МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Комп'ютерних наук і технологій
(повне найменування факультету)

Комп'ютерні системи та мережі
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)
бакалаврський
(ступінь вищої освіти)

на тему: РОЗРОБКА ВЕБПЛАТФОРМИ ДЛЯ ЗБЕРІГАННЯ ТА ОЦІНЮВАННЯ ФІЛЬМІВ

Виконав(ла): студент(ка) 4 курсу,
групи КНТ-512

Спеціальності
123 Комп'ютерна інженерія
(код і найменування спеціальності)

Освітня програма (спеціалізація)
Комп'ютерна інженерія
(назва освітньої програми (спеціалізації)

ДРУЖКО Р.В.
(ПРІЗВИЩЕ та ініціали)

Керівник КИРИЧЕК Г.Г.
(ПРІЗВИЩЕ та ініціали)
Рецензент ТЯГУНОВ Д.В.
(ПРІЗВИЩЕ та ініціали)

2026

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет Комп'ютерних наук і технологій
Кафедра комп'ютерних систем та мереж
Ступінь вищої освіти бакалаврський
Спеціальність 123 Комп'ютерна інженерія
(код і найменування)
Освітня програма (спеціалізація) Комп'ютерна інженерія
(назва освітньої програми (спеціалізації)

ЗАТВЕРДЖУЮ
Завідувач кафедри КУДЕРМЕТОВ Р.К.
_____
«14» квітня 2026 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

ДРУЖКО Романа Володимировича
(ПРІЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Розробка вебплатформи для зберігання та оцінювання фільмів
керівник проєкту (роботи) к.т.н., доцент, КИРИЧЕК Г.Г.,
(науковий ступінь, вчене звання, ПРІЗВИЩЕ, ім'я, по батькові)
затверджені наказом закладу вищої освіти від « 14 » квітня 2026 року № 160
2. Строк подання студентом проєкту (роботи) 01.06.2026 р.
3. Вихідні дані до проєкту (роботи) опис предметної області; мова програмування PHP; JavaScript, HTML, CSS; СУБД MySQL; технології AJAX, JSON; сервіс TMDB API; середовище розробки Visual Studio Code; локальне серверне середовище XAMPP, Apache, phpMyAdmin; UML-моделювання; стандарти оформлення ДСТУ 3008:2015, ДСТУ 8302:2015.
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1) Опис предметної області;
2) Розробка вимог до програмного забезпечення;
3) Моделювання програмного забезпечення;
4) Проєктування архітектури програмного забезпечення;
5) Проєктування інтерфейсу користувача.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, кількість слайдів, плакатів)
Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРІЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-5	КИРИЧЕК Г.Г.		
Нормоконтроль	ДЬЯЧУК Т.С.		

7. Дата видачі завдання «14» квітня 2026 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Аналіз предметної області	до 18.04.2026	
2	Визначення та аналіз вимог до програмного	до 22.04.2026	
	забезпечення		
3	Розробка специфікації вимог SRS	до 24.04.2026	
4	Розробка діаграми варіантів використання	до 28.04.2026	
5	Розробка діаграми послідовності	до 01.05.2026	
6	Розробка описів прецедентів	до 05.05.2026	
7	Проєктування архітектури програмного	до 12.05.2026	
	забезпечення		
8	Проєктування інтерфейсу користувача	до 22.05.2026	
9	Оформлення пояснювальної записки	до 25.05.2026	
10	Проходження нормоконтролю	до 27.05.2026	
11	Перевірка на наявність академічного плагіату	до 28.05.2026	
12	Проходження рецензування	до 01.06.2026	

Студент(ка) _____ Роман ДРУЖКО
(підпис) (Ім'я ПРІЗВИЩЕ)

Керівник проєкту (роботи) _____ Галина КИРИЧЕК
(підпис) (Ім'я ПРІЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 137 с., 25 рис., 5 табл., 2 дод., 25 джерел.

ВЕБПЛАТФОРМА, ФІЛЬМИ, СЕРІАЛИ, БАЗА ДАНИХ, СИСТЕМА ОЦІНЮВАННЯ, РЕЦЕНЗІЇ, PHP, MYSQL, JAVASCRIPT, AJAX, TMDB API

Об’єкт розроблення – вебплатформа для зберігання, пошуку та оцінювання фільмів і серіалів.

Мета роботи – розробка вебплатформи Spokuta для роботи з медіаконтентом із підтримкою каталогу, системи оцінювання та рецензування, рекомендацій та AJAX-взаємодії.

Під час виконання роботи було проаналізовано платформи IMDb, Letterboxd та TMDB, визначено основні проблеми подібних сервісів: складну навігацію, перевантажений інтерфейс і повільну роботу з каталогом.

У Spokuta реалізовано авторизацію користувачів, систему пошуку та фільтрації, систему оцінювання та рецензій, систему персональних списків, рекомендації та інтеграцію TMDB API.

Для розробки використовувались PHP, MySQL, JavaScript, AJAX та TMDB API. PHP застосовується для серверної логіки та роботи з базою даних, MySQL – для зберігання користувачів і каталогу, а JavaScript та AJAX – для оновлення інтерфейсу без перезавантаження сторінок.

У результаті створено вебплатформу Spokuta з каталогом фільмів і серіалів, системою оцінювання та рецензій, персональними списками, рекомендаціями, адміністративною панеллю та адаптивним інтерфейсом для різних пристроїв.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

AJAX	–	Asynchronous JavaScript and XML, технологія асинхронного обміну даними між клієнтською та серверною частинами вебплатформи
API	–	Application Programming Interface, інтерфейс програмної взаємодії
CRUD	–	Create, Read, Update, Delete, базові операції створення, читання, оновлення та видалення даних
CSS	–	Cascading Style Sheets, мова опису стилів вебсторінок
ER	–	Entity-Relationship, модель сутностей та зв’язків бази даних
HTML	–	HyperText Markup Language, мова розмітки вебсторінок
HTTP	–	HyperText Transfer Protocol, протокол передавання даних у вебсередовищі
JSON	–	JavaScript Object Notation, формат обміну структурованими даними
MySQL	–	Реляційна система керування базами даних
PHP	–	Hypertext Preprocessor, серверна мова програмування
SQL	–	Structured Query Language, мова запитів до реляційних баз даних
TMDB API	–	API сервісу The Movie Database для отримання метаданих про фільми та серіали
UI	–	User Interface, користувацький інтерфейс
UML	–	Unified Modeling Language, уніфікована мова моделювання програмного забезпечення
URL	–	Uniform Resource Locator, адреса ресурсу в мережі Інтернет
БД	–	База даних
СУБД	–	Система управління базами даних

ВСТУП

Розвиток цифрових сервісів суттєво змінив спосіб пошуку, перегляду та оцінювання фільмів і серіалів. Користувачі дедалі частіше потребують не лише короткої інформації про медіаконтент, а й зручного інструменту для зберігання власних списків, оцінювання переглянутих фільмів, написання рецензій і пошуку схожого контенту. Через постійне зростання кількості фільмів і серіалів актуальними є вебплатформи, що поєднують каталогізацію, систему оцінювання, персональні списки та адаптивний інтерфейс.

Існуючі сервіси для роботи з медіаконтентом мають різні переваги, однак не завжди забезпечують зручну взаємодію. Частина платформ орієнтована переважно на довідкову інформацію, інші – на соціальні функції або ведення персональних списків. При цьому користувачам часто доводиться працювати з перевантаженим інтерфейсом, складною навігацією, обмеженими можливостями фільтрації або недостатньо зручною мобільною адаптацією.

У межах дипломної роботи розробляється вебплатформа Spokuta, призначена для зберігання, пошуку та оцінювання фільмів і серіалів. Платформа поєднує каталог медіаконтенту, систему оцінок і рецензій, обране, список перегляду, базові рекомендації схожого контенту та адміністративне керування. Для підвищення зручності частина дій користувача виконується без повного перезавантаження сторінки за допомогою асинхронних запитів.

Метою дипломної роботи є розробка вебплатформи для зберігання, пошуку та оцінювання фільмів і серіалів із підтримкою рольового доступу, персональних списків, базових рекомендацій схожого контенту, адміністративного керування та адаптивного інтерфейсу.

Під час розробки використано PHP, MySQL, JavaScript, AJAX та TMDB API. PHP застосовано для серверної логіки, MySQL – для зберігання даних, JavaScript і AJAX – для динамічної взаємодії з інтерфейсом, а TMDB API – для отримання метаданих про фільми й серіали.

У першому розділі розглянуто предметну область, проаналізовано сучасні платформи для роботи з фільмами та серіалами, визначено проблеми існуючих рішень, обґрунтовано вибір технологій і сформовано постановку задачі розробки вебплатформи Spokuta. У другому розділі сформовано функціональні та нефункціональні вимоги до програмного забезпечення, описано специфікацію вимог і визначено ролі користувачів. У третьому розділі виконано UML-моделювання основних сценаріїв роботи системи, зокрема діаграми варіантів використання, послідовності та активності. У четвертому розділі спроєктовано архітектуру вебплатформи, структуру бази даних та взаємодію основних компонентів. У п’ятому розділі розглянуто проєктування інтерфейсу користувача, адаптивність сторінок, асинхронну взаємодію та результати ручного тестування.

Практична цінність роботи полягає у створенні працездатної вебплатформи Spokuta, яка може бути використана як основа для подальшого розвитку сервісів каталогізації та оцінювання медіаконтенту.

Оформлення пояснювальної записки виконано з урахуванням методичних вказівок до виконання дипломної роботи бакалавра для спеціальності 123 «Комп’ютерна інженерія», а також вимог ДСТУ 3008:2015 і ДСТУ 8302:2015 [1–3]. Окремі результати роботи, пов’язані з проєктуванням інтерфейсу користувача та концепцією вебплатформи для відеоресурсу, були апробовані у наукових публікаціях автора [4, 5].

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Аналіз сучасних платформ для роботи з фільмами та серіалами

Платформи для роботи з фільмами та серіалами давно перестали бути лише каталогами з назвами та короткими описами. Сучасні користувачі очікують від таких сервісів швидкого пошуку, персональних списків перегляду, системи оцінювання, рецензій, рекомендацій, адаптації до мобільних пристроїв та зручної взаємодії з інтерфейсом без зайвих переходів між сторінками.

Через велику кількість медіаконтенту однією з основних проблем є навігація. Користувачу важливо не лише знайти потрібний фільм або серіал, а й швидко відфільтрувати результати, зберегти контент в обране або список перегляду, переглянути оцінки, отримати рекомендації та повернутися до перегляду пізніше без складної послідовності дій.

Для аналізу предметної області було розглянуто популярні платформи для роботи з медіаконтентом: IMDb, Letterboxd та TMDB. Кожен із цих сервісів реалізує власний підхід до організації каталогу, персоналізації, користувацької взаємодії та роботи з метаданими. Аналіз їхніх переваг і недоліків став основою для формування концепції вебплатформи Spokuta.

IMDb є однією з найбільших платформ для роботи з інформацією про фільми та серіали. Сервіс містить значну кількість метаданих: описи фільмів і серіалів, акторський склад, режисерів, студії, трейлери, оцінки, рецензії, дати виходу та іншу інформацію. Аналіз функціональних можливостей IMDb виконано на основі відкритих матеріалів офіційного вебресурсу платформи [6].

Головною перевагою IMDb є масштаб каталогу. Платформа підтримує як популярні новинки, так і старі або маловідомі фільми. Система пошуку та фільтрації працює стабільно навіть за великого обсягу даних. Користувач може виконувати пошук за назвами, жанрами, роками випуску, акторами та іншими параметрами.

IMDb підтримує оцінки, рецензії, рекомендації, список перегляду,

персональні списки, сторінки акторів і режисерів, а також тематичні добірки контенту. Водночас інтерфейс сервісу місцями виглядає перевантаженим. На сторінці одного фільму може одночасно розміщуватися багато блоків: трейлери, акторський склад, рекомендації, цікаві факти, оцінки, медіаматеріали та інші секції. Через це користувачеві складніше зосередитися на основній інформації.

Особливо це помітно під час використання платформи на мобільних пристроях. Формально адаптивний інтерфейс присутній, однак навігація не завжди є зручною при швидкій роботі з каталогом. Частина елементів займає занадто багато простору, а окремі блоки виглядають перевантаженими навіть на екранах ПК. Ще однією проблемою IMDb є обмежена соціальна взаємодія. Оцінки та рецензії реалізовані достатньо добре, але взаємодія між користувачами не є основним напрямом розвитку сервісу.

Letterboxd використовує інший підхід. Якщо IMDb орієнтований переважно на довідкову інформацію про контент, то Letterboxd більше сфокусований на користувацькій активності, персональних списках, оцінках, рецензіях та соціальній взаємодії. Особливості організації користувацьких списків, оцінок і рецензій Letterboxd розглянуто за матеріалами офіційного вебресурсу платформи [7].

Основу платформи складають оцінки, рецензії, обране, список перегляду, користувацькі списки, рекомендації та активність користувачів. Користувач може вести власний список переглянутих фільмів, створювати добірки, вподобувати рецензії інших людей, підписуватися на профілі та відстежувати активність друзів. Через це Letterboxd більше нагадує соціальну платформу для любителів кіно.

Однією з найсильніших сторін Letterboxd є зручність інтерфейсу. Він виглядає легшим і менш перевантаженим, ніж інтерфейс IMDb. Більшість дій виконується швидко: користувач може поставити оцінку, додати фільм до списку перегляду або переглянути рекомендації за короткий час.

Сервіс також добре реалізує персоналізацію. Рекомендації формуються з урахуванням оцінок, вподобаних фільмів, активності користувача та створених списків. Проте Letterboxd має і свої обмеження. Платформа більше орієнтована саме на фільми, а робота із серіалами реалізована слабше. Крім того, метадані тут

менш деталізовані, ніж у IMDb або TMDB. Система пошуку та фільтрації також є менш гнучкою, що може ускладнювати роботу з великим каталогом.

TMDB, або The Movie Database, займає окреме місце серед платформ для роботи з медіаконтентом. Головною особливістю сервісу є наявність програмного інтерфейсу для отримання метаданих про фільми та серіали. Можливості TMDB як джерела метаданих про фільми та серіали визначено за матеріалами офіційного вебресурсу сервісу [8].

TMDB активно використовується сторонніми вебплатформами як джерело інформації про фільми та серіали. Через API можна отримувати назви, жанри, постери, фонові зображення, дати виходу, оцінки, інформацію про акторський склад, рекомендації та інші метадані. Саме ця особливість зробила TMDB важливою частиною вебплатформи Spokuta. Інтеграція TMDB API дозволяє автоматизувати роботу з каталогом та зменшити обсяг ручного заповнення даних.

У порівнянні з IMDb інтерфейс TMDB виглядає більш сучасним і візуально легким. Платформа використовує адаптивні сітки, великі зображення попереднього перегляду та зручне компонування елементів. Робота з каталогом виглядає швидшою, а навігація – простішою. Важливо також, що TMDB однаково підтримує як фільми, так і серіали.

Водночас соціальні функції TMDB реалізовані слабше. Оцінки та рецензії присутні, однак вони не є основною частиною сервісу. TMDB більше орієнтований на зберігання метаданих, роботу з каталогом та надання API для сторонніх застосунків.

Під час аналізу IMDb, Letterboxd та TMDB було виявлено спільні проблеми, характерні для великих платформ роботи з медіаконтентом. Однією з них є перевантаженість інтерфейсу. Частина сервісів містить занадто багато інформації на одній сторінці, через що навігація стає складнішою, особливо на мобільних пристроях.

Також проблемою є надмірна кількість переходів між сторінками під час виконання простих дій. Додавання контенту до списку перегляду, виставлення оцінки, відкриття рецензій або застосування фільтрів у деяких випадках

потребують повного перезавантаження сторінки чи переходу до іншого розділу. Це знижує швидкість взаємодії з каталогом.

Окрему проблему становить персоналізація. На деяких платформах рекомендації є занадто загальними та недостатньо враховують активність конкретного користувача. Через це система рекомендацій може втрачати практичну користь при регулярному використанні.

Для платформ із великим каталогом медіаконтенту важливою є автоматизація роботи з метаданими. Якщо вся інформація додається вручну, підтримка каталогу потребує значних витрат часу. Саме тому використання зовнішніх API є важливим для сучасних вебплатформ. Вони дають змогу автоматично отримувати назви фільмів, жанри, постери, роки випуску, оцінки, інформацію про акторський склад та рекомендації.

У вебплатформі Spokuta для цього використовується TMDB API. Через API система отримує метадані про фільми та серіали, що спрощує підтримку каталогу та роботу адміністративної панелі. Інтеграція зовнішнього API допомагає пришвидшити додавання нового контенту, зменшити кількість ручної роботи, підтримувати актуальність даних та уникати дублювання інформації.

Аналіз існуючих платформ показав, що жоден із розглянутих сервісів не поєднує всі потрібні можливості без компромісів. IMDb має великий каталог і деталізовані метадані, але його інтерфейс може бути перевантаженим. Letterboxd краще реалізує персональні списки та соціальну взаємодію, але слабше працює із серіалами та деталізованими метаданими. TMDB має потужний API і сучасний інтерфейс, проте менш орієнтований на користувацьку взаємодію.

Саме це стало основою для створення вебплатформи Spokuta. Під час проєктування основна увага приділялася швидкій роботі з каталогом, асинхронній взаємодії, базовим рекомендаціям, обраному, списку перегляду, адаптивному інтерфейсу та зменшенню кількості повних перезавантажень сторінки.

У Spokuta використовується асинхронна логіка роботи, тому оцінки, рецензії, пошук, фільтрація, обране та список перегляду можуть оновлюватися без повного перезавантаження сторінки. Це робить роботу з каталогом швидшою та зручнішою.

Також платформа поєднує локальну систему оцінок і рецензій з інтеграцією TMDB API, що дозволяє одночасно підтримувати користувацьку взаємодію та автоматизовану роботу з метаданими. Порівняльну характеристику розглянутих платформ наведено у таблиці 1.1.

Таблиця 1.1 – Порівняльна характеристика платформ для роботи з фільмами та серіалами

Критерій	IMDb	Letterboxd	TMDB	Spokuta
Каталог фільмів	Так	Так	Так	Так
Каталог серіалів	Так	Частково	Так	Так
Оцінки та рецензії	Так	Так	Частково	Так
Обране та список перегляду	Так	Так	Частково	Так
Рекомендації	Так	Так	Так	Базові
Соціальна взаємодія	Обмежена	Висока	Мінімальна	Середня
Інтеграція API	Часткова	Обмежена	Повноцінна	TMDB API
Адаптивний інтерфейс	Середній	Хороший	Хороший	Так
Пошук і фільтрація	Розширені	Середні	Швидкі	Асинхронні
Персоналізація	Середня	Висока	Низька	Базова
Адміністративне керування	Обмежене	Мінімальне	Часткове	Так

Порівняння показує, що існуючі сервіси мають різний акцент: IMDb – метадані та масштаб каталогу, Letterboxd – персональні списки й соціальна взаємодія, TMDB – API та робота з метаданими. Під час розробки Spokuta основна увага приділялася поєднанню цих підходів без перевантаження інтерфейсу та ускладнення логіки взаємодії з користувачем.

1.2 Аналіз технологій та засобів розробки

Під час розробки вебплатформи Spokuta увага приділялася не лише набору функцій, а й швидкості, стабільності та зручності роботи з каталогом. Для сервісу такого типу важливими є швидкий пошук і фільтрація, асинхронне оновлення даних, адаптивний інтерфейс та зручна взаємодія з великим обсягом медіаконтенту.

Для реалізації проєкту було обрано PHP, MySQL, JavaScript, AJAX та TMDB API. Такий набір технологій дозволяє поєднати серверну логіку, базу даних, динамічну роботу інтерфейсу та інтеграцію зовнішнього джерела метаданих без надмірного ускладнення структури проєкту.

Під час вибору інструментів враховувалися швидкість розробки, простота підтримки, робота з великим каталогом, підтримка асинхронних запитів, інтеграція зовнішнього API, адаптація до мобільних пристроїв, реалізація рекомендацій, обраного, списку перегляду та адміністративного керування.

PHP у Spokuta використовується як основа серверної частини. Вибір PHP обґрунтований його можливостями для обробки HTTP-запитів, роботи із сесіями, взаємодії з базами даних і реалізації серверної логіки вебзастосунків [9]. Через PHP реалізовано взаємодію між інтерфейсом користувача, базою даних та TMDB API.

Для такого вебпроєкту PHP є зручним рішенням завдяки простій інтеграції з MySQL, підтримці сесій та роботі з HTTP-запитами. Це дозволило реалізувати основний функціонал без використання великих серверних фреймворків або складної багаторівневої архітектури.

У Spokuta PHP відповідає за авторизацію та автентифікацію, роботу із сесіями, CRUD-операції, взаємодію з базою даних, оцінки й рецензії, обране, список перегляду, рекомендації, адміністративну панель, обробку асинхронних запитів та інтеграцію TMDB API.

Через PHP працюють сторінки каталогу, профілю користувача та адміністративної панелі. Серверна частина також обробляє асинхронні запити для оцінок, обраного, списку перегляду, пошуку та фільтрації. Окремо використовується логіка сесій: користувач може залишатися авторизованим після оновлення сторінки, а перевірка ролей користувача й адміністратора потрібна для обмеження доступу до адміністративної частини.

MySQL використовується як основа бази даних вебплатформи. У ній зберігаються користувачі, каталог фільмів і серіалів, оцінки з текстовими рецензіями, обране, список перегляду, метадані контенту та дані, потрібні для формування рекомендацій. Використання MySQL дає змогу організувати

реляційне зберігання даних, застосовувати SQL-запити, індекси, зв’язки між таблицями та обмеження цілісності [10].

Для Spokuta база даних є центральною частиною системи, оскільки більшість функцій пов’язана з каталогом контенту та діями користувачів. У MySQL зберігаються дані про користувачів, фільми й серіали, оцінки, рецензії, обране, список перегляду та метадані контенту. Також через MySQL реалізовано зв’язки між таблицями, пошук, сортування і фільтрацію.

Під час вибору СКБД було важливо, щоб система стабільно працювала з великою кількістю записів і підтримувала швидкий пошук та фільтрацію. MySQL підходить для таких задач завдяки підтримці SQL-запитів і реляційній структурі даних. Ще однією перевагою стала проста інтеграція з PHP, що дозволило реалізувати серверну логіку без додаткових ORM або складних проміжних програмних рішень.

Для захисту даних використовуються підготовлені SQL-запити та перевірка ролей користувачів. Це зменшує ризик SQL-ін’єкцій і несанкціонованого доступу до адміністративної панелі.

JavaScript у Spokuta використовується для логіки взаємодії та динамічної роботи інтерфейсу. Значна частина взаємодії побудована на стороні клієнта без повного перезавантаження сторінки. JavaScript забезпечує реалізацію клієнтської логіки, обробку подій користувача та динамічне оновлення елементів вебсторінки [11].

У проєкті JavaScript відповідає за динамічне оновлення інтерфейсу, асинхронну взаємодію, пошук і фільтрацію, роботу з обраним і списком перегляду, оновлення оцінок, адаптивну навігацію, модальні вікна, спливаючі повідомлення та логіку взаємодії адміністративної панелі.

Через JavaScript реалізовано швидке оновлення окремих елементів сторінки без повторного завантаження всього контенту. Це особливо важливо під час роботи з каталогом, коли користувач часто взаємодіє з оцінками, рецензіями, обраним або списком перегляду. Також JavaScript використовується для мобільної навігації та адаптивної взаємодії з інтерфейсом.

AJAX є однією з ключових технологій у Spokuta, оскільки саме через нього реалізовано більшість асинхронних оновлень без повного перезавантаження сторінки. Для виконання асинхронних HTTP-запитів у клієнтській частині може використовуватися Fetch API, що дозволяє обмінюватися даними із сервером без повного перезавантаження сторінки [12].

Для платформ із великим каталогом це особливо важливо, оскільки постійне перезавантаження сторінки погіршує зручність взаємодії та уповільнює навігацію. У Spokuta AJAX використовується для пошуку й фільтрації, оновлення оцінок, роботи з обраним і списком перегляду, рекомендацій, динамічного завантаження контенту, адміністративних дій та часткового оновлення сторінок.

Наприклад, під час додавання фільму до списку перегляду користувачу не потрібно повторно відкривати сторінку. Дані оновлюються асинхронно, а інтерфейс одразу показує зміни. Схожа логіка використовується для оцінок, рецензій та частини рекомендацій. Це робить роботу з каталогом швидшою і зручнішою.

TMDB API використовується у Spokuta як зовнішнє джерело метаданих для фільмів і серіалів. Документація TMDB API описує механізми отримання інформації про фільми, серіали, постери, жанри, рейтинги та інші метадані, що використовуються під час наповнення каталогу [13].

Без зовнішнього API підтримка великого каталогу вимагала б ручного додавання назв, описів, постерів, жанрів, років випуску та іншої інформації. Для вебплатформи з великою кількістю контенту це створювало б значний обсяг ручної роботи.

Через TMDB API система автоматично отримує назви фільмів і серіалів, постери, жанри, описи, оцінки, роки випуску, фонові зображення та інші метадані. Інтеграція API спрощує роботу адміністративної панелі: адміністратор може додати новий контент через TMDB ID або пошук за назвою, після чого основні поля каталогу заповнюються автоматично.

TMDB API також допомагає підтримувати каталог в актуальному стані та зменшує кількість дублювання інформації. Окремою перевагою є підтримка

фільмів і серіалів у межах єдиної API-структури, що дозволяє використовувати спільну логіку роботи для різних типів контенту.

HTML і CSS використовуються для побудови структури та оформлення сторінок вебплатформи. HTML визначає основні елементи сторінки, зокрема форми, таблиці, навігацію, картки контенту та блоки інтерфейсу, а CSS відповідає за стилізацію, компонування елементів і адаптацію сторінок до різних розмірів екрана [14, 15].

Для передавання відповідей між серверною частиною та клієнтським інтерфейсом використовується формат JSON. Він зручний для обміну структурованими даними у вебзастосунках, оскільки дозволяє передавати статус операції, повідомлення, оновлені рейтинги, списки та інші дані у компактному форматі [16].

Під час вибору технологій також враховувалася адаптація до мобільних пристроїв. Багато користувачів працюють із подібними сервісами саме через смартфони, тому адаптивний інтерфейс був важливою частиною проєкту.

Адаптивність інтерфейсу реалізовано через гнучку навігацію, адаптивні сітки та перебудову окремих елементів сторінки залежно від розміру екрана. JavaScript і AJAX дозволили зробити взаємодію швидшою навіть на мобільних пристроях, оскільки частина контенту оновлюється без повного перезавантаження сторінки. Це особливо помітно під час роботи з каталогом, рекомендаціями, обраним і списком перегляду.

Розробка та локальне тестування вебплатформи виконувалися у середовищі Visual Studio Code з використанням локального серверного середовища XAMPP. Для роботи серверної частини під час локального тестування використовувався Apache HTTP Server. Локальне серверне середовище XAMPP застосовувалося для запуску проєкту, Visual Studio Code – для редагування коду, а phpMyAdmin – для адміністрування бази даних [17–20].

У таблиці 1.2 наведено технології та засоби розробки, використані під час створення вебплатформи Spokuta.

Таблиця 1.2 – Технології та засоби розробки вебплатформи Spokuta

Технологія	Призначення	Роль у Spokuta
PHP	Серверна розробка	Реалізація серверної логіки
MySQL	База даних	Зберігання користувачів, фільмів, оцінок, рецензій і персональних списків
JavaScript	Логіка інтерфейсу	Динамічне оновлення елементів сторінки
AJAX	Асинхронна взаємодія	Оновлення даних без повного перезавантаження сторінки
TMDB API	Отримання метаданих	Автоматичне наповнення каталогу
HTML/CSS	Структура та стилізація	Структура сторінок і адаптивний інтерфейс

Обраний набір технологій дозволив реалізувати Spokuta як вебплатформу зі швидкою взаємодією, адаптивним інтерфейсом та зручною роботою з каталогом. PHP і MySQL забезпечують серверну логіку та роботу бази даних, JavaScript і AJAX відповідають за асинхронну взаємодію без перезавантаження сторінки, а TMDB API автоматизує роботу з метаданими та спрощує підтримку каталогу.

У результаті було реалізовано платформу з пошуком і фільтрацією, рекомендаціями, обраним, списком перегляду, оцінками, рецензіями та адміністративною панеллю без надмірного ускладнення структури проєкту або перевантаження інтерфейсу.

1.3 Постановка задачі розробки вебплатформи Spokuta

Під час аналізу існуючих платформ для роботи з фільмами та серіалами було визначено, що більшість подібних сервісів мають широкий набір можливостей, але не завжди залишаються зручними при щоденному використанні. Частина платформ перевантажена інтерфейсом і великою кількістю другорядних елементів, через що робота з каталогом стає менш комфортною. Інші сервіси добре підходять як джерело метаданих, але слабше реалізують персоналізацію або швидку взаємодію з контентом.

Окремою проблемою є логіка навігації у великих каталогах. Коли користувач

постійно переходить між сторінками, фільтрами, списками, оцінками та рецензіями, навіть невеликі затримки або повне перезавантаження сторінки починають помітно впливати на загальне враження від сервісу. Це особливо відчувається на мобільних пристроях, де перевантажений інтерфейс або складна структура навігації швидко роблять платформу незручною.

Ще одна проблема стосується роботи з рекомендаціями та особистими списками. На багатьох сервісах рекомендації виглядають відокремленими від реальної активності користувача або працюють недостатньо гнучко. Обране та список перегляду також часто сприймаються як прості списки без повної інтеграції з каталогом та історією переглядів. Через це користувачеві доводиться витрачати більше часу на пошук контенту або ручне керування власною бібліотекою.

Саме це стало однією з причин створення Spokuta. Ідея вебплатформи полягає не лише у формуванні каталогу фільмів і серіалів, а й у створенні швидкого та зручного середовища для роботи з медіаконтентом. Основний акцент зроблено на комфортній взаємодії з каталогом, асинхронному оновленні даних, адаптивному інтерфейсі та базовій персоналізації.

Під час формування задачі розробки було визначено основні напрями, які повинна підтримувати платформа:

- швидкий пошук і фільтрацію контенту;
- зручну систему оцінок і рецензій;
- роботу з обраним і списком перегляду;
- базові рекомендації схожого контенту;
- адаптивний інтерфейс для мобільних пристроїв;
- асинхронну взаємодію без постійного перезавантаження сторінки;
- адміністративне керування каталогом;
- інтеграцію TMDB API для автоматизації роботи з метаданими.

Одним із ключових завдань було створення каталогу, який залишався б зручним навіть за великої кількості контенту. Для цього необхідно було реалізувати систему пошуку та фільтрації з підтримкою сортування й динамічного оновлення результатів. Користувач не повинен постійно переходити між окремими

сторінками лише для зміни фільтра або перегляду результатів пошуку.

Також важливою задачею стала реалізація системи оцінок і рецензій. У Spokuta оцінки, рецензії, обране, список перегляду та рекомендації пов’язані між собою, тому активність користувача може впливати на подальшу роботу з каталогом і формування базових рекомендацій.

Окремо враховувалась необхідність адаптивного інтерфейсу. Значна частина користувачів працює з подібними сервісами через смартфони або планшети, тому інтерфейс повинен коректно адаптуватися до різних розмірів екрана. При цьому адаптація не повинна обмежуватися лише зміною ширини блоків. Потрібно було забезпечити зручну мобільну навігацію, компактні картки каталогу та швидку взаємодію з елементами інтерфейсу навіть на невеликих екранах.

Ще однією задачею стала мінімізація повних перезавантажень сторінок. Постійне перезавантаження негативно впливає на швидкість роботи з каталогом, особливо коли користувач часто взаємодіє з оцінками, рецензіями, обраним або списком перегляду. Саме тому значна частина логіки у Spokuta працює через асинхронні запити. Це дозволяє оновлювати окремі елементи сторінки без повторного завантаження всього інтерфейсу.

Асинхронна взаємодія використовується для таких функцій:

- пошук і фільтрація;
- оцінки та рецензії;
- рекомендації;
- обране та список перегляду;
- адміністративна панель;
- частина логіки навігації.

Під час розробки окремо враховувалась проблема ручного наповнення каталогу. Якщо додавати метадані для фільмів і серіалів вручну, підтримка великої бібліотеки контенту швидко стає незручною та займає багато часу. Для вирішення цієї проблеми у Spokuta використовується TMDB API.

Інтеграція TMDB API дозволяє автоматично отримувати:

- назви фільмів і серіалів;

- описи;
- жанри;
- постери;
- оцінки;
- роки випуску;
- інші метадані.

Завдяки цьому адміністратору не потрібно вручну заповнювати велику кількість полів під час додавання нового контенту. Частина інформації автоматично імпортується через API, після чого зберігається у локальній базі даних.

Для роботи вебплатформи також було реалізовано систему ролей користувачів. У Spokuta передбачено три основні ролі:

- гість;
- авторизований користувач;
- адміністратор.

Гість має доступ лише до базового функціоналу. Неавторизований користувач може переглядати каталог, використовувати пошук і фільтрацію та відкривати сторінки фільмів і серіалів. Водночас оцінки, рецензії, обране, список перегляду та рекомендації доступні лише після авторизації.

Авторизований користувач отримує доступ до персоналізованих функцій платформи. Він може виставляти оцінки, залишати рецензії, формувати обране й список перегляду, переглядати рекомендації та працювати з власним профілем.

Адміністратор має розширений доступ до керування контентом і користувачами. Через адміністративну панель адміністратор може:

- додавати нові фільми й серіали;
- редагувати контент;
- модерувати рецензії;
- працювати з каталогом;
- використовувати імпорт даних із TMDB;

– переглядати статистичні дані.

Наявність адміністративної панелі є важливою частиною постановки задачі, оскільки без окремого адміністративного інтерфейсу підтримка каталогу швидко стала б незручною. Особливо це стосується великих обсягів контенту та роботи з метаданими TMDB.

Під час формування логіки вебплатформи також враховувалась взаємодія між ролями та каталогом. Користувачі повинні мати швидкий доступ до основних функцій без складної структури навігації або перевантаженого інтерфейсу. Через це у Spokuta зроблено акцент на компактному адаптивному інтерфейсі, простій навігації та асинхронному оновленні даних.

Отже, основна задача розробки полягає у створенні вебплатформи, яка поєднує каталог фільмів і серіалів, базову персоналізацію, оцінки, рецензії, обране, список перегляду, рекомендації та адміністративне керування в єдиному середовищі без перевантаження інтерфейсу та зайвих переходів між сторінками.

Надалі логіка взаємодії користувачів із платформою, ролі доступу та основні сценарії роботи будуть детальніше показані за допомогою діаграми варіантів використання та опису сценаріїв використання системи.

1.4 Основні завдання

Для досягнення поставленої мети у межах дипломної роботи необхідно виконати такі основні завдання:

– проаналізувати сучасні платформи для роботи з фільмами та серіалами;
– визначити функціональні та нефункціональні вимоги до вебплатформи;
– розробити специфікацію вимог до програмного забезпечення;
– змоделювати основні сценарії взаємодії користувачів із системою;
– спроєктувати архітектуру вебплатформи та структуру бази даних;
– реалізувати каталог фільмів і серіалів, систему оцінювання, рецензії,

обране та список перегляду;

- інтегрувати TMDB API для отримання метаданих про фільми й серіали;
- реалізувати адміністративну панель для керування контентом, користувачами та рецензіями;
- забезпечити адаптивність інтерфейсу для роботи з різних пристроїв;
- провести ручне тестування основних функцій вебплатформи.

Визначені завдання охоплюють повний цикл розробки вебплатформи Spokuta: від аналізу предметної області та формування вимог до проєктування архітектури, реалізації основного функціоналу, адаптації інтерфейсу та перевірки працездатності системи.

1.5 Висновки до розділу

Під час аналізу існуючих платформ для роботи з фільмами та серіалами було визначено, що навіть популярні сервіси мають певні обмеження. Частина з них орієнтується переважно на великі бази метаданих, інші – на соціальну активність користувачів або систему оцінок і рецензій. При цьому не всі платформи однаково добре реалізують персоналізацію, швидку навігацію та взаємодію з каталогом без постійного перезавантаження сторінок.

Окремо було виявлено проблеми перевантаженого інтерфейсу, складної логіки навігації та недостатньо зручної адаптації до мобільних пристроїв. Для платформ із великою кількістю контенту це відчувається особливо сильно, оскільки користувач постійно працює з пошуком, фільтрацією, обраним, списком перегляду, оцінками та рецензіями. Навіть невеликі затримки або надлишок елементів інтерфейсу можуть впливати на комфорт роботи з каталогом.

Порівняння IMDb, Letterboxd та TMDB дозволило визначити сильні сторони кожного підходу. IMDb є прикладом великої інформаційної бази з деталізованими метаданими, Letterboxd демонструє переваги персональних списків і соціальної

взаємодії, а TMDB є корисним джерелом метаданих завдяки наявності API. Водночас кожна з цих платформ має певні компроміси, пов’язані з інтерфейсом, персоналізацією, роботою із серіалами або рівнем взаємодії користувача з каталогом.

