

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)

на тему:

«Вебплатформа для рецензування фільмів та серіалів»

Виконав студент групи ІПЗ-22-4
ДЕНИСЕНКО Денис Олександрович

Керівник роботи:
ЄФРЕМОВ Юрій Миколайович

Рецензент:
МОРОЗОВ Андрій Васильович

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»

Зав. кафедри інженерії
програмного забезпечення
Тетяна Вакалюк
«28» лютого 2026 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **ДЕНИСЕНКО Денис Олександрович**

Керівник роботи: **ЄФРЕМОВ Юрій Миколайович**

Тема роботи: **«Вебплатформа для рецензування фільмів та серіалів»**,
затверджена Наказом закладу вищої освіти від **«27» лютого 2026 р., № 92/с**

Вихідні дані для роботи: об'єктом дослідження є процес взаємодії користувачів з кіноконтентом та механізми обміну соціальними відгуками у вебсередовищі; предметом дослідження є методи та інструментальні засоби розробки вебплатформи для публікації та модерації рецензій на фільми та серіали.

Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Єфремов Ю.М.	01.03.2026р.	01.03.2026р
2	Єфремов Ю.М.	01.03.2026р	01.03.2026р
3	Єфремов Ю.М.	01.03.2026р	01.03.2026р

РЕФЕРАТ

Звіт з випускної кваліфікаційної роботи бакалавра був зроблений на тему «Вебплатформа для рецензування фільмів та серіалів». Звіт містить вступ, три розділи, підсумки та перелік використаних джерел. Загальний обсяг звіту складає 113 сторінок, робота містить 104 рисунків, 47 таблиць та 31 джерел літератури.

У звіті розглянуто предметну галузь «Вебплатформа для рецензування фільмів та серіалів» і процеси взаємодії користувачів з медіаконтентом. Проаналізовано головні функціональні спроможності подібних сервісів, зокрема публікацію відгуків, оцінювання контенту, формування рейтингів та персоналізованих порад. Досліджено взаємодію між користувачами вебплатформи, авторами відгуків та адміністраторами системи, а також виявлено обмеження сучасних аналогів. На підставі проведеного аналізу сформульовано концептуальні вимоги до програмного майданчика для рецензування фільмів та серіалів, який буде реалізовано в межах випускної кваліфікаційної роботи бакалавра.

Ключові слова: РЕЦЕНЗУВАННЯ, ФІЛЬМИ, СЕРІАЛИ, РЕЙТИНГИ, КОРИСТУВАЦЬКІ ОЦІНКИ, РЕКОМЕНДАЦІЙНА СИСТЕМА, ВЕБЗАСТОСУНОК, КОРИСТУВАЦЬКИЙ КОНТЕНТ.

					ІПЗ.КР.Б – 121 – 26 – ПЗ			
Змн.	Арк.	№ докум.	Підпис	Дата	Вебплатформа для рецензування фільмів та серіалів	Літ.	Арк.	Аркушів
Розроб.		Денисенко Д.О.						
Перевір.		Єфремов Ю.М.					2	156
Керівник						Житомирська політехніка, група ІПЗ-22-4		
Н. контр.		Тичина О.П.						
Зав. каф.		Вакалюк Т.А.						

ABSTRACT

The bachelor's qualification work report was written on the topic "Web platform for reviewing films and TV series". The report contains an introduction, three sections, a summary and a list of sources used. The total volume of the report is 113 pages, the work contains 104 figures, 47 tables and 31 sources of literature.

The report considers the subject area "Web platform for reviewing films and TV series" and the processes of user interaction with media content. The main functional capabilities of such services are analyzed, in particular, publishing reviews, evaluating content, forming ratings and personalized advice. The interaction between web platform users, review authors and system administrators is studied, and the limitations of modern analogues are also identified. Based on the analysis, conceptual requirements for a software platform for reviewing films and TV series, which will be implemented within the framework of the thesis, are formulated.

Keywords: REVIEWS, FILMS, SERIES, RATINGS, USER RATINGS, RECOMMENDATION SYSTEM, WEB APPLICATION, USER CONTENT.

					ІПЗ.КР.Б – 121 – 26 – ІЗ			
Змн.	Арк.	№ докум.	Підпис	Дата	Вебплатформа для рецензування фільмів та серіалів	Літ.	Арк.	Аркушів
Розроб.		Денисенко Д.О.						
Перевір.		Єфремов Ю.М.					2	156
Керівник						Житомирська політехніка, група ІПЗ-22-4		
Н. контр.		Тичина О.П.						
Зав. каф.		Вакалюк Т.А.						

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ.....	7
ВСТУП	8
РОЗДІЛ 1. АНАЛІЗ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ	
.....	10
1.1. Загальна характеристика проблеми	10
1.2. Аналіз аналогів програмного продукту.....	12
1.3. Вибір архітектури системи	18
1.4. Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення.....	21
Висновки до першого розділу	24
РОЗДІЛ 2. ПРОЄКТУВАННЯ СИСТЕМИ ДЛЯ РЕЦЕНЗУВАННЯ ФІЛЬМІВ ТА СЕРІАЛІВ	25
2.1. Визначення варіантів використання та об'єктно-орієнтованої структури системи.....	25
2.2. Розробка бази даних системи	35
2.3. Проєктування та реалізація алгоритмів роботи системи.....	51
2.4. Реалізація функціоналу системи рецензування фільмів та серіалів.....	59
Висновки до другого розділу.....	62
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ВЕБПЛАТФОРМОЮ ДЛЯ РЕЦЕНЗУВАННЯ ФІЛЬМІВ ТА СЕРІАЛІВ	63
3.1. Визначення варіантів використання та об'єктно-орієнтованої структури системи.....	63
3.2. Структура інтерфейсу вебплатформи.....	66
3.3. Статистика користувача	103
3.4. Тестування вебплатформи	106
Висновки до третього розділу	109

		Денисенко Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		5

ВИСНОВКИ	110
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	111
ДОДАТКИ.....	114

		Денисенко Д.О.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Сфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		6

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API - прикладний програмний інтерфейс

