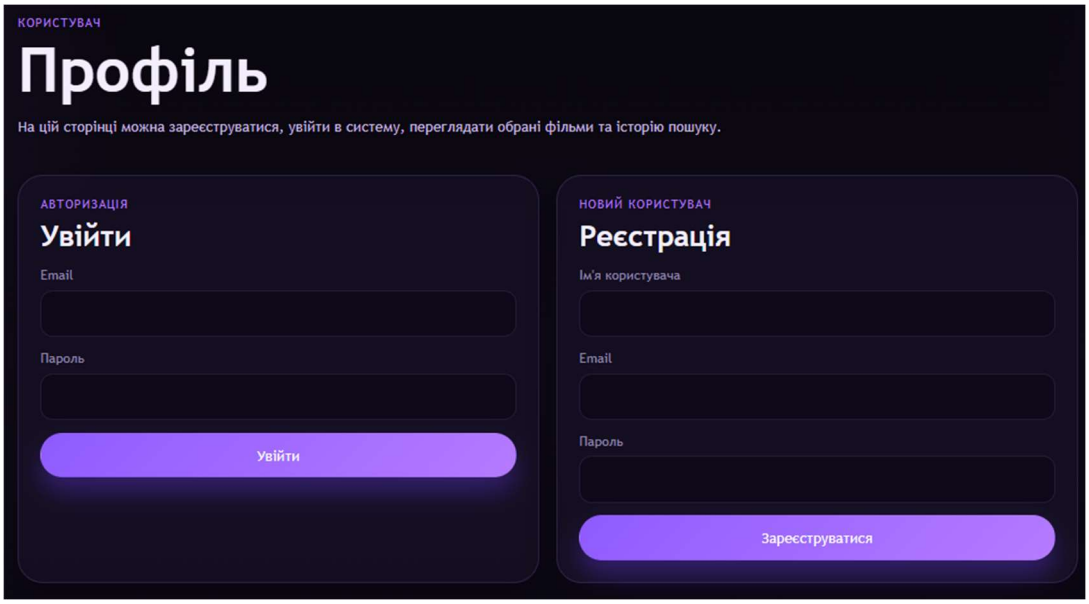

Рис. 3.3. Сторінка профілю у неавторизованому стані з формами входу та реєстрації

Після авторизації користувачу стають доступними функції, пов’язані з обраним та історією пошуку. Список обраних фільмів дає змогу зберігати цікаві позиції для подальшого перегляду. Це дозволяє не втрачати знайдені фільми та

повернутися до них у зручний момент. У свою чергу, історія пошуку зберігає раніше введені запити, що дає змогу повторно використовувати їх без необхідності повторного введення тексту.

Сторінка профілю (див. рис. 3.4) після авторизації виконує роль центрального блоку персоналізованої взаємодії. У ній відображаються ім’я користувача, адреса електронної пошти, дата реєстрації, кількість елементів у списку обраного, кількість записів в історії пошуку, список обраних фільмів та останні пошукові запити. Така структура дає змогу користувачу швидко оцінити власну активність у системі та виконати основні дії, не переходячи до інших сторінок.