Саме це стало основою для формування задачі розробки Spokuta. Платформа створювалась як компактне та швидке середовище для роботи з фільмами й серіалами, де акцент зроблено на зручній взаємодії з контентом, адаптивному інтерфейсі та базовій персоналізації. Важливою частиною стала асинхронна взаємодія, оскільки значна частина дій користувача пов’язана з оновленням даних без повного перезавантаження сторінки.

Під час аналізу технологій було визначено, що PHP, MySQL, JavaScript, AJAX та TMDB API добре підходять для вебпроєкту такого типу. PHP забезпечує серверну логіку та взаємодію з базою даних, MySQL використовується для збереження користувачів, каталогу, оцінок, рецензій і персональних списків, а JavaScript разом із AJAX відповідають за динамічне оновлення інтерфейсу та швидшу роботу з каталогом.

Інтеграція TMDB API дозволила автоматизувати отримання метаданих і спростити наповнення каталогу. Це особливо важливо для платформ, де кількість контенту постійно збільшується. Адміністративна панель також стала необхідною частиною структури проєкту, оскільки без окремого адміністративного інтерфейсу підтримка каталогу, модерація рецензій та робота з контентом були б значно складнішими.

У результаті першого розділу було сформовано основні вимоги до вебплатформи Spokuta, визначено задачі розробки та обґрунтовано вибір технологій. Проведений аналіз став основою для подальшого моделювання сценаріїв взаємодії, проєктування архітектури, структури бази даних і розподілу ролей користувачів.

2 РОЗРОБКА ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Аналіз та формування вимог до системи

Під час розробки вебплатформи Spokuta формування вимог до системи базувалося не лише на стандартному наборі функцій для каталогу фільмів і серіалів, а й на проблемах, які були розглянуті у попередньому розділі. Основна увага приділялася швидкості роботи з каталогом, зручності навігації, базовій персоналізації та зменшенню кількості зайвих дій під час роботи з контентом.

Через велику кількість медіаконтенту користувачу часто складно швидко знайти потрібний фільм або серіал. Особливо це помітно у сервісах із перевантаженим інтерфейсом або повільною логікою роботи сторінок. Саме тому для Spokuta потрібно було реалізувати не просто каталог, а вебсервіс зі швидким пошуком, асинхронною взаємодією та персональними функціями.

Вимоги до проєкту формувалися з урахуванням трьох основних напрямів:

- роботи з каталогом;
- взаємодії користувача з контентом;
- підтримки адміністративної частини.

Функціональна частина вебплатформи повинна забезпечувати повноцінну роботу з фільмами та серіалами в межах єдиного каталогу. Користувач має отримувати швидкий доступ до контенту, а також можливість взаємодіяти з ним без складної логіки навігації або постійного перезавантаження сторінок.

До основних функціональних вимог належать:

- реєстрація та авторизація користувачів;
- перегляд каталогу фільмів і серіалів;
- пошук і фільтрація контенту;
- оцінки та рецензії;
- обране та список перегляду;
- базові рекомендації схожого контенту;
- робота з профілем користувача;

– інтеграція TMDB API;
– адміністративна панель;
– асинхронне оновлення даних.

Авторизація повинна підтримувати створення облікового запису, вхід до системи, вихід із системи та збереження сесії користувача. Після входу в систему користувач отримує доступ до персоналізованих функцій платформи, які недоступні для неавторизованого користувача.

У Spokuta передбачено три основні ролі:

– гість;
– авторизований користувач;
– адміністратор.

Гість має доступ лише до базових функцій платформи. Неавторизований користувач може відкривати сторінки фільмів і серіалів, використовувати пошук і фільтрацію та переглядати каталог без можливості взаємодії з персональними функціями.

Авторизований користувач отримує доступ до оцінок, рецензій, обраного, списку перегляду, рекомендацій та власного профілю. Після авторизації користувач може виставляти оцінки, залишати рецензії, додавати контент у персональні списки та отримувати рекомендації на основі власної активності.

Адміністратор має розширені права доступу. Через адміністративну панель адміністратор повинен мати можливість:

– додавати новий контент;
– редагувати інформацію про фільми та серіали;
– видаляти контент;
– модерувати рецензії;
– працювати з імпортом через TMDB API;
– керувати каталогом;
– переглядати статистичні дані.

Окремий акцент робився на швидкості роботи пошуку та фільтрації. Для

платформи з великою кількістю контенту звичайного пошуку недостатньо. Користувач повинен мати можливість швидко сортувати каталог, працювати з жанрами, роками випуску, оцінками та типом контенту без переходу між окремими сторінками.

Система пошуку та фільтрації повинна:

– швидко обробляти запити;
– підтримувати динамічне оновлення результатів;
– працювати без повного перезавантаження сторінки;
– коректно працювати як на ПК, так і на мобільних пристроях;
– не перевантажувати інтерфейс великою кількістю елементів.

Для цього у платформі використовується асинхронна взаємодія. Вона стала важливою частиною логіки платформи, оскільки більшість дій користувача пов’язана з невеликими змінами всередині каталогу.

Через асинхронні запити повинні оновлюватися:

– оцінки;
– обране та список перегляду;
– рекомендації;
– результати пошуку та фільтрації;
– частина логіки навігації;
– окремі елементи адміністративної панелі.

Такий підхід дозволяє зменшити кількість повних перезавантажень сторінок і зробити взаємодію з каталогом швидшою. Особливо це помітно під час роботи з мобільних пристроїв, де навіть невеликі затримки інтерфейсу швидко стають відчутними.

Окремо формувалися вимоги до логіки персоналізації. Платформа не повинна працювати лише як статичний каталог контенту. Рекомендації, обране, список перегляду, оцінки та рецензії мають бути пов’язані між собою і впливати на подальшу взаємодію користувача з платформою.

Базові рекомендації повинні формуватися з урахуванням:

– виставлених оцінок;

– обраного та списку перегляду;
– історії активності;
– типу контенту;
– взаємодії з каталогом.

Також під час формування вимог враховувалася інтеграція TMDB API. Підтримка великого каталогу без автоматизації роботи з метаданими створює значний обсяг ручної роботи для адміністратора. Саме тому інтеграція TMDB стала важливою частиною структури платформи.

Через TMDB API система повинна отримувати:

– назви фільмів і серіалів;
– жанри;
– постери;
– описи;
– оцінки;
– роки випуску;
– фонові зображення;
– інші метадані.

Інтеграція API також спрощує роботу адміністративної панелі та пришвидшує додавання нового контенту до каталогу.

Крім функціональних вимог, під час проєктування враховувалися нефункціональні характеристики вебплатформи. Для сервісу такого типу важливими є швидкість роботи, стабільність інтерфейсу, адаптивність, зручність навігації та базовий захист даних користувачів. Узагальнені функціональні та нефункціональні вимоги до системи наведено у таблиці 2.1.

Таблиця 2.1 – Функціональні та нефункціональні вимоги системи

Категорія	Вимога
Функціональні	Реєстрація та авторизація
Функціональні	Пошук і фільтрація
Функціональні	Оцінки та рецензії
Функціональні	Обране та список перегляду
Функціональні	Базові рекомендації

Продовження таблиці 2.1

Категорія	Вимога
Функціональні	Інтеграція TMDB API
Функціональні	Адміністративна панель
Функціональні	Асинхронна взаємодія
Нефункціональні	Адаптивний інтерфейс
Нефункціональні	Швидке оновлення інтерфейсу
Нефункціональні	Підтримка мобільних пристроїв
Нефункціональні	Стабільна робота каталогу
Нефункціональні	Зручність використання
Нефункціональні	Безпечна авторизація
Нефункціональні	Захист даних користувачів

Під час проєктування окремо враховувалася продуктивність системи. Пошук, фільтрація, рекомендації та оцінювання повинні працювати швидко навіть при збільшенні кількості контенту та активності користувачів.

Адаптивний інтерфейс також став важливою частиною вимог до системи. Інтерфейс повинен коректно адаптуватися до різних розмірів екрана та не створювати проблем із навігацією або виходом елементів за межі сторінки.

Для адаптації до мобільних пристроїв враховувалися:

- адаптивні сітки;
- адаптивна навігація;
- компактні картки каталогу;
- адаптивне меню;
- зручне розташування кнопок взаємодії;
- швидке завантаження сторінок.

Під час формування вимог до інтерфейсу також потрібно було уникнути його перевантаження. Користувач не повинен постійно переходити між окремими сторінками або втрачати контекст під час роботи з каталогом.

Окрему увагу приділено безпеці вебплатформи. Оскільки Spokuta підтримує облікові записи користувачів, рецензії та адміністративний функціонал, потрібно було реалізувати базові механізми захисту даних і перевірки доступу.

Система повинна підтримувати:

- хешування паролів;

- авторизацію через сесії;
- перевірку ролей користувачів;
- підготовлені SQL–запити;
- перевірку вхідних даних;
- обмеження доступу до адміністративної панелі.

Під час формування вимог до безпеки враховувалися базові рекомендації щодо захисту від SQL–ін’єкцій та організації автентифікації користувачів. Для цього в системі передбачено використання підготовлених SQL–запитів, перевірки вхідних даних, хешування паролів, сесій і перевірки ролей доступу [21, 22].

Також враховувалася безпека роботи з TMDB API. Токен API не повинен зберігатися у відкритому вигляді або передаватися через клієнтську частину вебплатформи.

Під час формування вимог до Spokuta акцент робився не лише на кількості функцій, а саме на логіці взаємодії користувача з каталогом. Через це у проєкті поєднуються асинхронна взаємодія, адаптивний інтерфейс, базові рекомендації, обране, список перегляду, оцінки та рецензії без перевантаження інтерфейсу й зайвих переходів між сторінками.

Сформовані вимоги стали основою для подальшого опису специфікації вимог до програмного забезпечення та основних сценаріїв роботи вебплатформи Spokuta.

2.2 Розробка специфікації вимог SRS

Після формування основних вимог до вебплатформи наступним етапом стала підготовка специфікації вимог до програмного забезпечення. Її основна задача полягає у фіксації логіки роботи Spokuta, описі ролей користувачів, доступного функціоналу та сценаріїв взаємодії з платформою до переходу до UML–моделювання та детального проєктування.

Специфікація вимог для Spokuta формувалася не як повний формальний документ міжнародного стандарту, а як структурований опис майбутньої вебплатформи з урахуванням її реального функціоналу. Основний акцент зроблено на логіці роботи каталогу, взаємодії користувача з контентом, системі ролей та підтримці асинхронної взаємодії без постійного перезавантаження сторінок.

Під час підготовки специфікації враховувалися:

- структура каталогу фільмів і серіалів;
- пошук і фільтрація контенту;
- оцінки та рецензії;
- обране та список перегляду;
- базові рекомендації схожого контенту;
- авторизація та автентифікація користувачів;
- адміністративна панель;
- інтеграція TMDB API;
- адаптивний інтерфейс.

Специфікація також описує основні сценарії використання вебплатформи та межі доступу для різних категорій користувачів. Це дозволяє ще на етапі проєктування визначити, які функції доступні гостю, які можливості отримує авторизований користувач та які інструменти повинні бути доступні адміністратору.

У Spokuta використовується три основні ролі користувачів:

- гість;
- авторизований користувач;
- адміністратор.

Окремо у специфікації враховується TMDB API як зовнішній сервіс, через який платформа отримує метадані фільмів і серіалів.

Гість взаємодіє лише з відкритою частиною каталогу. Для нього доступний перегляд сторінок контенту, пошук, фільтрація та навігація по каталогу без авторизації. При цьому персональні функції залишаються обмеженими.

Авторизований користувач після входу до системи отримує доступ до оцінок,

рецензій, обраного, списку перегляду, рекомендацій та власного профілю. Логіка доступу побудована так, щоб авторизований користувач міг працювати з контентом без зайвих переходів між сторінками. Частина дій виконується асинхронно, тому оцінки, рецензії або зміни у персональних списках оновлюються динамічно.

Адміністратор має розширений набір функцій. Через адміністративну панель адміністратор може:

- додавати контент;
- редагувати інформацію про фільми та серіали;
- видаляти записи;
- модерувати рецензії;
- працювати з імпортом даних через TMDB API;
- керувати каталогом і частиною користувацької активності.

Основних учасників системи та їхні функції наведено у таблиці 2.2.

Таблиця 2.2 – Основні учасники системи та їх функції

Учасник системи	Основні можливості
Гість	Перегляд каталогу, пошук і фільтрація, відкриття сторінок контенту
Авторизований користувач	Оцінки, рецензії, обране, список перегляду, рекомендації, профіль користувача
Адміністратор	Керування каталогом, адміністративна панель, модерація рецензій, імпорт даних із TMDB
TMDB API	Надання метаданих для фільмів і серіалів

У специфікації також описуються основні сценарії роботи вебплатформи. Найбільш важливим серед них є взаємодія користувача з каталогом контенту. Користувач повинен мати можливість швидко знаходити фільми або серіали, використовуючи пошук і фільтрацію без складної логіки навігації.

Пошук повинен підтримувати:

- назви контенту;
- жанри;
- тип контенту;

– сортування;
– фільтрацію;
– швидке оновлення результатів.

Окремо у специфікації описується сценарій авторизації. Після входу в обліковий запис користувач отримує доступ до персоналізованих функцій, а також до власної історії активності. Логіка доступу базується на ролях і авторизації через сесію.

Сценарії оцінювання та написання рецензій у специфікації розглядаються як одна з основних частин взаємодії з платформою. Авторизований користувач повинен мати можливість:

– виставляти оцінки;
– змінювати оцінки;
– залишати рецензії;
– переглядати рецензії інших користувачів;
– взаємодіяти з контентом без повного перезавантаження сторінки.

Через обране та список перегляду користувач формує персональні списки контенту. Це дозволяє швидше працювати з каталогом та зберігати фільми або серіали для подальшого перегляду.

Базові рекомендації у специфікації описуються як частина логіки персоналізації. Система рекомендацій повинна враховувати:

– оцінки;
– обране та список перегляду;
– історію взаємодії;
– тип контенту;
– активність користувача.

Окремий блок специфікації присвячений роботі адміністративної панелі. Адміністративна частина платформи повинна забезпечувати керування каталогом без прямої роботи з базою даних. Для цього у специфікації описуються сценарії:

– додавання контенту;

- редагування;
- видалення;
- перевірки рецензій;
- імпорту даних через TMDB API.

TMDB API у специфікації розглядається як зовнішній сервіс, який взаємодіє із серверною частиною платформи. Через API система отримує:

- назви;
- жанри;
- постери;
- описи;
- оцінки;
- рік випуску;
- інші метадані.

Завдяки цьому адміністратору не потрібно вручну заповнювати велику кількість полів під час додавання нового контенту.

Під час формування специфікації також враховувалася логіка доступу між ролями. Авторизований користувач успадковує базові можливості гостя, а адміністратор отримує доступ до керування платформою. Така структура пізніше використовується під час побудови діаграми варіантів використання.

На основі сформованої специфікації вимог була підготовлена діаграма варіантів використання вебплатформи Spokuta, яка детальніше розглядається у розділі 3 під час UML–моделювання програмного забезпечення.

У діаграмі використовуються зв’язки включення для сценаріїв, які залежать від інших дій користувача. Наприклад, частина адміністративних операцій пов’язана з отриманням даних через TMDB API, а функції авторизованого користувача залежать від авторизації та активної сесії.

Під час підготовки специфікації основна увага приділялася не формальному опису UML–нотації, а зрозумілій логіці роботи вебплатформи та поведінці користувачів. Саме специфікація стала базою для подальшого UML–моделювання, проєктування структури бази даних та опису архітектури Spokuta.

2.3 Висновки до розділу

У другому розділі було сформовано основні вимоги до вебплатформи Spokuta та визначено загальну логіку її роботи. Основна увага приділялася зручності взаємодії користувача з контентом, швидкості навігації, базовій персоналізації та роботі з різними ролями доступу.

Функціональні вимоги охоплюють пошук і фільтрацію контенту, оцінки та рецензії, обране, список перегляду, базові рекомендації, авторизацію користувачів, інтеграцію TMDB API та роботу адміністративної панелі. Нефункціональні вимоги пов’язані зі швидкодією, стабільністю, адаптивністю інтерфейсу, підтримкою мобільних пристроїв і асинхронним оновленням даних.

Під час підготовки специфікації вимог було описано структуру ролей користувачів і межі доступу між гостем, авторизованим користувачем та адміністратором. Це дозволило відокремити функції звичайного користувача від адміністративної частини та сформувати зрозумілу логіку роботи платформи.

Окремо було визначено роль TMDB API, який використовується для автоматизованого отримання метаданих про фільми й серіали та спрощення наповнення каталогу. Також було враховано потребу в адаптивному інтерфейсі, швидкій роботі каталогу й базовій персоналізації.

Діаграма варіантів використання, підготовлена на основі специфікації вимог, стала базою для подальшого UML–моделювання та опису сценаріїв взаємодії користувачів із вебплатформою Spokuta.

3 МОДЕЛЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Діаграми варіантів використання

Після формування специфікації вимог наступним етапом проєктування стала побудова UML–моделі вебплатформи Spokuta. Для опису взаємодії користувачів із системою використано діаграму варіантів використання, яка дає змогу показати основні сценарії роботи платформи, зв’язки між учасниками системи та межі доступу для різних ролей.

Для моделювання структури та поведінки вебплатформи використано нотацію UML, яка є поширеним стандартом опису варіантів використання, послідовностей взаємодії та компонентів програмної системи [23].

Діаграма варіантів використання у Spokuta застосовується не лише як формальна UML–діаграма, а як спосіб візуально показати структуру взаємодії користувача з каталогом, логіку авторизації, базової персоналізації та адміністративного керування. Це особливо важливо для вебплатформи, де частина функцій доступна всім користувачам, а частина – лише після авторизації або через адміністративну панель.

У UML–моделі Spokuta використовується чотири основні учасники системи:

- гість;
- авторизований користувач;
- адміністратор;
- TMDB API.

На рисунку 3.1 представлена діаграма варіантів використання вебплатформи Spokuta, яка демонструє основних учасників системи, їхню взаємодію з платформою та ключові сценарії роботи.

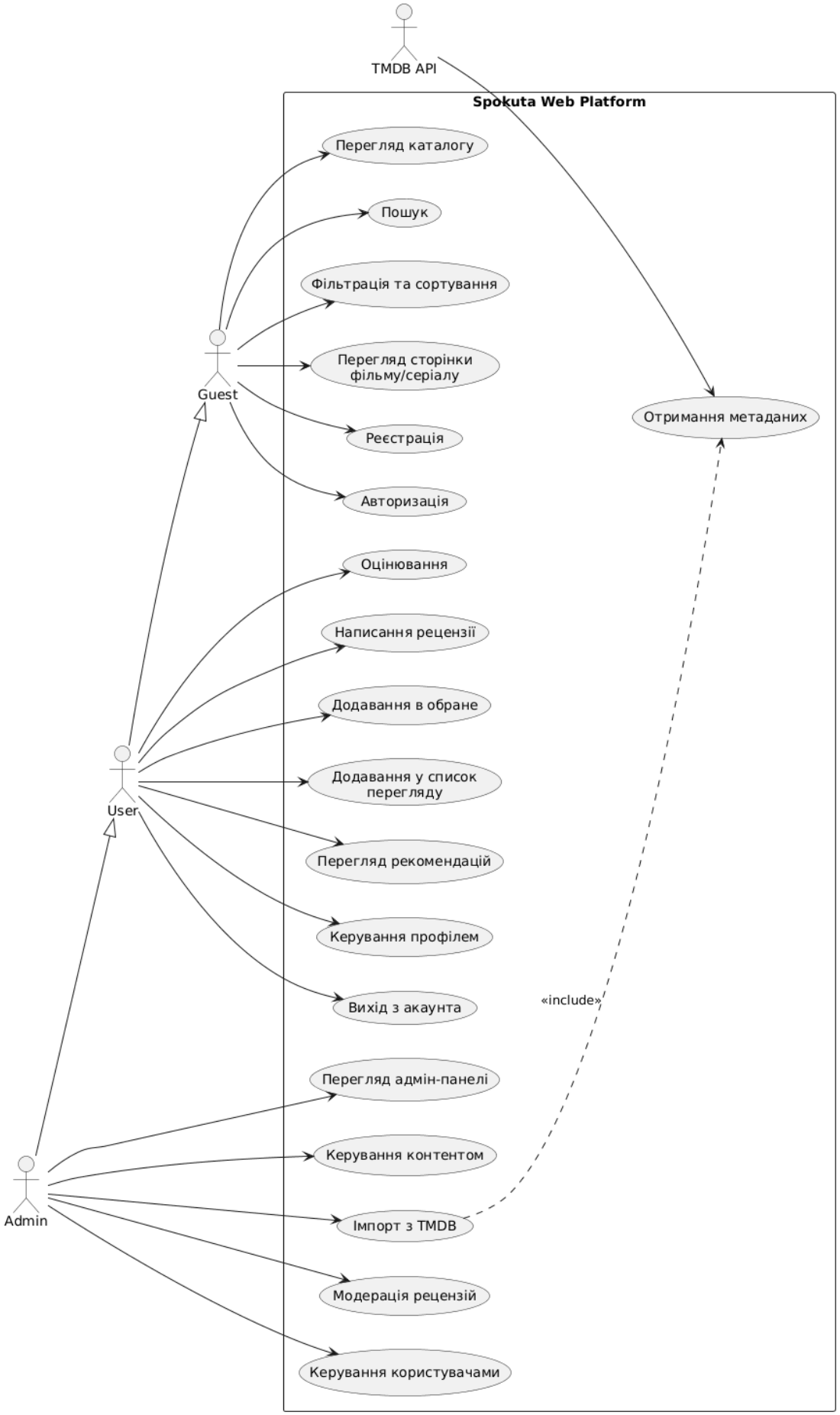

Рисунок 3.1 – Діаграма варіантів використання вебплатформи Spokuta

Гість є базовою роллю користувача. Для нього доступна лише відкрита частина вебплатформи без авторизації. Гість може переглядати каталог, використовувати пошук і фільтрацію, відкривати сторінки фільмів і серіалів, переглядати оцінки та рецензії, а також виконувати навігацію по контенту.

У діаграмі гість пов’язаний із такими варіантами використання:

- перегляд каталогу;
- пошук контенту;
- фільтрація та сортування;
- перегляд сторінки фільму або серіалу;
- реєстрація;
- авторизація.

Таке розділення дозволяє відокремити відкритий функціонал платформи від персоналізованих можливостей авторизованого користувача.

Авторизований користувач у UML–моделі успадковує можливості гостя. Це показано через зв’язок узагальнення між учасниками системи. Така структура була обрана тому, що авторизований користувач зберігає весь базовий функціонал гостя, але додатково отримує доступ до персональних дій.

Після авторизації користувач може:

- виставляти оцінки;
- залишати рецензії;
- працювати з обраним і списком перегляду;
- переглядати рекомендації;
- керувати власним профілем;
- переглядати історію активності;
- виходити з облікового запису.

У діаграмі ці сценарії пов’язані вже з авторизованим користувачем, оскільки вони потребують активної сесії та персональних даних.

Окремо у діаграмі відображається логіка оцінок і рецензій. Вона є однією з основних частин взаємодії користувача з платформою. Через систему оцінювання

користувач формує власну активність, а рекомендації можуть використовувати ці дані для базової персоналізації каталогу.

Обране та список перегляду у UML–моделі також винесені в окремі варіанти використання. Це дозволяє показати, що користувач може працювати з власними списками незалежно від інших сценаріїв. При цьому логіка взаємодії з каталогом залишається спільною як для гостя, так і для авторизованого користувача.

Адміністратор є найвищою роллю у структурі доступу. У діаграмі він успадковує можливості авторизованого користувача, тому має доступ не лише до адміністративних функцій, а й до звичайного користувацького функціоналу.

Через адміністративну панель адміністратор може:

- додавати контент;
- редагувати інформацію про фільми та серіали;
- видаляти записи;
- модерувати рецензії;
- керувати користувачами;
- працювати з імпортом через TMDB API;
- переглядати статистичні дані.

Така структура ролей дозволяє уникнути дублювання сценаріїв у UML–моделі та робить діаграму компактнішою. Завдяки зв’язку узагальнення не потрібно повторно пов’язувати адміністратора з базовими функціями каталогу або авторизації.

Окрему роль у діаграмі варіантів використання виконує TMDB API. У моделі він представлений як зовнішній сервіс, який взаємодіє з платформою під час імпорту контенту. Через TMDB API система отримує метадані фільмів і серіалів:

- назви;
- жанри;
- постери;
- описи;
- оцінки;

– рік випуску;

– інформацію про тип контенту.

У діаграмі TMDB API пов’язаний із варіантом використання імпорту контенту. Це показує, що частина адміністративних сценаріїв залежить від зовнішнього API.

У Spokuta також використовуються зв’язки включення між окремими варіантами використання. Вони застосовуються для сценаріїв, які логічно залежать від інших дій. Наприклад, імпорт контенту через TMDB API включає отримання метаданих із зовнішнього сервісу. Аналогічно деякі персональні функції авторизованого користувача залежать від авторизації та активної сесії.

Під час побудови діаграми варіантів використання основна увага приділялася не максимальній деталізації, а читабельності UML–моделі. Через це у діаграму не додавалися другорядні технічні процеси або дрібні внутрішні дії системи. Основний акцент зроблено саме на логіці взаємодії учасників системи з платформою.

Діаграма варіантів використання також допомогла сформувати загальну структуру подальшого UML–моделювання. На її основі стало простіше визначити ключові процеси для діаграм послідовності, діаграм активності та ER–моделі бази даних.

У наступному підрозділі буде розглянуто діаграми послідовності, які детальніше показують обмін даними між користувачем, серверною частиною, асинхронними запитами та зовнішніми сервісами під час роботи вебплатформи Spokuta.

3.2 Діаграми послідовності

Після побудови діаграми варіантів використання наступним етапом UML–моделювання стало формування діаграм послідовності та діаграми активності.

Якщо діаграма варіантів використання показує загальну логіку взаємодії учасників системи з платформою, то діаграми послідовності детальніше демонструють порядок обміну даними між користувачем, браузером, клієнтською логікою, серверною частиною та зовнішніми сервісами.

Для вебплатформи Spokuta такі UML–діаграми є важливими через значну кількість асинхронних запитів і взаємодію без повного перезавантаження сторінки. Частина дій користувача виконується майже миттєво: оновлення оцінок, додавання до обраного або списку перегляду, відображення спливаючих повідомлень, оновлення рекомендацій або імпорт контенту через TMDB API. Через це було важливо окремо показати порядок передачі даних між компонентами платформи.

На рисунку 3.2 представлена діаграма послідовності взаємодії користувача із системою оцінювання через асинхронний запит.

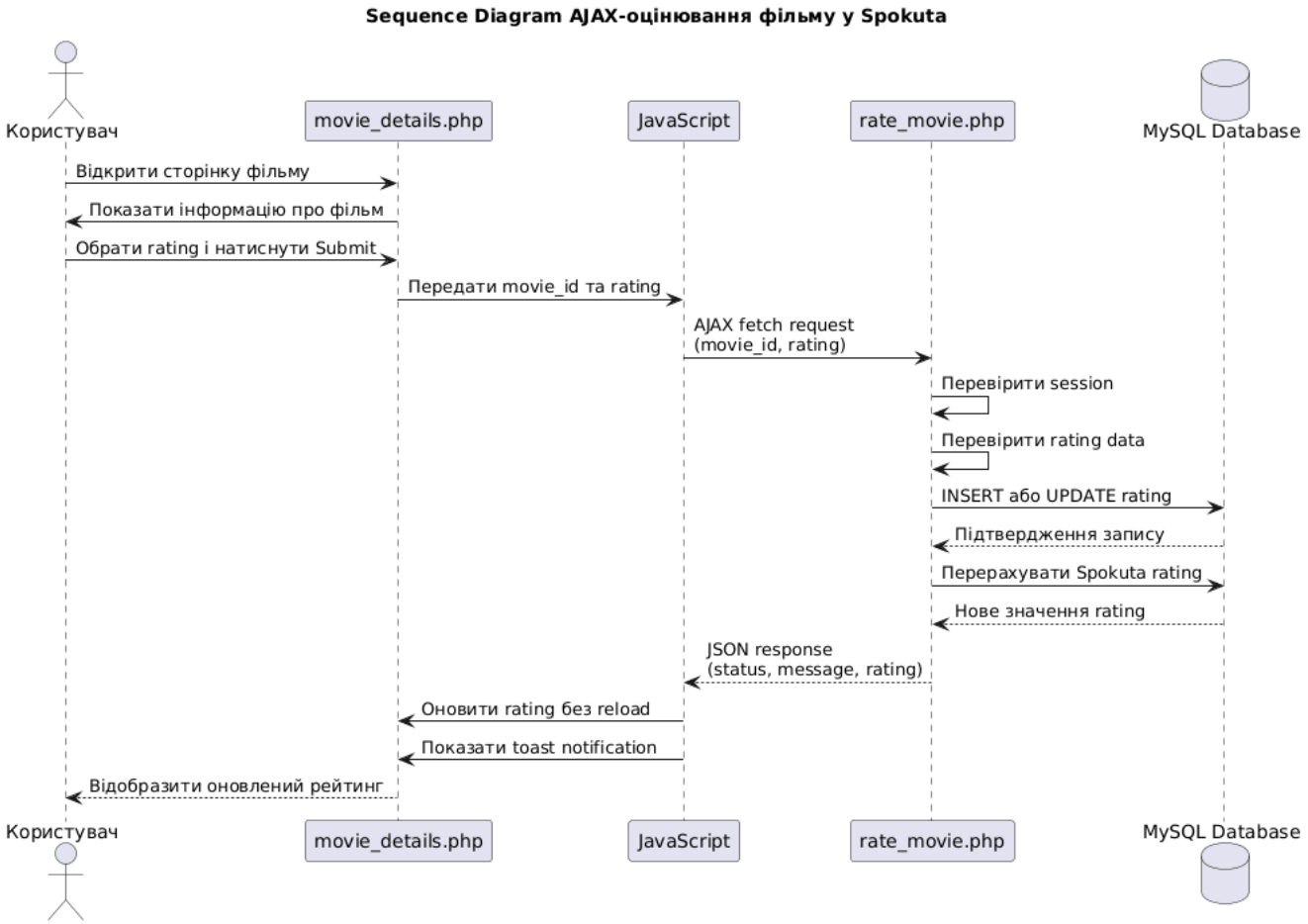

Рисунок 3.2 – Діаграма послідовності взаємодії із системою оцінювання

У підрозділі розглядаються:

– діаграма послідовності взаємодії системи оцінювання через асинхронні запити;
– діаграма послідовності імпорту контенту через TMDB API;
– діаграма активності авторизації користувача.

У цій UML–моделі показано процес виставлення оцінки фільму або серіалу без перезавантаження сторінки. Основними учасниками взаємодії є:

– авторизований користувач;
– браузер;
– клієнтська логіка JavaScript;
– серверна частина PHP;
– база даних MySQL.

Процес починається з відкриття сторінки контенту користувачем. Після цього браузер завантажує інформацію про фільм або серіал, а також поточний рейтинг Spokuta. Коли користувач виставляє оцінку, JavaScript формує асинхронний запит і надсилає його до серверної частини PHP.

Далі серверна частина перевіряє активну сесію користувача. Це необхідно для того, щоб уникнути анонімного оцінювання та пов’язати оцінку з конкретним обліковим записом. Після перевірки сесії серверна частина аналізує отримані дані: ідентифікатор контенту та значення оцінки.

Після успішної перевірки серверна частина взаємодіє з MySQL і виконує оновлення оцінки. Якщо користувач уже оцінював цей фільм раніше, відбувається оновлення попереднього значення. Якщо оцінки ще не було – створюється новий запис.

Окремим етапом у діаграмі послідовності показано перерахунок локального рейтингу Spokuta. Сервер повторно обчислює середнє значення оцінок для контенту та повертає оновлені дані у форматі JSON.

Після отримання відповіді JavaScript оновлює інтерфейс без перезавантаження сторінки. Користувач одразу бачить новий рейтинг, а також спливаюче повідомлення про успішне збереження оцінки. Такий підхід зробив

взаємодію з каталогом швидшою та зручнішою, особливо під час роботи з великим обсягом контенту.

Схожа логіка використовується і для рецензій, обраного, списку перегляду та інших асинхронних запитів у Spokuta. Різниця полягає лише у типі даних та сценарії оновлення інтерфейсу.

На рисунку 3.3 представлена діаграма послідовності імпорту контенту через TMDB API.

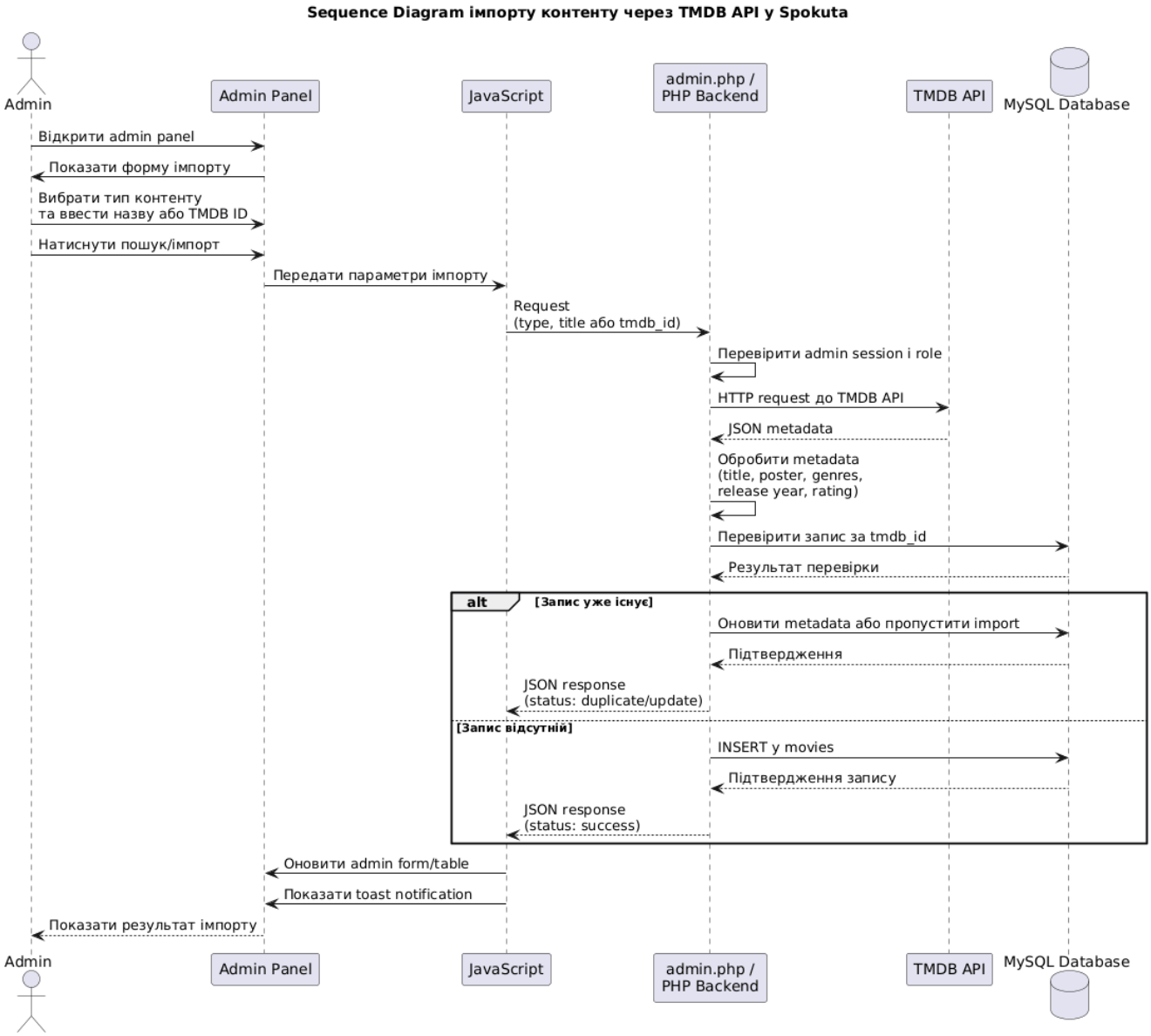

Рисунок 3.3 – Діаграма послідовності імпорту контенту через TMDB API

Ця UML–діаграма описує процес автоматичного отримання метаданих

фільмів і серіалів через зовнішній API–сервіс. Основними учасниками взаємодії є:

- адміністратор;
- адміністративна панель;
- клієнтська логіка JavaScript;
- серверна частина PHP;
- TMDB API;
- база даних MySQL.