БД - база даних

ПЗ - програмне забезпечення

СУБД - система управління базами даних

MVC - схема розділення даних додатків і керуючої логіки на три окремі компоненти Model-View-Controller

CRUD - основні операції, що використовуються для управління даними (Create, Read, Update, Delete)

DTO - об'єкт передачі даних, що використовується для інкапсуляції даних та їх відправлення між підсистемами

HTTP / HTTPS - протокол передавання гіпертексту

JSON - текстовий формат обміну даними

JWT - відкритий стандарт для створення токенів доступу (використовується для автентифікації)

		Денисенко Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		7

ВСТУП

Актуальність теми. На сьогоднішній день онлайн-кінотеатрів та цифрових медіа стало дуже багато, через це ми витрачаємо на пошук хорошого фільму більше часу, ніж на перегляд. Через це людям простіше орієнтуватися на відгуки та оцінки інших глядачів. Але знайти чесну рецензію дуже важко, через те що популярні сайти зазвичай завантаженою рекламою, мають заплутаний інтерфейс або через свої закриті алгоритми підсовують нам проплачений контент який їм вигідний замість того, що дійсно варто подивитися. Також більшість платформ незручні для спілкування з іншими кінолюбителями. Саме тому зараз дуже потрібна сучасна та проста вебплатформа, де люди могли незалежно ділитися думками про кіно та легко знаходити те, що їм справді сподобається. Майже всі існуючі сайти перевантажені надокучливими рекламними банерами та складними інтерфейсами. Це перетворює процес пошуку на стомлюючий процес, тоді як користувачу потрібно просто та швидко вирішити що переглянути. На даний момент спостерігається чіткий алгоритм де люди більше довіряють відгукам, а не офіційним рейтингам. Отже, розробка такої вебплатформи є вкрай важливою, оскільки вона перетворює процес пошуку кіно з виснажливого обов'язку на приємний досвід, побудований на довірі, простоті та якісному спілкуванні.

Метою випускної кваліфікаційної роботи бакалавра є розробка сайту рецензування фільмів та серіалів, що розробляється в межах випускної кваліфікаційної роботи бакалавра. Для цього досліджено механізми публікації відгуків і формування рейтингів на існуючих сервісах, визначено функціонал майбутньої системи.

Об'єктом дослідження є процес взаємодії користувачів під час пошуку, оцінювання фільмів в інтернет середовищі.

Предметом дослідження є застосування сучасних вебтехнологій, необхідних для створення платформи, що забезпечує публікацію рецензій, розрахунок рейтингів та зручну взаємодію користувачів.

		Денисенко Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		8

Для розв'язання поставлених завдань та розробки платформи застосовуються сучасні методи системного аналізу які досліджують вимоги до продукта, принципи об'єктно-орієнтованого програмування для побудови архітектури, а також методи візуального моделювання для проєктування бази даних і логіки роботи вебзастосунку [29].

Завданням проєкту став аналіз існуючих аналогів сайтів для рецензування фільмів та серіалів, виявити переваги та недоліки, та усунути їх. Спроектувати архітектуру системи та базу даних, використовуючи методи візуального моделювання. Розробити програмну реалізацію основних модулів платформи. Налаштувати систему сповіщень та інтерактивної взаємодії між користувачами. Здійснити розгортання застосунку в хмарному середовищі Azure та провести тестування працездатності системи.

Методикою роботи стало проєктування архітектури системи, розділивши її на фронтенд та бекенд, далі потрібно було створити чітку модель бази даних, щоб всі дані користувачів та фільмів зберігалися впорядковано. Для розробки коду було обрано перевірені архітектурні шаблони, для забезпечення миттєвого спілкування в чатах було використано технологію SignalR. Для якісної роботи вебплатформи потрібно постійно контролювати багаторівневе тестування від перевірки окремих функцій коду до повної імітації дій користувача у браузері. Також окрему увагу було виділено на безпеку, щоб надійно захищати дані користувачів та персоналу. Для того щоб система працювала швидко, було проаналізовано складність алгоритмів і продуктивність API. Розгортання відбувається автоматизовано завдяки Docker та CI/CD пайплайнів, які самі збираються перед кожним оновленням. Весь цей підхід дозволив створити надійну та масштабовану вебплатформу для рецензування фільмів та серіалів.

За темою випускної кваліфікаційної роботи бакалавра було опубліковано тези на “Міжнародної науково-технічної конференції Інформаційно-комп'ютерні технології” - 2026, 02-03 квітня 2026 року [1, 2].

		Денисенко Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		9

РОЗДІЛ 1. АНАЛІЗ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

1.1. Загальна характеристика проблеми

Під час підготовки до розробки необхідно було дослідити сферу онлайн платформ для рецензування та чітко сформулювати завдання для майбутнього проєкту.

Створювана вебплатформа має орієнтуватися на потреби одразу кількох ключових груп користувачів: звичайних глядачів, активних авторів рецензій та адміністраторів системи. На багатьох існуючих ресурсах процеси пошуку фільмів, публікації відгуків, їхньої модерації та розрахунку рейтингів реалізовані незручно або працюють відокремлено. Це ускладнює взаємодію аудиторії та знижує загальну корисність таких сервісів для користувача [3].

Тому головним завданням стало формування концепції зручного та швидкого середовища. Платформа повинна забезпечувати надійне централізоване збереження бази даних про медіаконтент та профілі користувачів. Необхідно реалізувати зручний засобів для публікації та редагування відгуків, прозору систему оцінювання для формування рейтингів, а також закласти архітектурну основу для аналізу вподобань глядачів і впровадження системи рекомендацій.

Процеси пошуку медіавмісту, публікація рецензій, оцінення, модерації та формування показників які втілюються з використанням обмежених або роздроблених засобів, які ускладнюють зручну взаємодію користувачів та зменшують користь використання майданчика.

Завданням випускної кваліфікаційної роботи бакалавра є формування основ програмного середовища, які забезпечать підтримку головних процесів рецензування фільмів та серіалів.