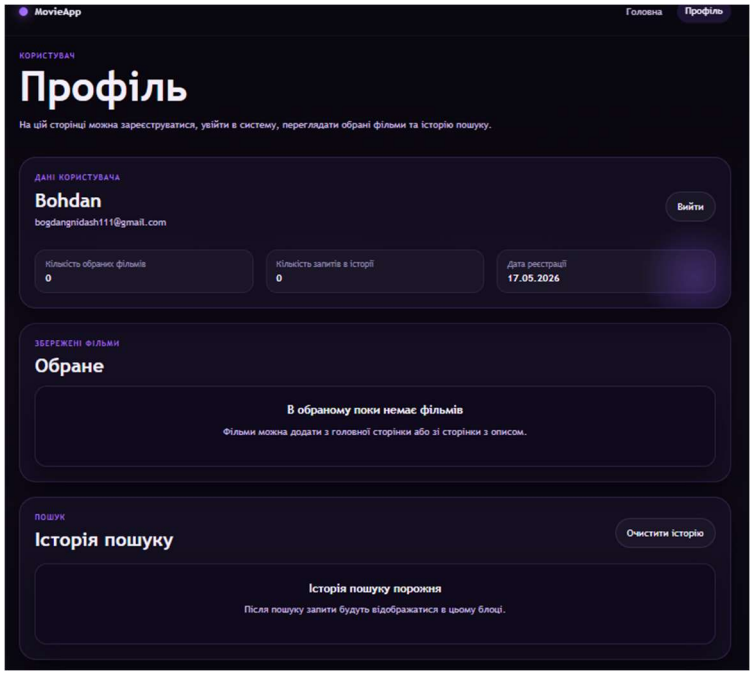

Рис. 3.4. Сторінка профілю авторизованого користувача

У межах функціоналу також реалізовано базову систему повідомлень і перевірки введених даних. Застосунок інформує користувача про успішний вхід, завершення реєстрації, додавання фільму до обраного, видалення з обраного, очищення історії пошуку або виникнення помилки. Такий підхід підвищує зрозумілість роботи системи та формує більш передбачувану поведінку інтерфейсу.

Важливою характеристикою реалізованого функціоналу є його узгодженість із серверною архітектурою. Клієнтська частина не взаємодіє із TMDB API безпосередньо, а працює через серверну частину, яка виконує проміжну обробку даних. Це дозволяє приховати ключ доступу до API, централізувати логіку обробки та поєднати зовнішні дані з локальним збереженням персоналізованої інформації.

Функціональні можливості розробленого вебзастосунку охоплюють як базові інформаційні операції з кінематографічним контентом, так і індивідуалізовані сервіси, орієнтовані на конкретного користувача. У результаті створено вебресурс, який відповідає темі дипломної роботи та забезпечує практичну реалізацію поставлених завдань.

3.2 Опис взаємодії користувача з розробленим вебзастосунком

Взаємодію користувача з вебзастосунком доцільно розглядати як послідовний сценарій, що починається з відкриття головної сторінки і поступово переходить до використання персоналізованих функцій системи.

Початковою точкою взаємодії є головна сторінка вебзастосунку. Після її відкриття користувач отримує доступ до переліку популярних фільмів, що автоматично завантажуються через серверну частину. На цьому етапі користувач може переглядати доступні картки фільмів, ознайомлюватися з короткими відомостями та здійснювати первинний вибір контенту. Саме головна сторінка виконує роль основної точки входу до системи, оскільки поєднує функції навігації, пошуку та попереднього перегляду вмісту.

У межах першого сценарію користувач може просто переглядати популярні фільми без виконання додаткових дій. Такий режим використання підходить для швидкого ознайомлення з популярним контентом та не потребує авторизації. На цьому етапі користувач бачить роботу відкритого функціоналу вебзастосунку, який доступний одразу після відкриття сайту.

Наступним логічним сценарієм є використання пошуку див. рис. 3.5. Користувач вводить назву фільму в поле пошуку та надсилає відповідний запит. Після цього клієнтська частина передає введений текст на сервер, сервер звертається до TMDB API та повертає сформований результат у вигляді нового списку карток. Якщо введений запит не дає результату, система відображає відповідне повідомлення, що дозволяє користувачу скоригувати параметри пошуку.