Процес починається після відкриття адміністративної панелі адміністратором. Користувач вибирає тип контенту, вводить назву або TMDB ID та запускає процес імпорту.

JavaScript формує запит і надсилає його до серверної частини PHP. Після цього сервер перевіряє сесію адміністратора та його роль доступу. Лише після успішної перевірки серверна частина формує HTTP–запит до TMDB API.

У відповідь TMDB API повертає JSON–об’єкт із метаданими контенту. Серед отриманих даних:

- назва;
- оригінальна назва;
- жанри;
- постери;
- фонові зображення;
- рік випуску;
- оцінка TMDB;
- тип контенту.

Після отримання відповіді серверна частина PHP обробляє ці дані та виконує перевірку дублювання через tmdb_id. Це дозволяє уникнути повторного додавання однакових фільмів або серіалів до каталогу.

Якщо запис уже існує, система повертає повідомлення про наявність дубліката або оновлює окремі поля. Якщо контенту немає у MySQL, серверна частина створює новий запис у каталозі.

Після завершення операції сервер повертає JSON–відповідь, а JavaScript оновлює таблицю або форму в адміністративній панелі без перезавантаження сторінки. Користувач отримує спливаюче повідомлення з результатом імпорту.

Такий підхід значно спрощує адміністрування каталогу. Без інтеграції TMDB API адміністратору довелося б вручну вводити назви, жанри, постери, оцінки та інші метадані для кожного фільму або серіалу.

На рисунку 3.4 представлена діаграма активності авторизації користувача.

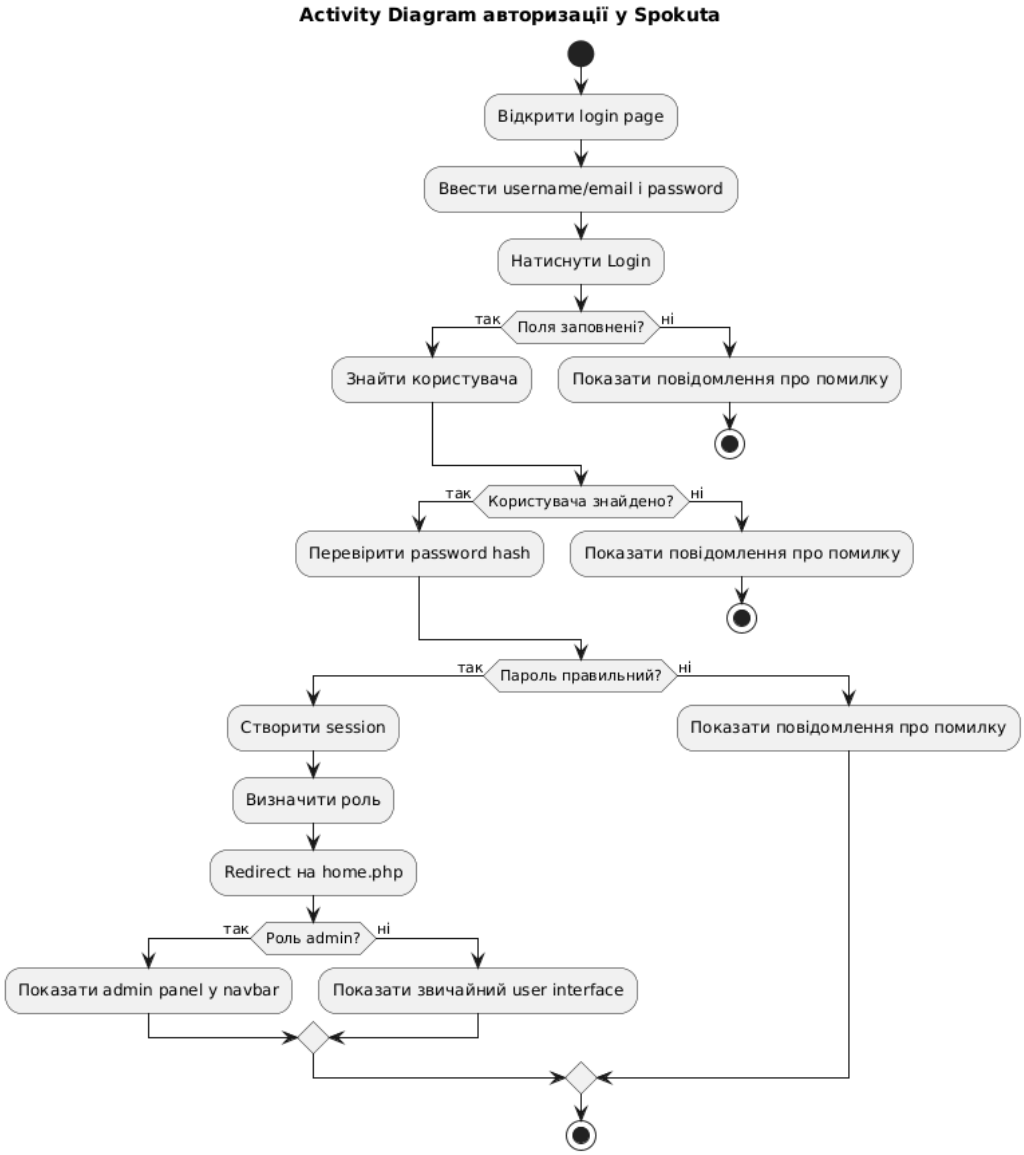

Рисунок 3.4 – Діаграма активності авторизації користувача

На відміну від діаграм послідовності, діаграма активності показує не обмін

повідомленнями між компонентами, а послідовність станів та умов переходу під час авторизації.

Процес починається після відкриття сторінки входу користувачем. Далі користувач вводить логін або електронну пошту та пароль, після чого надсилає форму авторизації.

Першим етапом перевірки є аналіз коректності введених полів. Якщо користувач залишив порожні значення, система одразу показує повідомлення про помилку й завершує поточний сценарій.

Якщо поля заповнені правильно, серверна частина PHP виконує пошук користувача в MySQL. У випадку відсутності облікового запису система також повертає повідомлення про помилку авторизації.

Після успішного пошуку користувача виконується перевірка хешу пароля. Якщо пароль неправильний, діаграма активності переходить у гілку помилки. При успішній перевірці серверна частина створює сесію користувача та визначає його роль доступу. Після цього користувач перенаправляється на головну сторінку платформи.

Окремо в діаграмі активності показується логіка ролей:

– адміністратор отримує доступ до адміністративної панелі;

– авторизований користувач працює зі стандартним інтерфейсом каталогу.

Така структура діаграми активності дозволяє наочно показати всі ключові стани процесу авторизації без перевантаження технічними деталями. Через блоки прийняття рішень добре видно логіку перевірки полів, пароля та ролей користувача.

Усі розглянуті UML–діаграми допомогли сформувати більш цілісне уявлення про роботу Spokuta. Діаграми послідовності показують порядок передачі даних між компонентами платформи, а діаграма активності – логіку переходів між станами користувача під час виконання окремих сценаріїв.

У наступному підрозділі буде розглянуто більш детальний опис прецедентів використання та логіку окремих сценаріїв роботи вебплатформи Spokuta.

3.3 Розгорнуті описи прецедентів

Після побудови UML–діаграм наступним етапом стало формування розгорнутих описів основних сценаріїв роботи вебплатформи Spokuta. На відміну від діаграми варіантів використання або діаграм послідовності, у цьому підрозділі основна увага приділяється не структурі UML–моделі, а поведінці користувача та реакції платформи під час виконання окремих дій.

Такий опис дозволяє краще показати логіку взаємодії між користувачем, інтерфейсом, асинхронними запитами, серверною частиною та базою даних. Також це допомагає зрозуміти, як працюють ролі доступу, динамічне оновлення сторінок і взаємодія з TMDB API.

Одним із базових сценаріїв у Spokuta є авторизація користувача. Процес починається після переходу на сторінку входу. Користувач вводить ім’я користувача або електронну пошту та пароль, після чого надсилає форму авторизації.

Після отримання даних серверна частина PHP перевіряє коректність введених полів. Якщо частина полів порожня або введені дані неправильні, користувач отримує повідомлення про помилку без переходу на інші сторінки.

Далі система виконує перевірку облікового запису у MySQL. Якщо користувача знайдено, серверна частина перевіряє хеш пароля та створює сесію. Після цього визначається роль доступу.

У випадку звичайного авторизованого користувача відкривається доступ до каталогу, оцінок, рецензій, обраного, списку перегляду та рекомендацій. Якщо обліковий запис має роль адміністратора, у навігаційній панелі додатково з’являється доступ до адміністративної панелі.

Авторизація працює достатньо швидко й не перевантажує інтерфейс зайвими переходами. Логіка побудована так, щоб користувач одразу отримував результат перевірки без складної навігації або великої кількості додаткових дій.

Іншим важливим сценарієм є виставлення оцінок і створення рецензій. Після відкриття сторінки фільму або серіалу користувач може поставити оцінку та залишити власну рецензію.

Після натискання кнопки підтвердження JavaScript формує асинхронний запит і передає його до серверної частини PHP. Сервер перевіряє сесію користувача, ідентифікатор контенту та значення оцінки.

Якщо перевірка проходить успішно, серверна частина записує нову оцінку у MySQL або оновлює попередню. Далі система автоматично перераховує локальний рейтинг Spokuta для конкретного фільму чи серіалу.

Після завершення обробки JavaScript отримує JSON–відповідь і оновлює частину інтерфейсу без перезавантаження сторінки. Користувач одразу бачить змінений рейтинг та спливаюче повідомлення про успішне збереження.

Схожий сценарій використовується і для рецензій. Після відправлення рецензії новий запис автоматично з’являється у списку без повного перезавантаження сторінки. Це робить роботу з каталогом помітно швидшою, особливо під час частого перемикання між сторінками контенту.

Окремо у Spokuta реалізований сценарій роботи з обраним і списком перегляду. Після відкриття сторінки контенту користувач може додати фільм або серіал до одного з персональних списків.

Під час натискання кнопки JavaScript надсилає асинхронний запит до серверної частини. Сервер перевіряє сесію користувача та визначає, чи не був цей контент доданий раніше.

Якщо запис уже існує, система не створює дубль у базі даних. Якщо контент відсутній у списку, створюється новий запис у MySQL. Після завершення операції інтерфейс оновлюється без перезавантаження сторінки. Кнопка може змінити свій стан або текст, а користувач отримує коротке повідомлення про успішне додавання.

Обране та список перегляду використовуються не лише як звичайні списки. На основі активності користувача система також формує базові рекомендації та персоналізує частину контенту на головній сторінці.

Ще одним важливим сценарієм є імпорт контенту через TMDB API. Ця функція доступна лише адміністратору через адміністративну панель.

Після відкриття адміністративної панелі адміністратор може ввести назву фільму, серіалу або TMDB ID. Далі JavaScript надсилає запит до серверної частини PHP.

Перед початком імпорту серверна частина перевіряє сесію користувача та його роль доступу. Якщо користувач не має прав адміністратора, система блокує виконання запиту.

Після успішної перевірки сервер надсилає HTTP–запит до TMDB API та отримує JSON–відповідь із метаданими контенту. Серед них:

- назва;
- жанри;
- постери;
- фонові зображення;
- рейтинг TMDB;
- рік випуску;
- тип контенту.

Після цього серверна частина аналізує отримані дані та перевіряє, чи існує контент із таким tmdb_id у MySQL. Якщо запис уже є в каталозі, система повідомляє адміністратора про наявність дубліката або виконує оновлення окремих полів.

Якщо контент відсутній, створюється новий запис у каталозі Spokuta. Після завершення імпорту JavaScript оновлює таблицю або форму в адміністративній панелі без перезавантаження сторінки.

Такий сценарій суттєво спрощує роботу адміністратора. Без TMDB API увесь каталог довелося б заповнювати вручну, включаючи жанри, рейтинги, постери та інші метадані.

Останній важливий сценарій пов’язаний з адміністративними діями. Доступ до цієї частини платформи має лише адміністратор.

Після авторизації адміністратор отримує можливість працювати з контентом і користувачами через окремий інтерфейс. Через адміністративну панель можна:

- додавати новий контент;
- редагувати інформацію про фільми та серіали;
- видаляти записи;
- модерувати рецензії;
- працювати з імпортом через TMDB API;
- переглядати статистичні дані.

Більшість адміністративних дій також використовує асинхронну взаємодію. Завдяки цьому оновлення таблиць, модерація рецензій або зміна даних відбуваються без повного перезавантаження сторінки.

Окремо реалізована перевірка доступу до адміністративних функцій. Навіть якщо користувач вручну відкриє адресу адміністративної панелі, серверна частина додатково перевіряє роль через сесію. Якщо користувач не має прав адміністратора, доступ блокується.

Під час роботи над Spokuta особлива увага приділялася саме логіці взаємодії користувача з платформою. Більшість основних сценаріїв побудована навколо швидкого оновлення інтерфейсу, мінімальної кількості перезавантажень сторінок та зручної роботи з каталогом.

Розглянуті прецеденти також допомогли точніше сформувати логіку ролей, асинхронної взаємодії та поведінку інтерфейсу під час роботи з великим обсягом контенту.

У наступному підрозділі будуть сформовані висновки до розділу 3 та підсумована роль UML–моделювання у процесі проєктування вебплатформи Spokuta.

3.4 Висновки до розділу

Третій розділ був присвячений UML–моделюванню вебплатформи Spokuta та опису основних сценаріїв взаємодії користувачів із системою. Побудовані UML–діаграми дозволили структурувати логіку роботи платформи до переходу до архітектури та реалізації окремих модулів.

Діаграма варіантів використання показала загальну модель взаємодії між гостем, авторизованим користувачем, адміністратором та TMDB API. За її допомогою було розділено права доступу, логіку ролей та основні сценарії роботи з каталогом, оцінками, рецензіями, обраним, списком перегляду й адміністративними функціями.

Діаграми послідовності відобразили порядок обміну даними між браузером, клієнтською логікою JavaScript, серверною частиною PHP, базою даних MySQL та зовнішнім API. Особливу увагу приділено асинхронній взаємодії, яка використовується для оновлення оцінок, рецензій, рекомендацій та частини адміністративного функціоналу без повного перезавантаження сторінки.

Окремо була змодельована логіка імпорту контенту через TMDB API, що показує взаємодію між адміністративною панеллю, серверною частиною та зовнішнім сервісом метаданих. Також діаграма активності авторизації дала можливість відобразити послідовність перевірки логіна, пароля, сесії користувача, ролі доступу та обробку помилкових сценаріїв.

Розгорнуті описи прецедентів допомогли точніше сформувати поведінку платформи під час взаємодії користувача з каталогом. Особлива увага приділялася реакції інтерфейсу, роботі асинхронних запитів та оновленню сторінок без повного перезавантаження.

Побудовані UML–моделі стали основою для подальшого проєктування архітектури Spokuta, структури компонентів і логіки взаємодії між клієнтською частиною, серверною частиною та базою даних.

4 ПРОЄКТУВАННЯ АРХІТЕКТУРИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Загальна архітектура системи

Після побудови UML–моделей та опису основних сценаріїв роботи наступним етапом стало формування загальної архітектури вебплатформи Spokuta. Архітектура визначає спосіб взаємодії між клієнтською частиною, серверною логікою, базою даних та зовнішніми сервісами. Від цього залежить не лише швидкість роботи платформи, а й зручність підтримки, розподіл ролей користувачів та стабільність роботи каталогу при великій кількості запитів.

Spokuta побудована за класичною клієнт–серверною архітектурою. Користувач працює з платформою через браузер, який відповідає за відображення інтерфейсу, обробку дій користувача та надсилання запитів до серверної частини. Сервер, у свою чергу, обробляє отримані дані, взаємодіє з MySQL та повертає готову відповідь у вигляді HTML–сторінки або JSON–даних для асинхронних запитів.

На рисунку 4.1 представлена діаграма компонентів вебплатформи Spokuta, яка демонструє структуру основних частин системи та зв’язки між ними.

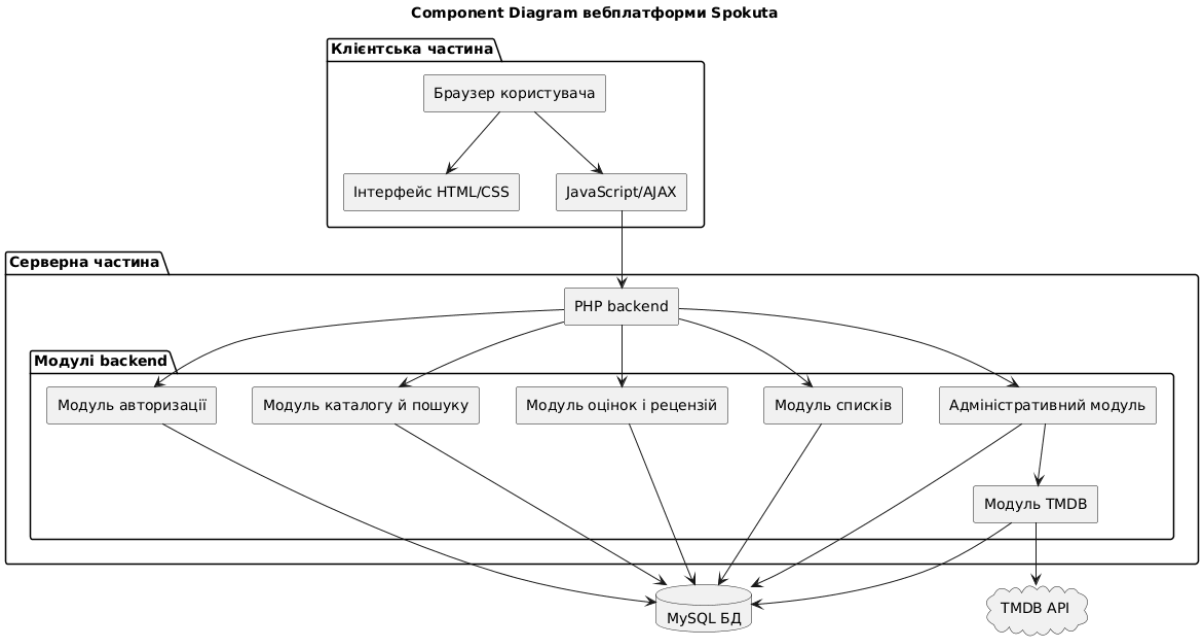

Рисунок 4.1 – Діаграма компонентів вебплатформи Spokuta

Після відкриття сторінки браузер завантажує HTML–розмітку, CSS–стилі та JavaScript–логіку. Частина контенту формується сервером одразу під час генерації сторінки, але значна кількість дій у Spokuta працює через асинхронні запити без повного перезавантаження. Такий підхід використовується для оцінок, рецензій, обраного, списку перегляду, рекомендацій, пошуку, фільтрації та частини адміністративних функцій.

Архітектура платформи умовно поділяється на кілька основних частин:

- клієнтська частина;
- серверна частина PHP;
- база даних MySQL;
- модуль інтеграції TMDB API;
- система авторизації та ролей;
- адміністративний модуль.

Клієнтська частина відповідає за взаємодію користувача з інтерфейсом. Вона включає сторінки каталогу, навігацію, форми авторизації, сторінки контенту, профіль користувача та адміністративну панель. Для динамічного оновлення даних використовується JavaScript. Саме він формує асинхронні запити та оновлює окремі елементи інтерфейсу після отримання відповіді від сервера.

Наприклад, під час виставлення оцінки користувачу не потрібно повторно відкривати сторінку фільму. JavaScript надсилає запит до серверної частини PHP, після чого сервер оновлює дані у MySQL та повертає новий рейтинг. Далі інтерфейс змінюється динамічно, а користувач бачить спливаюче повідомлення про успішне збереження оцінки.

Схожа логіка використовується і для роботи з обраним та списком перегляду. Після натискання кнопки додавання контенту JavaScript виконує асинхронний запит, а сервер перевіряє сесію користувача та створює або оновлює запис у базі даних. Після цього кнопка змінює свій стан без перезавантаження сторінки.

Серверна частина Spokuta реалізована на PHP. Вона відповідає за обробку HTTP–запитів, перевірку авторизації, роботу з ролями користувачів, обробку асинхронних запитів та взаємодію з MySQL. Також через PHP реалізована логіка

рекомендацій, формування каталогу та перевірка доступу до адміністративних функцій.

Окремо сервер відповідає за роботу сесій. Після успішної авторизації PHP створює сесію користувача та зберігає інформацію про його роль. Надалі це використовується для перевірки доступу до оцінок, рецензій, обраного, списку перегляду та адміністративної панелі.

Логіка ролей у Spokuta побудована навколо трьох основних ролей:

– гість;
– авторизований користувач;
– адміністратор.

Гість має доступ лише до відкритої частини каталогу, пошуку та фільтрації. Авторизований користувач після входу до системи отримує доступ до персоналізованих функцій: оцінок, рецензій, обраного, списку перегляду, рекомендацій та профілю користувача. Адміністратор додатково має можливість працювати з адміністративною панеллю, імпортом контенту та керуванням каталогом.

Перевірка ролей виконується на серверній стороні. Навіть якщо користувач вручну відкриє адресу адміністративної панелі, PHP додатково перевіряє сесію та роль доступу. Якщо прав недостатньо, сервер блокує запит і перенаправляє користувача на головну сторінку або сторінку авторизації.

База даних MySQL використовується як центральне сховище всієї інформації платформи. У ній зберігаються:

– користувачі;
– оцінки та рецензії;
– обране та список перегляду;
– інформація про фільми та серіали;
– історія активності;
– частина метаданих TMDB.

PHP взаємодіє з MySQL через SQL–запити та отримує дані для формування каталогу, профілю користувача або рекомендацій. Під час роботи з каталогом

сервер виконує сортування, фільтрацію та пошук контенту за жанрами, рейтингом або типом контенту.

Окремий компонент архітектури пов’язаний із TMDB API. Інтеграція зовнішнього API дозволила автоматизувати наповнення каталогу та зменшити кількість ручної роботи під час додавання контенту. Через TMDB API система отримує:

- назви;
- жанри;
- постери;
- фонові зображення;
- описи;
- рік випуску;
- оцінки TMDB;
- тип контенту.

Під час імпорту адміністратор запускає процес через адміністративну панель, після чого PHP надсилає HTTP–запит до TMDB API. Отримані дані проходять перевірку та записуються у MySQL. Перед додаванням сервер також перевіряє tmdb_id, щоб уникнути дублювання записів у каталозі.

Для Spokuta була обрана модульна структура організації компонентів. Через це окремі частини платформи працюють відносно незалежно одна від одної. Наприклад, модуль оцінок і рецензій не впливає на логіку імпорту через TMDB API, а пошук і фільтрація можуть працювати незалежно від рекомендацій, обраного або списку перегляду.

Такий підхід спростив підтримку проєкту та зробив структуру більш зрозумілою під час розробки. Також це дозволило поступово розширювати функціонал без повного перепроєктування всієї архітектури.

Під час проєктування основна увага приділялася не лише розділенню компонентів, а й швидкості взаємодії між ними. Завдяки асинхронним запитам значна частина дій обробляється окремо від повного завантаження сторінки. Це зменшує кількість зайвих переходів та робить роботу з каталогом більш плавною,

особливо на мобільних пристроях.

Адаптивний інтерфейс також вплинув на архітектуру клієнтської частини. Інтерфейс адаптується до різних розмірів екрана, а навігація та блоки контенту перебудовуються залежно від типу пристрою. Завдяки цьому користувач може комфортно працювати з каталогом як на ПК, так і на смартфоні.

Сформована архітектура стала основою для подальшого проєктування структури бази даних, зв’язків між сутностями та організації зберігання інформації у MySQL. У наступному підрозділі буде детальніше розглянуто проєктування бази даних вебплатформи Spokuta та структуру основних таблиць.

4.2 Проєктування бази даних

Після формування загальної архітектури вебплатформи наступним етапом стало проєктування структури бази даних. Для Spokuta база даних виконує роль центрального сховища інформації, пов’язаної з користувачами, каталогом контенту, оцінками, рецензіями, обраним, списком перегляду та адміністративною частиною платформи. Від структури таблиць та організації зв’язків між ними залежить швидкість роботи каталогу, коректність збереження даних і стабільність роботи всієї системи.

У Spokuta використовується MySQL. Цю систему керування базами даних було обрано через просту інтеграцію з PHP, підтримку реляційної структури та зручну роботу зі зв’язками між сутностями. Також MySQL добре підходить для вебпроєктів із великою кількістю однотипних запитів, пов’язаних із каталогом, пошуком і взаємодією користувачів із контентом.

Структура бази даних проєктувалася так, щоб окремі сутності не дублювали одна одну та могли працювати незалежно. Через це інформацію було розділено на окремі таблиці: користувачі, контент, оцінки, обране та список перегляду. Текстові рецензії не винесені в окрему таблицю, а зберігаються разом з оцінкою у таблиці

ratings через поле review_text. Такий підхід спрощує оновлення даних і зменшує кількість дубльованих записів.

На рисунку 4.2 представлена ER–діаграма бази даних вебплатформи Spokuta, яка показує основні сутності, їхні поля та зв’язки між таблицями.

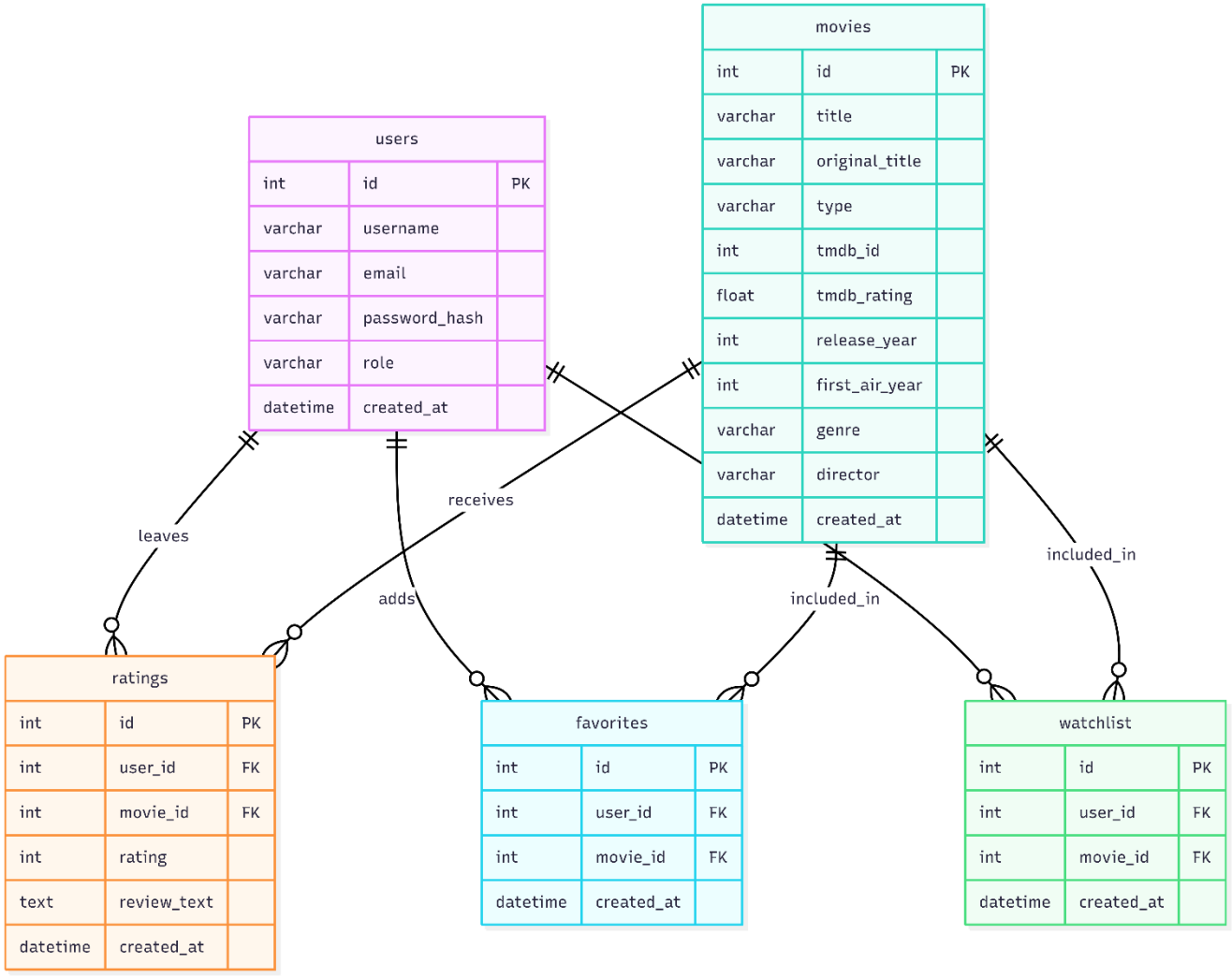

Рисунок 4.2 – ER Diagram бази даних вебплатформи Spokuta

Під час проєктування бази даних основна увага приділялася зв’язкам між користувачем та контентом. Значна частина функціоналу платформи побудована саме навколо цієї взаємодії: оцінювання фільмів, написання текстових рецензій у межах таблиці ratings, формування обраного, списку перегляду та робота рекомендацій.

Основною таблицею для збереження користувачів є users. У ній зберігаються:

– ім’я користувача;
– електронна пошта;
– хеш пароля;
– роль користувача;
– дата створення облікового запису.

Таблиця users використовується під час авторизації, автентифікації та перевірки ролей. Поле role визначає права доступу користувача до окремих частин платформи. Звичайний авторизований користувач має доступ до оцінок, рецензій, обраного, списку перегляду та рекомендацій, тоді як адміністратор додатково отримує доступ до адміністративної панелі та функцій керування каталогом.

Інформація про фільми та серіали зберігається у таблиці movies. Вона містить як локальні дані платформи, так і частину метаданих, отриманих через TMDB API. Для кожного запису зберігаються:

– назва;
– оригінальна назва;
– тип контенту;
– жанр;
– режисер;
– рік випуску;
– ідентифікатор TMDB;
– оцінка TMDB;
– дата додавання контенту.

Поле tmdb_id використовується для синхронізації з TMDB API та перевірки дублювання під час імпорту контенту. Якщо в базі вже існує запис із таким tmdb_id, система не створює новий фільм або серіал повторно. Це дозволяє уникнути дублювання контенту в каталозі.

Окремо в таблиці movies зберігається тип контенту – фільм або серіал. Це використовується під час пошуку, фільтрації та формування рекомендацій.

Для роботи з оцінками та рецензіями використовується таблиця ratings. У ній

зберігається не лише числова оцінка користувача, а й текст рецензії у полі review_text. Через це рецензії не є окремою сутністю бази даних, а працюють як частина запису про оцінювання контенту.

Таблиця ratings містить:

- ідентифікатор користувача;
- ідентифікатор фільму або серіалу;
- оцінку;
- текст рецензії;
- дату створення запису.

Кожен запис у ratings пов’язаний із конкретним користувачем та конкретним фільмом через зовнішні ключі. Один користувач може залишати багато оцінок і рецензій до різних фільмів, а один фільм може мати багато оцінок і рецензій від різних користувачів. Через це між users і ratings, а також між movies і ratings використовується зв’язок типу «один до багатьох».

Такий підхід спрощує структуру бази даних, оскільки оцінка та текстова рецензія зберігаються в одному записі. Якщо користувач оновлює свою оцінку або змінює текст рецензії, серверна частина PHP працює з однією таблицею ratings, а інтерфейс може оновити дані через асинхронний запит без повного перезавантаження сторінки.

Для обраного та списку перегляду були створені окремі таблиці favorites та watchlist. Хоча обидві сутності мають схожу структуру, їх розділення зробило логіку роботи платформи більш зрозумілою. Таблиця favorites використовується для збереження улюбленого контенту, а watchlist – для фільмів або серіалів, які користувач планує переглянути пізніше.

У таблицях favorites та watchlist зберігаються:

- ідентифікатор користувача;
- ідентифікатор фільму або серіалу;
- дата додавання запису.

Кожен запис пов’язаний із користувачем та конкретним елементом контенту через зовнішні ключі. Один користувач може додати багато фільмів до обраного

або списку перегляду, а один фільм може одночасно бути у списках багатьох користувачів.

Рекомендації у Spokuta частково формуються динамічно на основі оцінок, обраного, списку перегляду та історії активності користувача. Через це окрема таблиця recommendations не використовується як основне постійне сховище. Значна частина логіки рекомендацій формується під час обробки запитів на серверній стороні.

Окрему роль у структурі бази даних відіграють дані, з якими працює адміністративна частина. Адміністративний модуль використовує ті самі таблиці movies, users та ratings, але має розширені права доступу. Через адміністративну панель адміністратор може:

- додавати контент;
- редагувати записи;
- видаляти фільми та серіали;
- модерувати текстові рецензії в межах таблиці ratings;
- імпортувати контент через TMDB API.

Такий підхід дозволив уникнути створення окремої дубльованої структури для адміністративної частини платформи.

Під час проєктування ER–діаграми також враховувалася нормалізація структури бази даних. Основну інформацію було розділено на окремі сутності, а зв’язки між ними реалізовано через зовнішні ключі. Це дозволило:

- зменшити дублювання інформації;
- спростити оновлення записів;
- покращити стабільність роботи каталогу;
- зробити пошук і фільтрацію швидшими.

Зовнішні ключі використовуються для підтримки цілісності даних між таблицями. Наприклад, запис у ratings не може існувати без відповідного користувача чи фільму. Якщо запис про контент видаляється, пов’язані оцінки та рецензії також повинні оброблятися коректно, оскільки числова оцінка і текст рецензії зберігаються в межах однієї таблиці.

Структура бази даних також враховує роботу асинхронної взаємодії. Під час динамічного оновлення оцінок, рецензій, обраного або списку перегляду сервер отримує невеликі окремі запити та працює лише з необхідними таблицями. Через це навантаження на сервер та кількість зайвих запитів до MySQL залишаються відносно невеликими навіть при активній роботі з каталогом.

Окремо під час проєктування враховувалася робота пошуку та фільтрації. Частина полів у таблиці movies використовується для сортування та фільтрації контенту за жанром, типом, роком випуску або рейтингом. Це дозволяє швидко формувати вибірки контенту без складної додаткової обробки.

Сформована структура бази даних стала основою для взаємодії між усіма компонентами вебплатформи. Через зв’язки між таблицями система може коректно працювати з авторизацією, оцінками, рецензіями, рекомендаціями, обраним, списком перегляду, а інтеграція TMDB API спрощує наповнення каталогу.

У наступному підрозділі буде детальніше розглянуто взаємодію компонентів системи, обмін даними між клієнтською і серверною частинами та організацію асинхронних запитів у вебплатформі Spokuta.

4.3 Взаємодія компонентів системи

Після формування архітектури вебплатформи та структури бази даних наступним етапом стало проєктування взаємодії між основними компонентами Spokuta. Робота платформи побудована навколо постійного обміну даними між браузером користувача, серверною частиною на PHP, MySQL та зовнішнім TMDB API. Саме ця взаємодія забезпечує роботу каталогу, авторизації, оцінок, рецензій, обраного, списку перегляду та адміністративних функцій.

Основою роботи Spokuta є клієнт–серверна взаємодія. Користувач працює з платформою через браузер, де відображається інтерфейс і виконується клієнтська логіка JavaScript. Після виконання певної дії браузер формує HTTP–запит до

серверної частини PHP, а сервер обробляє дані, звертається до MySQL та повертає відповідь.

Частина сторінок формується сервером одразу під час відкриття каталогу або сторінки фільму. Проте значна кількість дій працює через асинхронні запити. Це дозволяє оновлювати окремі елементи інтерфейсу без повного перезавантаження сторінки.

Такий підхід використовується для:

- оцінок і рецензій;
- обраного та списку перегляду;
- рекомендацій;
- пошуку та фільтрації;
- частини адміністративної панелі.

Після відкриття сторінки браузер отримує HTML–структуру, CSS–стилі та JavaScript–файли. Далі JavaScript реагує на дії користувача: натискання кнопок, пошук, фільтрацію або взаємодію з каталогом.

Коли користувач, наприклад, виставляє оцінку, браузер формує асинхронний запит та надсилає його до серверної частини PHP. Сервер перевіряє сесію користувача, коректність даних та права доступу. Після цього PHP звертається до MySQL, оновлює або створює запис у таблиці ratings та повертає відповідь у форматі JSON.

Отримавши відповідь, JavaScript динамічно оновлює рейтинг на сторінці. Додатково користувач бачить спливаюче повідомлення про успішне збереження оцінки або помилку авторизації.

Схожа логіка використовується для рецензій. Після надсилання рецензії сервер перевіряє:

- чи авторизований користувач;
- чи не порожній текст;
- чи існує контент у базі даних.

Після успішної перевірки рецензія записується у MySQL, а новий запис одразу з’являється в інтерфейсі без повторного відкриття сторінки.

Обране та список перегляду працюють за схожою схемою. Користувач натискає кнопку додавання, JavaScript формує асинхронний запит, а серверна частина PHP перевіряє сесію та роль користувача. Якщо користувач авторизований, сервер створює запис у таблиці favorites або watchlist.