Окрім чисто технічних вимог та проблем існує великий брак простого людського спілкування. Більшість платформ працюють за принципом

		Денисенко Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремов Ю.М.				10
Змн.	Арк.	№ докум.	Підпис	Дата		

швидкого оцінення, але кіно це не просто витрата часу це відчуття яким хочеться ділитися.

В глядачів не вистачає місця, де можна не просто поставити зірочку фільму, а вступити в дискусію з людьми, які мають схожі смаки, або знайти нові цікаві добірки, сформовані не роботами, а реальними кінолюбителями. Саме тому потрібно поєднати функціональність потужної вебплатформи з духом невеликого, але активного клубу за інтересами. У роботі над цим проєктом я приділив особливу увагу саме інструментам для авторів. Важливо людина яка хоче написати детальну рецензію, не має відчувати технічні обмеження. Важливо закласти таку основу, яка дозволить вебплатформі зростати разом з її аудиторією.

Сьогодні це може бути проста система відгуків, а вже завтра це повноцінна база для аналізу вподобань, яка допоможе користувачам відкривати для себе те кіно, яке вони самі ніколи б не знайшли. Вибір технологій для цієї вебплатформи теж був вибраний не випадково. Використання сучасного стека C# та React, дає можливість зробити інтерфейс швидким та чуйним - щоб сторінки завантажувалися миттєво, а будь-яка взаємодія з сайтом приносила задоволення, а не роздратування. Це стало фундаментом, який дозволяє впевнено дивитися в майбутнє коли система повинна працювати так само стабільно, як і на старті. Потрібно перестати витрачати час на боротьбу з інтерфейсами чи нечесними рейтингами.

Головною задачею стала можливість надати інструмент, який поверне людям задоволення від вибору кіно, зробивши цей процес максимально прозорим, швидким і справжнім. Саме такий персоналізований та відкритий підхід здатний змінити правила гри, перетворюючи звичайний перегляд на справжню пригоду.

Це стало першим кроком до створення простору, де технології нарешті починають служити не алгоритмам, а живим людям та їхнім щирим враженням.

		Денисенко Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремюв Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		11

1.2. Аналіз аналогів програмного продукту

На сьогоднішній день вебплатформи для глядачів та рецензентів стали дуже популярними. Вони допомагають людям шукати фільми та серіали, а також зберігати в улюблене, ділитися думками та переглядати оцінки інших. Ці сервіси сильно впливають саме на глядачі які обирають їх. Щоб довести доцільність розробки власної вебплатформи для відгуків та рецензій у межах бакалаврської роботи, потрібно розібратися, які схожі проєкти вже доступні на просторах інтернету.

Вивчення аналогів допоможе зрозуміти їхні головні фішки та інструменти. За допомогою детального аналізу популярних сайтів можна виділити важливі функції такі як, система оцінювання, спілкування між користувачами та рейтингова система. Це допоможе зробити власний продукт зручним та конкурентно спроможним.

Для порівняння було обрано три найвідоміші світові платформи, які мають величезні бази відеоконтенту та збирають мільйони відгуків: IMDb, Rotten Tomatoes та Letterboxd.

IMDb (Internet Movie Database) - це найбільших і найстаріших сайт про кінематограф. Його головна ідея це величезна й чітка база даних, в якій можна знайти буквально все про фільми та серіали: акторський склад, команди творців, дати виходу, тривалість, жанри, касові збори та інші.

Посилання на IMDb - (<https://www.imdb.com>)

Таблиця 1.1.

Переваги та недоліки платформи IMDb

Переваги	Недоліки
Велика та постійно оновлювана база даних фільмів і серіалів.	Обмежені можливості персоналізації рекомендацій.
Наявність детальної інформації про медіаконтент (жанри, акторський склад, рейтинги, дати релізів).	Складний інтерфейс для нових користувачів через велику кількість інформації.

Переваги	Недоліки
Популярна система користувацького рейтингу, що базується на великій кількості оцінок.	Відсутність гнучких інструментів модерації користувацького контенту.
Висока впізнаваність платформи серед користувачів у всьому світі.	Неможливість детального аналізу рецензій та користувацьких уподобань.