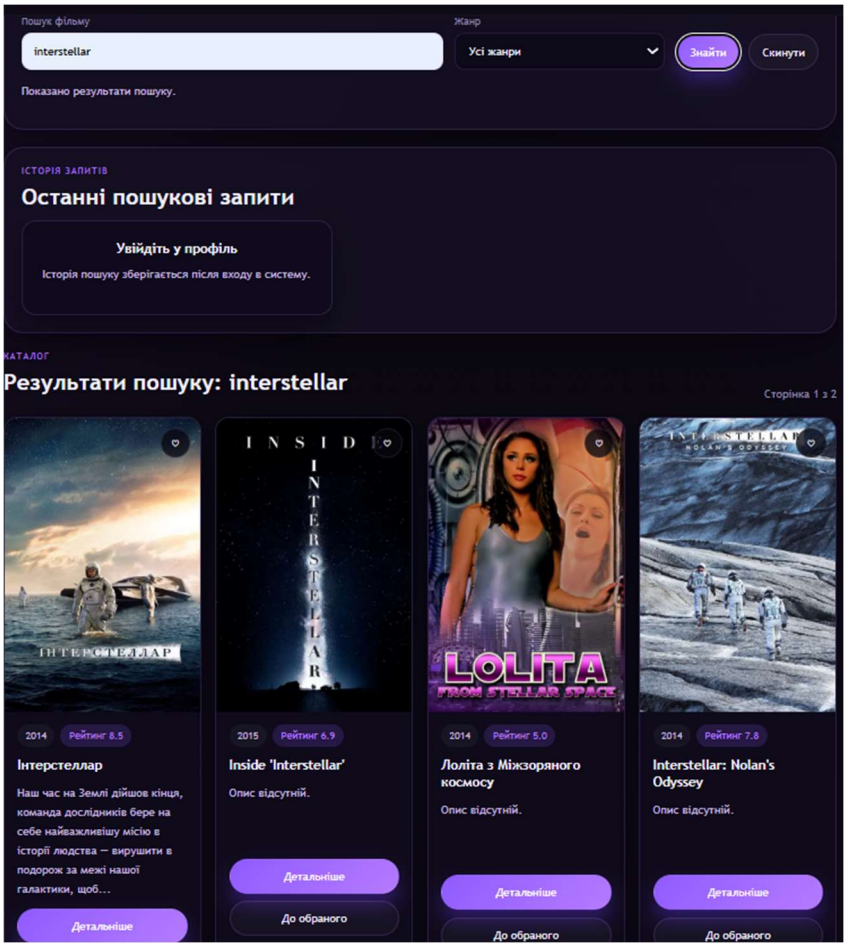

Рис. 3.5. Скріншот результатів пошуку за назвою фільму

Пошуковий сценарій може бути доповнений використанням жанрового фільтра. У цьому випадку користувач або обирає жанр до виконання пошуку, або використовує фільтрацію після отримання результатів. Такий підхід є зручним у ситуаціях, коли потрібно уточнити перелік знайдених фільмів і звузити його до

певної категорії. Унаслідок цього вебзастосунок забезпечує не лише загальний пошук, а й більш цілеспрямований відбір контенту.

Ще одним важливим сценарієм є перехід на сторінку детальної інформації. Після натискання на картку фільму користувач потрапляє на окрему сторінку, де містяться розширені відомості про вибраний фільм. На цій сторінці відображаються базові характеристики, опис, жанри, акторський склад, а також блок схожих фільмів. Завдяки цьому користувач переходить від короткого перегляду до більш глибокого ознайомлення з об’єктом.

У межах сторінки деталей користувач може або завершити ознайомлення з інформацією, або продовжити навігацію через блок схожих фільмів. Наявність такого блоку формує природне продовження взаємодії із системою, оскільки користувачеві не потрібно повертатися на головну сторінку для пошуку нового контенту. Це підвищує зв’язність користувацького сценарію та зручність використання вебзастосунку.

Окремий сценарій пов’язаний із переходом до сторінки профілю. Якщо користувач не авторизований, на цій сторінці йому пропонуються форми входу та реєстрації. Відповідно, наступним кроком стає створення нового облікового запису або введення вже наявних облікових даних. На цьому етапі користувач взаємодіє з формами, вводить ім’я, електронну адресу та пароль, після чого система перевіряє коректність даних.

Після успішної реєстрації або входу користувач переходить до персоналізованого етапу взаємодії. У профілі відображаються збережені фільми, історія пошукових запитів, дата реєстрації та інші дані облікового запису. Таким чином, сторінка профілю стає центром керування персональними даними користувача в межах системи.