Після цього інтерфейс одразу оновлюється:

– змінюється стан кнопки;

– оновлюється кількість елементів;

– показується повідомлення про успішне додавання.

Завдяки асинхронній взаємодії користувачеві не потрібно постійно перезавантажувати сторінку або повторно відкривати каталог після кожної дії. Це помітно спрощує роботу з платформою, особливо під час активного перегляду контенту.

Окрему роль у взаємодії компонентів відіграє логіка авторизації. Після успішного входу PHP створює сесію користувача та зберігає інформацію про його роль. Надалі кожен запит до серверної частини перевіряє:

– чи існує сесія;

– чи авторизований користувач;

– які права доступу він має.

Якщо користувач не авторизований, сервер блокує доступ до оцінок, рецензій, обраного, списку перегляду та рекомендацій. У такому випадку PHP повертає повідомлення про помилку або перенаправляє користувача на сторінку авторизації.

Перевірка ролей використовується і для адміністративної панелі. Навіть якщо користувач вручну спробує відкрити адресу адміністративної частини, сервер додатково перевірить роль у сесії. Якщо користувач не є адміністратором, доступ до адміністративної панелі буде заборонений.

Адміністративна частина платформи також активно використовує асинхронну взаємодію. Це застосовується під час:

– імпорту контенту;

– редагування записів;

– модерації рецензій;

– оновлення таблиць в адміністративній панелі.

Під час імпорту контенту адміністратор вводить назву або TMDB ID, після чого JavaScript надсилає запит до серверної частини PHP. Сервер перевіряє сесію та права доступу, а потім звертається до TMDB API.

TMDB API повертає JSON–дані з метаданими фільму або серіалу. Сервер обробляє отриману інформацію та перевіряє, чи існує такий запис у базі даних через tmdb_id. Якщо контент уже є в каталозі, сервер повертає повідомлення про дублювання запису. Якщо запис відсутній, інформація зберігається у MySQL.

Після завершення імпорту PHP повертає відповідь у форматі JSON, а адміністративна панель динамічно оновлює таблицю контенту. Адміністратору не потрібно повторно відкривати сторінку для перевірки результату.

Пошук і фільтрація також побудовані навколо взаємодії клієнтської та серверної частин. Під час введення пошукового запиту або зміни фільтрів JavaScript надсилає асинхронний запит із параметрами пошуку. Серверна частина PHP формує запит до MySQL та повертає список відповідного контенту.

Далі браузер оновлює каталог без перезавантаження сторінки. Завдяки цьому навігація між фільмами та серіалами працює швидше, а кількість зайвих переходів зменшується.

Рекомендації формуються на серверній стороні. PHP аналізує оцінки, обране, список перегляду та частину історії активності користувача, після чого створює список рекомендованого контенту. Далі ці дані передаються у браузер та відображаються у відповідних блоках інтерфейсу.

Під час взаємодії компонентів також використовується перевірка помилок та валідація даних. Сервер перевіряє:

– коректність вхідних даних;

– наявність сесії;

– права доступу;

– існування контенту;

– дублювання записів.

Якщо виникає помилка, PHP повертає відповідь у форматі JSON із відповідним повідомленням. JavaScript отримує цю відповідь та показує користувачу спливаюче повідомлення або повідомлення в інтерфейсі.

Окремо варто згадати адаптивний інтерфейс. Хоча він напряму не змінює серверну логіку, він впливає на взаємодію компонентів на стороні клієнта. Частина елементів каталогу, навігації, обраного та списку перегляду перебудовується залежно від розміру екрана. Завдяки цьому взаємодія з платформою залишається зручною як на ПК, так і на мобільних пристроях.

Під час проєктування взаємодії компонентів головною задачею було зменшення кількості зайвих переходів та повторних завантажень сторінок. Через використання асинхронних запитів більшість дій виконується окремо від повного завантаження сторінки. Це дозволило зробити роботу каталогу більш плавною та прискорити взаємодію користувача з контентом.

Сформована логіка взаємодії між браузером, серверною частиною PHP, MySQL та TMDB API забезпечує стабільну роботу основних функцій Spokuta та дозволяє обробляти дії користувача без складних повторних запитів або дублювання даних.

У наступному підрозділі будуть підведені загальні висновки до розділу 4 та підсумовані результати проєктування архітектури вебплатформи Spokuta.

4.4 Висновки до розділу

Побудована архітектура вебплатформи Spokuta дозволила організувати стабільну взаємодію між клієнтською частиною, серверною частиною PHP, MySQL та TMDB API. Поділ системи на окремі компоненти спростив роботу з каталогом, авторизацією, рекомендаціями та адміністративними функціями. Завдяки клієнт–серверній логіці основна обробка даних виконується на серверній стороні, тоді як браузер відповідає за відображення інтерфейсу та динамічну взаємодію з

користувачем.

Діаграма компонентів допомогла сформувати загальну структуру платформи та показати зв’язки між основними модулями. Окремо була організована логіка ролей із підтримкою гостя, авторизованого користувача та адміністратора. Сесії використовуються для перевірки авторизації та доступу до оцінок, рецензій, обраного, списку перегляду й адміністративної панелі.

Структура бази даних була побудована навколо зв’язків між користувачами та контентом. ER–діаграма показує зв’язки між таблицями users, movies, ratings, favorites та watchlist. Таблиця ratings зберігає як числову оцінку, так і текст рецензії у полі review_text. Використання зовнішніх ключів дозволило підтримувати цілісність даних і уникати дублювання записів у каталозі.

Асинхронна взаємодія стала однією з основних частин логіки платформи. Значна кількість дій виконується без повного перезавантаження сторінки: виставлення оцінок, додавання контенту до обраного або списку перегляду, пошук, фільтрація та частина адміністративних функцій. Для обміну даними між клієнтською і серверною частинами використовуються відповіді у форматі JSON, що спрощує оновлення окремих елементів інтерфейсу.

Окремо була реалізована інтеграція TMDB API. Вона дозволила автоматизувати наповнення каталогу та отримання метаданих про фільми й серіали без ручного заповнення великої кількості інформації. Через адміністративну панель адміністратор може імпортувати контент, перевіряти дублікати записів та швидше працювати з каталогом.

Сформована архітектура забезпечує стабільну взаємодію між компонентами системи та дозволяє підтримувати роботу каталогу навіть при активній взаємодії користувачів із платформою. Наступний розділ буде присвячений проєктуванню інтерфейсу користувача, адаптивному інтерфейсу та організації взаємодії користувача з вебплатформою Spokuta.

5 ПРОЄКТУВАННЯ ІНТЕРФЕЙСУ КОРИСТУВАЧА

5.1 Концепція інтерфейсу

Після проєктування архітектури та бази даних наступним етапом стала розробка концепції інтерфейсу вебплатформи Spokuta. Інтерфейс повинен не просто відображати каталог фільмів і серіалів, а давати змогу швидко працювати з контентом: шукати потрібні записи, фільтрувати результати, переглядати інформацію, залишати оцінки й рецензії та додавати фільми або серіали до обраного чи списку перегляду.

Основна ідея інтерфейсу Spokuta полягає у простій та зрозумілій побудові сторінок без зайвого перевантаження. Для платформи з каталогом медіаконтенту це особливо важливо, оскільки на одній сторінці може відображатися багато карток, фільтрів, кнопок і допоміжної інформації. Якщо додати занадто багато другорядних елементів, навігація стає складнішою, а робота з каталогом – повільнішою.

Через це під час проєктування інтерфейсу головний акцент робився на:

- швидкому доступі до каталогу;
- зрозумілій навігації між сторінками;
- зручній роботі з пошуком і фільтрами;
- простому додаванні контенту до обраного та списку перегляду;
- швидкому виставленні оцінок і написанні рецензій;
- адаптації до ПК та мобільних пристроїв;
- зрозумілому зворотному зв’язку після дій.

Каталог побудований за картковим принципом. Кожен фільм або серіал відображається як окремий елемент із основною інформацією: назвою, постером, типом контенту, роком випуску та рейтингом. Такий підхід дозволяє швидко переглядати багато записів і не відкривати сторінку кожного фільму лише для базового ознайомлення.

Картки контенту виконують не тільки інформаційну, а й навігаційну

функцію. Через них можна перейти до детальної сторінки фільму або серіалу, де вже доступні розширений опис, оцінювання, текстова рецензія, дії зі списками та рекомендації. Це спрощує загальний сценарій роботи: спочатку короткий перегляд каталогу, потім – детальна взаємодія з обраним контентом.

На рисунку 5.1 представлена загальна структура головної сторінки вебплатформи Spokuta, яка демонструє побудову каталогу, картки фільмів і серіалів, навігацію та основні елементи взаємодії.

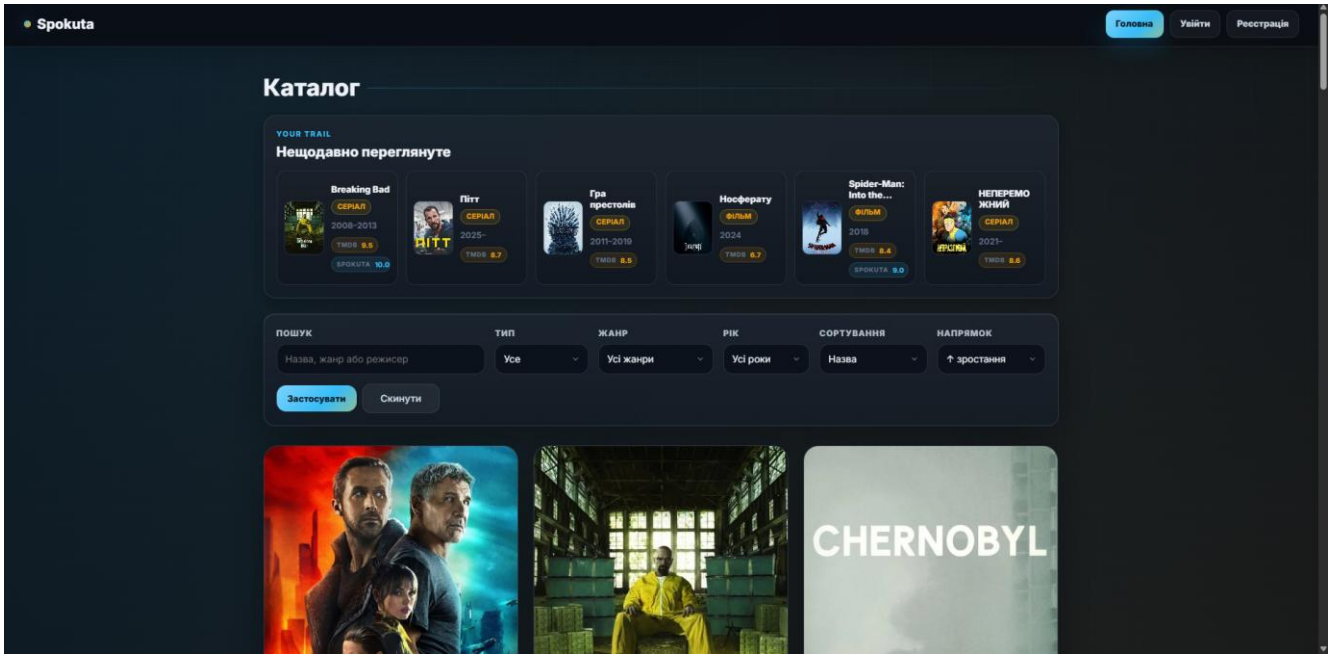

Рисунок 5.1 – Загальна структура головної сторінки вебплатформи Spokuta

Навігація в інтерфейсі побудована так, щоб основні розділи були доступні без зайвих переходів. Для неавторизованого відвідувача доступні каталог, пошук, фільтрація та сторінки окремих фільмів або серіалів. Цього достатньо, щоб ознайомитися з платформою та зрозуміти її можливості.

Після авторизації інтерфейс розширюється. Авторизований користувач отримує доступ до персональних функцій: оцінок, рецензій, обраного, списку перегляду, рекомендацій та профілю. При цьому базова структура сторінок не змінюється повністю, а лише доповнюється додатковими діями. Завдяки цьому після авторизації інтерфейс не змінюється кардинально, а залишається зрозумілим для користувача.

Для адміністратора передбачений окремий доступ до адміністративної панелі. Вона використовується для керування каталогом, додавання нового контенту, редагування записів та роботи з даними, отриманими через TMDB API. Адміністративна частина відокремлена від звичайного користувацького інтерфейсу, оскільки має інші задачі та потребує більшої кількості службових елементів.

Пошук і фільтрація є одними з головних елементів інтерфейсу. При великій кількості фільмів і серіалів звичайний перегляд каталогу стає незручним, тому користувач повинен мати змогу швидко звузити список результатів. У Spokuta для цього використовуються пошуковий рядок і фільтри, які дозволяють швидше знайти потрібний контент.

Важливо, що взаємодія з каталогом не повинна кожного разу супроводжуватися повним перезавантаженням сторінки. Для цього використовується асинхронна взаємодія. Під час зміни фільтрів, додавання до списків або оновлення оцінок і рецензій сервер обробляє запит окремо, а JavaScript оновлює тільки потрібну частину інтерфейсу.

Такий підхід робить роботу з платформою більш плавною. Наприклад, після додавання фільму до списку перегляду кнопка може одразу змінити стан, а на сторінці з’являється повідомлення про успішну дію. Користувачу не потрібно чекати повного перезавантаження сторінки або заново шукати той самий фільм у каталозі.

Для зворотного зв’язку після дій використовуються спливаючі повідомлення. Вони повідомляють про успішне збереження оцінки, додавання до обраного або списку перегляду, а також про помилку, якщо дія недоступна. Це зручно, бо повідомлення не перекривають основний інтерфейс і не змушують переходити на окрему сторінку з результатом.

Для позначення завантаження даних використовуються тимчасові візуальні блоки. Замість порожньої сторінки або різкої появи контенту користувач бачить елементи, які показують, що дані ще завантажуються. Це робить інтерфейс більш передбачуваним, особливо під час роботи з каталогом або рекомендаціями.

Окрему увагу було приділено адаптивності. Spokuta повинна залишатися зручною як на ПК, так і на смартфонах. Для цього картки каталогу, навігація, форми та кнопки перебудовуються залежно від ширини екрана. На великих екранах каталог показується у кілька колонок, а на мобільних пристроях елементи стають компактнішими та зручнішими для натискання.

Мобільна навігація має бути простою, оскільки на невеликому екрані складні меню швидко перевантажують сторінку. Тому основні дії повинні залишатися доступними, але не займати надто багато місця. Особливо це стосується пошуку, фільтрів, кнопок додавання до списків та переходу до профілю.

Концепція інтерфейсу Spokuta побудована навколо швидкої взаємодії з контентом. Користувач спочатку бачить каталог, далі може перейти до фільму або серіалу, поставити оцінку, залишити текстову рецензію, додати запис до обраного чи списку перегляду або переглянути рекомендації. Більшість цих дій виконується без зайвих переходів і без повного оновлення сторінки.

Такий підхід дозволяє поєднати простий зовнішній вигляд із достатньою кількістю функцій. Інтерфейс не перевантажується службовими елементами, але при цьому підтримує основні сценарії роботи з медіаконтентом. У наступному підрозділі буде детальніше розглянуто проєктування окремих сторінок вебплатформи Spokuta та їх роль у загальній логіці взаємодії з користувачем.

5.2 Проєктування сторінок вебплатформи

Після визначення загальної концепції інтерфейсу було спроєктовано основні сторінки вебплатформи Spokuta. Їхня структура побудована навколо кількох головних сценаріїв: перегляд каталогу, пошук контенту, робота з окремим фільмом або серіалом, оцінювання, додавання до персональних списків та адміністрування. Кожна сторінка має власне призначення, але всі вони пов’язані між собою єдиною логікою навігації.

Під час проєктування сторінок враховувалися ролі гостя, авторизованого користувача та адміністратора. Гість може переглядати відкриту частину платформи: каталог, пошук, фільтри та сторінки фільмів або серіалів. Після авторизації користувач отримує доступ до оцінок, рецензій, обраного, списку перегляду, рекомендацій і профілю. Адміністратор працює з окремою адміністративною панеллю, де доступні функції керування каталогом, імпорту через TMDB API та модерації.

Першою сторінкою, з якою може взаємодіяти неавторизований відвідувач, є сторінка авторизації. Вона використовується для входу в обліковий запис і має просту структуру без зайвих елементів. Основну частину сторінки займає форма входу з полями для електронної пошти або логіна та пароля. Після введення даних форма надсилається на сервер, де PHP перевіряє правильність облікових даних і створює сесію.

Якщо введені дані некоректні, користувач бачить повідомлення про помилку. Це важливо, оскільки людина одразу розуміє, що саме пішло не так: неправильний пароль, відсутній обліковий запис або порожні поля. На сторінці також передбачені навігаційні посилання для переходу до реєстрації або повернення до відкритої частини платформи.

На рисунку 5.2 представлена сторінка авторизації вебплатформи Spokuta.

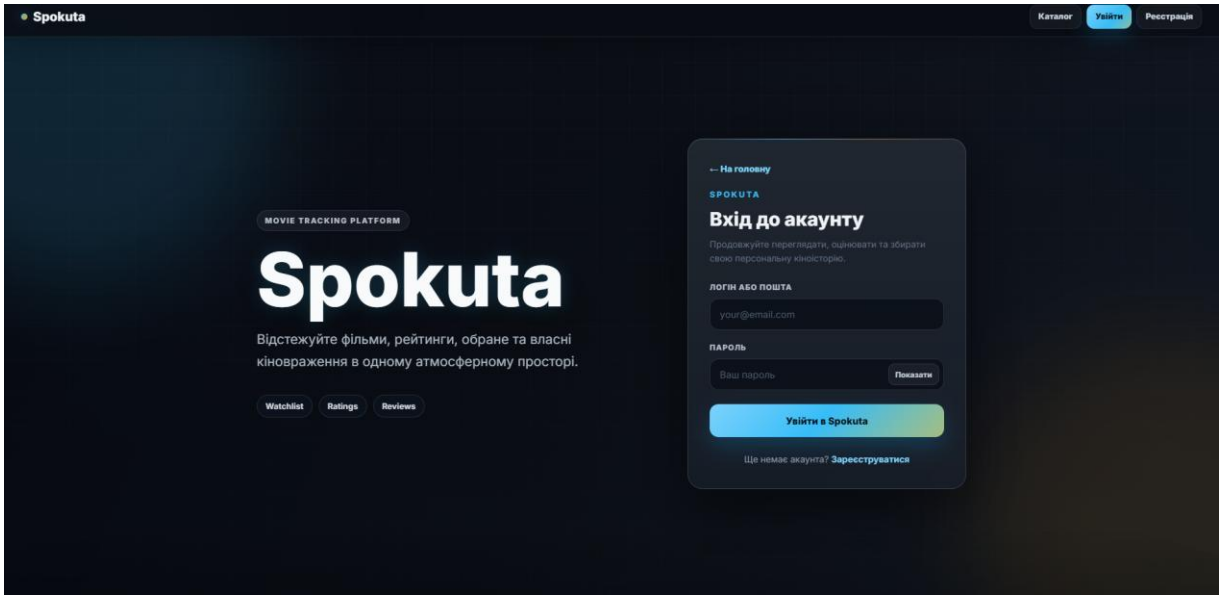

Рисунок 5.2 – Сторінка авторизації вебплатформи Spokuta

Сторінка реєстрації призначена для створення нового облікового запису. Вона побудована схожим чином, щоб користувач не витрачав час на пошук потрібних дій. У формі вводяться базові дані: ім’я користувача, електронна пошта та пароль. Перед збереженням серверна частина перевіряє коректність введених значень і наявність уже зареєстрованого облікового запису з такими даними.

Після успішної реєстрації користувач може перейти до авторизації та почати працювати з персональними функціями. Такий поділ між реєстрацією і входом робить логіку зрозумілою: спочатку створення облікового запису, потім вхід у систему. Для гостя ця сторінка є переходом від простого перегляду каталогу до повноцінної взаємодії з платформою.

На рисунку 5.3 представлена сторінка реєстрації вебплатформи Spokuta.

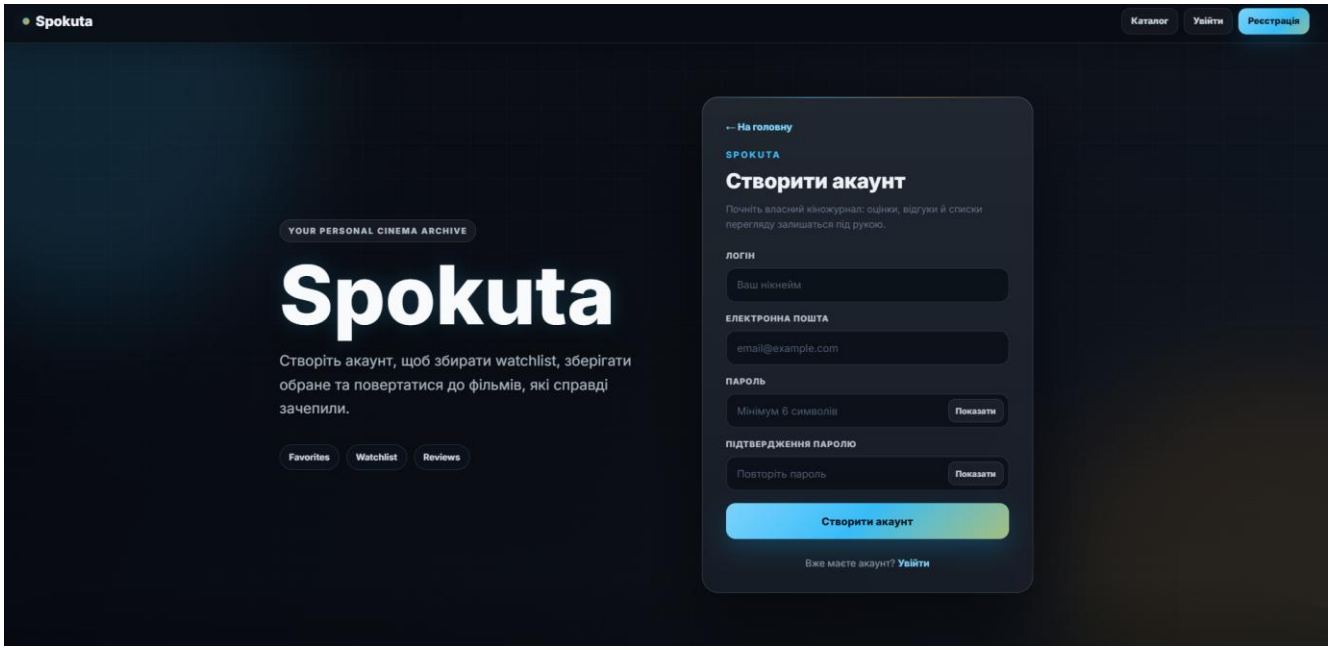

Рисунок 5.3 – Сторінка реєстрації вебплатформи Spokuta

Головна сторінка виконує роль каталогу фільмів і серіалів. Саме тут зібрані основні елементи для швидкого перегляду контенту: картки, постери, назви, тип контенту, рейтинг і елементи переходу до детальної сторінки. Такий підхід дозволяє швидко оцінити каталог візуально і не відкривати кожен запис окремо лише для базової інформації.

Картки контенту є головним елементом цієї сторінки. Вони мають достатньо

компактний вигляд, але при цьому містять основну інформацію про фільм або серіал. Постер допомагає швидко впізнати контент, а рейтинг і назва дають базове уявлення про запис. При натисканні на картку або відповідний елемент навігації відкривається сторінка конкретного фільму чи серіалу.

Система пошуку та фільтрації потрібна для роботи з великим каталогом. Якщо записів багато, звичайне прокручування сторінки стає незручним. Через пошуковий рядок можна знайти контент за назвою або частиною назви, а фільтри допомагають звузити результати за типом, жанром чи іншими параметрами, які використовуються в каталозі.

Частина взаємодії з головною сторінкою працює через асинхронні запити. Завдяки цьому зміна пошукового запиту або фільтрів не обов’язково потребує повного перезавантаження сторінки. Оновлюється лише потрібна частина каталогу, тому робота виглядає швидшою і менш різкою для користувача.

На рисунку 5.4 представлена головна сторінка каталогу.

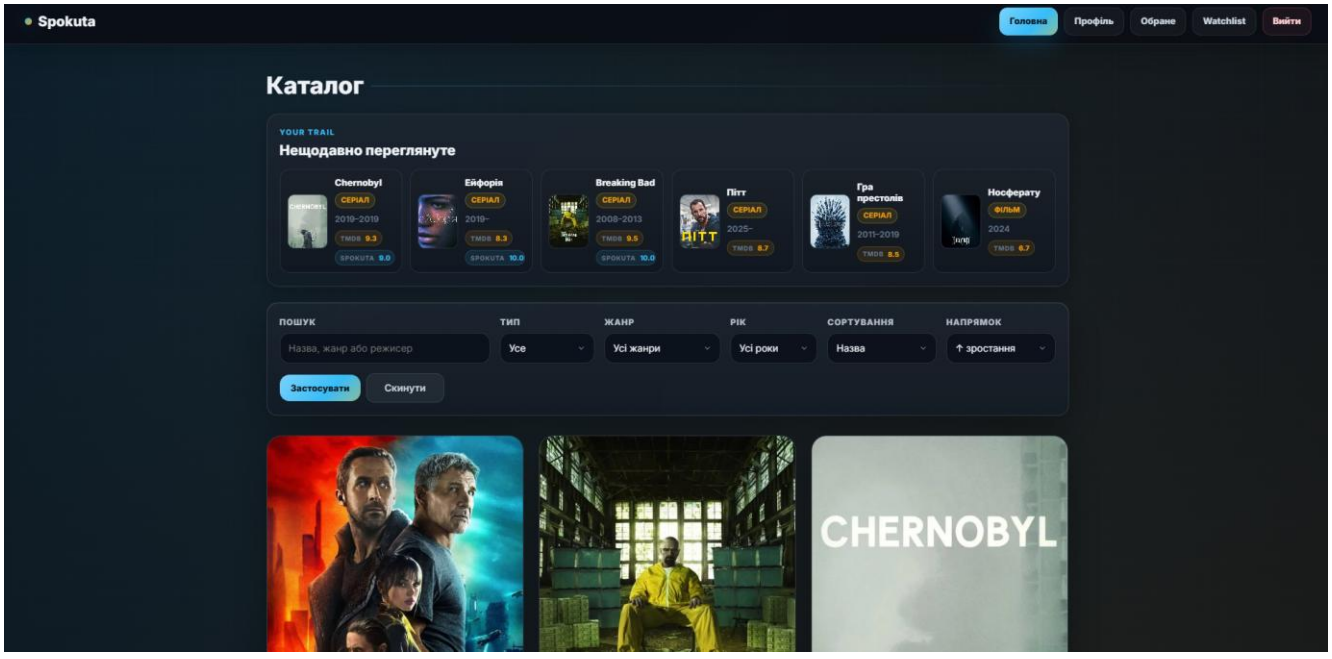

Рисунок 5.4 – Головна сторінка каталогу

Сторінка фільму або серіалу призначена для детального перегляду інформації про конкретний елемент каталогу. Вона відкривається після вибору запису з головної сторінки або з блоку рекомендацій. На цій сторінці розміщується

постер, назва, опис, жанри, рік випуску, рейтинг та інша інформація, яка зберігається локально або отримується через TMDB API.

Ця сторінка є центральною для взаємодії з контентом. Тут авторизований користувач може поставити оцінку, залишити текстову рецензію, додати фільм або серіал до обраного чи списку перегляду і перейти до схожого контенту через рекомендації. Для гостя сторінка працює у режимі перегляду: він може ознайомитися з інформацією, але персональні дії залишаються недоступними до авторизації.

На рисунку 5.5 представлена сторінка фільму або серіалу.

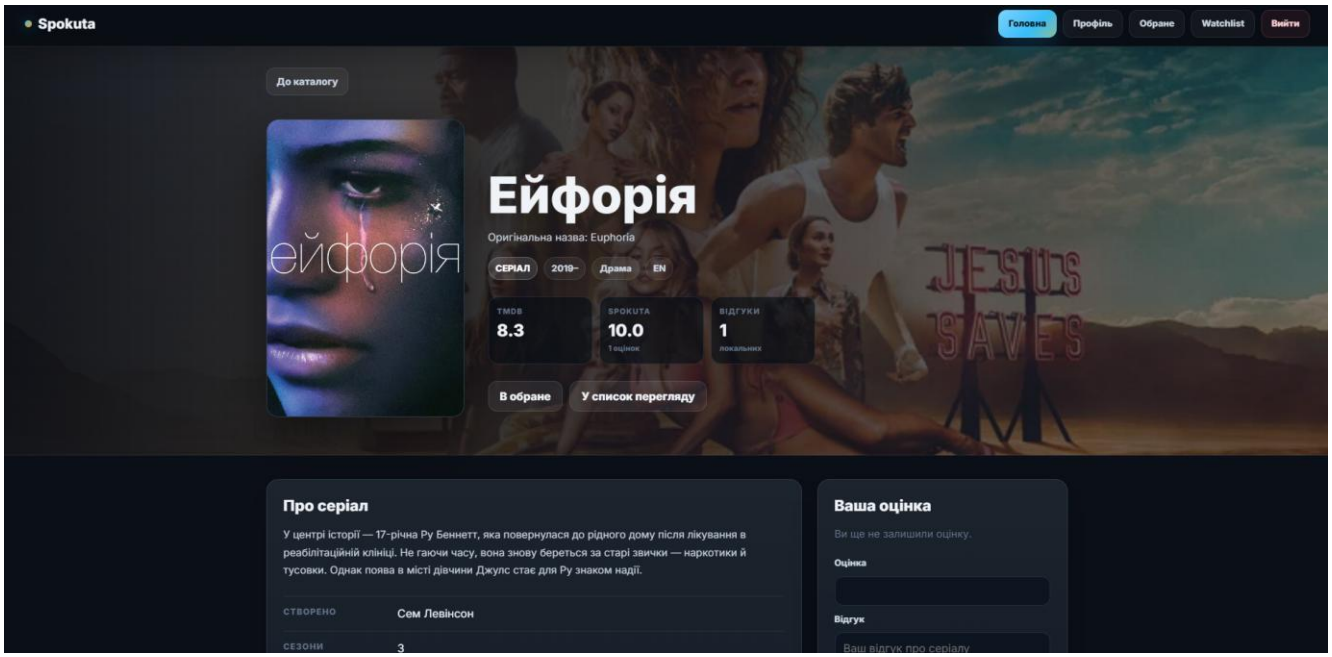

Рисунок 5.5 – Сторінка фільму або серіалу

Блок оцінок і рецензій дозволяє авторизованому користувачу взаємодіяти з контентом після перегляду. Він включає виставлення числової оцінки та можливість залишити текстову рецензію. Для користувача це виглядає як єдина функція оцінювання і рецензування, хоча на рівні бази даних текст рецензії не винесений в окрему таблицю.

У MySQL числова оцінка та текст рецензії зберігаються в таблиці ratings. Поле rating відповідає за оцінку, а review_text – за текстовий відгук. Така структура не створює зайвої сутності рецензій і дозволяє працювати з оцінкою та рецензією

як з одним записом, пов’язаним із конкретним користувачем і конкретним фільмом або серіалом.

Після виставлення оцінки або збереження рецензії JavaScript надсилає асинхронний запит до серверної частини. PHP перевіряє сесію, коректність movie_id, значення rating та текст review_text. Якщо все правильно, дані записуються або оновлюються у таблиці ratings. Потім інтерфейс показує оновлений результат без повного перезавантаження сторінки.

У цьому ж блоці можуть відображатися рецензії інших користувачів. Це дає можливість не лише поставити власну оцінку, а й побачити загальну реакцію на фільм або серіал. Після успішної дії з’являється спливаюче повідомлення, яке повідомляє про збереження оцінки або тексту рецензії.

На рисунку 5.6 представлений блок оцінок і рецензій.

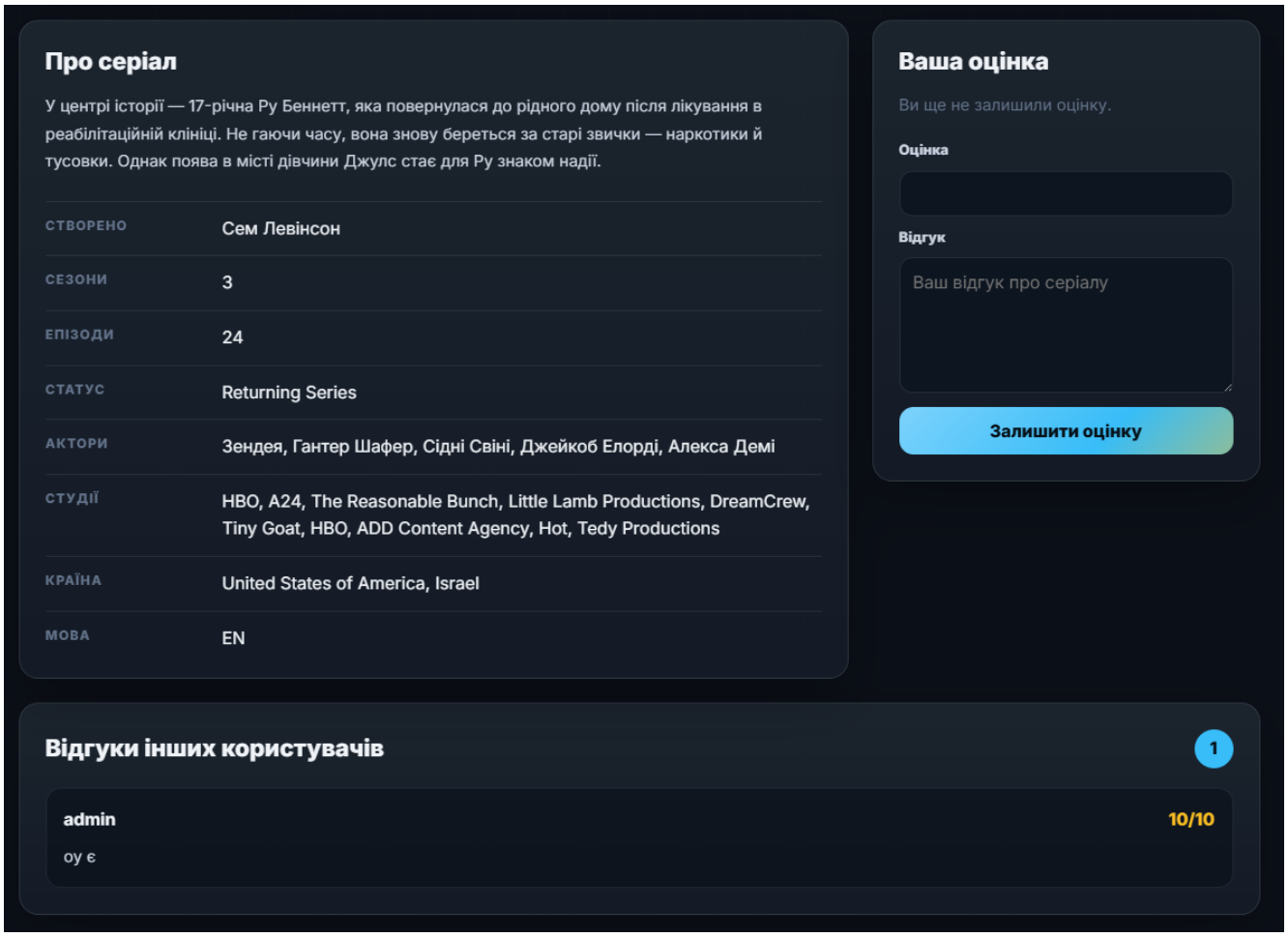

Рисунок 5.6 – Блок оцінок і рецензій

Блок рекомендацій використовується для продовження взаємодії з каталогом. Після перегляду сторінки фільму або серіалу користувач може одразу перейти до схожого або потенційно цікавого контенту. Це зменшує потребу повертатися на головну сторінку та знову вручну використовувати пошук.

Рекомендації можуть враховувати жанри, тип контенту, оцінки, рецензії або інші доступні дані платформи. У цій роботі рекомендації не розглядаються як складна система штучного інтелекту або машинного навчання. Це більш проста логіка добору контенту на основі вже наявних даних у каталозі та активності користувача.

З погляду інтерфейсу блок рекомендацій повинен бути помітним, але не заважати перегляду основної інформації. Тому він розміщується як додаткова секція на сторінці контенту або у профілі. Кожен рекомендований елемент може бути поданий у вигляді картки з постером, назвою та швидким переходом до сторінки фільму або серіалу.

На рисунку 5.7 представлений блок рекомендацій.