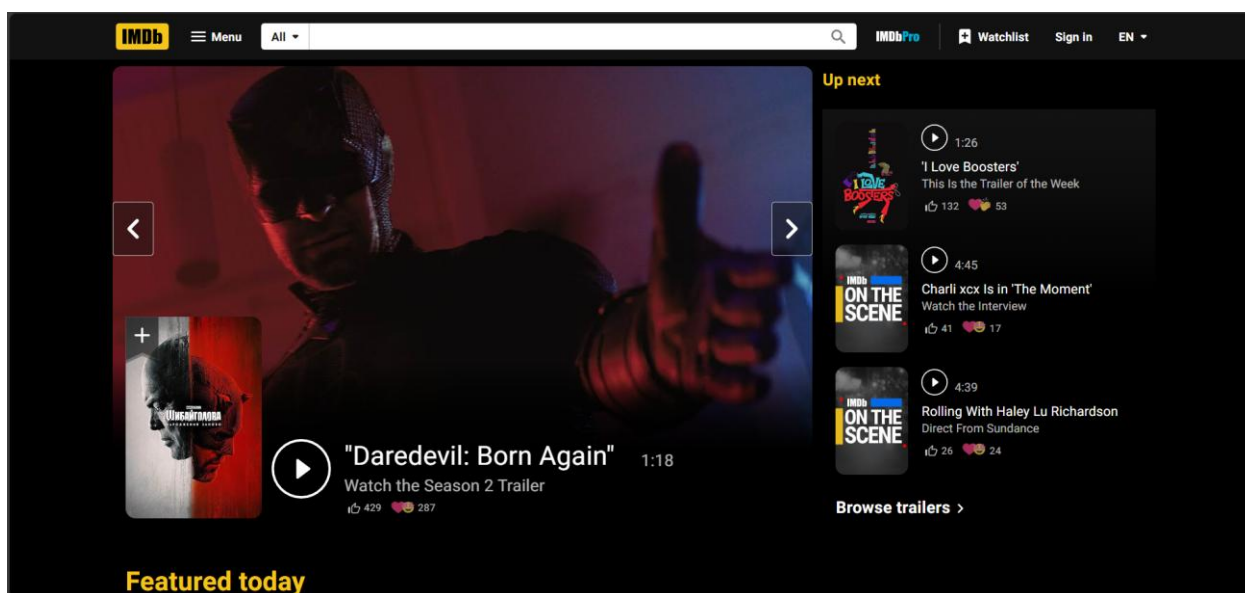


Рис.1.1. Інтерфейс сайту IMDb

Rotten Tomatoes це ще один дуже популярний сайт з відгуками, але його головна задумка в тому, що він збирає рецензії як від професійних кінокритиків так і від звичайних глядачів. Система оцінювання тут влаштована незвично, замість звичного середнього бала сервіс показує відсоток позитивних відгуків. Якщо більшості критиків фільм сподобався, він отримує статус «свіжий», а якщо ні то «гнилий». Такий підхід дозволяє користувач буквально за секунду зрозуміти, чи варто витратити час на кіно. Крім того, на вебплатформі можна порівняти думку професіоналів із враженнями звичайних людей. Часто ці оцінки сильно відрізняються, що допомагає подивитися на фільм з різних боків і зробити власні висновки.

		Денисенко Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		13

Переваги та недоліки платформи Rotten Tomatoes

Переваги	Недоліки
Чітка та зрозуміла система оцінювання якості контенту.	Обмежені можливості взаємодії між користувачами.
Розмежування професійних рецензій і відгуків звичайних користувачів.	Мінімальна персоналізація контенту.
Зручний інструмент швидкої оцінки фільмів і серіалів.	Орієнтація переважно на професійних критиків.
Висока довіра до рейтингу серед аудиторії.	Відсутність соціальних функцій та інструментів формування спільнот.

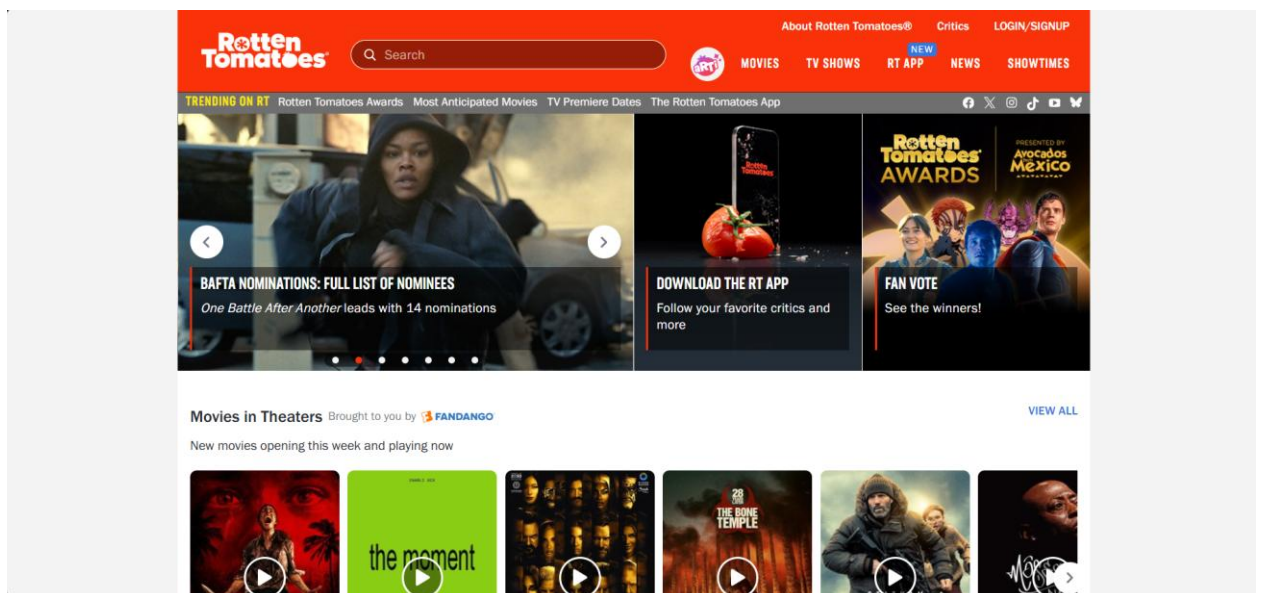


Рис.1.2. Інтерфейс сайту Rotten Tomatoes

Letterboxd це платформа, яка поєднує в собі сайт для рецензування та соціальну мережу, яка повністю реалізує логіку соціальних мереж. Головна особливість це орієнтація на спільноту. Користувачі можуть не тільки оцінювати фільми, а й писати рецензії, створювати тематичні списки, підписуватися на профілі інших користувачів та інший функціонал. На сайті легко знайти однодумців та обговорити кіно в коментарях. Вебплатформа дає

багато можливостей для персоналізації свого профілю, але водночас вона має мінуси, для прикладу майже немає інструментів для глибокого аналізу даних чи розширеного керування контентом. Крім того, сервіс лише частково підтримує серіали.

Посилання на Letterboxd (<https://letterboxd.com>)

Таблиця 1.3.

Переваги та недоліки платформи Letterboxd

Переваги	Недоліки
Соціальна спрямованість платформи та активна спільнота користувачів.	Обмежена підтримка серіалів порівняно з фільмами.
Зручні інструменти для публікації рецензій і ведення особистого кіножурналу.	Обмежена підтримка серіалів порівняно з фільмами.
Можливість створення списків фільмів і підписки на інших користувачів.	Обмежені можливості з адміністративного керування контентом.
Простий та інтуїтивно зрозумілий інтерфейс.	Залежність рекомендацій від активності користувачів без глибокого аналізу даних.