Практичне значення має сценарій додавання фільму до обраного. Якщо користувач авторизований, відповідна дія може виконуватися як із головної сторінки, так і зі сторінки деталей фільму. Після натискання на кнопку додавання клієнтська частина передає запит серверу, сервер зберігає відповідні дані у

локальній базі, а система оновлює стан інтерфейсу. У результаті користувач бачить, що фільм успішно збережено, а пізніше може знайти його у профілі.

Подібним чином працює і сценарій видалення фільму з обраного. Він виконується переважно зі сторінки профілю, де користувач переглядає свій список збережених позицій. Після виконання дії список оновлюється, а система повідомляє користувача про результат операції. Завдяки цьому користувач має контроль над власною колекцією обраних фільмів.

Ще одним важливим сценарієм є робота з історією пошуку. Після авторизації кожен новий пошуковий запит зберігається у локальній базі даних. У профілі ці запити відображаються окремим списком, який може бути використаний повторно або очищений повністю. Така функція підвищує зручність роботи з застосунком, особливо у випадках багаторазового повернення до раніше знайденого контенту.

У цілому користувацький сценарій взаємодії з вебзастосунком є послідовним і логічно побудованим. Він охоплює відкриття головної сторінки, пошук або фільтрацію фільмів, перехід до сторінки детальної інформації, авторизацію та використання персоналізованих функцій. Саме така послідовність демонструє повний цикл роботи користувача із системою та підтверджує її функціональну завершеність.

3.3 Аналіз ефективності розробленого вебзастосунку та можливих проблем у його функціонуванні

Аналіз результатів реалізації вебзастосунку показує, що обраний підхід до його побудови є доцільним для вирішення поставленого завдання. Застосунок поєднує клієнт-серверну архітектуру, використання зовнішнього джерела даних та локальне збереження персоналізованої інформації користувача.

З позиції функціональної ефективності вебзастосунок забезпечує реалізацію всіх базових можливостей, визначених у межах теми дипломної роботи. Користувач може переглядати популярні фільми, виконувати пошук,

застосовувати жанрову фільтрацію, переглядати детальну інформацію про окремі позиції, реєструватися, входити у систему, працювати зі списком обраного та історією пошуку. Отже, створене рішення є не лише демонстраційним, а й функціонально завершеним у межах базової версії.

Важливою перевагою застосунку є використання TMDB API як джерела кінематографічних даних. Це рішення дозволяє уникнути самостійного наповнення великої локальної бази фільмів і водночас працювати з актуальними відомостями про контент. Завдяки цьому користувач отримує інформацію про популярні фільми у форматі, максимально наближеному до реального інформаційного сервісу.

Ефективність системи також проявляється в організації серверної частини. Клієнтська частина не звертається до TMDB API безпосередньо, а працює через серверний прошарок. Це дозволяє приховати ключ доступу, централізувати обробку даних, уніфікувати відповіді для фронтенду та забезпечити взаємодію з локальною базою даних. У практичному сенсі такий підхід підвищує безпеку та керованість вебзастосунку.

Окремо слід відзначити використання SQLite для локального збереження даних користувачів. Обрана база даних є достатньою для навчального вебпроєкту та дозволяє реалізувати персоналізовані функції без ускладнення системи додатковим сервером баз даних.

Зручність використання вебзастосунку забезпечується за рахунок зрозумілого інтерфейсу та логіки побудови сторінок. Головна сторінка виконує роль стартового блоку, сторінка деталей надає розширену інформацію про вибраний фільм, а сторінка профілю забезпечує доступ до персоналізованого контенту.

До сильних сторін застосунку слід віднести поєднання кількох механізмів роботи з контентом. Це не лише пошук за назвою, але й фільтрація за жанром, перегляд схожих фільмів, збереження фільмів до обраного та фіксація історії пошуку. У сукупності ці функції формують більш повний сервіс, ніж простий

список популярних фільмів, і підвищують практичну значущість реалізованого рішення.