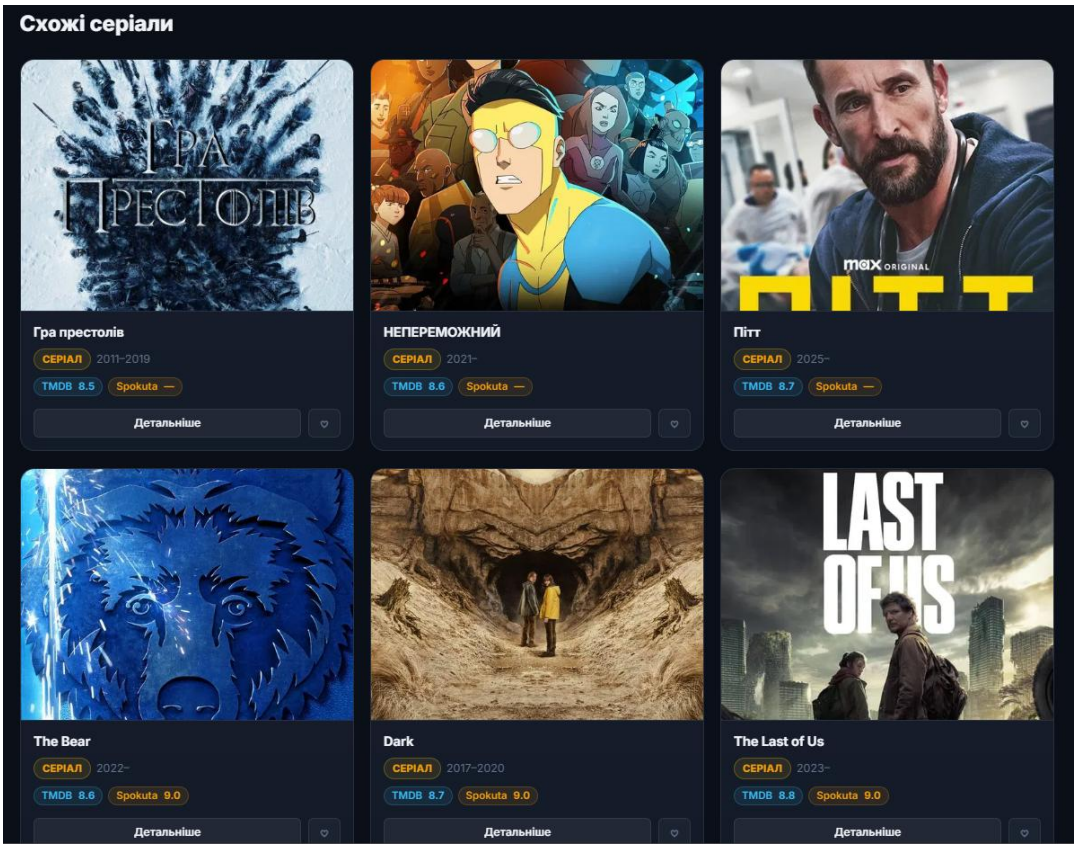

Рисунок 5.7 – Блок рекомендацій

Профіль користувача об’єднує персональну активність в одному місці. Після авторизації користувач може переглядати власне обране, список перегляду, раніше залишені оцінки й рецензії, рекомендації та частину історії взаємодії. Така сторінка потрібна для того, щоб користувач міг повернутися до свого контенту без повторного пошуку в каталозі.

Обране використовується для збереження улюблених фільмів і серіалів. Список перегляду потрібний для контенту, який користувач планує переглянути пізніше. Обидва списки мають схожу логіку, але різне призначення. Це робить профіль більш корисним, оскільки він працює як особиста бібліотека.

Оцінки й рецензії у профілі показують, з яким контентом користувач уже взаємодіяв. Якщо людина залишила оцінку або текстову рецензію, ця інформація може використовуватися для відображення активності та формування рекомендацій. Історія взаємодії допомагає краще показати шлях користувача всередині платформи, але не перевантажує інтерфейс зайвими деталями.

На рисунку 5.8 представлений профіль користувача.

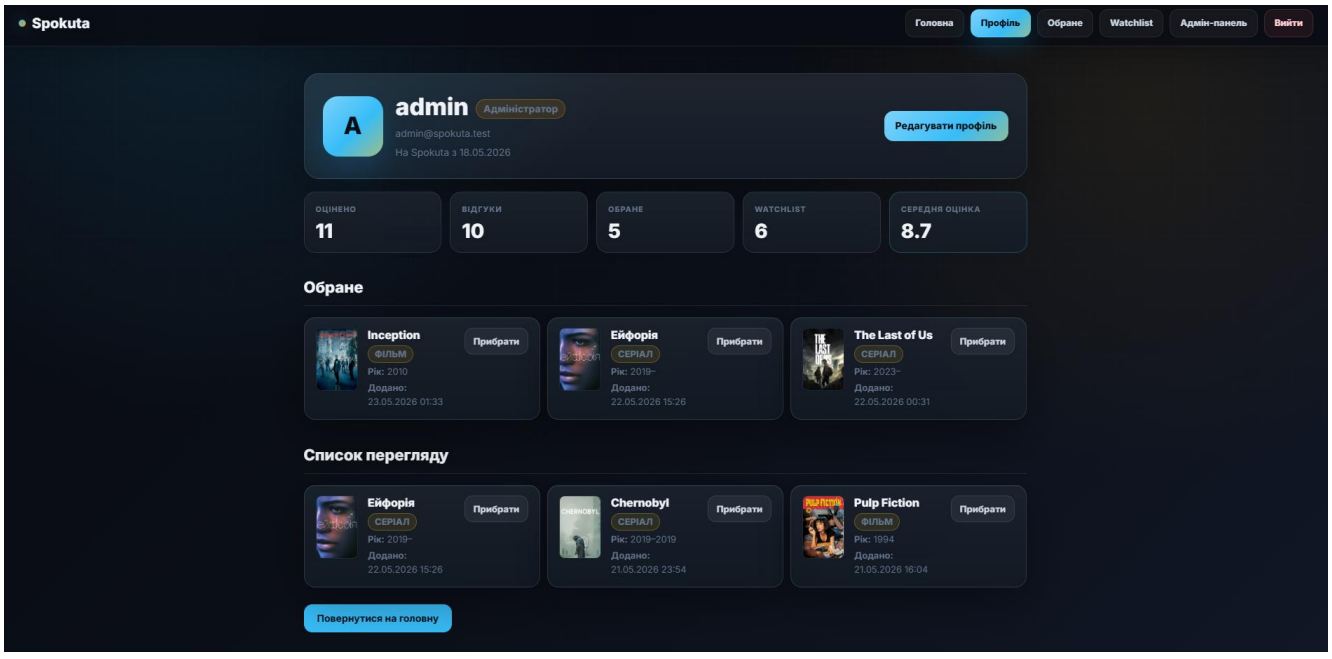

Рисунок 5.8 – Профіль користувача

Для виходу з облікового запису передбачене окреме вікно підтвердження. Воно потрібне, щоб користувач випадково не завершив сесію під час роботи з

платформою. Такий елемент особливо корисний, якщо кнопка виходу розміщена поруч із іншими навігаційними діями.

Вікно підтвердження містить коротке повідомлення і дві дії: підтвердити вихід або залишитися в обліковому записі. Після підтвердження PHP завершує сесію, а користувач повертається до відкритої частини платформи. Далі він знову має статус гостя і може переглядати каталог, але без доступу до персональних функцій.

На рисунку 5.9 представлено вікно підтвердження виходу з облікового запису.

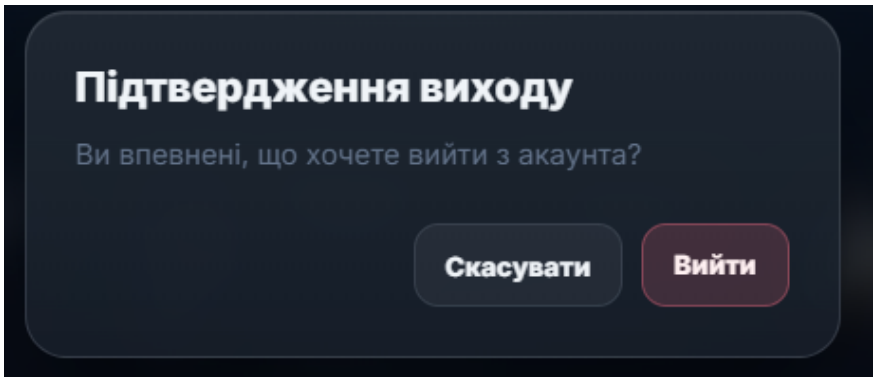

Рисунок 5.9 – Вікно підтвердження виходу з акаунта

Адміністративна панель призначена для керування основними даними вебплатформи. Доступ до неї має лише адміністратор. Перевірка ролі виконується на серверній стороні, тому звичайний авторизований користувач або гість не може перейти до адміністративних функцій навіть через пряме відкриття адреси сторінки.

Адміністративна панель має іншу логіку, ніж звичайна користувацька частина. Тут головний акцент зроблено не на перегляді контенту, а на керуванні ним. Адміністратор може додавати нові фільми та серіали, редагувати вже наявні записи, видаляти помилковий або зайвий контент, а також працювати з рецензіями користувачів через засоби модерації.

Для візуального подання статистичних даних в адміністративній панелі використовується бібліотека Chart.js. Вона дає змогу відображати статистику

платформи у вигляді діаграм і робить адміністративний інтерфейс більш наочним [24].

На рисунку 5.10 представлена адміністративна панель вебплатформи Spokuta.

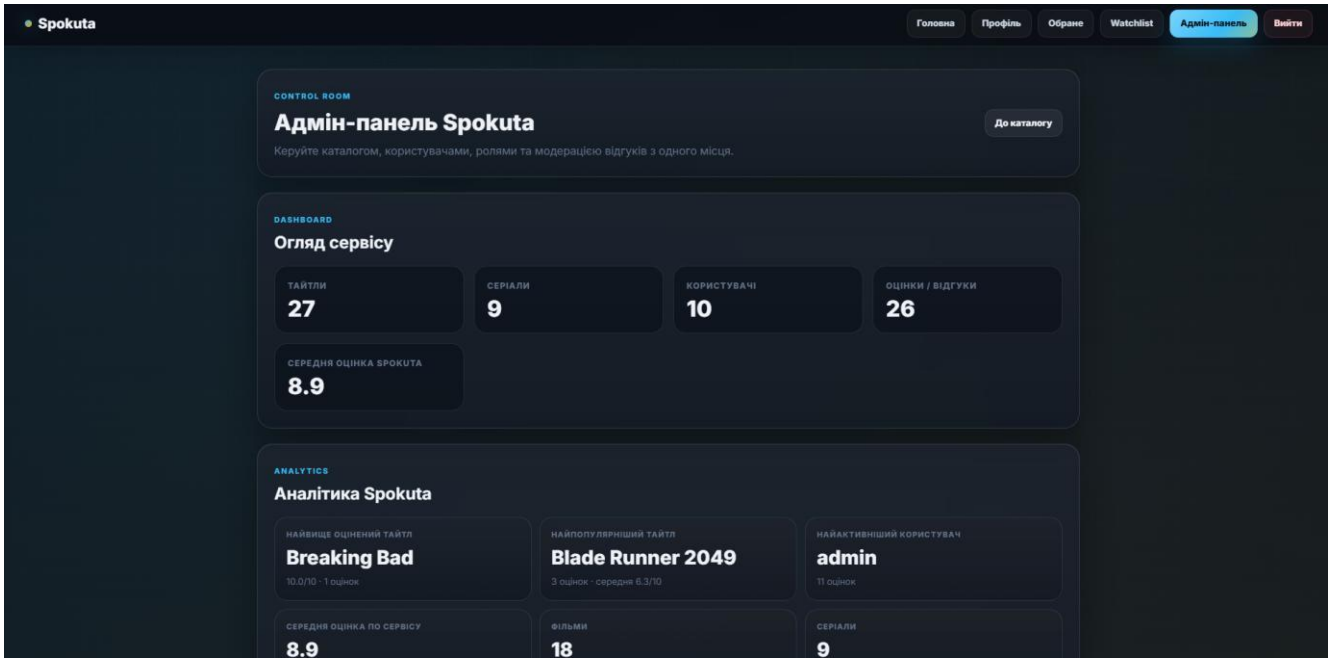

Рисунок 5.10 – Адміністративна панель вебплатформи Spokuta

Додавання контенту через адміністративну панель виконується за допомогою форми. У ній адміністратор може заповнити основні поля, потрібні для створення запису в каталозі. Це може бути назва, тип контенту, рік, жанр, рейтинг, постер або інші дані, які використовуються на сторінках платформи. Після надсилання форми PHP обробляє запит, перевіряє дані та зберігає запис у MySQL.

Якщо дія виконується через асинхронний запит, сторінка не перезавантажується повністю. Після успішного додавання інтерфейс може показати спливаюче повідомлення або оновити список записів у таблиці. Це пришвидшує роботу адміністратора, особливо коли потрібно додати або відредагувати кілька елементів каталогу поспіль.

На рисунку 5.11 представлено додавання контенту через адміністративну панель.

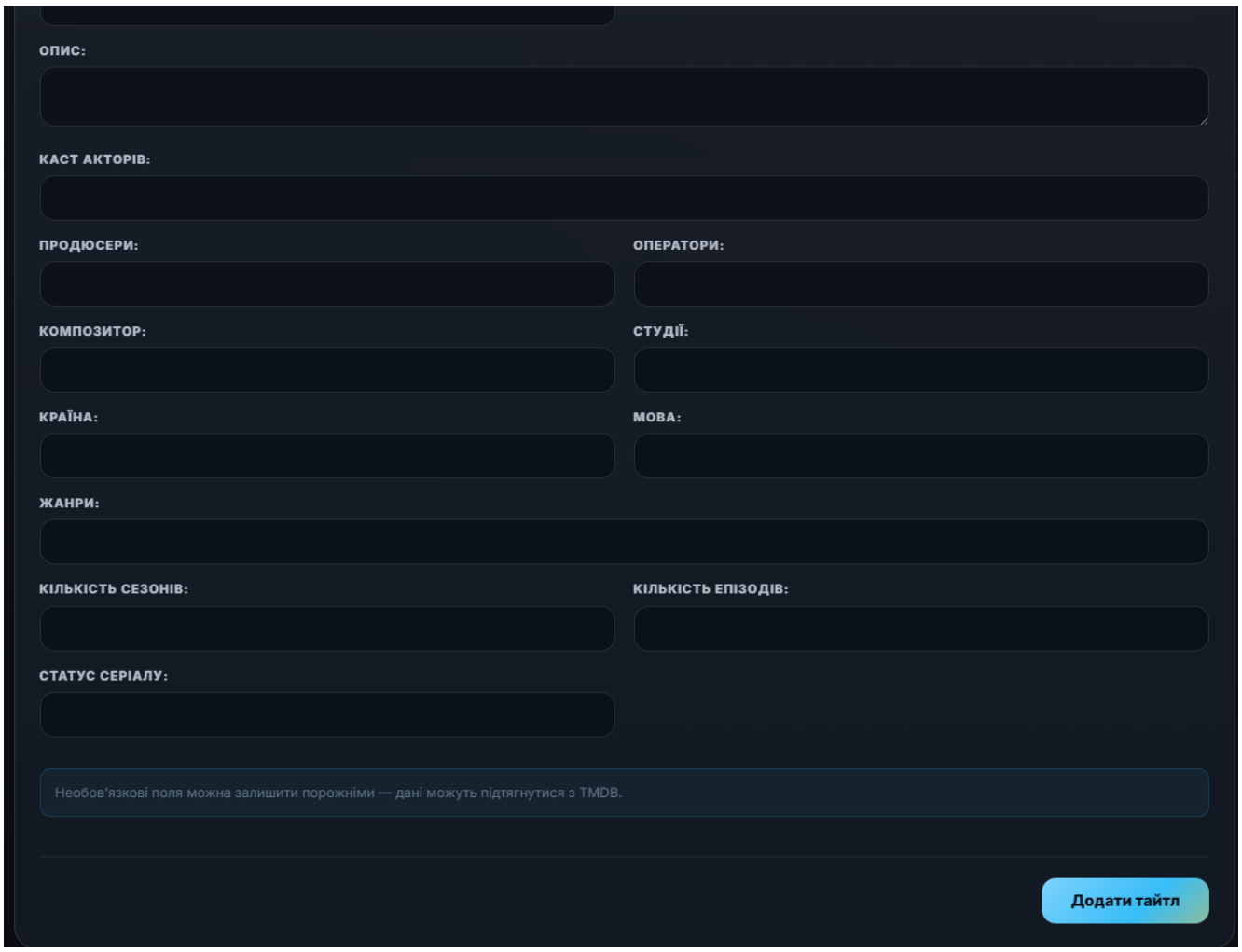

Рисунок 5.11 – Додавання контенту через адміністративну панель

Імпорт метаданих через TMDB API спрощує наповнення каталогу. Замість ручного введення всіх даних адміністратор може використати TMDB ID або пошук за назвою. Після цього PHP надсилає запит до TMDB API, отримує метадані та заповнює потрібні поля.

Через TMDB API можуть отримуватися назва, оригінальна назва, постер, жанри, рік випуску, рейтинг TMDB, тип контенту та інша інформація. Перед збереженням система перевіряє tmdb_id, щоб не створювати дублікати записів. Це важливо для підтримки чистого каталогу, де один і той самий фільм або серіал не дублюється кілька разів.

На рисунку 5.12 представлено імпорт метаданих через TMDB API.

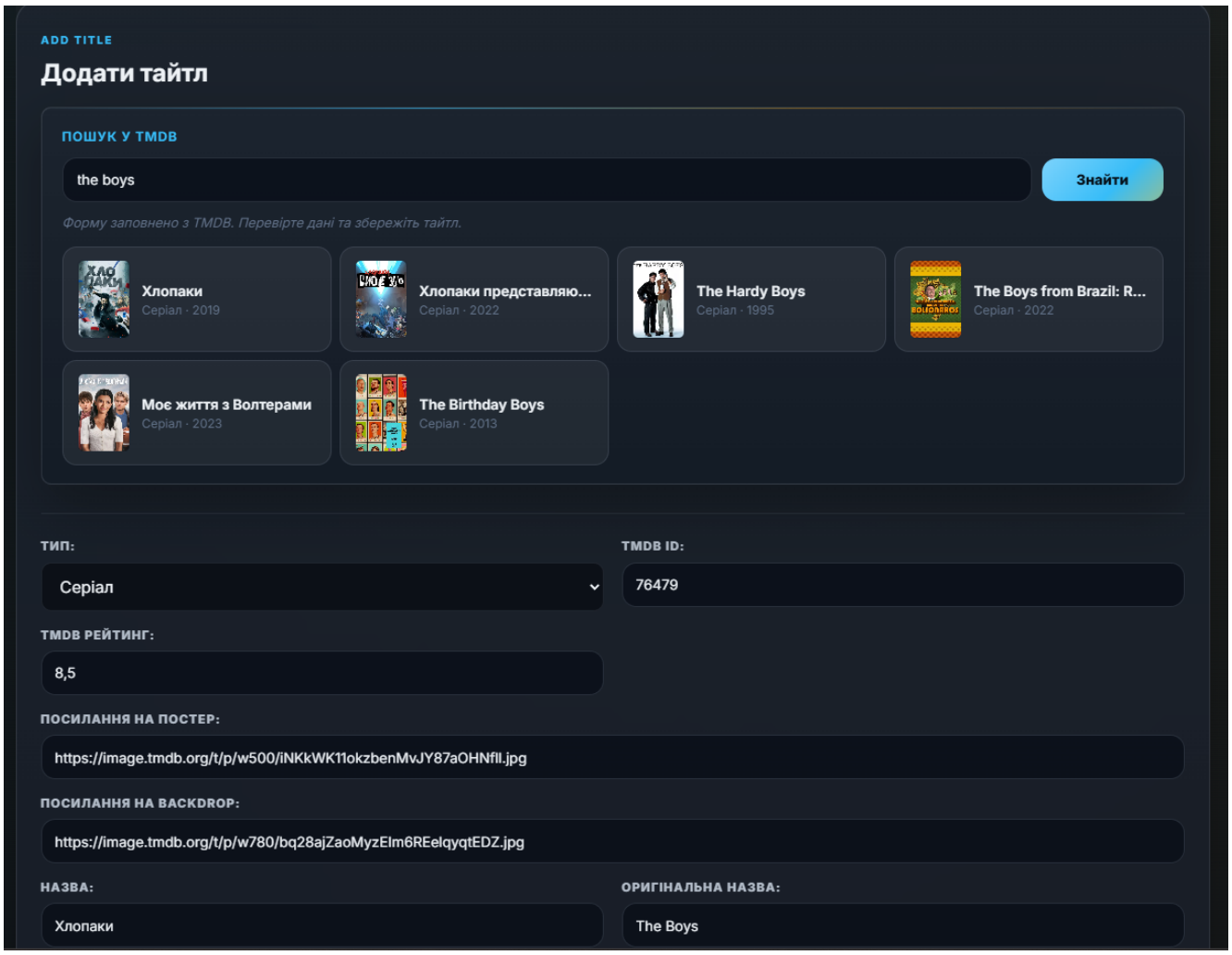

Рисунок 5.12 – Імпорт метаданих через TMDB API

Логіка навігації між сторінками побудована так, щоб користувач не губився під час роботи з платформою. З головної сторінки можна перейти до конкретного фільму або серіалу, зі сторінки контенту – до рекомендацій або персональних дій, з профілю – до власних списків і оцінених записів. Для адміністратора основний шлях проходить через адміністративну панель, де доступні форми та таблиці керування.

Загальна структура сторінок Spokuta побудована навколо практичних сценаріїв. Авторизація відкриває доступ до персональних функцій, каталог забезпечує швидкий пошук контенту, сторінка фільму або серіалу дає змогу оцінити й зберегти запис, профіль збирає активність, а адміністративна панель відповідає за керування даними. Така організація не перевантажує інтерфейс і

дозволяє поступово переходити від перегляду контенту до більш активної взаємодії.

У наступному підрозділі буде розглянуто адаптивний дизайн та адаптацію вебплатформи Spokuta до мобільних пристроїв, оскільки всі описані сторінки повинні залишатися зручними не лише на ПК, а й на мобільних пристроях.

5.3 Адаптивність інтерфейсу та мобільна адаптація

Під час проєктування інтерфейсу Spokuta окремо враховувалася робота сторінок на різних розмірах екрана. Вебплатформа орієнтована не лише на перегляд із ПК, а й на використання зі смартфонів, оскільки каталог фільмів і серіалів часто переглядають у швидких сценаріях: знайти фільм, додати його до списку перегляду, поставити оцінку або переглянути рекомендації. Через це адаптивний інтерфейс був закладений як одна з основних вимог до платформи.

Адаптація інтерфейсу реалізована без створення окремої мобільної версії сайту. Одна й та сама структура сторінок перебудовується залежно від ширини екрана. На великих екранах каталог може відображатися у кілька колонок, а на смартфонах елементи розміщуються компактніше, щоб не створювати горизонтальне прокручування і не ускладнювати взаємодію.

Під час проєктування адаптивного інтерфейсу також враховувалися загальні рекомендації щодо доступності вебконтенту, зокрема читабельність тексту, достатній розмір елементів взаємодії, зрозуміла навігація та коректне відображення сторінок на різних пристроях [25].

Головна сторінка каталогу найбільше залежить від адаптивного дизайну, оскільки саме на ній одночасно відображається багато карток контенту. Для ПК зручним є варіант із кількома колонками, де користувач бачить більше фільмів і серіалів одразу. На мобільних пристроях така структура була б занадто дрібною, тому картки перебудовуються в більш вузький формат і займають ширину, зручну

для перегляду зі смартфона.

Картки контенту адаптуються так, щоб постер, назва, рейтинг і основні дії залишалися читабельними. На малому екрані важливо не просто зменшити всі елементи, а змінити їх розташування. Якщо кнопки або текст стають надто дрібними, сторінка формально відкривається, але працювати з нею незручно. Тому в Spokuta картки зберігають достатній розмір для натискання і не виходять за межі екрана.

На рисунку 5.13 представлена головна сторінка Spokuta на мобільному пристрої.

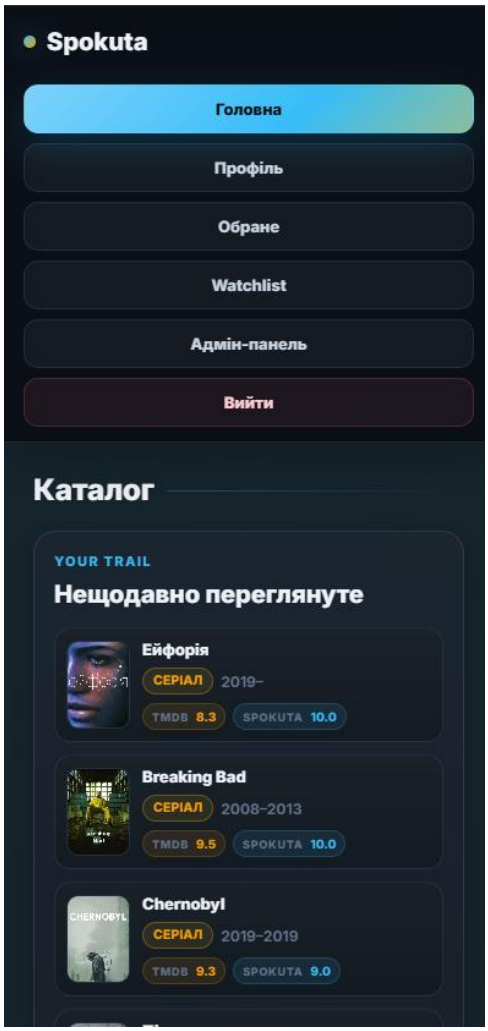

Рисунок 5.13 – Головна сторінка Spokuta на мобільному пристрої

Навігація на мобільних пристроях повинна займати менше простору, ніж на ПК. У версії для великих екранів можна використати ширшу навігаційну панель,

більше посилань і додаткові елементи. На смартфоні така структура швидко перевантажує верхню частину сторінки. Через це навігація адаптується до малого екрана: основні переходи залишаються доступними, але не займають надто багато місця.

Особливо важливо, щоб користувач міг швидко повернутися до каталогу, перейти до профілю, скористатися пошуком або вийти з облікового запису. Якщо такі дії сховані занадто глибоко, робота із сайтом стає повільнішою. Тому мобільна навігація у Spokuta будується навколо найчастіших сценаріїв, а другорядні елементи не повинні заважати перегляду контенту.

Сторінка фільму або серіалу також має іншу структуру на смартфоні. На великому екрані постер, опис, рейтинг, кнопки обраного й списку перегляду та блок оцінок і рецензій можуть розміщуватися поруч або у ширших секціях. На мобільному пристрої ці елементи краще розташовувати послідовно зверху вниз. Так користувач поступово переглядає інформацію: спочатку постер і назву, потім опис, жанри, рейтинг, дії зі списками, рецензії та рекомендації.

Такий підхід зменшує візуальне перевантаження. Сторінка не змушує користувача масштабувати екран або прокручувати її вбік. Взаємодія з оцінками та рецензіями також адаптується: поля введення, кнопки оцінювання та збереження рецензії повинні мати достатній розмір для натискання. Це особливо важливо, оскільки на смартфоні помилки при натисканні виникають частіше, ніж при роботі мишею.

Система пошуку та фільтрації також потребує адаптації. На ПК фільтри можуть розміщуватися ширше або бути доступними поруч із каталогом. На смартфоні велика кількість полів і кнопок може займати занадто багато місця, тому елементи фільтрації повинні бути компактними та зрозумілими. Користувач має швидко ввести назву, вибрати потрібний параметр і побачити оновлений список.

На рисунку 5.14 представлена сторінка фільму або серіалу на смартфоні.

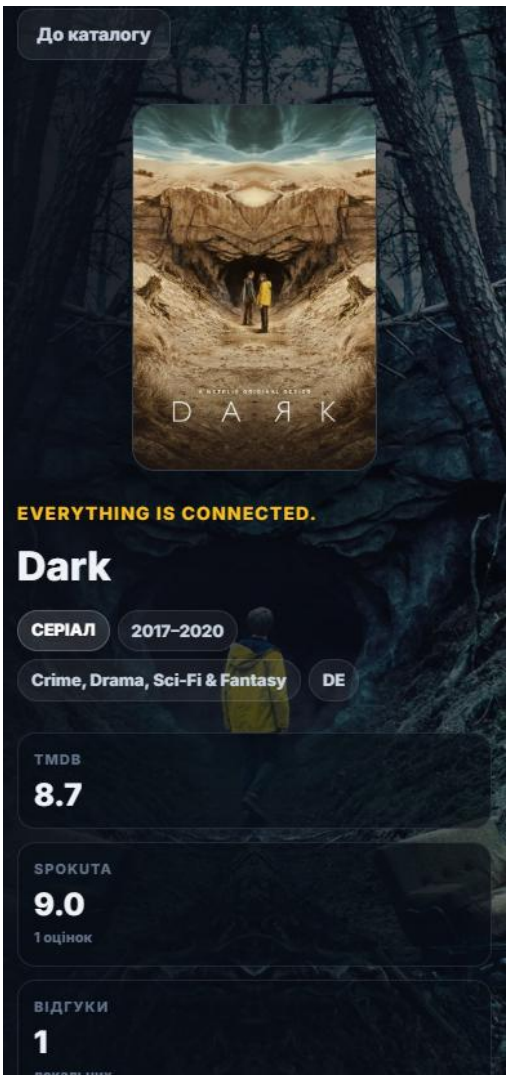

Рисунок 5.14 – Сторінка фільму або серіалу на смартфоні

Асинхронна взаємодія допомагає зробити мобільну роботу з каталогом швидшою. Повне перезавантаження сторінки на смартфоні відчувається сильніше, особливо якщо з’єднання нестабільне або каталог містить багато елементів. Коли оновлюється лише частина сторінки, взаємодія виглядає плавнішою. Це помітно під час фільтрації, додавання контенту до обраного або списку перегляду, а також під час збереження оцінок і рецензій.

Для позначення завантаження даних використовуються тимчасові візуальні блоки. Завдяки цьому під час завантаження сторінка не виглядає порожньою. На мобільному пристрої це сприймається краще, ніж різке появлення блоків після затримки. Тимчасові елементи показують, що дані завантажуються, а структура сторінки вже зрозуміла. Для каталогу та рекомендацій це особливо корисно,

оскільки там може підвантажуватися кілька карток одночасно.

Спливаючі повідомлення також адаптовані під швидкі дії. Після додавання фільму до списку перегляду, збереження оцінки або появи помилки користувач отримує коротке повідомлення. Воно не повинно перекривати основний контент або заважати подальшій роботі зі сторінкою. На смартфоні такі повідомлення мають бути помітними, але компактними.

На рисунку 5.15 представлена адаптація навігації та карток контенту.

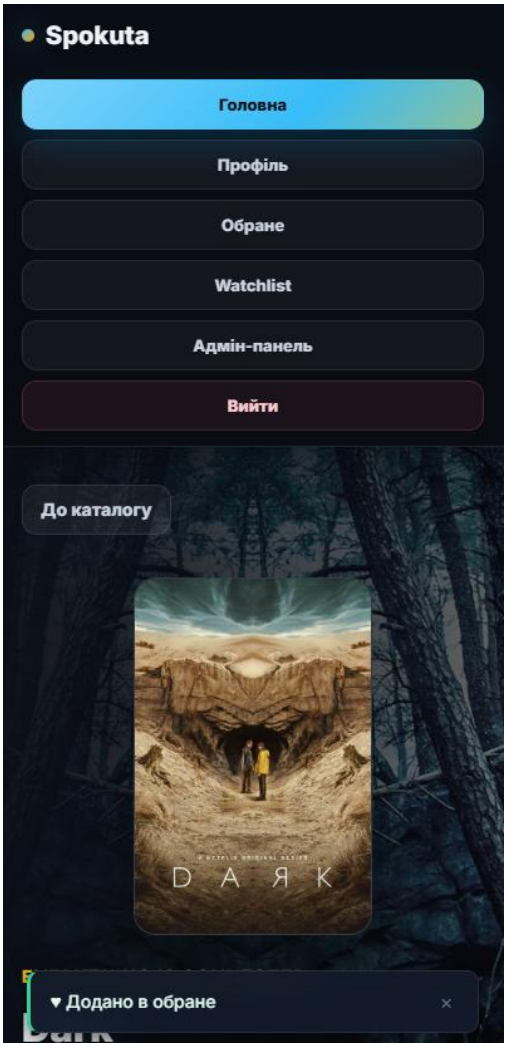

Рисунок 5.15 – Адаптація навігації та карток контенту

Форми авторизації та реєстрації також були враховані під час мобільної адаптації. На смартфоні поля введення повинні бути достатньо широкими, а кнопки – зручними для натискання. Форма не повинна виходити за межі екрана або вимагати горизонтального прокручування. Повідомлення про помилки

розміщуються поруч із формою або в помітному місці, щоб користувач одразу бачив причину невдалої дії.

Профіль користувача на мобільному пристрої будується послідовно. Обране, список перегляду, оцінки, рецензії та рекомендації краще розділяти на окремі блоки, які легко переглядати вертикально. Якщо показати все одразу в широкій таблиці або складній сітці, сторінка стане незручною. Тому персональна активність повинна бути подана у вигляді зрозумілих секцій.

Адміністративна панель є складнішою для адаптації, оскільки містить форми, таблиці, кнопки редагування, видалення та імпорту через TMDB API. На ПК така панель може бути ширшою і показувати більше службової інформації. На мобільному екрані частину елементів потрібно розміщувати вертикально, а таблиці та форми робити компактнішими.

При адаптації адміністративної панелі важливо зберегти доступність основних дій. Адміністратор повинен мати змогу додати контент, відредагувати запис, виконати імпорт через TMDB API або скористатися модерацією без порушення структури сторінки. При цьому службові елементи не мають перекривати одне одного або створювати горизонтальне прокручування.

Адаптивний дизайн у Spokuta побудований навколо гнучкої структури сторінок. Змінюється не сам функціонал, а спосіб його відображення. Каталог, сторінка контенту, профіль, форми та адміністративна панель залишаються тими самими, але їх елементи перебудовуються під конкретний розмір екрана. Це дозволяє підтримувати єдину логіку роботи для різних пристроїв.

Завдяки такому підходу користувач може почати роботу з ПК, а потім відкрити платформу зі смартфона і не вчитися заново користуватися інтерфейсом. Основні дії залишаються на своїх логічних місцях: пошук у каталозі, перехід до фільму, оцінки, рецензії, обране, список перегляду, рекомендації і профіль. Різниця полягає лише у тому, як ці блоки розміщуються на екрані.

На рисунку 5.16 представлена мобільна адаптація адміністративної панелі.

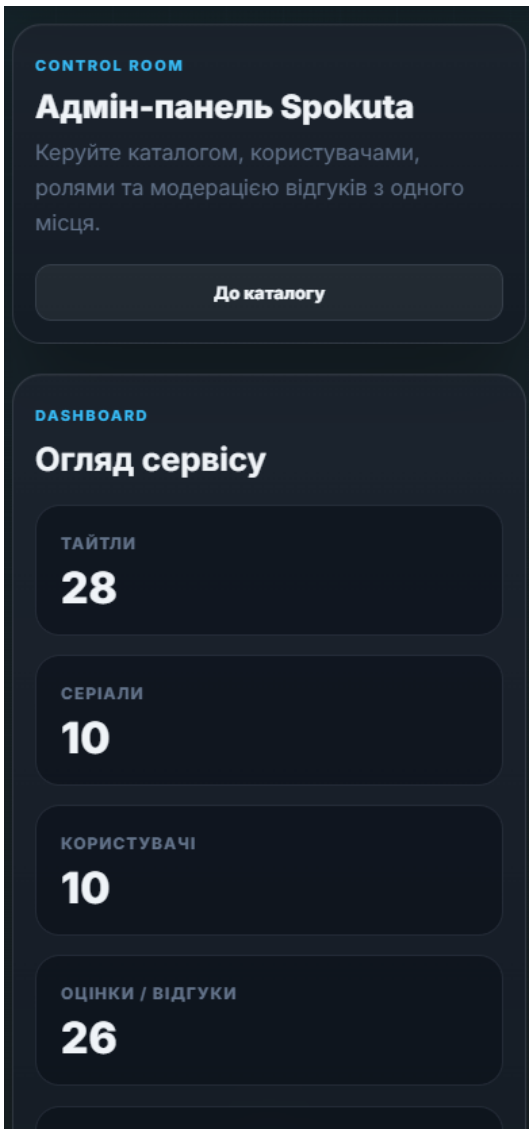

Рисунок 5.16 – Мобільна адаптація адміністративної панелі

Адаптивність напряму впливає на зручність роботи з вебплатформою. Якщо каталог нормально виглядає лише на ПК, значна частина сценаріїв стає менш комфортною. Для Spokuta це особливо важливо, бо взаємодія з фільмами та серіалами часто відбувається швидко: знайти, оцінити, додати до списку або переглянути рекомендацію. Адаптивний інтерфейс дозволяє виконувати ці дії без зайвого масштабування, горизонтального прокручування та постійного повернення до попередніх сторінок.