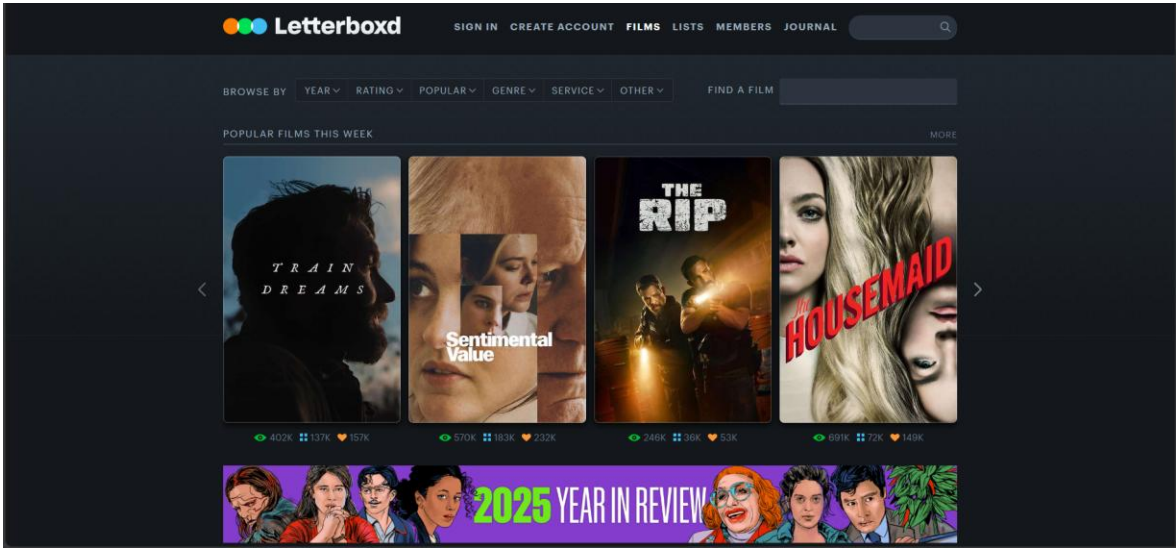


Рис.1.3. Інтерфейс сайту Letterboxd

На основі виконаного аналізу існуючих програмних рішень було складено порівняльну таблицю ключових характеристик досліджуваних платформ.

Таблиця 1.4.

Порівняльна таблиця ключових характеристик

Характеристика	Kovix	IMDb	Rotten Tomatoes	Letterboxd
Тип платформи	Гібридна соціально-інформаційна платформа	Інформаційно-рецензійна база даних	Агрегатор професійних та користувацьких рецензій	Соціальна платформа для кінолюбителів
Підтримка фільмів	Так	Так	Так	Так
Підтримка серіалів	Так	Так	Так	Обмежено
Користувацькі рецензії	Ні	Так	Так	Так
Професійні рецензії	Так	Обмежено	Так	Ні
Система оцінювання	Користувацький рейтинг (1-10) / Зіркова система	Користувацький рейтинг (1-10)	Відсотковий рейтинг	Зіркова система (0,5-5)
Формування рейтингів	Так	Так	Так	Так
Соціальна взаємодія	Середня	Обмежена	Мінімальна	Розширена
Персоналізовані рекомендації	Планується	Обмежено	Ні	Частково
Створення списків	Так	Так	Ні	Так
Аналітичні інструменти	Середні	Обмежені	Орієнтовані на оцінки	Відсутні
Модерація контенту	Адміністративна	Часткова	Централізована	Обмежена
Інтерфейс користувача	Сучасний та інтуїтивний	Інформаційно насичений	Простий	Мінімалістичний

продовж. табл.1.4

Характеристика	Kovix	IMDb	Rotten Tomatoes	Letterboxd
API / Доступ до даних	Ні	Так	Так	Частково
Мобільні додатки	Ні	Так	Так	Так
Імпорт/експорт даних	Ні	Обмежено	Ні	Так
Де подивитися	Ні	Так	Так	Так
Редакційний контент	Базовий	Так	Так	Обмежено
Бейджі / Верифікація критиків	Так	Ні	Так	Ні
Мультимовність інтерфейсу	Ні	Так	Ні	Ні
Вхід через соцмережі	Ні	Так	Так	Так
Українська мова	Так	Ні	Ні	Ні
Орієнтація на спільноту	Висока	Середня	Низька	Висока

Порівняння платформ (табл.1.4) показує, що жоден із відомих сайтів не є ідеальним та не здатний закрити абсолютно всі потреби сучасного глядача. Більшість сервісів роблять ставку на щось одне, збирання великої бази даних, фокусування на професійних критиках, або розвиток соціальної мережі. Через це користувачам часто бракує точних персональних рекомендацій, авторам - зручної аналітики, а адміністраторам - гнучких інструментів для модерації коментарів та відгуків.

Саме тому розробка власної вебплатформи є цілком виправданою та актуальною [31]. Проєкт дозволить об'єднати найкращі ідеї конкурентів, зручну систему рецензування, активну спільноту та розумні рекомендації. При цьому система матиме простий і зрозумілий інтерфейс, у якому легко розбереться будь-який користувач.

1.3. Вибір архітектури системи

Вибір архітектурного стилю - це один із найважливіших кроків під час розробки платформи. Від цього рішення залежить не лише те, як взаємодітимуть частини коду між собою, а й наскільки зручно буде розвивати проєкт у майбутньому: масштабувати його, додавати новий функціонал та забезпечувати стабільну роботу без збоїв.

Враховуючи специфіку проєкту - зокрема, необхідність швидко обробляти великі обсяги даних у реальному часі - ми проаналізували сучасні підходи до розробки. Для побудови системи було виділено два основні варіанти: монолітну та мікросервісну архітектури.

Монолітна архітектура цей підхід передбачає створення системи як єдиного, суцільного блоку. У такому випадку всі компоненти (користувацький інтерфейс, серверна логіка та робота з базою даних) знаходяться разом і тісно пов'язані всередині одного застосунку.