Разом з тим, у роботі вебзастосунку можливі певні обмеження. Передусім вони пов’язані із залежністю від зовнішнього сервісу TMDB API. Якщо API тимчасово недоступне, повертає помилку або працює з затримкою, це безпосередньо впливає на швидкість завантаження і повноту відображених результатів. Подібна залежність є характерною для вебзастосунків, що інтегруються із зовнішніми інформаційними сервісами.

Ще одним можливим обмеженням є неповнота або відсутність окремих відомостей у відповідях TMDB API. На практиці це може проявлятися у вигляді відсутнього постера, неповного опису, відсутності трейлера або недостатньої локалізації окремих назв і описів. У межах реалізованого проєкту такі ситуації частково враховано за допомогою резервних текстових повідомлень, однак повністю усунути їх без контролю над зовнішнім джерелом даних неможливо.

Певним обмеженням є й те, що локальна база даних не містить повного каталогу фільмів, а використовується лише для збереження облікових даних користувача, обраного та історії пошуку. Такий підхід є доцільним у межах навчального проєкту, однак він означає, що застосунок постійно залежить від зовнішнього API при завантаженні фільмів. У разі подальшого розвитку системи це може бути частково компенсовано впровадженням механізмів кешування.

З технічної точки зору потенційною проблемою може бути також обробка одночасних запитів при значному навантаженні. Для поточної версії застосунку ця проблема не є критичною, оскільки проєкт має навчальний характер і не орієнтований на масове комерційне використання. Проте при масштабуванні вебресурсу постало б питання переходу до більш потужної системи збереження даних та розширення серверної інфраструктури.

З погляду безпеки реалізоване рішення містить базові захисні механізми. До них належать хешування паролів, використання сесій для підтримки авторизації та перевірка доступу до персоналізованих функцій. Разом з тим, подальший розвиток системи міг би передбачати розширені заходи захисту,

зокрема більш деталізовану політику обмеження доступу, додаткові механізми захисту сесій та розширене журналювання користувацьких дій.

Практична цінність вебзастосунку полягає не лише в реалізації поставленого функціоналу, а й у демонстрації сучасного підходу до побудови інформаційного сервісу. У проєкті поєднано інтеграцію із зовнішнім API, локальне збереження персоналізованих даних, розмежування серверної та клієнтської частин, а також базові засоби авторизації. Усе це свідчить про відповідність створеного рішення сучасним підходам до розробки вебзастосунків.

Серед перспектив подальшого розвитку системи можна виділити розширення механізмів пошуку, додавання сортування результатів, створення рекомендаційного блоку на основі дій користувача, упровадження кешування відповідей TMDB API, розширення структури профілю користувача та покращення інтерфейсу. Також доцільним напрямом є розширення аналітичної складової, наприклад ведення статистики переглядів або частоти використання пошукових запитів.

У цілому розроблений вебзастосунок можна оцінити як функціонально достатній, логічно структурований та придатний до подальшого розвитку. Реалізоване програмне забезпечення відповідає темі дипломної роботи, демонструє практичне застосування сучасних вебтехнологій і може бути використане як базова реалізація інформаційного сервісу для перегляду популярних фільмів.

Висновки до розділу 3

Розглянуто функціонування та основні елементи інтерфейсу розробленого вебзастосунку для перегляду популярних фільмів. Було подано розгорнутий огляд функціональних можливостей системи, до яких належать перегляд популярних фільмів, пошук, жанрова фільтрація, перегляд детальної інформації, реєстрація, авторизація, робота з обраним та історією пошуку.

Також послідовно описано базовий користувацький сценарій взаємодії із системою, який охоплює відкриття головної сторінки, виконання пошуку, перехід до сторінки детальної інформації, авторизацію та використання персоналізованих функцій у профілі користувача. Окрему увагу приділено оцінці ефективності реалізованого рішення, його сильним сторонам, можливим обмеженням та перспективам подальшого розвитку.