У наступному підрозділі буде розглянуто тестування вебплатформи Spokuta, зокрема перевірку основних сценаріїв роботи, асинхронної взаємодії, адаптивності інтерфейсу та функцій для різних ролей користувачів.

5.4 Тестування вебплатформи

Після проєктування інтерфейсу та адаптації сторінок було виконано ручне тестування вебплатформи Spokuta. Основна мета тестування полягала у перевірці того, чи коректно працюють ключові сценарії взаємодії з платформою: авторизація, пошук, фільтрація, оцінки, рецензії, обране, список перегляду, рекомендації, адміністративна панель та інтеграція TMDB API.

Тестування проводилось у локальному середовищі розробки з використанням PHP, MySQL, JavaScript та асинхронних запитів. Перевірялася не тільки наявність потрібних сторінок, а й поведінка інтерфейсу після дій користувача. Особлива увага приділялася сценаріям, у яких дані оновлюються без повного перезавантаження сторінки, оскільки асинхронна взаємодія є однією з основних частин Spokuta.

Під час тестування перевірялися такі напрями:

- авторизація та автентифікація;
- рольовий доступ гостя, авторизованого користувача та адміністратора;
- робота пошуку та фільтрації;
- збереження оцінок і рецензій;
- додавання та видалення обраного і списку перегляду;
- формування рекомендацій;
- робота адміністративної панелі;
- імпорт метаданих через TMDB API;
- асинхронне оновлення інтерфейсу;
- адаптивність інтерфейсу на різних розмірах екрана;
- обробка помилкових або неповних даних.

Спочатку перевірялися позитивні сценарії. До них належать успішна реєстрація, вхід у систему з правильними даними, пошук фільму за назвою, фільтрація каталогу, додавання контенту до списку перегляду, збереження оцінки та текстової рецензії. Такі сценарії показують, чи працює основна логіка

платформи у звичайних умовах.

Окремо виконувалися негативні сценарії. Наприклад, перевірявся вхід із неправильним паролем, спроба залишити оцінку без авторизації, відкриття адміністративної панелі звичайним користувачем або гостем, надсилання порожніх форм та некоректних значень. Такі перевірки потрібні для того, щоб система не тільки виконувала правильні дії, а й коректно реагувала на помилки.

Для авторизації та автентифікації перевірялася робота сторінок входу та реєстрації. При правильних даних створювалася сесія, після чого користувач отримував доступ до персональних функцій. При неправильному паролі або порожніх полях система не повинна авторизувати користувача. У таких випадках інтерфейс показує повідомлення про помилку, а доступ до закритих функцій не відкривається.

Рольовий доступ перевірявся окремо для гостя, авторизованого користувача та адміністратора. Гість може переглядати каталог, користуватися пошуком і відкривати сторінки контенту, але не може залишати оцінки, рецензії або додавати фільми до персональних списків. Авторизований користувач після входу отримує доступ до оцінок, рецензій, обраного, списку перегляду, рекомендацій і профілю. Адміністратор додатково має доступ до адміністративної панелі і функцій керування каталогом.

Система пошуку та фільтрації перевірялася через різні варіанти пошукових запитів. Використовувалися повні назви, часткові назви та порожній пошук. Також перевірялася фільтрація за доступними параметрами каталогу. Очікуваний результат полягав у тому, що система повинна показувати релевантні записи й не порушувати структуру сторінки при відсутності результатів.

Під час тестування оцінок і рецензій перевірялося виставлення числової оцінки та збереження текстової рецензії. У структурі бази даних ці дані зберігаються в таблиці ratings: поле rating відповідає за оцінку, а review_text – за текст рецензії. Окрема таблиця reviews не використовується. Це було важливо перевірити, щоб логіка інтерфейсу відповідала реальній структурі MySQL.

На рисунку 5.17 представлена перевірка роботи оцінок і рецензій.

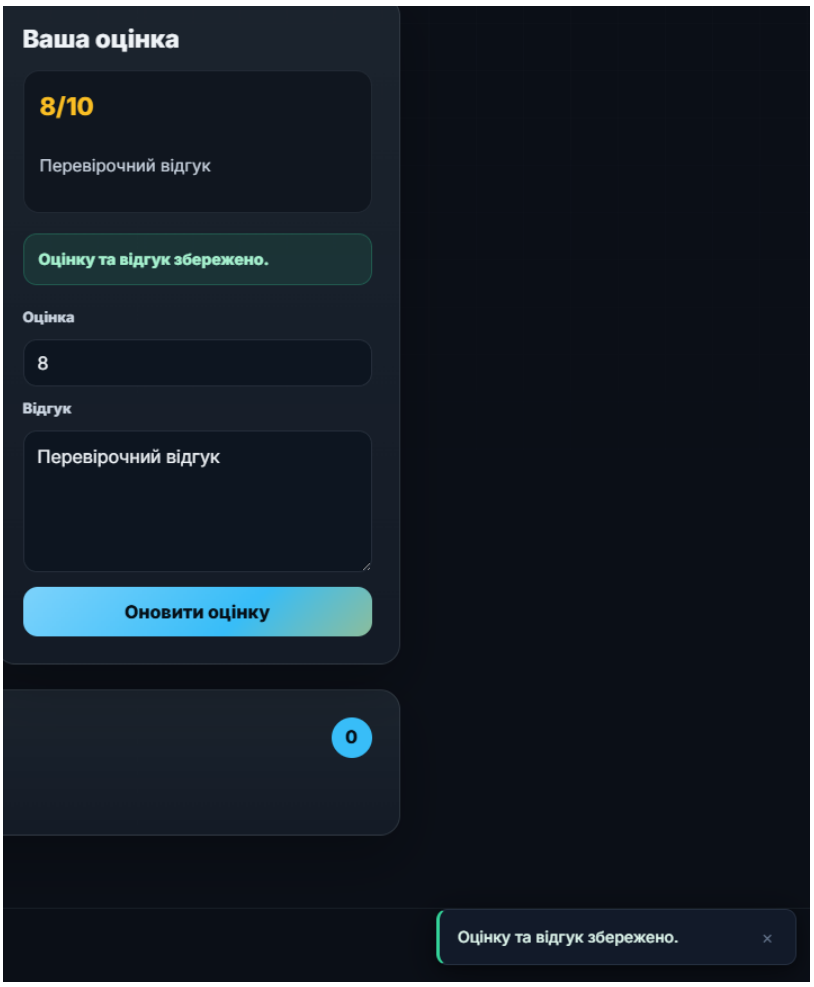

Рисунок 5.17 – Перевірка роботи оцінок і рецензій

Обране та список перегляду тестувалися через додавання та видалення контенту з персональних списків. Після натискання відповідної кнопки система повинна була зберегти зв’язок між користувачем і фільмом або серіалом. При повторній дії стан кнопки мав змінюватися, а інтерфейс – показувати результат без повного перезавантаження сторінки.

Рекомендації перевірялися через перегляд блоків рекомендованого контенту на сторінці фільму або в профілі. Основна перевірка полягала в тому, щоб рекомендовані записи відкривалися коректно, не дублювали помилкові посилання і дозволяли продовжити роботу з каталогом без повернення на головну сторінку.

Адміністративна панель тестувалася окремо, оскільки вона має розширені права доступу. Перевірялися додавання нового контенту, редагування записів, видалення фільмів або серіалів, модерація рецензій та робота форм. Також перевірявся доступ до адміністративної панелі: звичайний авторизований

користувач і гість не повинні мати можливості відкрити її функції.

Інтеграція TMDB API перевірялася через імпорт метаданих. Під час тестування адміністратор вводив назву або TMDB ID, після чого система отримувала інформацію про фільм або серіал. Перевірялося, чи заповнюються потрібні поля, чи зберігається tmdb_id, чи не створюються дубльовані записи при повторному імпорті одного й того самого контенту.

Перевірка асинхронного оновлення виконувалася для дій, які повинні оновлювати інтерфейс без повного перезавантаження сторінки. До таких дій належать оцінки, рецензії, обране, список перегляду, частина пошуку й фільтрації та окремі дії в адміністративній панелі. Перевірялося, чи змінюється стан кнопок, чи з’являються спливаючі повідомлення і чи не скидається поточна сторінка після виконання дії.

Перевірка адаптивності проводилася через зміну розміру вікна браузера та перегляд сторінок у режимі мобільного екрана. Перевірялися головна сторінка каталогу, сторінка фільму або серіалу, форми авторизації та реєстрації, профіль і адміністративна панель. Основна вимога – відсутність горизонтального прокручування, коректне відображення карток, зручні кнопки та читабельні форми.

На рисунку 5.18 представлена перевірка мобільної адаптації.

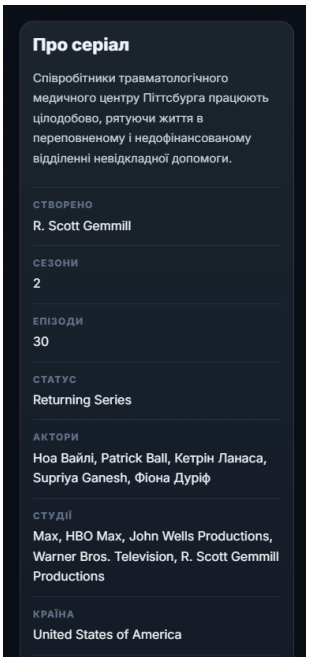

Рисунок 5.18 – Перевірка мобільної адаптації

Повторне тестування виконувалося після змін у коді або інтерфейсі. Після виправлення окремої функції повторно перевірялися пов’язані сценарії, щоб переконатися, що зміни не порушили інші частини платформи. Наприклад, після змін у роботі оцінок і рецензій додатково перевірялися профіль, рекомендації і відображення рейтингу на сторінці контенту.

Основні результати ручного тестування вебплатформи Spokuta наведено у таблиці 5.1.

Таблиця 5.1 – Результати тестування вебплатформи Spokuta

№	Сценарій тестування	Очікуваний результат	Фактичний результат	Статус
1	Вхід із правильними обліковими даними	Користувач авторизується, створюється сесія, відкриваються персональні функції	Авторизація виконана, доступ до профілю та персональних дій відкрито	Успішно
2	Вхід із неправильним паролем	Система не виконує авторизацію та показує повідомлення про помилку	Вхід заблоковано, повідомлення про помилку відображається	Успішно
3	Реєстрація нового користувача	Новий обліковий запис створюється після заповнення коректних даних	Користувач створений, дані збережені у MySQL	Успішно
4	Реєстрація з порожніми або некоректними полями	Система не створює обліковий запис і повідомляє про помилку	Реєстрація не виконується, помилка показується	Успішно
5	Пошук фільму за назвою	Каталог показує записи, що відповідають пошуковому запиту	Результати пошуку відображаються коректно	Успішно
6	Фільтрація каталогу	Після вибору фільтра список контенту оновлюється	Каталог оновлюється відповідно до вибраного фільтра	Успішно
7	Збереження оцінки	Оцінка записується у поле `rating` таблиці `ratings`	Оцінка збережена, інтерфейс оновився без повного перезавантаження	Успішно
8	Збереження текстової рецензії	Текст рецензії записується у поле `review_text` таблиці `ratings`	Рецензія збережена в межах таблиці `ratings`, повідомлення показано	Успішно
9	Додавання фільму до обраного	Запис додається до персонального списку обраного	Фільм додано, стан кнопки оновлено	Успішно

Продовження таблиці 5.1

№	Сценарій тестування	Очікуваний результат	Фактичний результат	Статус
10	Додавання фільму до списку перегляду	Запис додається до списку перегляду	Фільм додано до списку перегляду, інтерфейс оновився	Успішно
11	Перегляд рекомендацій	Відображаються рекомендовані фільми або серіали з переходом на сторінку контенту	Блок рекомендацій працює, переходи відкривають відповідні сторінки	Успішно
12	Імпорт контенту через TMDB API	Дані про фільм або серіал отримуються з API та заповнюють поля каталогу	Метадані отримані та підготовлені до збереження	Успішно
13	Спроба гостя виконати персональну дію	Система обмежує доступ і пропонує авторизацію	Персональна дія недоступна без входу	Успішно
14	Спроба авторизованого користувача відкрити адміністративну панель	Доступ до адміністративної панелі заборонено	Користувач не отримує доступ до адміністративної панелі	Успішно
15	Перевірка адаптивного інтерфейсу	Сторінки коректно адаптуються до мобільного екрана без горизонтального прокручування	Каталог, форми та сторінка контенту відображаються коректно	Успішно
16	Асинхронне оновлення без перезавантаження сторінки	Після дії оновлюється лише потрібний елемент інтерфейсу	Стан кнопок, оцінки, рецензії та повідомлення оновлюються без повного перезавантаження	Успішно

Результати тестування показали, що основні сценарії роботи вебплатформи виконуються коректно. Авторизація, реєстрація, пошук, фільтрація, оцінки, рецензії, обране, список перегляду, рекомендації та адміністративна панель працюють відповідно до заданої логіки. Негативні сценарії також обробляються очікувано: система не дозволяє виконувати закриті дії без авторизації, не відкриває адміністративну панель для звичайного користувача та реагує на некоректні дані у формах.

На рисунку 5.19 представлено тестування адміністративної панелі.

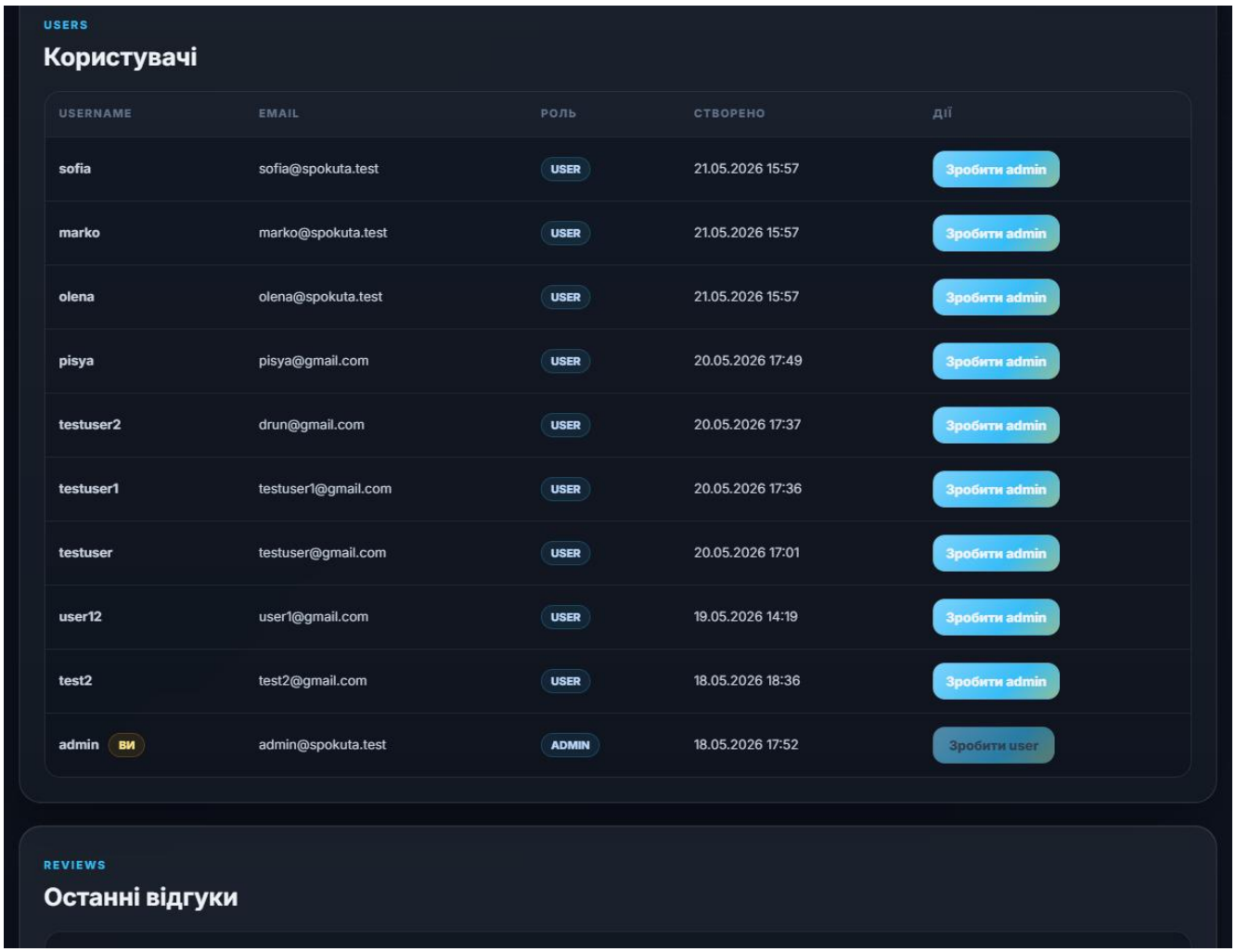

Рисунок 5.19 – Тестування адміністративної панелі

Після перевірки окремих функцій додатково оцінювалася стабільність логіки навігації. Користувач може переходити від каталогу до сторінки фільму або серіалу, далі – до рекомендацій, профілю або персональних списків. При цьому основна структура інтерфейсу не порушується, а асинхронне оновлення не скидає поточний сценарій роботи.

У наступному підрозділі буде сформовано висновки до розділу, де буде коротко підсумовано проєктування інтерфейсу, адаптацію сторінок та результати тестування вебплатформи Spokuta.

5.5 Висновки до розділу

У п’ятому розділі було розглянуто проєктування інтерфейсу користувача вебплатформи Spokuta. Основна увага приділялася тому, щоб інтерфейс був зручним для роботи з каталогом фільмів і серіалів, не перевантажував сторінки зайвими елементами та дозволяв швидко виконувати основні дії: пошук, фільтрацію, оцінювання, написання рецензій і додавання контенту до персональних списків.

Було сформовано загальну концепцію інтерфейсу, у якій головний акцент зроблено на простій навігації, картковому відображенні контенту та швидкій взаємодії без зайвих переходів між сторінками. Каталог Spokuta побудований так, щоб користувач міг одразу бачити основну інформацію про фільми та серіали, переходити до детальної сторінки контенту, працювати з оцінками, рецензіями, обраним, списком перегляду та рекомендаціями.

Окремо були описані основні сторінки вебплатформи: сторінки авторизації та реєстрації, головна сторінка каталогу, сторінка фільму або серіалу, профіль користувача, блоки оцінок, рецензій і рекомендацій, вікно підтвердження виходу з облікового запису та адміністративна панель. Для кожної сторінки було визначено її призначення, основні елементи інтерфейсу та місце в загальній логіці навігації.

Таке розділення сторінок дозволило зробити структуру вебплатформи більш послідовною. Користувач поступово переходить від перегляду каталогу до детальної взаємодії з конкретним фільмом або серіалом, а після авторизації отримує доступ до персональних функцій без зміни загальної логіки інтерфейсу.

У розділі також було враховано різницю між ролями гостя, авторизованого користувача та адміністратора. Гість має доступ до відкритої частини платформи, зокрема каталогу, пошуку та сторінок контенту. Авторизований користувач після входу в систему отримує доступ до персональних функцій. Адміністратор працює з адміністративною панеллю, через яку може додавати, редагувати й видаляти контент, виконувати імпорт через TMDB API та працювати з модерацією рецензій.

Значна увага була приділена адаптивному інтерфейсу та мобільній адаптації. Сторінки Spokuta проєктувалися так, щоб зберігати зручність як на ПК, так і на мобільних пристроях. Для цього структура каталогу, картки контенту, форми, навігація, профіль користувача та адміністративна панель адаптуються до різних розмірів екрана без створення окремої мобільної версії.

Асинхронна взаємодія стала важливою частиною інтерфейсу. Завдяки їй окремі дії виконуються без повного перезавантаження сторінки: оновлення оцінок і рецензій, робота з обраним і списком перегляду, частина пошуку, фільтрації та дій в адміністративній панелі. Це робить взаємодію з каталогом швидшою і зручнішою, особливо під час активної роботи з фільмами та серіалами.

У підрозділі тестування було перевірено основні сценарії роботи вебплатформи. Ручне тестування охопило авторизацію та автентифікацію, рольовий доступ, пошук і фільтрацію, оцінки й рецензії, обране та список перегляду, рекомендації, інтеграцію TMDB API, адміністративну панель, асинхронне оновлення інтерфейсу та адаптивність сторінок.

Також були перевірені негативні сценарії: некоректний пароль, порожні форми, спроби гостя виконати закриті дії та спроби звичайного користувача отримати доступ до адміністративної панелі. Результати тестування показали, що система коректно реагує на такі ситуації, не відкриває закриті функції без авторизації та перевіряє роль користувача перед доступом до адміністративної частини.

За результатами розділу було підтверджено, що інтерфейс Spokuta відповідає основним вимогам до вебплатформи для зберігання та оцінювання фільмів і серіалів. Він підтримує зручну роботу з каталогом, персональні функції для авторизованих користувачів, адміністративне керування, адаптацію до різних пристроїв та швидке оновлення даних через асинхронні запити. Проведене проєктування і тестування показало, що користувацька частина платформи є логічно завершеною та готовою до використання в межах поставлених задач дипломної роботи.

ВИСНОВКИ

У межах дипломної роботи була розроблена вебплатформа Spokuta для зберігання, пошуку та оцінювання фільмів і серіалів. Мета роботи була досягнута шляхом створення вебзастосунку, який поєднує каталог медіаконтенту, систему оцінок і рецензій, персональні списки користувача, базові рекомендації, адміністративне керування та інтеграцію із зовнішнім API.

Під час виконання роботи було проаналізовано існуючі сервіси для роботи з медіаконтентом, зокрема IMDb, Letterboxd та TMDB. На основі аналізу були сформовані функціональні й нефункціональні вимоги до вебплатформи, визначені основні ролі користувачів та обґрунтована необхідність створення власного рішення з підтримкою каталогу, оцінювання, персональних списків і адаптивного інтерфейсу.

У процесі проєктування було сформовано UML–моделі вебплатформи, зокрема діаграму варіантів використання, діаграми послідовності та діаграму активності. Це дозволило описати основні сценарії взаємодії користувачів із платформою, логіку авторизації, роботу з каталогом, оцінками, рецензіями, персональними списками та адміністративними функціями.

Для збереження даних була спроєктована база даних MySQL, яка охоплює користувачів, каталог фільмів і серіалів, оцінки, рецензії, обране, список перегляду та частину метаданих, отриманих через TMDB API. Текстові рецензії зберігаються в таблиці ratings через поле review_text, що дозволяє працювати з оцінкою та рецензією як з одним пов’язаним записом користувацької активності.

У вебплатформі реалізовано авторизацію та автентифікацію користувачів із підтримкою реєстрації, входу, сесій і розмежування прав доступу. Логіка ролей передбачає розділення можливостей гостя, авторизованого користувача та адміністратора. Для захисту даних використовуються хешування паролів, підготовлені SQL–запити та перевірка сесій.

Основною частиною Spokuta став каталог фільмів і серіалів із підтримкою

пошуку, фільтрації та сортування. Також реалізовано систему локальних оцінок і рецензій, обране, список перегляду, профіль користувача та базові рекомендації. Значна частина взаємодії виконується через асинхронні запити без повного перезавантаження сторінки, що робить роботу з каталогом швидшою та зручнішою.

Для автоматизації наповнення каталогу була реалізована інтеграція TMDB API. Вона використовується для імпорту фільмів і серіалів, отримання постерів, жанрів, описів, рейтингів та іншої інформації про контент. Окремо реалізовано адміністративну панель, через яку адміністратор може додавати, редагувати й видаляти контент, модерувати рецензії, керувати користувачами, імпортувати дані через TMDB API та переглядати статистику. Для візуального подання статистичних даних використовується бібліотека Chart.js [24].

Під час реалізації інтерфейсу значна увага приділялася адаптивності та зручності роботи на різних пристроях. Вебплатформа адаптується до ПК, планшетів і смартфонів завдяки гнучкій структурі сторінок та зміні розташування елементів залежно від розміру екрана. Для покращення взаємодії використовуються асинхронні запити, спливаючі повідомлення та тимчасові візуальні блоки завантаження.

У процесі розробки було проведено ручне, повторне та адаптивне тестування. Перевірялася робота авторизації, рольового доступу, пошуку, фільтрації, оцінок, рецензій, обраного, списку перегляду, рекомендацій, інтеграції TMDB API, адміністративної панелі та адаптивності інтерфейсу. Результати тестування показали, що основні сценарії роботи вебплатформи виконуються коректно, а критичних помилок у роботі основних модулів виявлено не було.

Розроблена вебплатформа має практичну цінність як приклад повноцінного вебзастосунку для роботи з медіаконтентом. Проєкт демонструє поєднання PHP, MySQL, JavaScript, асинхронних запитів та TMDB API в єдиній системі, яка підтримує каталог фільмів і серіалів, персональну активність користувачів, адміністративне керування та адаптивний інтерфейс для різних пристроїв.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Методичні вказівки до виконання дипломної роботи бакалавра для студентів спеціальності 123 «Комп’ютерна інженерія» всіх форм навчання / уклад.: Р. К. Кудерметов, О. В. Польська, Н. В. Луценко, Н. В. Щербак, О. В. Зелік. Запоріжжя : НУ «Запорізька політехніка», 2020. 54 с.

2. ДСТУ 3008:2015. Інформація та документація. Звіти у сфері науки і техніки. Структура та правила оформлювання. На заміну ДСТУ 3008–95 ; чинний від 2017–07–01. Вид. офіц. Київ : УкрНДНЦ, 2016. 32 с.

3. ДСТУ 8302:2015. Інформація і документація. Бібліографічне посилання. Загальні положення та правила складання. Чинний від 2016–07–01. Вид. офіц. Київ : УкрНДНЦ, 2016. 30 с.

4. Киричек Г. Г., Тягунова М. Ю., Дружко Р. В. Інтерфейс користувача як основа взаємодії з інформаційною системою. International Scientific and Practical Conference “Science, Education, and Society: From Theory to Practice in the Context of Contemporary Global Changes” : Conference Proceedings, San Francisco, USA, January 13, 2026. San Francisco : Golden Quill Publishing, 2026. P. 178–182. DOI: https://doi.org/10.64076/GQP–13.01.2026.021.

5. Дружко Р. В., Киричек Г. Г. Вебплатформа для відеоресурсу. Тиждень науки–2026. Факультет комп’ютерних наук і технологій : тези доповідей наук.–практ. конф., м. Запоріжжя, 13–17 квіт. 2026 р. / редкол.: В. Шаломєєв (відп. ред.). Запоріжжя : НУ «Запорізька політехніка», 2026.

6. IMDb: Ratings, Reviews, and Where to Watch the Best Movies & TV Shows. URL: https://www.imdb.com/ (дата звернення: 25.05.2026).

7. Letterboxd – Social film discovery. URL: https://letterboxd.com/ (дата звернення: 25.05.2026).

8. The Movie Database (TMDB). URL: https://www.themoviedb.org/ (дата звернення: 25.05.2026).

9. PHP Manual. URL: https://www.php.net/manual/en/ (дата звернення: 25.05.2026).

10. MySQL 8.4 Reference Manual. URL: https://dev.mysql.com/doc/refman/8.4/en/ (дата звернення: 25.05.2026).

11. JavaScript Guide. MDN Web Docs. URL: https://developer.mozilla.org/en–US/docs/Web/JavaScript/Guide (дата звернення: 25.05.2026).

12. Fetch API. MDN Web Docs. URL: https://developer.mozilla.org/en–US/docs/Web/API/Fetch_API (дата звернення: 25.05.2026).

13. Getting Started. TMDB API Documentation. URL: https://developer.themoviedb.org/reference/intro/getting–started (дата звернення: 25.05.2026).

14. HTML Living Standard. WHATWG. URL: https://html.spec.whatwg.org/ (дата звернення: 25.05.2026).

15. CSS: Cascading Style Sheets. MDN Web Docs. URL: https://developer.mozilla.org/en–US/docs/Web/CSS (дата звернення: 25.05.2026).

16. ECMA–404. The JSON Data Interchange Syntax. Ecma International. URL: https://ecma–international.org/publications–and–standards/standards/ecma–404/ (дата звернення: 25.05.2026).

17. Visual Studio Code Documentation. Microsoft. URL: https://code.visualstudio.com/docs (дата звернення: 25.05.2026).

18. XAMPP Documentation. Apache Friends. URL: https://www.apachefriends.org/docs/ (дата звернення: 25.05.2026).

19. Apache HTTP Server Documentation. The Apache Software Foundation. URL: https://httpd.apache.org/docs/ (дата звернення: 25.05.2026).

20. phpMyAdmin Documentation. URL: https://www.phpmyadmin.net/docs/ (дата звернення: 25.05.2026).

21. SQL Injection Prevention Cheat Sheet. OWASP Cheat Sheet Series. URL: https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html (дата звернення: 25.05.2026).

22. Authentication Cheat Sheet. OWASP Cheat Sheet Series. URL: https://cheatsheetseries.owasp.org/cheatsheets/Authentication_Cheat_Sheet.html (дата звернення: 25.05.2026).

23. Unified Modeling Language. Object Management Group. URL: https://www.omg.org/spec/UML/ (дата звернення: 25.05.2026).

24. Chart.js Documentation. URL: https://www.chartjs.org/docs/ (дата звернення: 25.05.2026).

25. Web Content Accessibility Guidelines (WCAG) 2.2. W3C. URL: https://www.w3.org/TR/WCAG22/ (дата звернення: 25.05.2026).

ДОДАТОК А
ЛІСТИНГИ ПРОГРАМ

Лістинг А.1 – Структура бази даних вебплатформи Spokuta

```
CREATE DATABASE IF NOT EXISTS movie_service
    CHARACTER SET utf8mb4
    COLLATE utf8mb4_unicode_ci;

USE movie_service;

CREATE TABLE IF NOT EXISTS users (
    id INT AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(100) NOT NULL,
    email VARCHAR(255) NOT NULL,
    password VARCHAR(255) NOT NULL,
    role ENUM('user', 'admin') NOT NULL DEFAULT 'user',
    created_at TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
    UNIQUE KEY uq_users_username (username),
    UNIQUE KEY uq_users_email (email)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

CREATE TABLE IF NOT EXISTS movies (
    id INT AUTO_INCREMENT PRIMARY KEY,
    title VARCHAR(255) NOT NULL,
    original_title VARCHAR(255) DEFAULT NULL,
    english_title VARCHAR(255) DEFAULT NULL,
    original_language VARCHAR(20) DEFAULT NULL,
    release_year SMALLINT UNSIGNED DEFAULT NULL,
    first_air_year SMALLINT UNSIGNED DEFAULT NULL,
    last_air_year SMALLINT UNSIGNED DEFAULT NULL,
    description TEXT DEFAULT NULL,
    poster_path VARCHAR(500) DEFAULT NULL,
    backdrop_path VARCHAR(500) DEFAULT NULL,
    director VARCHAR(255) DEFAULT NULL,
    genre VARCHAR(255) DEFAULT NULL,
    slogan VARCHAR(500) DEFAULT NULL,
    type VARCHAR(50) NOT NULL DEFAULT 'movie',
    tmdb_id INT DEFAULT NULL,
    avg_rating DECIMAL(3,1) NOT NULL DEFAULT 0.0,
    tmdb_rating DECIMAL(3,1) DEFAULT NULL,
    created_at TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
    KEY idx_movies_type (type),
    KEY idx_movies_tmdb_type (tmdb_id, type),
    KEY idx_movies_titles (title, original_title, english_title)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

CREATE TABLE IF NOT EXISTS ratings (
    id INT AUTO_INCREMENT PRIMARY KEY,
```

```
    user_id INT NOT NULL,
    movie_id INT NOT NULL,
    rating TINYINT UNSIGNED NOT NULL,
    review_text TEXT DEFAULT NULL,
    created_at TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
    UNIQUE KEY uq_ratings_user_movie (user_id, movie_id),
    KEY idx_ratings_movie_id (movie_id),
    CONSTRAINT fk_ratings_user
        FOREIGN KEY (user_id) REFERENCES users (id)
        ON DELETE CASCADE
        ON UPDATE CASCADE,
    CONSTRAINT fk_ratings_movie
        FOREIGN KEY (movie_id) REFERENCES movies (id)
        ON DELETE CASCADE
        ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

CREATE TABLE IF NOT EXISTS favorites (
    id INT AUTO_INCREMENT PRIMARY KEY,
    user_id INT NOT NULL,
    movie_id INT NOT NULL,
    created_at TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
    UNIQUE KEY uq_favorites_user_movie (user_id, movie_id),
    KEY idx_favorites_movie_id (movie_id),
    CONSTRAINT fk_favorites_user
        FOREIGN KEY (user_id) REFERENCES users (id)
        ON DELETE CASCADE
        ON UPDATE CASCADE,
    CONSTRAINT fk_favorites_movie
        FOREIGN KEY (movie_id) REFERENCES movies (id)
        ON DELETE CASCADE
        ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

CREATE TABLE IF NOT EXISTS watchlist (
    id INT AUTO_INCREMENT PRIMARY KEY,
    user_id INT NOT NULL,
    movie_id INT NOT NULL,
    created_at TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
    UNIQUE KEY uq_watchlist_user_movie (user_id, movie_id),
    KEY idx_watchlist_movie_id (movie_id),
    CONSTRAINT fk_watchlist_user
        FOREIGN KEY (user_id) REFERENCES users (id)
        ON DELETE CASCADE
        ON UPDATE CASCADE,
    CONSTRAINT fk_watchlist_movie
        FOREIGN KEY (movie_id) REFERENCES movies (id)
        ON DELETE CASCADE
        ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

Лістинг А.2 – Підключення до бази даних

```
<?php
$conn = new mysqli('localhost', 'root', '', 'movie_service');

if (!$conn->connect_error && !$conn->set_charset('utf8mb4')) {
    die('Тимчасова помилка налаштування кодування бази даних.');
}
?>
```

Лістинг А.3 – Авторизація та перевірка ролей користувачів

```
<?php
if (session_status() === PHP_SESSION_NONE) {
    session_start();
}

function requireLogin()
{
    if (!isLoggedIn()) {
        header("Location: login.html");
        exit();
    }
}

function isLoggedIn()
{
    return isset($_SESSION['user_id']);
}

function requireAdmin()
{
    requireLogin();

    if (!isset($_SESSION['role']) || $_SESSION['role'] !== 'admin') {
        header("Location: home.php");
        exit();
    }
}

function getCsrfToken()
{
    if (empty($_SESSION['csrf_token'])) {
        $_SESSION['csrf_token'] = bin2hex(random_bytes(32));
    }

    return $_SESSION['csrf_token'];
}

function csrfInput()
{
    return '<input type="hidden" name="csrf_token" value="' .
        htmlspecialchars(getCsrfToken(), ENT_QUOTES, 'UTF-8') . '">';
```

```
}

function isValidCsrfToken($token)
{
    return isset($_SESSION['csrf_token'])
        && is_string($token)
        && hash_equals($_SESSION['csrf_token'], $token);
}
?>
```

Лістинг А.4 – Відкрита робота каталогу для Guest та User

```
<?php
require_once __DIR__ . '/includes/auth.php';
require_once __DIR__ . '/includes/navbar.php';
require_once __DIR__ . '/includes/db.php';

$user_id = isLoggedIn() ? (int) $_SESSION['user_id'] : 0;

function bindStatementParams($stmt, $types, &$params)
{
    if ($types === '') {
        return;
    }

    $refs = array($types);
    foreach ($params as &$value) {
        $refs[] = &$value;
    }

    call_user_func_array(array($stmt, 'bind_param'), $refs);
}

$search = trim($_GET['search'] ?? '');
$selected_genre = trim($_GET['genre'] ?? '');
$selected_year = trim($_GET['year'] ?? '');
$selected_type = trim($_GET['type'] ?? '');

$where = array();
$params = array();
$types = '';

if ($search !== '') {
    $search_like = '%' . $search . '%';

    $where[] = "(COALESCE(m.title, '') LIKE ?
        OR COALESCE(m.original_title, '') LIKE ?
        OR COALESCE(m.english_title, '') LIKE ?
        OR COALESCE(m.genre, '') LIKE ?
        OR COALESCE(m.director, '') LIKE ?)";

    $params[] = $search_like;
    $params[] = $search_like;
```

```
    $params[] = $search_like;
    $params[] = $search_like;
    $params[] = $search_like;
    $types .= 'sssss';
}

if ($selected_genre !== '') {
    $where[] = "COALESCE(m.genre, '') LIKE ?";
    $params[] = '%' . $selected_genre . '%';
    $types .= 's';
}

if ($selected_type !== '') {
    $where[] = "m.type = ?";
    $params[] = $selected_type;
    $types .= 's';
}

if ($selected_year !== '') {
    $where[] = "(m.release_year = ? OR m.first_air_year = ?)";
    $params[] = (int) $selected_year;
    $params[] = (int) $selected_year;
    $types .= 'ii';
}