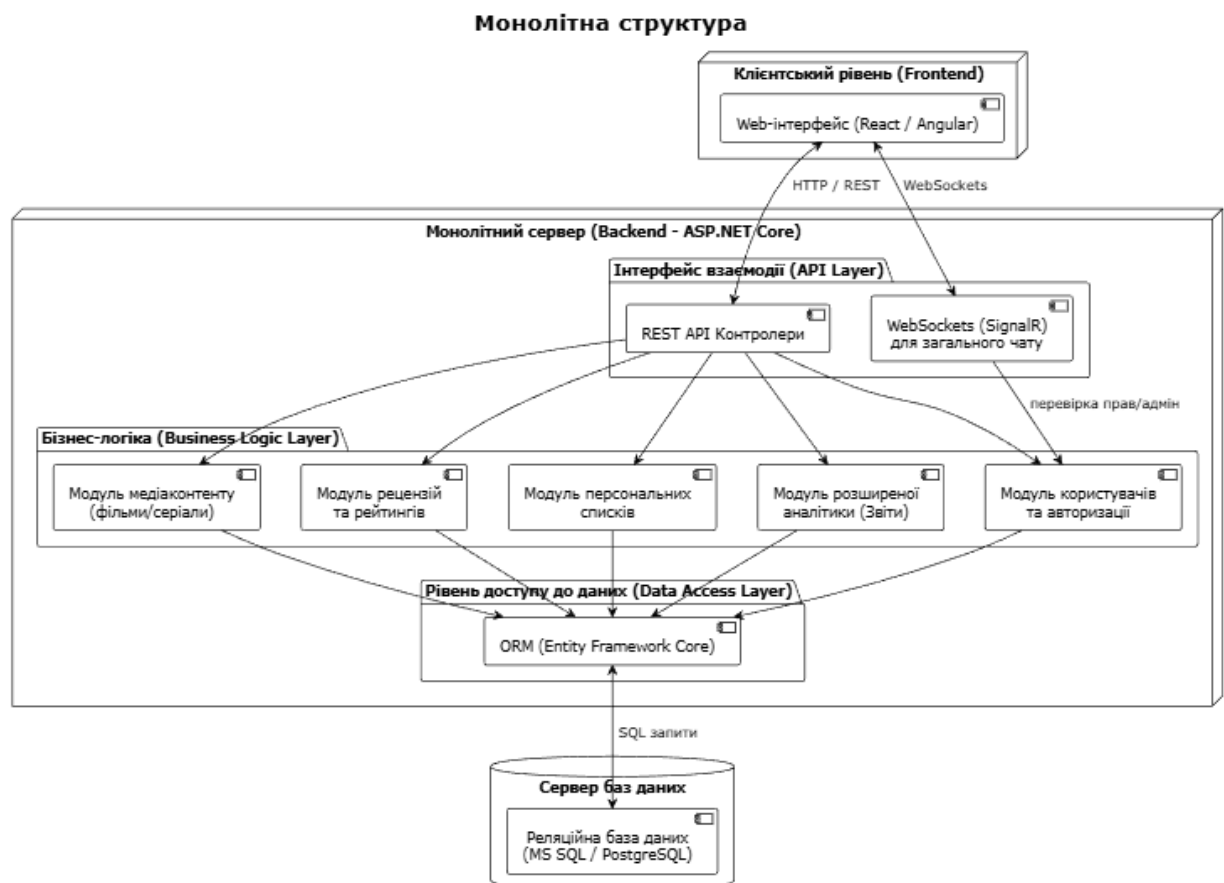


Рис.1.4. Схема монолітної архітектури

Мікросервісна архітектура передбачає розділення системи на набір невеликих незалежних сервісів. Кожен з них відповідає за свою конкретну задачу і спілкується з рештою системи через API.

Головна перевага цього підходу висока гнучкість, платформу легко масштабувати, а її окремі частини можна оновлювати паралельно, не впливаючи на роботу інших. Але розробка розгортання та підтримка мікросервісів є технічно набагато складнішими. Саме тому цей підхід іноді буває виправданим на початкових етапах розробки продукту для систем середньої складності.



Рис.1.5. Схема мікросервісної архітектури

Зваживши всі переваги та недоліки класичних підходів, було обрано для вебплатформи модульну монолітну архітектуру. Цей варіант дозволяє розробляти й запускати систему як один цілісний застосунок. Завдяки цьому зберігаються головні переваги звичайного моноліту, простіше тестування, розгортання на сервері та розробка йде набагато швидше.

Водночас усередині коду діє суворе правило, кожна велика функція проєкту живе у своєму власному незалежному модулі. Ці модулі спілкуються між собою лише через чітко визначені інтерфейси, а вся їхня внутрішня логіка прихована від інших частин системи.

Такий підхід не дає коду перетворитися на заплутаний і хаотичний «клубок». Але найголовніше це фундамент на майбутнє. Оскільки всі

компоненти слабко пов'язані між собою, у разі зростання кількості користувачів будь-який модуль можна буде легко дістати з моноліту й перетворити на окремий мікросервіс без переписування всієї системи з нуля [4].