Отримані результати підтверджують, що створений вебзастосунок є працездатним програмним продуктом, який відповідає поставленому завданню та може слугувати практичною основою для підсумкової кваліфікаційної роботи.

ВИСНОВКИ

У кваліфікаційній бакалаврській роботі розв’язано актуальне завдання, яке полягало у розробці вебзастосунку для перегляду популярних фільмів з використанням API TMDB. У ході виконання роботи було розглянуто теоретичні аспекти побудови сучасних вебресурсів для роботи з медіаконтентом, проаналізовано наявні інформаційні платформи кінематографічного спрямування та визначено основні вимоги до розроблюваної системи.

Встановлено, що сучасні вебресурси для перегляду фільмів повинні забезпечувати швидкий доступ до структурованої інформації, підтримувати зручні механізми пошуку та фільтрації, а також враховувати вимоги до адаптивності, зручності користування та персоналізації. Проведений аналіз існуючих ресурсів, зокрема IMDb, JustWatch, Letterboxd, Rotten Tomatoes і TMDB, дав змогу визначити сильні сторони сучасних кінематографічних сервісів і використати ці підходи при формуванні вимог до власного вебзастосунку.

Було виконано проєктування системи, побудовано інформаційну модель вебзастосунку, визначено основні сутності та зв’язки між ними, а також спроєктовано локальну базу даних для збереження персоналізованих даних користувача. Для відображення логічної та фізичної моделей було використано UML-діаграми та ER-діаграму. Крім того, було обґрунтовано використання клієнт-серверної архітектури, зовнішнього джерела даних у вигляді TMDB API та локальної бази даних SQLite.

У межах реалізації програмного рішення було створено серверну частину вебзастосунку на основі Node.js та Express, яка забезпечує обробку запитів користувача, взаємодію із TMDB API, виконання авторизації та роботу з локальною базою даних. Клієнтську частину реалізовано засобами HTML, CSS та JavaScript. У результаті забезпечено відображення популярних фільмів, пошук за назвою, жанрову фільтрацію, пагінацію результатів, перегляд детальної інформації про фільм, а також роботу зі сторінкою профілю користувача.

Розглянуто функціонування створеного вебзастосунку, описано основні елементи інтерфейсу та послідовність взаємодії користувача з системою. Встановлено, що реалізоване рішення забезпечує виконання базових функцій, передбачених темою дипломної роботи, а саме перегляд популярних фільмів, пошук, фільтрацію, реєстрацію, вхід, збереження обраного та історії пошуку. Також визначено сильні сторони застосунку, можливі обмеження його функціонування та перспективи подальшого розвитку.

Отже, поставлену мету досягнуто, а завдання кваліфікаційної бакалаврської роботи виконано в повному обсязі. Розроблений вебзастосунок є працездатним програмним продуктом, який відповідає темі дослідження, демонструє практичне використання сучасних вебтехнологій і може бути використаний як основа для подальшого вдосконалення інформаційного сервісу для перегляду популярних фільмів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ericsson. Exploring mobile traffic trends 2020–2024. Ericsson Mobility Report, 2024. URL: https://www.ericsson.com/en/reports-and-papers/mobility-report/articles/exploring-mobile-traffic-trends-2020-2024

2. Ericsson. Streaming video – from megabits to gigabytes. Ericsson Mobility Report, 2024. URL: https://www.ericsson.com/en/reports-and-papers/mobility-report/articles/streaming-video

3. Broocks A., Studnicka Z. Gravity and trade in video on demand services. Review of World Economics, 2025. URL: https://link.springer.com/article/10.1007/s10290-024-00561-5

4. Ameri M., Honka E., Xie Y. Watching intensity and media franchise engagement. Quantitative Marketing and Economics, 2024. URL: https://link.springer.com/article/10.1007/s11129-024-09283-2

5. Romero Meza L., D'Urso G. User's Dilemma: A Qualitative Study on the Influence of Netflix Recommender Systems on Choice Overload. Psychological Studies, 2024. URL: https://link.springer.com/article/10.1007/s12646-024-00807-0