$where_sql = '';
if (!empty($where)) {
    $where_sql = 'WHERE ' . implode(' AND ', $where);
}

$sql = "
    SELECT m.*,
           COALESCE(AVG(r.rating), 0) AS spokuta_avg_rating,
           COUNT(r.id) AS rating_count,
           SUM(CASE
               WHEN r.review_text IS NOT NULL AND TRIM(r.review_text)
!= ''
               THEN 1 ELSE 0
           END) AS review_count,
           CASE WHEN f.id IS NULL THEN 0 ELSE 1 END AS is_favorite,
           CASE WHEN w.id IS NULL THEN 0 ELSE 1 END AS is_watchlisted
    FROM movies m
    LEFT JOIN ratings r ON r.movie_id = m.id
    LEFT JOIN favorites f ON f.movie_id = m.id AND f.user_id = ?
    LEFT JOIN watchlist w ON w.movie_id = m.id AND w.user_id = ?
    $where_sql
    GROUP BY m.id
    ORDER BY m.title ASC
    LIMIT 100
";

$stmt = $conn->prepare($sql);
```

```
$movie_params = array($user_id, $user_id);
foreach ($params as $param) {
    $movie_params[] = $param;
}

$movie_types = 'ii' . $types;
bindStatementParams($stmt, $movie_types, $movie_params);

$stmt->execute();
$result = $stmt->get_result();

$movies = array();
while ($movie = $result->fetch_assoc()) {
    $movies[] = $movie;
}

$stmt->close();
?>
```

Лістинг А.5 – Сторінка фільму або серіалу

```
<?php
require_once __DIR__ . '/includes/auth.php';
require_once __DIR__ . '/includes/navbar.php';
require_once __DIR__ . '/includes/db.php';

$is_logged_in = isLoggedIn();
$user_id = $is_logged_in ? (int) $_SESSION['user_id'] : 0;

$movie_id = filter_input(INPUT_GET, 'id', FILTER_VALIDATE_INT);
if ($movie_id === false || $movie_id === null || $movie_id <= 0) {
    die('Некоректний ідентифікатор фільму.');
}

$movie_sql = "
    SELECT m.*,
           COALESCE(rs.avg_rating, 0) AS spokuta_avg_rating,
           COALESCE(rs.rating_count, 0) AS rating_count,
           COALESCE(rs.review_count, 0) AS review_count,
           CASE WHEN f.id IS NULL THEN 0 ELSE 1 END AS is_favorite,
           CASE WHEN w.id IS NULL THEN 0 ELSE 1 END AS is_watchlisted
    FROM movies m
    LEFT JOIN (
        SELECT movie_id,
               AVG(rating) AS avg_rating,
               COUNT(*) AS rating_count,
               SUM(CASE
                   WHEN review_text IS NOT NULL AND TRIM(review_text)
!= ''
                   THEN 1 ELSE 0
               END) AS review_count
        FROM ratings
        GROUP BY movie_id
```

```
    ) rs ON rs.movie_id = m.id
    LEFT JOIN favorites f ON f.movie_id = m.id AND f.user_id = ?
    LEFT JOIN watchlist w ON w.movie_id = m.id AND w.user_id = ?
    WHERE m.id = ?
    LIMIT 1
";

$movie_stmt = $conn->prepare($movie_sql);
$movie_stmt->bind_param("iii", $user_id, $user_id, $movie_id);
$movie_stmt->execute();

$movie = $movie_stmt->get_result()->fetch_assoc();
$movie_stmt->close();

if (!$movie) {
    die('Фільм або серіал не знайдено.');
}

$own_review = null;

if ($is_logged_in) {
    $own_review_sql = "
        SELECT rating, review_text, created_at
        FROM ratings
        WHERE user_id = ? AND movie_id = ?
        LIMIT 1
    ";

    $own_review_stmt = $conn->prepare($own_review_sql);
    $own_review_stmt->bind_param("ii", $user_id, $movie_id);
    $own_review_stmt->execute();

    $own_review = $own_review_stmt->get_result()->fetch_assoc();
    $own_review_stmt->close();
}

$reviews_sql = "
    SELECT u.username, r.rating, r.review_text, r.created_at
    FROM ratings r
    JOIN users u ON u.id = r.user_id
    WHERE r.movie_id = ?
      AND r.review_text IS NOT NULL
      AND TRIM(r.review_text) != ''
    ORDER BY r.created_at DESC, r.id DESC
";

$reviews_stmt = $conn->prepare($reviews_sql);
$reviews_stmt->bind_param("i", $movie_id);
$reviews_stmt->execute();

$reviews_result = $reviews_stmt->get_result();
$reviews = array();
```

```
while ($review = $reviews_result->fetch_assoc()) {
    $reviews[] = $review;
}

$reviews_stmt->close();

$recommendations_sql = "
    SELECT id, title, poster_path, type, release_year, first_air_year,
tmdb_rating
    FROM movies
    WHERE id != ?
      AND type = ?
    ORDER BY tmdb_rating DESC, title ASC
    LIMIT 6
";

$recommendations_stmt = $conn->prepare($recommendations_sql);
$recommendations_stmt->bind_param("is", $movie_id, $movie['type']);
$recommendations_stmt->execute();

$recommendations_result = $recommendations_stmt->get_result();
$recommendations = array();

while ($item = $recommendations_result->fetch_assoc()) {
    $recommendations[] = $item;
}

$recommendations_stmt->close();
?>
```

Лістинг А.6 – Збереження оцінки та текстової рецензії

```
<?php
require_once __DIR__ . '/includes/auth.php';

function wantsJsonResponse()
{
    $accept = $_SERVER['HTTP_ACCEPT'] ?? '';
    $requested_with = $_SERVER['HTTP_X_REQUESTED_WITH'] ?? '';

    return stripos($accept, 'application/json') !== false
        || strtolower($requested_with) === 'fetch';
}

function finishRateJson($success, $payload = array(), $status_code =
200)
{
    if (!wantsJsonResponse()) {
        return;
    }

    http_response_code($status_code);
    header('Content-Type: application/json; charset=utf-8');
```

```
    echo   json_encode(array_merge(array('success'   =>   $success),
$payload));
    exit();
}

function finishRateAuthRequired()
{
    if (wantsJsonResponse()) {
        http_response_code(401);
        header('Content-Type: application/json; charset=utf-8');
        echo json_encode(array(
            'success' => false,
            'auth_required' => true,
            'redirect' => 'login.html',
            'message' => 'Увійдіть, щоб залишати оцінки та відгуки.'
        ));
        exit();
    }

    header("Location: login.html");
    exit();
}

if ($_SERVER['REQUEST_METHOD'] !== 'POST') {
    finishRateJson(false,    array('message'    =>    'Метод    не
підтримується.'), 405);
    header("Location: home.php");
    exit();
}

if (!isLoggedIn()) {
    finishRateAuthRequired();
}

if (!isValidCsrfToken($_POST['csrf_token'] ?? '')) {
    finishRateJson(false, array(
        'message' => 'Сесія застаріла. Оновіть сторінку й спробуйте
ще раз.'
    ), 403);
    exit();
}

$user_id = (int) $_SESSION['user_id'];
$movie_id       =       filter_input(INPUT_POST,       'movie_id',
FILTER_VALIDATE_INT);
$rating = filter_input(INPUT_POST, 'rating', FILTER_VALIDATE_INT);
$review_text = trim($_POST['review_text'] ?? '');

if ($movie_id === false || $movie_id <= 0 || $rating === false ||
$rating < 1 || $rating > 10) {
    finishRateJson(false, array('message' => 'Оцінка має бути числом
від 1 до 10.'), 422);
    exit();
```

```
}

require_once __DIR__ . '/includes/db.php';

$movie_sql = "SELECT id FROM movies WHERE id = ? LIMIT 1";
$stmt = $conn->prepare($movie_sql);
$stmt->bind_param("i", $movie_id);
$stmt->execute();

if (!$stmt->get_result()->fetch_assoc()) {
    $stmt->close();
    $conn->close();

    finishRateJson(false, array('message' => 'Тайтл не знайдено.'),
404);
    exit();
}

$stmt->close();

$check_sql = "SELECT id FROM ratings WHERE user_id = ? AND movie_id =
? LIMIT 1";
$stmt = $conn->prepare($check_sql);
$stmt->bind_param("ii", $user_id, $movie_id);
$stmt->execute();

$result = $stmt->get_result();

if ($result->num_rows > 0) {
    $update_sql = "
        UPDATE ratings
        SET rating = ?, review_text = ?, created_at = NOW()
        WHERE user_id = ? AND movie_id = ?
    ";

    $stmt = $conn->prepare($update_sql);
    $stmt->bind_param("isii",   $rating,   $review_text,   $user_id,
$movie_id);
    $stmt->execute();
} else {
    $insert_sql = "
        INSERT INTO ratings (user_id, movie_id, rating, review_text)
        VALUES (?, ?, ?, ?)
    ";

    $stmt = $conn->prepare($insert_sql);
    $stmt->bind_param("iiis",    $user_id,    $movie_id,    $rating,
$review_text);
    $stmt->execute();
}

$stmt->close();
```

```
$stats_sql = "
    SELECT AVG(rating) AS avg_rating,
           COUNT(*) AS rating_count,
           SUM(CASE
               WHEN review_text IS NOT NULL AND TRIM(review_text) !=
''
               THEN 1 ELSE 0
           END) AS review_count
    FROM ratings
    WHERE movie_id = ?
";

$stats_stmt = $conn->prepare($stats_sql);
$stats_stmt->bind_param("i", $movie_id);
$stats_stmt->execute();

$stats = $stats_stmt->get_result()->fetch_assoc();
$stats_stmt->close();

$conn->close();

finishRateJson(true, array(
    'message' => 'Оцінку та відгук збережено.',
    'movie_id' => $movie_id,
    'rating' => $rating,
    'review_text' => $review_text,
    'spokuta_rating' => number_format((float) $stats['avg_rating'],
1),
    'rating_count' => (int) $stats['rating_count'],
    'review_count' => (int) $stats['review_count']
));

header("Location: movie_details.php?id=" . $movie_id);
exit();
?>
```

Лістинг А.7 – Робота з favorites

```
<?php
require_once __DIR__ . '/includes/auth.php';

function finishToggle($success, $payload = array(), $status_code =
200)
{
    http_response_code($status_code);
    header('Content-Type: application/json; charset=utf-8');
    echo json_encode(array_merge(array('success' => $success),
$payload));
    exit();
}

if ($_SERVER['REQUEST_METHOD'] !== 'POST') {
```

```
    finishToggle(false, array('message' => 'Метод не підтримується.'),
405);
}

if (!isLoggedIn()) {
    finishToggle(false, array(
        'auth_required' => true,
        'redirect' => 'login.html',
        'message' => 'Увійдіть, щоб додавати тайтли в обране.'
    ), 401);
}

if (!isValidCsrfToken($_POST['csrf_token'] ?? '')) {
    finishToggle(false, array('message' => 'Некоректний CSRF token.'),
403);
}

require_once __DIR__ . '/includes/db.php';

$user_id = (int) $_SESSION['user_id'];
$movie_id = isset($_POST['movie_id']) ? (int) $_POST['movie_id'] : 0;

if ($movie_id <= 0) {
    finishToggle(false,  array('message'  =>  'Некоректний  тайтл.'),
422);
}

$check_sql = "SELECT id FROM favorites WHERE user_id = ? AND movie_id
= ?";
$check_stmt = $conn->prepare($check_sql);
$check_stmt->bind_param("ii", $user_id, $movie_id);
$check_stmt->execute();

$check_result = $check_stmt->get_result();
$is_favorite = false;

if ($favorite = $check_result->fetch_assoc()) {
    $delete_sql = "DELETE FROM favorites WHERE id = ? AND user_id =
?";
    $delete_stmt = $conn->prepare($delete_sql);
    $delete_stmt->bind_param("ii", $favorite['id'], $user_id);
    $delete_stmt->execute();
    $delete_stmt->close();
} else {
    $insert_sql = "INSERT INTO favorites (user_id, movie_id) VALUES
(?, ?)";
    $insert_stmt = $conn->prepare($insert_sql);
    $insert_stmt->bind_param("ii", $user_id, $movie_id);
    $insert_stmt->execute();
    $insert_stmt->close();

    $is_favorite = true;
}
```

```
$check_stmt->close();
$conn->close();

finishToggle(true, array(
    'movie_id' => $movie_id,
    'action' => 'favorite',
    'active' => $is_favorite,
    'label' => $is_favorite ? 'В обраному' : 'В обране'
));
?>
```

Лістинг А.8 – Робота з watchlist

```
<?php
require_once __DIR__ . '/includes/auth.php';

function finishToggle($success, $payload = array(), $status_code =
200)
{
    http_response_code($status_code);
    header('Content-Type: application/json; charset=utf-8');
    echo   json_encode(array_merge(array('success'   =>   $success),
$payload));
    exit();
}

if ($_SERVER['REQUEST_METHOD'] !== 'POST') {
    finishToggle(false, array('message' => 'Метод не підтримується.'),
405);
}

if (!isLoggedIn()) {
    finishToggle(false, array(
        'auth_required' => true,
        'redirect' => 'login.html',
        'message'  =>  'Увійдіть,  щоб  додавати  тайтли  у  список
перегляду.'
    ), 401);
}

if (!isValidCsrfToken($_POST['csrf_token'] ?? '')) {
    finishToggle(false, array('message' => 'Некоректний CSRF token.'),
403);
}

require_once __DIR__ . '/includes/db.php';

$user_id = (int) $_SESSION['user_id'];
$movie_id = isset($_POST['movie_id']) ? (int) $_POST['movie_id'] : 0;

if ($movie_id <= 0) {
```

```
    finishToggle(false, array('message' => 'Некоректний тайтл.'),
422);
}

$check_sql = "SELECT id FROM watchlist WHERE user_id = ? AND movie_id
= ?";
$check_stmt = $conn->prepare($check_sql);
$check_stmt->bind_param("ii", $user_id, $movie_id);
$check_stmt->execute();

$check_result = $check_stmt->get_result();
$is_watchlisted = false;

if ($watchlist_item = $check_result->fetch_assoc()) {
    $delete_sql = "DELETE FROM watchlist WHERE id = ? AND user_id =
?";
    $delete_stmt = $conn->prepare($delete_sql);
    $delete_stmt->bind_param("ii", $watchlist_item['id'], $user_id);
    $delete_stmt->execute();
    $delete_stmt->close();
} else {
    $insert_sql = "INSERT INTO watchlist (user_id, movie_id) VALUES
(?, ?)";
    $insert_stmt = $conn->prepare($insert_sql);
    $insert_stmt->bind_param("ii", $user_id, $movie_id);
    $insert_stmt->execute();
    $insert_stmt->close();

    $is_watchlisted = true;
}

$check_stmt->close();
$conn->close();

finishToggle(true, array(
    'movie_id' => $movie_id,
    'action' => 'watchlist',
    'active' => $is_watchlisted,
    'label' => $is_watchlisted ? 'У списку перегляду' : 'У список
перегляду'
));
?>
```

Лістинг А.9 – Інтеграція TMDB API

```
<?php
define('TMDB_API_BASE_URL', 'https://api.themoviedb.org/3');
define('TMDB_IMAGE_BASE_URL', 'https://image.tmdb.org/t/p');

$tmdb_bearer_token = getenv('TMDB_BEARER_TOKEN') ?:
'<TMDB_API_TOKEN>';
define('TMDB_BEARER_TOKEN', $tmdb_bearer_token);
```

```
function tmdbRequest($endpoint)
{
    static $request_cache = array();

    if (TMDB_BEARER_TOKEN === '' || !function_exists('curl_init')) {
        return null;
    }

    $endpoint = '/' . ltrim($endpoint, '/');

    if (array_key_exists($endpoint, $request_cache)) {
        return $request_cache[$endpoint];
    }

    $url = TMDB_API_BASE_URL . $endpoint;

    $ch = curl_init($url);
    curl_setopt_array($ch, array(
        CURLOPT_RETURNTRANSFER => true,
        CURLOPT_HTTPHEADER => array(
            'Authorization: Bearer ' . TMDB_BEARER_TOKEN,
            'Accept: application/json'
        ),
        CURLOPT_CONNECTTIMEOUT => 4,
        CURLOPT_TIMEOUT => 8
    ));

    $response = curl_exec($ch);
    $status_code = curl_getinfo($ch, CURLINFO_HTTP_CODE);
    $curl_error = curl_errno($ch);
    curl_close($ch);

    if ($curl_error || $status_code < 200 || $status_code >= 300 ||
$response === false) {
        $request_cache[$endpoint] = null;
        return null;
    }

    $data = json_decode($response, true);
    $request_cache[$endpoint] = is_array($data) ? $data : null;

    return $request_cache[$endpoint];
}

function tmdbImageUrl($path, $size = 'w500')
{
    if (empty($path)) {
        return '';
    }

    if (preg_match('/^https?:\/\//i', (string) $path)) {
        return (string) $path;
    }
```

```
    return TMDB_IMAGE_BASE_URL . '/' . $size . $path;
}

function tmdbGetTitleEnrichmentById($tmdb_id, $type = 'movie')
{
    $tmdb_id = (int) $tmdb_id;
    $type = $type === 'series' ? 'series' : 'movie';

    if ($tmdb_id <= 0) {
        return array();
    }

    if ($type === 'movie') {
        $details = tmdbRequest('/movie/' . $tmdb_id . '?language=uk-
UA');

        return array(
            'tmdb_id' => $tmdb_id,
            'type' => 'movie',
            'title' => $details['title'] ?? '',
            'original_title' => $details['original_title'] ?? '',
            'release_year' => isset($details['release_date'])
                ? substr($details['release_date'], 0, 4)
                : '',
            'poster_path' => tmdbImageUrl($details['poster_path'] ??
'', 'w500'),
            'backdrop_path' => tmdbImageUrl($details['backdrop_path']
?? '', 'w780'),
            'description' => $details['overview'] ?? '',
            'tmdb_rating' => isset($details['vote_average'])
                ? number_format((float) $details['vote_average'], 1,
'.', '')
                : ''
        );
    }

    $details = tmdbRequest('/tv/' . $tmdb_id . '?language=uk-UA');

    return array(
        'tmdb_id' => $tmdb_id,
        'type' => 'series',
        'title' => $details['name'] ?? '',
        'original_title' => $details['original_name'] ?? '',
        'first_air_year' => isset($details['first_air_date'])
            ? substr($details['first_air_date'], 0, 4)
            : '',
        'poster_path' => tmdbImageUrl($details['poster_path'] ?? '',
'w500'),
        'backdrop_path' => tmdbImageUrl($details['backdrop_path'] ??
'', 'w780'),
        'description' => $details['overview'] ?? '',
        'tmdb_rating' => isset($details['vote_average'])
```

```
            ? number_format((float) $details['vote_average'], 1, '.',
'')
            : ''
    );
}
?>
```

Лістинг А.10 – Адміністративна панель

```
<?php
require_once __DIR__ . '/includes/auth.php';
requireAdmin();

require_once __DIR__ . '/includes/db.php';

$admin_message = null;

function adminMoviePayload()
{
    $type = $_POST['type'] ?? 'movie';
    $type = $type === 'series' ? 'series' : 'movie';

    return array(
        'title' => trim($_POST['title'] ?? ''),
        'original_title' => trim($_POST['original_title'] ?? ''),
        'english_title' => trim($_POST['english_title'] ?? ''),
        'release_year'  =>  filter_input(INPUT_POST,  'release_year',
FILTER_VALIDATE_INT),
        'first_air_year'          =>          filter_input(INPUT_POST,
'first_air_year', FILTER_VALIDATE_INT),
        'description' => trim($_POST['description'] ?? ''),
        'poster_path' => trim($_POST['poster_path'] ?? ''),
        'backdrop_path' => trim($_POST['backdrop_path'] ?? ''),
        'director' => trim($_POST['director'] ?? ''),
        'genre' => trim($_POST['genre'] ?? ''),
        'slogan' => trim($_POST['slogan'] ?? ''),
        'type' => $type,
        'tmdb_id'     =>     filter_input(INPUT_POST,     'tmdb_id',
FILTER_VALIDATE_INT),
        'tmdb_rating'   =>   filter_input(INPUT_POST,   'tmdb_rating',
FILTER_VALIDATE_FLOAT)
    );
}

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    if (!isValidCsrfToken($_POST['csrf_token'] ?? '')) {
        $admin_message = array(
            'type' => 'error',
            'text' => 'Не вдалося підтвердити запит. Оновіть сторінку
і спробуйте ще раз.'
        );
    } else {
        $admin_action = $_POST['admin_action'] ?? 'add_movie';
```

```
        if ($admin_action === 'add_movie') {
            $movie = adminMoviePayload();

            $sql = "
                INSERT INTO movies (
                    title, original_title, english_title,
                    release_year, first_air_year,
                    description, poster_path, backdrop_path,
                    director, genre, slogan, type,
                    tmdb_id, tmdb_rating
                ) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)
            ";

            $stmt = $conn->prepare($sql);

            if ($stmt) {
                $stmt->bind_param(
                    "sssissssssssid",
                    $movie['title'],
                    $movie['original_title'],
                    $movie['english_title'],
                    $movie['release_year'],
                    $movie['first_air_year'],
                    $movie['description'],
                    $movie['poster_path'],
                    $movie['backdrop_path'],
                    $movie['director'],
                    $movie['genre'],
                    $movie['slogan'],
                    $movie['type'],
                    $movie['tmdb_id'],
                    $movie['tmdb_rating']
                );

                $admin_message = $stmt->execute()
                    ? array('type' => 'success', 'text' => 'Новий
контент успішно додано.')
                    : array('type' => 'error', 'text' => 'Не вдалося
додати контент.');

                $stmt->close();
            }
        }

        if ($admin_action === 'delete_movie') {
            $movie_id    =    filter_input(INPUT_POST,    'movie_id',
FILTER_VALIDATE_INT);

            if ($movie_id) {
                $stmt = $conn->prepare("DELETE FROM movies WHERE id =
?");
                $stmt->bind_param("i", $movie_id);
```

```
                $stmt->execute();
                $stmt->close();
            }
        }

        if ($admin_action === 'delete_review') {
            $rating_id   =   filter_input(INPUT_POST,   'rating_id',
FILTER_VALIDATE_INT);

            if ($rating_id) {
                $stmt = $conn->prepare("
                    UPDATE ratings
                    SET review_text = NULL
                    WHERE id = ?
                ");
                $stmt->bind_param("i", $rating_id);
                $stmt->execute();
                $stmt->close();
            }
        }
    }
}

$reviews_sql = "
    SELECT r.id,
           r.rating,
           r.review_text,
           r.created_at,
           m.title AS movie_title,
           u.username
    FROM ratings r
    JOIN movies m ON m.id = r.movie_id
    JOIN users u ON u.id = r.user_id
    WHERE r.review_text IS NOT NULL
      AND TRIM(r.review_text) != ''
    ORDER BY r.created_at DESC, r.id DESC
    LIMIT 30
";

$reviews_result = $conn->query($reviews_sql);

$reviews = array();
while ($row = $reviews_result->fetch_assoc()) {
    $reviews[] = $row;
}
?>
```

Лістинг А.11 – AJAX–взаємодія на клієнтській стороні

```
(function () {
    function showToast(message, type) {
        if (!message) {
            return;
```

```
        }

        type = type || 'success';

        var container = document.getElementById('toast-container');
        if (!container) {
            container = document.createElement('div');
            container.id = 'toast-container';
            container.setAttribute('aria-live', 'polite');
            document.body.appendChild(container);
        }

        var toast = document.createElement('div');
        toast.className = 'toast toast-' + type;

        var msgEl = document.createElement('span');
        msgEl.className = 'toast-message';
        msgEl.textContent = message;

        toast.appendChild(msgEl);
        container.appendChild(toast);

        requestAnimationFrame(function () {
            toast.classList.add('is-visible');
        });

        window.setTimeout(function () {
            toast.classList.remove('is-visible');
            toast.remove();
        }, 3000);
    }

    function updateRatingBadges(data) {
        var ratingValue = document.querySelector('[data-spokuta-
rating-value]');
        var ratingCount = document.querySelector('[data-spokuta-
rating-count]');
        var reviewCount = document.querySelector('[data-spokuta-
review-count]');

        if (ratingValue) {
            ratingValue.textContent = data.spokuta_rating || 'немає
оцінок';
        }

        if (ratingCount) {
            ratingCount.textContent = String(data.rating_count || 0)
+ ' оцінок';
        }

        if (reviewCount) {
            reviewCount.textContent = String(data.review_count || 0);
        }
```

```
    }

    function updateOwnReview(form, data) {
        var panel = form.closest('.movie-details-rate-panel');
        var container = panel ? panel.querySelector('[data-own-review-
container]') : null;

        if (!container) {
            return;
        }

        var reviewText = String(data.review_text || '').trim();

        container.innerHTML =
            '<div class="movie-details-own-review">' +
            '<strong>' + data.rating + '/10</strong>' +
            '<p>' + escapeHtml(reviewText || 'Ви оцінили тайтл без
текстового відгуку.') + '</p>' +
            '</div>';
    }

    document.addEventListener('submit', function (event) {
        var form = event.target;

        if (!form.matches('[data-ajax-rating]')) {
            return;
        }

        event.preventDefault();

        fetch(form.action, {
            method: 'POST',
            body: new FormData(form),
            headers: {
                'Accept': 'application/json',
                'X-Requested-With': 'fetch'
            },
            credentials: 'same-origin'
        })
            .then(function (response) {
                return response.json().then(function (data) {
                    if (!response.ok || !data.success) {
                        var requestError = new Error(data.message ||
'Не вдалося зберегти оцінку.');
                        requestError.authRequired                    =
!!data.auth_required;
                        requestError.redirect   =   data.redirect   ||
'login.html';
                        throw requestError;
                    }

                    return data;
                });
```

```
            })
            .then(function (data) {
                updateOwnReview(form, data);
                updateRatingBadges(data);
                showToast(data.message || 'Оцінку та відгук
збережено.', 'success');
            })
            .catch(function (error) {
                if (error.authRequired) {
                    showToast(error.message || 'Увійдіть, щоб
виконати цю дію.', 'error');
                    window.location.href = error.redirect ||
'login.html';
                    return;
                }

                showToast(error.message || 'Не вдалося зберегти
оцінку.', 'error');
            });
    });

    document.addEventListener('submit', function (event) {
        var form = event.target;

        if (!form.matches('[data-ajax-toggle]')) {
            return;
        }

        event.preventDefault();

        var button = form.querySelector('button[type="submit"]');

        fetch(form.action, {
            method: 'POST',
            body: new FormData(form),
            headers: {
                'Accept': 'application/json',
                'X-Requested-With': 'fetch'
            },
            credentials: 'same-origin'
        })
            .then(function (response) {
                return response.json().then(function (data) {
                    if (!response.ok || !data.success) {
                        var requestError = new Error(data.message ||
'Дію не виконано.');
                        requestError.authRequired =
!!data.auth_required;
                        requestError.redirect = data.redirect ||
'login.html';
                        throw requestError;
                    }
```

```
                    return data;
                });
            })
            .then(function (data) {
                button.classList.toggle('is-active', !!data.active);
                button.textContent = data.label;
                button.setAttribute('aria-pressed',   data.active   ?
'true' : 'false');

                if (data.action === 'favorite') {
                    showToast(data.active  ?  'Додано  в  обране'  :
'Прибрано з обраного', 'success');
                }

                if (data.action === 'watchlist') {
                    showToast(data.active   ?   'Додано   у   список
перегляду' : 'Прибрано зі списку перегляду', 'success');
                }
            })
            .catch(function (error) {
                if (error.authRequired) {
                    showToast(error.message       ||       'Потрібна
авторизація.', 'error');
                    window.location.href    =    error.redirect    ||
'login.html';
                    return;
                }

                showToast(error.message || 'Не вдалося виконати дію.',
'error');
            });
    });
})();
```

Лістинг А.12 – Основні стилі інтерфейсу

```
.home-page .movie-filter-bar,
.admin-page .tmdb-import-panel,
.admin-page .container,
.profile-page .profile-info,
.profile-page .profile-movie-item,
.home-page .movie {
    border-color: rgba(203, 213, 225, 0.14);
    background:
        linear-gradient(180deg, rgba(255, 255, 255, 0.055), rgba(255,
255, 255, 0.02)),
        rgba(13, 21, 32, 0.80);
    box-shadow:
        0 22px 58px rgba(0, 0, 0, 0.34),
        inset 0 1px 0 rgba(255, 255, 255, 0.045);
    backdrop-filter: blur(18px);
}
```

```
.home-page .movie {
    overflow: hidden;
    border-radius: 22px;
}

.home-page .rating-duo {
    display: grid;
    grid-template-columns: repeat(2, minmax(0, 1fr));
    gap: 8px;
    margin-top: 2px;
}

.home-page .rating-pill {
    min-width: 0;
    padding: 9px 10px;
    background-color: rgba(8, 13, 22, 0.54);
    border: 1px solid rgba(148, 163, 184, 0.16);
    border-radius: 14px;
}

.home-page .movie-toggle-form button.is-active {
    color: #061018;
    background:
        linear-gradient(135deg, #86efac 0%, #34d399 100%);
    border-color: transparent;
}

@media (min-width: 981px) {
    .home-page .movie-filter-bar {
        grid-template-columns:
            minmax(220px, 1.45fr)
            minmax(100px, 0.65fr)
            minmax(120px, 0.8fr)
            minmax(100px, 0.6fr)
            minmax(125px, 0.75fr)
            minmax(125px, 0.75fr);
    }
}

@media (max-width: 720px) {
    .app-header,
    .home-page header.app-header,
    .profile-page header.app-header,
    .admin-page header.app-header {
        position: relative;
        padding: 14px 16px;
    }

    .home-page .container,
    .profile-page .container,
    .admin-page .container {
        padding-top: 24px;
    }
```

```
}

@media (max-width: 420px) {
    .home-page .rating-duo {
        grid-template-columns: 1fr;
    }
}

.catalog-skeleton-overlay {
    position: absolute;
    inset: 0;
    z-index: 10;
    display: grid;
    grid-template-columns: repeat(3, minmax(0, 1fr));
    gap: 20px;
    pointer-events: none;
}

.skeleton-card {
    display: flex;
    flex-direction: column;
    background: var(--bg-elevated);
    border: 1px solid var(--border);
    border-radius: var(--radius);
    overflow: hidden;
}

.skeleton-poster,
.skeleton-line,
.skeleton-btn {
    background: linear-gradient(
        90deg,
        var(--bg-soft) 25%,
        var(--bg-panel) 50%,
        var(--bg-soft) 75%
    );
    background-size: 200% 100%;
    animation: skeleton-shimmer 1.5s ease-in-out infinite;
}

@media (max-width: 980px) {
    .catalog-skeleton-overlay {
        grid-template-columns: repeat(2, minmax(0, 1fr));
    }
}

@media (max-width: 720px) {
    .catalog-skeleton-overlay {
        grid-template-columns: 1fr;
    }
}

#toast-container {
```

```
    position: fixed;
    bottom: 24px;
    right: 24px;
    z-index: 9999;
    display: flex;
    flex-direction: column;
    gap: 10px;
    width: 320px;
    pointer-events: none;
}

.toast {
    display: flex;
    align-items: center;
    gap: 10px;
    padding: 13px 14px 13px 16px;
    border-radius: var(--radius-sm);
    background: rgba(18, 26, 42, 0.95);
    border: 1px solid rgba(148, 163, 184, 0.16);
    border-left: 3px solid transparent;
    opacity: 0;
    transform: translateY(10px);
    transition:
        opacity 220ms ease,
        transform 220ms ease;
}