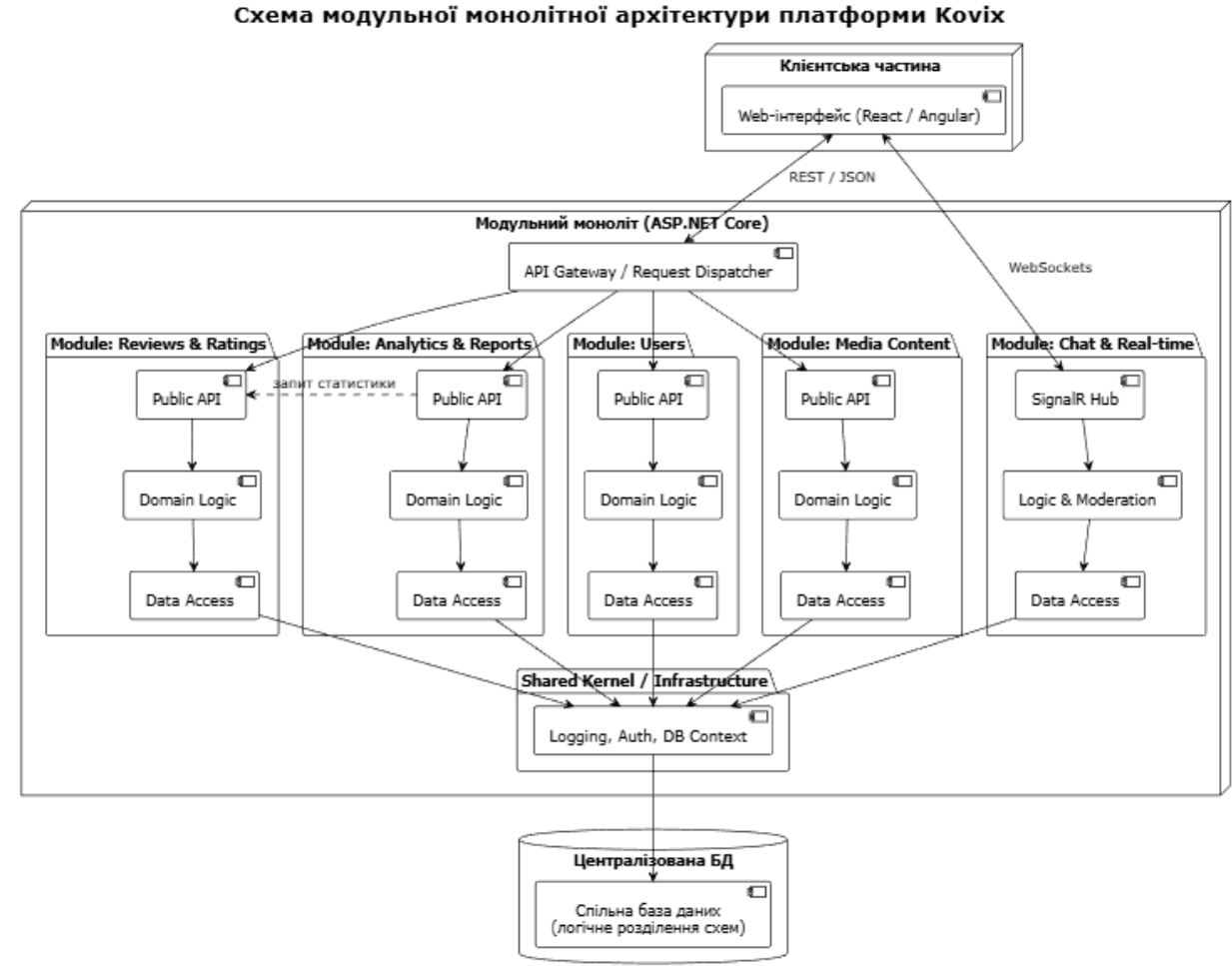


Рис.1.6. Схема архітектури модульного моноліту

Загальна архітектура вебплатформи базується на перевірених клієнт-серверній моделі, яка сьогодні є стандартом для подібних вебпроектів. Вона дозволяє чітко розділити інтерфейс користувача та внутрішню логіку системи [6].

Клієнтська частина - це сучасний вебзастосунок. Завдяки такому підходу користувачі можуть заходити на вебплатформу через будь-який браузер на комп'ютері чи смартфоні, незалежно від операційної системи. Серверна ж частина бере на себе всю складну роботу, обробку запитів, виконання бізнес-логіки та безпечне збереження інформації. Такий поділ обов'язків значно

спрощує підтримку проєкту, гарантує захист даних та підвищує загальну швидкодію системи.

Зважаючи на проведені архітектурні дослідження, поєднання клієнт-серверної моделі із модульним монолітом на сервері стало найкращим рішенням для проєкту. Це дозволяє зробити інтерфейс легким і швидким для звичайного глядача, тоді як добре структурована серверна частина ефективно й надійно обробляє оцінки, рецензії та профілі користувачів.

1.4. Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення

Вебплатформа Kovix поєднує в собі інформаційну базу кіноконтенту, можливість перегляду трейлерів, інструменти для створення списків, а також інтерактивні елементи такі як, загальні чати та систему рецензій. Реалізація такого набору функцій вимагає обробки даних у реальному часі, збереження складної реляційної структури та створення швидкого інтерфейсу. З огляду на це, для розробки системи за принципом модульного моноліту було обрано збалансований стек технологій [5].

Основою серверної логіки було обрано мову програмування C# та платформу ASP.NET. C# - сучасна об'єктно-орієнтована мова програмування з суворою типізацією, що допомагає виявляти помилки ще на етапі розробки, це гарантує стабільну роботу на рівні бізнес-логіки. Використання фреймворку .NET Core дозволяє реалізувати швидкий кросплатформний REST API, здатний витримувати високі навантаження [28]. Також вбудована технологія SignalR забезпечує постійне з'єднання між сервером і клієнтом для миттєвого обміну повідомленнями в чатах у режимі реального часу.

Робота з даними. Для збереження інформації використовується реляційна SQL-база даних. Такий вибір зумовлений чіткою структурою предметної області, де всі сутності такі як профіль користувача, фільми, відгуки та оцінки суворо пов'язані між собою. SQL-база забезпечує цілісність цих зв'язків на рівні зовнішніх ключів, гарантує надійність і дозволяє швидко виконувати складні запити для збору аналітики.

		Денисенко Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремюв Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		21

Взаємодія сервера з базою даних реалізована через Entity Framework Core. Ця система дозволяє працювати з таблицями як з звичайними об'єктами у коді, автоматизовує написання одноманітних запитів і надає зручний механізм міграцій для безпечного оновлення структури бази даних.

Клієнтську частину побудовано на бібліотеці React [15]. Головним плюсом є компонентний підхід. Він дозволяє створити елемент інтерфейсу один раз, а далі використовувати його на різних сторінках це сильно спрощує роботу з кодом і прискорює розробку. React оновлює інтерфейс динамічно, через це сторінки не перезавантажуються повністю. Завдяки цьому сайт працює швидко та плавно. Також у проєкті для переходів між сторінками було використано React Router DOM, для відправки запитів на сервер Axios та Vite для швидкої збірки проєкту.