6. Lu G., Qu S., Chen Y. Understanding user experience for mobile applications: a systematic literature review. Discover Applied Sciences, 2025. URL: https://link.springer.com/article/10.1007/s42452-025-07170-3

7. Sulong S., Yusof R. J. Raja. Developing a blind user mental model (BlUMM) for web browsing. Universal Access in the Information Society, 2024. URL: https://link.springer.com/article/10.1007/s10209-023-01035-5

8. Roy D., Dutta M. A systematic review and research perspective on recommender systems. Journal of Big Data, 2022. URL: https://link.springer.com/article/10.1186/s40537-022-00592-5

9. Kuo C.-S., Hsu C.-C. Continuance Intention to Use and Perceived Net Benefits as Perceived by Streaming Platform Users: An Application of the Updated IS Success Model. Behavioral Sciences, 2022. URL: https://www.mdpi.com/2076-328X/12/5/124

10. Kulkarni A., Shivananda A., Kulkarni A., Krishnan V. A. Applied Recommender Systems with Python: Build Recommender Systems with Deep Learning, NLP and Graph-Based Techniques. Apress, 2023. URL: https://link.springer.com/book/10.1007/978-1-4842-8954-9

11. Ravanmehr R., Mohamadrezaei R. Session-Based Recommender Systems Using Deep Learning. Springer, 2024. URL: https://link.springer.com/book/10.1007/978-3-031-42559-2

12. IMDb. Ratings FAQ. URL: https://help.imdb.com/article/imdb/track-movies-tv/ratings-faq/G67Y87TFYYP6TWAV

13. JustWatch. What is JustWatch? URL: https://support.justwatch.com/hc/en-us/articles/360020520097-What-is-JustWatch

14. The Movie Database. Getting Started. URL: https://developer.themoviedb.org/docs/getting-started

15. The Movie Database. Finding Data. URL: https://developer.themoviedb.org/docs/finding-data

16. Letterboxd. Social film discovery. URL: https://letterboxd.com/

17. Rotten Tomatoes. About Rotten Tomatoes. URL: https://www.rottentomatoes.com/about

18. The Movie Database. Search & Query For Details. URL: https://developer.themoviedb.org/docs/search-and-query-for-details