.toast.is-visible {
    opacity: 1;
    transform: translateY(0);
}

.toast-success {
    border-left-color: var(--success);
}

.toast-error {
    border-left-color: var(--danger);
}

.toast-info {
    border-left-color: var(--primary);
}

@media (max-width: 600px) {
    #toast-container {
        bottom: 16px;
        left: 50%;
        right: auto;
        transform: translateX(-50%);
        width: min(92vw, 360px);
    }
}
```

ДОДАТОК Б
СЛАЙДИ ПРЕЗЕНТАЦІЇ

Рисунок Б.1 – Слайд 1

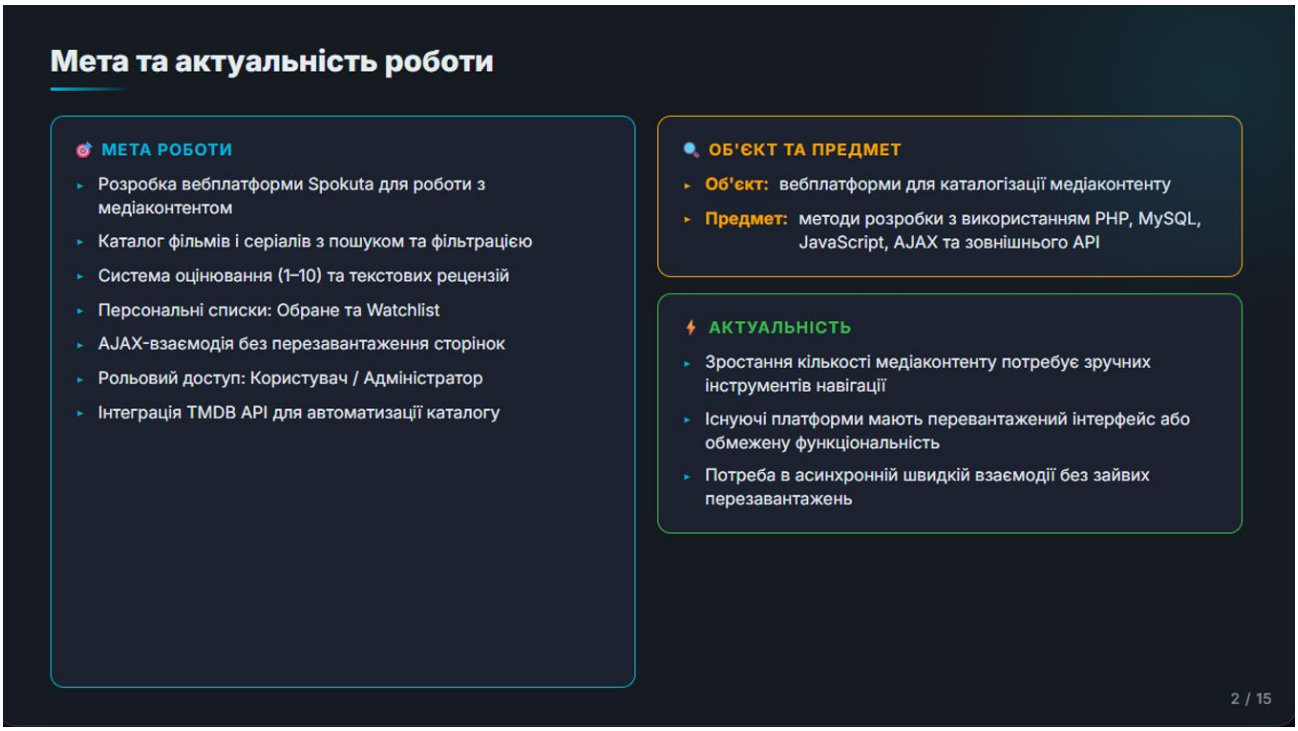

Рисунок Б.2 – Слайд 2

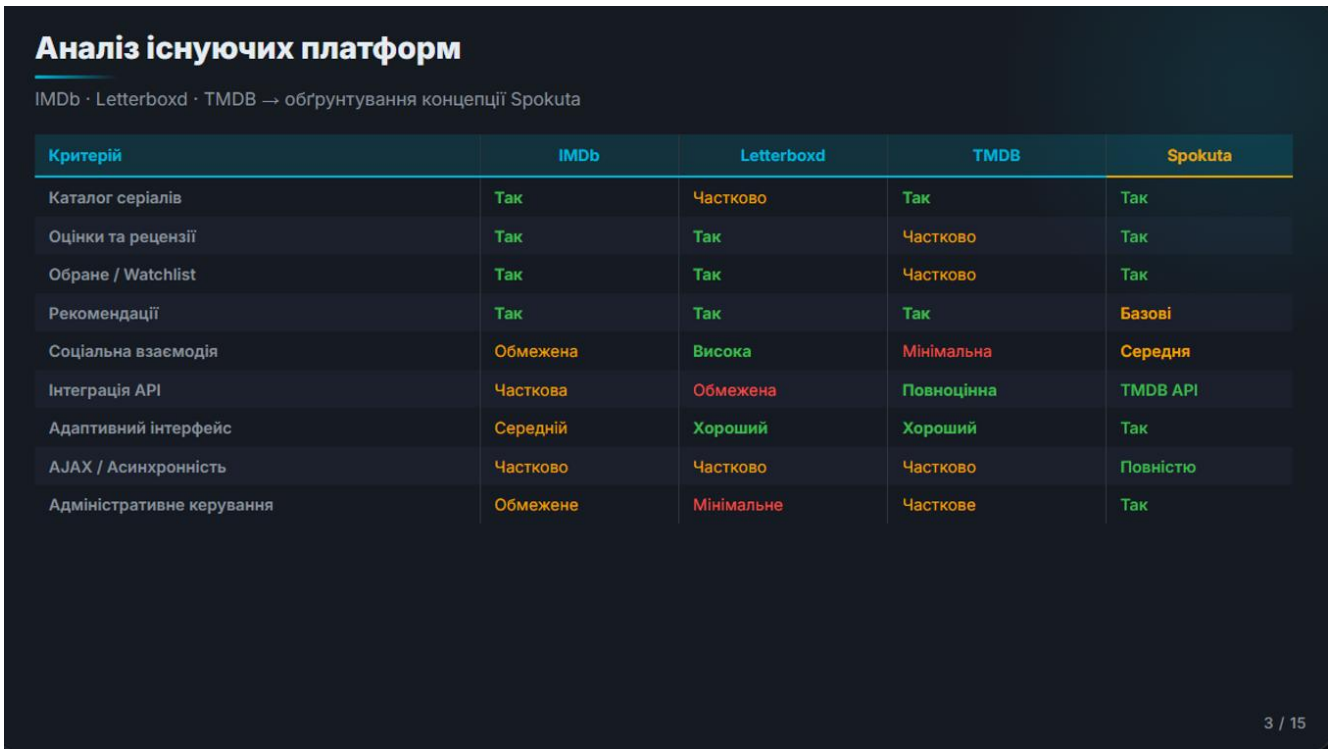

Рисунок Б.3 – Слайд 3

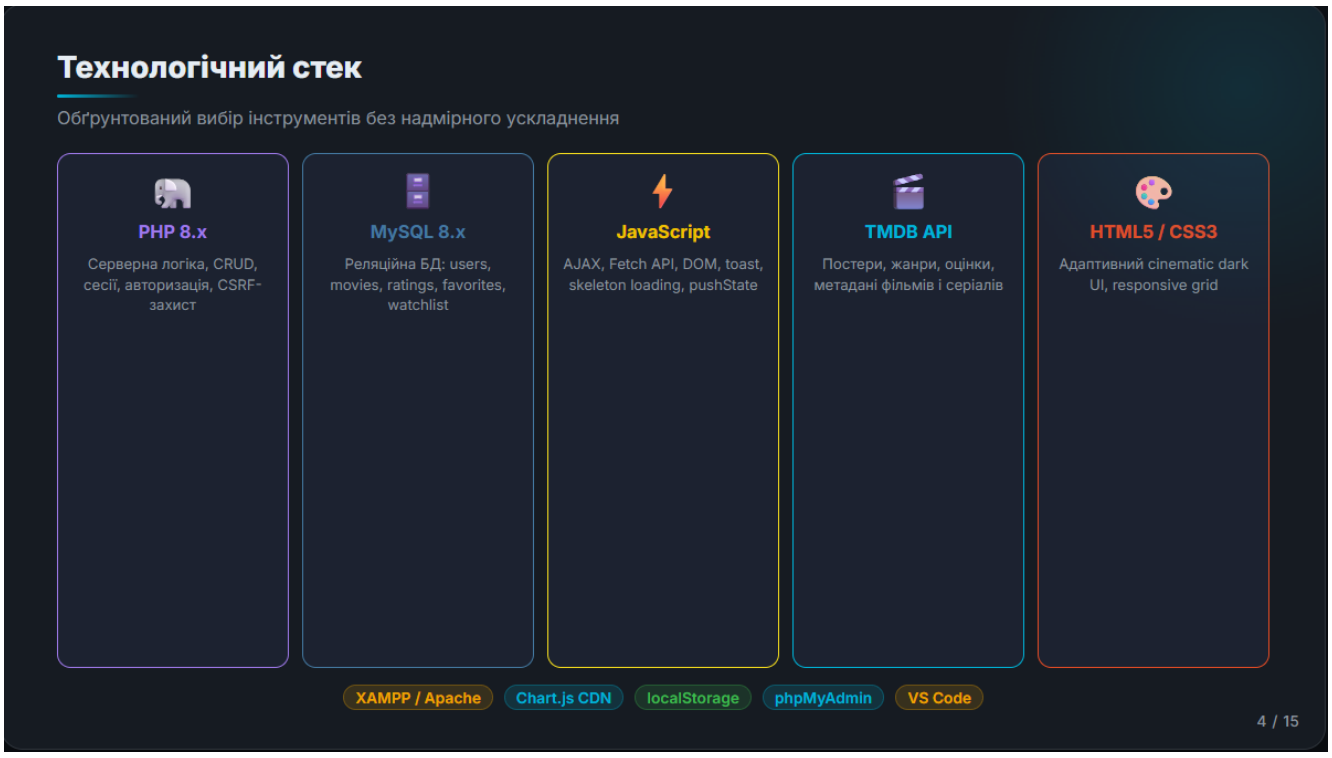

Рисунок Б.4 – Слайд 4

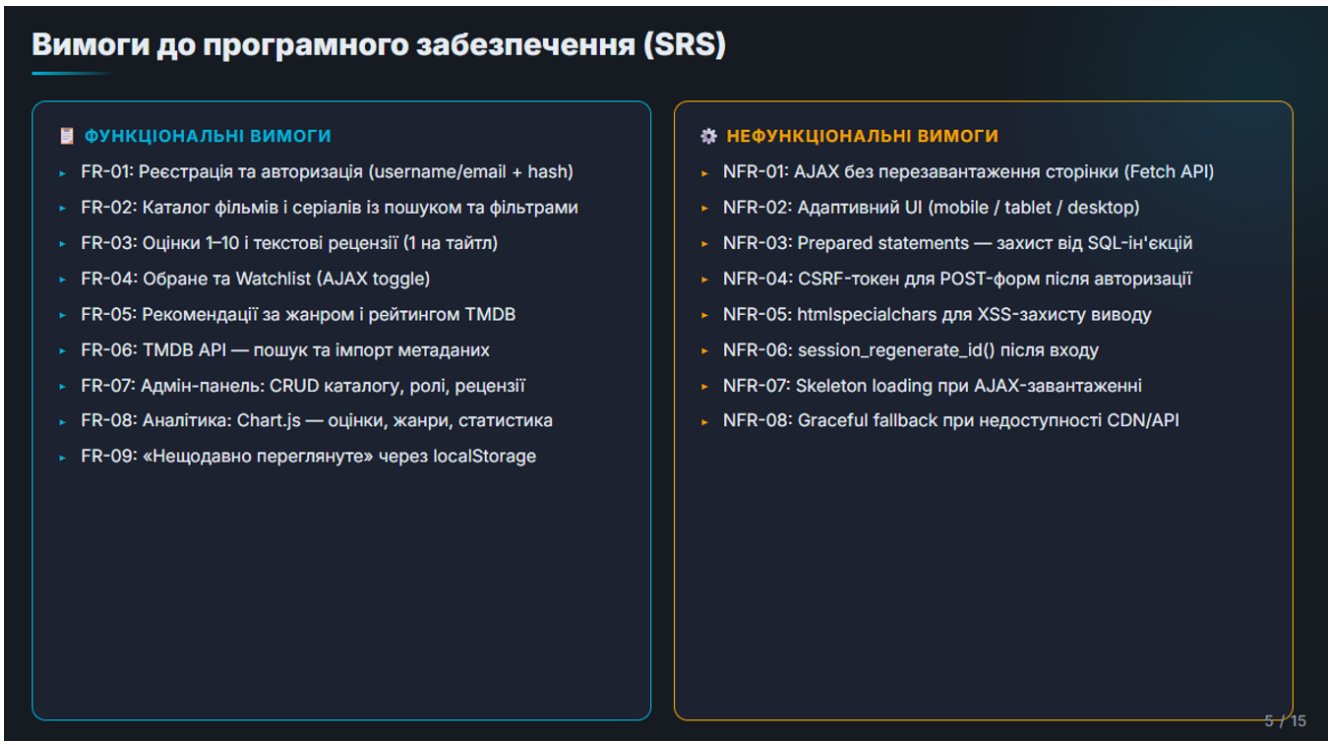

Рисунок Б.5 – Слайд 5

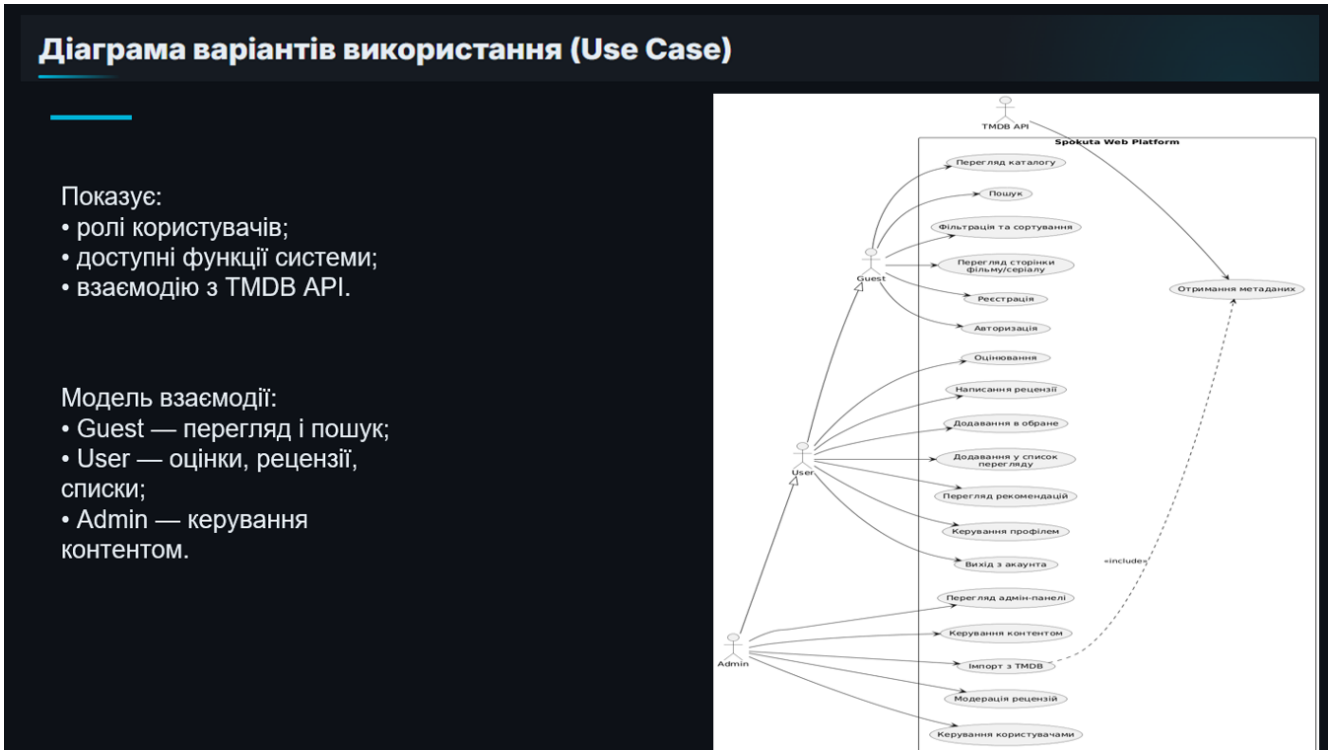

Рисунок Б.6 – Слайд 6

Рисунок Б.7 – Слайд 7

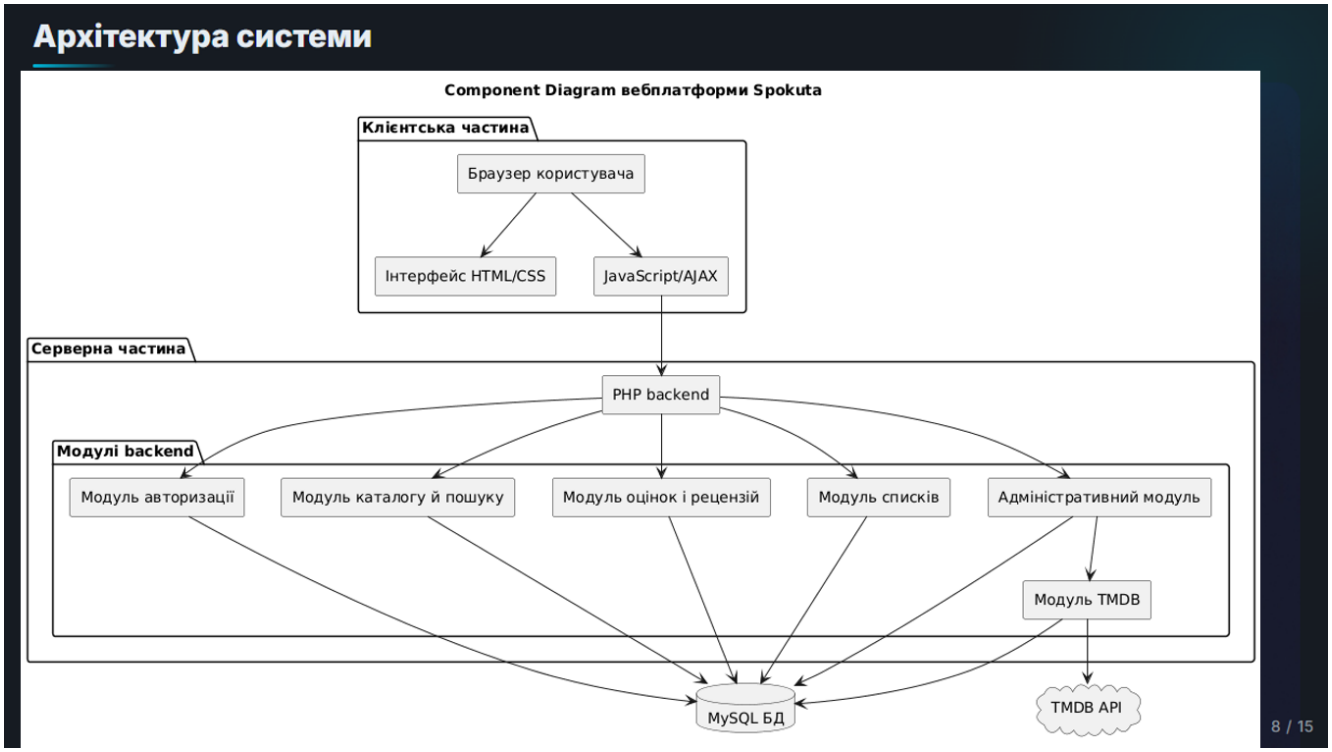

Рисунок Б.8 – Слайд 8

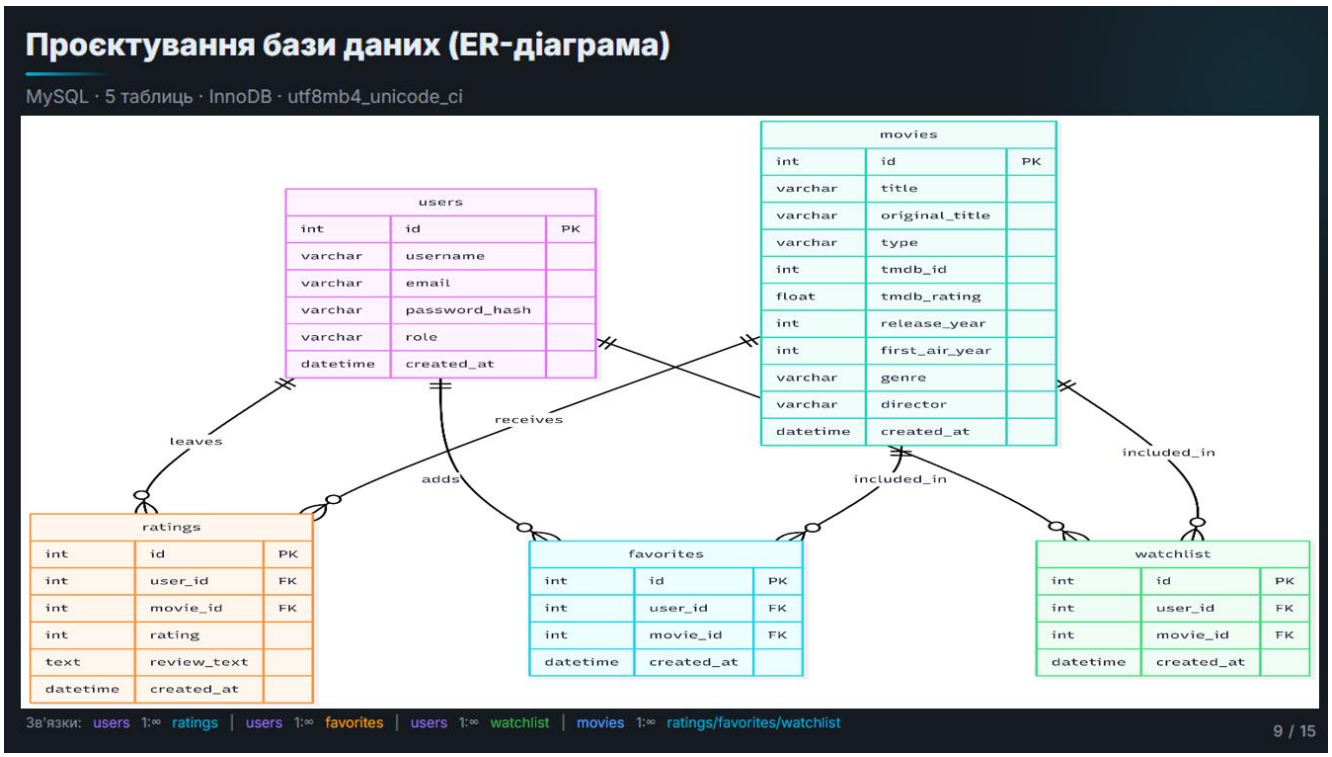

Рисунок Б.9 – Слайд 9

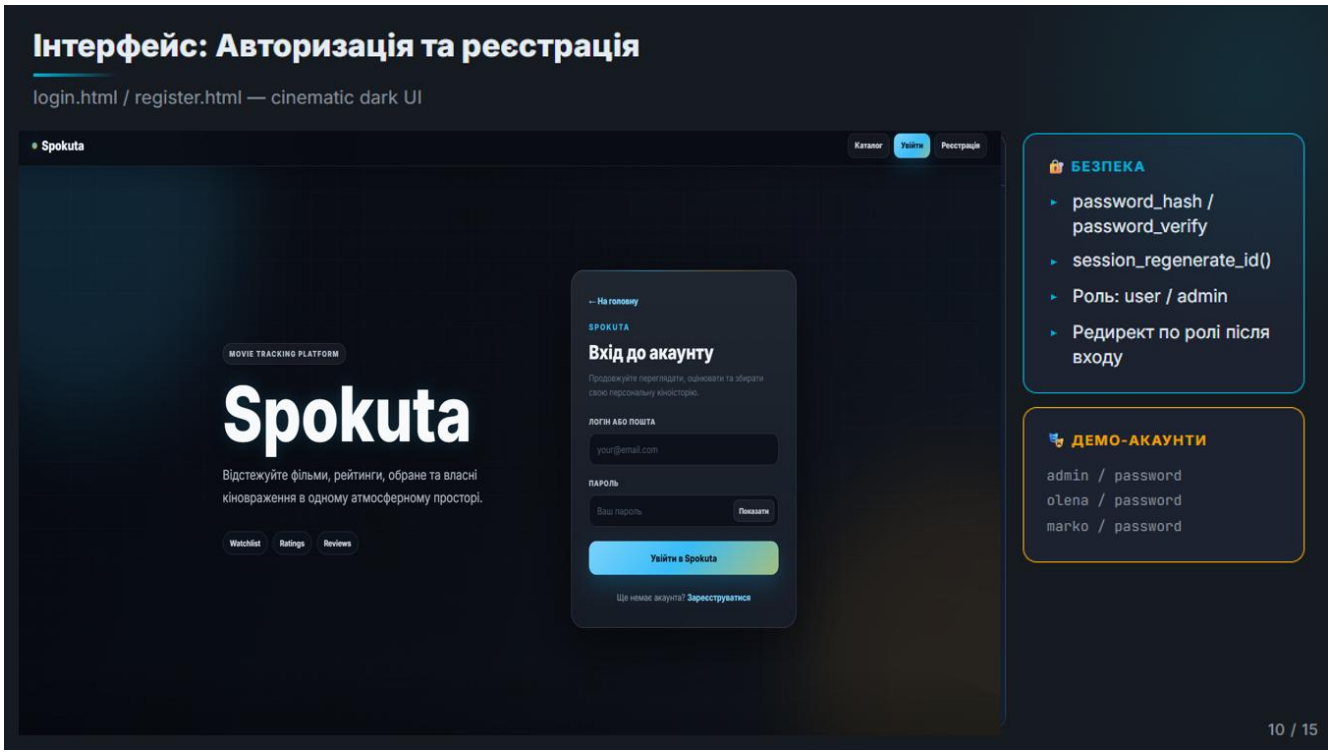

Рисунок Б.10 – Слайд 10

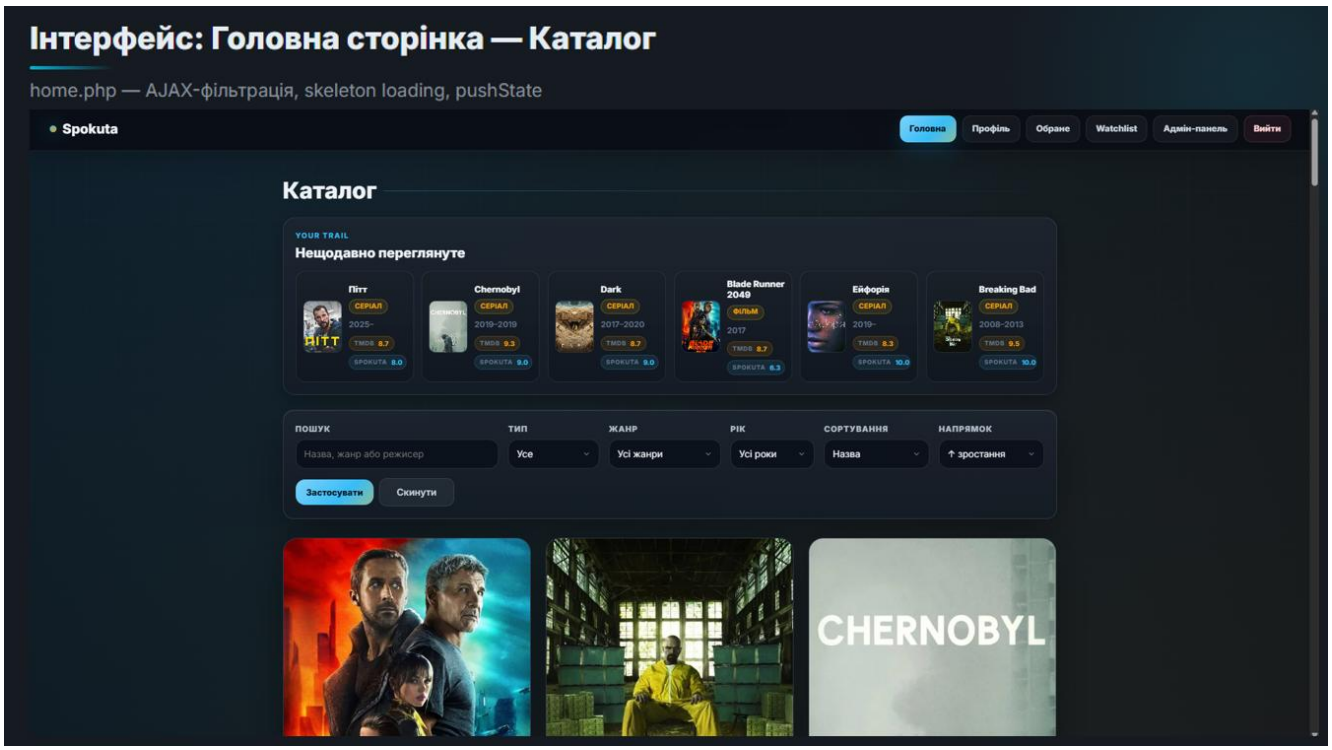

Рисунок Б.11– Слайд 11

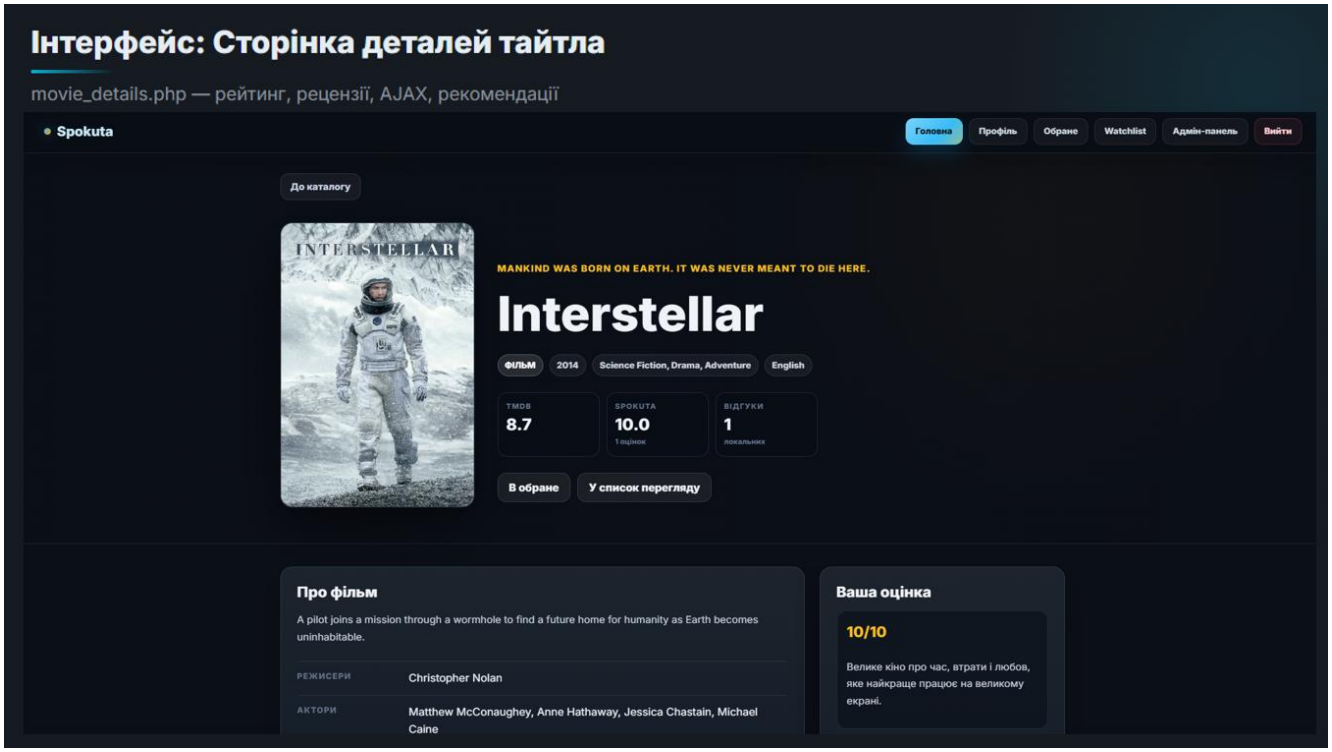

Рисунок Б.12– Слайд 12

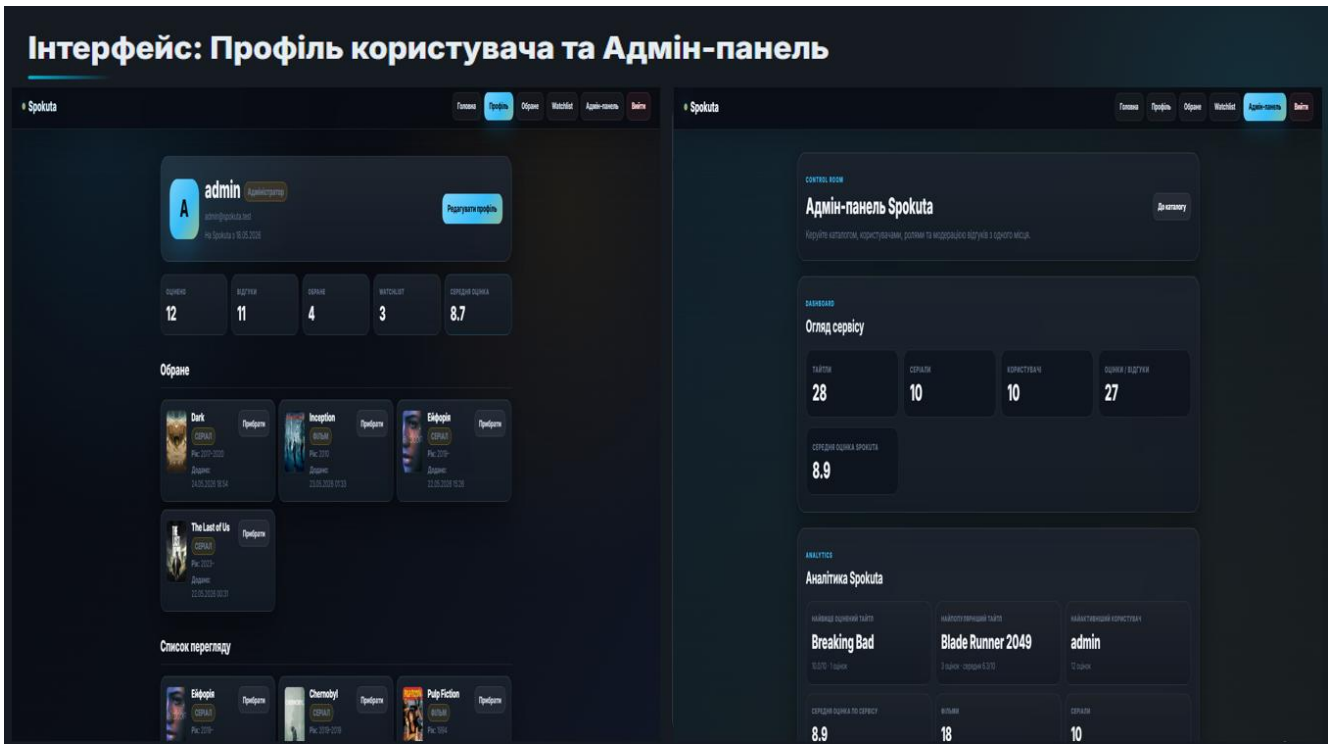

Рисунок Б.13– Слайд 13

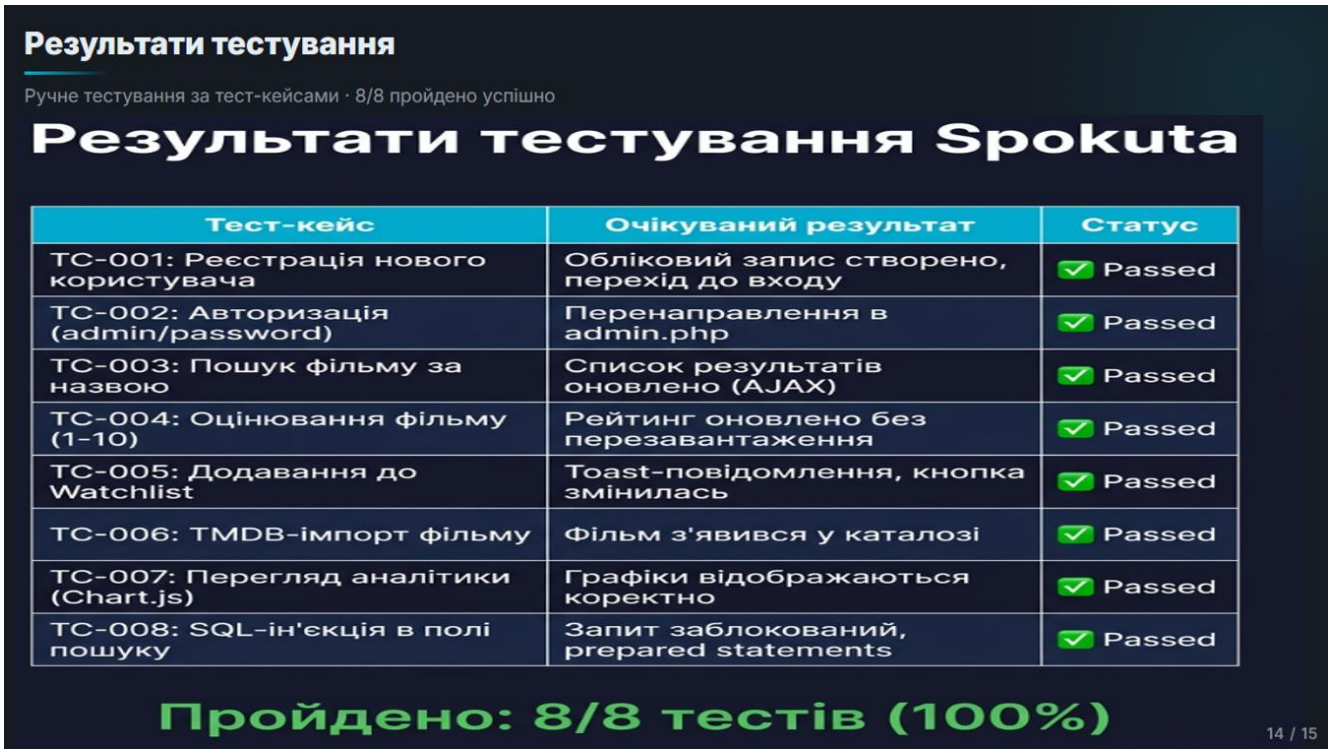

Рисунок Б.14– Слайд 14

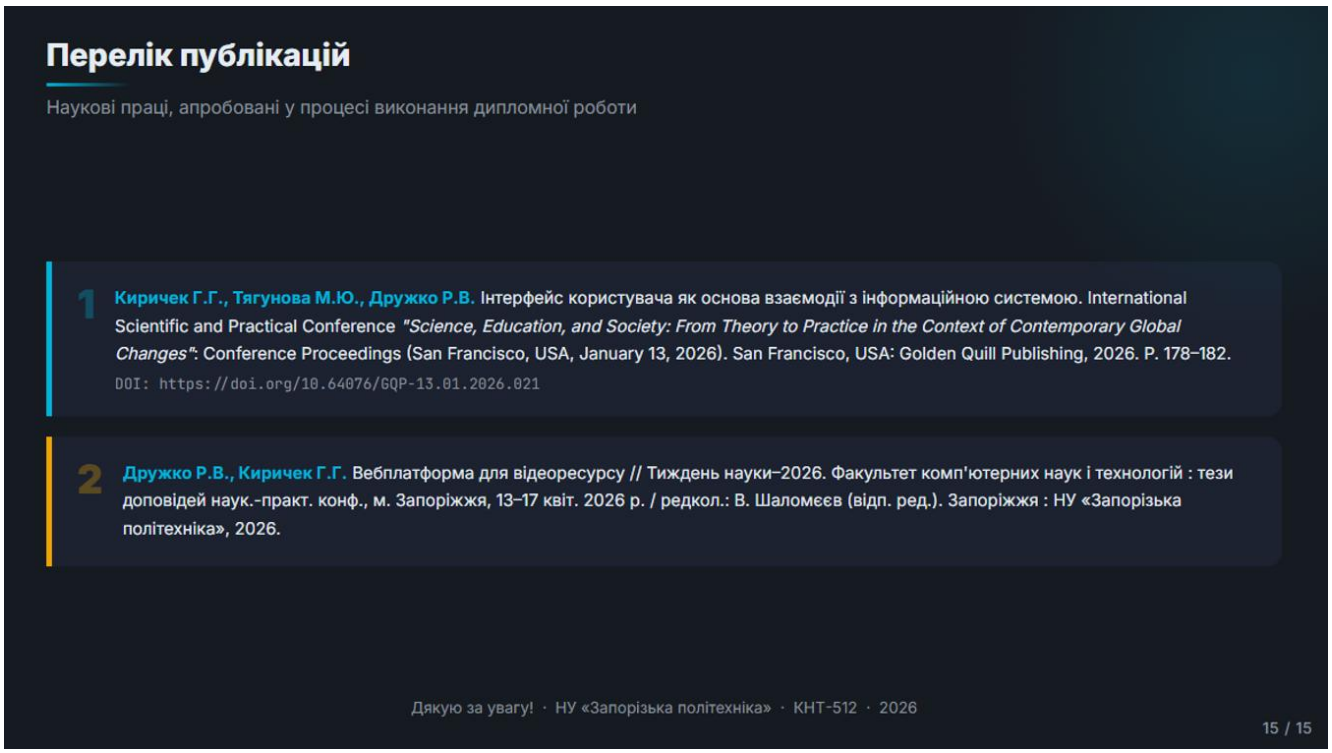

Рисунок Б.15– Слайд 15