19. The Movie Database. Popularity & Trending. URL: https://developer.themoviedb.org/docs/popularity-and-trending

20. The Movie Database. API Reference. URL: https://developer.themoviedb.org/reference/intro/getting-started

21. Node.js. Node.js Documentation. URL: https://nodejs.org/docs/latest/api/

22. Express.js. Installing. URL: https://expressjs.com/en/starter/installing.html

23. Express.js. Routing. URL: https://expressjs.com/en/guide/routing/

24. Express.js. Session middleware. URL: https://expressjs.com/en/resources/middleware/session.html

25. SQLite Documentation. URL: https://www.sqlite.org/docs.html

26. SQLite. SQL As Understood By SQLite. URL: https://www.sqlite.org/lang.html

27. SQLite. CREATE TABLE. URL: https://www.sqlite.org/lang_createtable.html

28. MDN Web Docs. HTML: HyperText Markup Language. URL: https://developer.mozilla.org/en-US/docs/Web/HTML

29. MDN Web Docs. CSS: Cascading Style Sheets. URL: https://developer.mozilla.org/en-US/docs/Web/CSS

30. MDN Web Docs. JavaScript. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript

31. MDN Web Docs. Fetch API. URL: https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API

32. MDN Web Docs. HTTP response status codes. URL: https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Status

Додаток А

Код, що відповідає за відображення головної сторінки

```
<!DOCTYPE html>
<html lang="uk">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Вебзастосунок для перегляду фільмів</title>
    <link rel="stylesheet" href="/css/style.css" />
  </head>
  <body>
    <div class="page-shell">
      <header class="site-header">
        <div class="container header-bar">
          <a class="brand" href="/">
            <span class="brand__mark"></span>
            <span class="brand__text">MovieApp</span>
          </a>
          <nav class="nav">
            <a class="nav__link nav__link--active" href="/">Головна</a>
            <a id="profile-link" class="nav__link" href="/profile.html">Профіль</a>
          </nav>
        </div>
      </header>

      <main>
        <section class="hero">
          <div class="container hero__content">
            <p class="eyebrow">TMDB API</p>
            <h1>Вебзастосунок для перегляду популярних фільмів</h1>
            <p class="hero__text">
              На головній сторінці відображаються популярні фільми. Також можна
виконати
              пошук за назвою, переглянути детальну інформацію про фільм і додати його
              до обраного після входу в систему.
            </p>
          </div>
        </section>

        <section class="container section-stack">
          <div class="panel">
            <form id="search-form" class="search-form">
              <label class="field">
                <span class="field__label">Пошук фільму</span>
                <input
                  id="search-input"
                  class="field__control"
                  type="search"
```

Продовження додатку А

```
        placeholder="Наприклад: Interstellar"
      />
    </label>

    <label class="field">
      <span class="field__label">Жанр</span>
      <select id="genre-select" class="field__control">
        <option value="">Усі жанри</option>
      </select>
    </label>

    <div class="search-form__actions">
      <button class="button button--primary" type="submit">Знайти</button>
      <button id="reset-button" class="button button--ghost" type="button">
        Скинути
      </button>
    </div>
  </form>

  <div class="panel__meta">
    <p id="status-text" class="status-text">Завантаження даних...</p>
  </div>
</div>

<section class="panel">
  <div class="panel__header">
    <div>
      <p class="eyebrow">Історія запитів</p>
      <h2 class="panel__title">Останні пошукові запити</h2>
    </div>
  </div>
  <div id="history-chips" class="chips"></div>
</section>

<section class="section-stack">
  <div class="section-header">
    <div>
      <p class="eyebrow">Каталог</p>
      <h2 id="section-title">Популярні фільми</h2>
    </div>
    <div id="pagination-info" class="pagination-info"></div>
  </div>

  <div id="movies-grid" class="movies-grid"></div>

  <div class="pagination">
    <button id="prev-page" class="button button--ghost" type="button">
      Попередня
    </button>
    <button id="next-page" class="button button--primary" type="button">
```

Продовження додатку А

```
            Наступна
          </button>
        </div>
      </section>
    </section>
  </main>

  <footer class="site-footer">
    <div class="container footer-content">
      <p>This product uses the TMDB API but is not endorsed or certified by TMDB.</p>
    </div>
  </footer>
 </div>

 <script type="module" src="/js/home.js"></script>
</body>
</html>
```

ЗГОДА здобувача вищої освіти

Державного університету економіки і технологій про
перевірку кваліфікаційної роботи на прояви
академічного плагіату
та розміщення в Репозитарії Університету

Я, **Гнідаш Богдан Максимович**,

підтримую політику Державного університету економіки і технологій з академічної доброчесності і відкритого доступу.

Засвідчую, що кваліфікаційна бакалаврська робота

«Розробка вебзастосунку для перегляду популярних фільмів з використанням API TMDB»

виконана самостійно та не містить академічного плагіату. Я не надавав і не одержував недозволену допомогу під час підготовки цієї роботи. Робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

Із чинним Положенням про запобігання та виявлення академічного плагіату в роботах здобувачів вищої освіти Державного університету економіки і технологій ознайомлений. Чітко усвідомлюю, що в разі виявлення у кваліфікаційній роботі порушення норм академічної доброчесності робота не допускається до захисту або оцінюється незадовільно.

Також я поінформований, що відповідно до «Положення про Репозитарій (електронну базу даних) Державного університету економіки і технологій» зазначена робота буде розміщена в Електронному архіві Університету (Репозитарії ДУЕТ). З умовами такого розміщення ознайомлений.

15.06.26 р  Гнідаш Б. М.