Для розгортання платформи та налаштування середовища використовується Docker [26]. Він дозволяє запакувати весь вебзастосунок разом з його залежностями й конфігураціями в один ізольований контейнер. Завдяки цьому зникає проблема сумісності, система працюватиме однаково стабільно як на локальному комп'ютері під час розробки, так і на будь-якому Linux- чи Windows-сервері.

Для стабільної роботи вебплатформи Kovix у розгорнутому вигляді встановлено такі мінімальні та рекомендовані системні вимоги:

Мінімальні вимоги для розгортання системи:

- Оперативна пам'ять: 8 ГБ RAM (рекомендовано 16 ГБ)
- Процесор: 4-ядерний процесор (рекомендовано 8 ядер)
- Дискове сховище: 50 ГБ вільного простору на SSD
- Мережа: стабільне з'єднання 100 Мбіт/с, HTTPS-сертифікація
- Операційна система: Linux (Ubuntu 22.04+) або Windows Server 2019+ з підтримкою Docker

Обраний набір технологій є оптимальним для бакалаврського проєкту, оскільки він забезпечує високу продуктивність, швидкість розробки та дозволяє легко масштабувати платформу в майбутньому.

		Денисенко Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремюв Ю.М.				22
Змн.	Арк.	№ докум.	Підпис	Дата		

Окрім технічних переваг ця сукупність технологій дуже спрощує підтримку проєкту по мірі його зростання. Завдяки Entity Framework Core та його міграціям базі даних дуже легко змінювати структуру даних без зупинки роботи вебплатформи та не хвилюватися за можливість втрати персональні дані.

Для фронтенду був обраний потужна бібліотека React яка разом з всіма інструментами Vite забезпечує максимально швидку збірку проєкту, що робить його дуже зручним для постійного тестування нових ідей та швидкого редагування. Цей підхід до розробки, який починається від вибору мови програмування до методів розгортання Docker, робить Kovix гнучким та надійним. Це дозволило отримати вебплатформу, яка легко масштабується, швидко оновлюється та не потребує складного налаштування на новому сервері це критично важливо для реального використання на сьогоднішній день.

Також впевненість дає вибір технологій які підвищують стабільність системи при масштабуванні кількості користувачів. Оскільки кожен компонент проєкту відповідає за свою частину роботи це допоможе покращувати продуктивність окремих частин вебплатформи, не переписуючи весь код з нуля. Завдяки зрозумілій архітектурі, новим розробникам буде легше зрозуміти структуру проєкту та долучитися до його розробки, що важливо для сервісної підтримки. В результаті було отримано гнучкий інструмент, який дозволяє швидко адаптуватися до нових запитів аудиторії та сучасних тенденцій веброзробки, залишаючись при цьому надійним рішенням для щоденного використання.

У майбутньому це дозволить зосередитися на розробці функціональності, покращення алгоритмів рекомендацій або додавання соціальних функцій, не витрачаючи час на можливі проблеми з технічними обмеженнями чи конфліктами середовища. Це збалансоване рішення дозволяє випустити готовий та високоякісний продукт у рамках випускної кваліфікаційної роботи бакалавра.

		Денисенко Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		23

Висновки до першого розділу

У першому розділі було визначено мету проєкту та сформовано основні завдання для створення вебплатформи Kovix. На основі аналізу предметної галузі було виділено головні функції майбутньої системи, серед яких, пошук відеоконтенту, виставлення оцінок, написання рецензій та коментарів і ведення персональної статистики переглядів.

Також було проведено детальний аналіз популярних сайтів аналогів, пошук їхніх слабких місць та сильних сторін. Це дозволило чітко зрозуміти, які саме інструменти потрібні нашому проєкту, щоб виділитися на ринку, а всі результати порівняння було зібрано у зведену таблицю. Окрему увагу було приділено вивченню архітектурних підходів для побудови вебплатформ, які працюють із великою кількістю користувачів.

Для розробки Kovix було обрано модульну монолітну архітектуру. Таке рішення дало змогу швидко створити й запустити систему як один застосунок, спростило його підтримку, але водночас воно гарантує, що в майбутньому будь-яку велику функцію можна буде легко перетворити на окремий мікросервіс або модуль.

Зважаючи на архітектуру та завдання проєкту, було затверджено оптимальний технологічний стек ASP.NET Core на бекенді та React на фронтенді. Наприкінці розділу було визначено мінімальні вимоги до серверного обладнання, які потрібні для стабільного розгортання вебплатформи в ізольованих Docker-контейнерах.

		Денисенко Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		24

РОЗДІЛ 2. ПРОЄКТУВАННЯ СИСТЕМИ ДЛЯ РЕЦЕНЗУВАННЯ ФІЛЬМІВ ТА СЕРІАЛІВ

2.1. Визначення варіантів використання та об'єктно-орієнтованої структури системи

Головним завданням при розробці вебплатформи це було створення стабільного застосунку, який має працювати під навантаженням та швидко обробляти запити користувачів. Це робиться для того щоб розмежувати права доступу та гнучко керувати контентом на вебплатформі Kovix, в таблицях нижче буде наведено короткий опис функціоналу для кожної з ролей для більш детального розуміння роботи програми [7].

У системі передбачено п'ять основних ролей:

Таблиця 2.1.

Ролі вебплатформи

Гість (неавторизований користувач)	Зареєстрований користувач	Критик (експерт)	Модератор	Адміністратор
---------------------------------------	---------------------------	---------------------	-----------	---------------

Для неавторизованих гостей та зареєстрованих користувачів платформа виконує роль інформаційного та інтерактивного хабу. Без реєстрації на сайті доступні такі базові функції:

Таблиця 2.2.

Базові функції для користувачів

Швидкий пошук фільмів та серіал за ключовими словами	Фільтрація та сортування контенту за жанрами, роком виходу, країною чи рейтингом	Перегляд детальної картки медіаконтенту	Перегляд офіційних трейлерів у модальному вікні та ознайомлення з глобальними трендами кінопрокату.
--	--	---	---

Після того як користувач створить особистий профіль, зареєструється або заїде на сайт він отримус розширений функціонал який продемонстрований в таблиці нижче:

Таблиця 2.3.

Функціонал для зареєстрованих користувачів

Можливість створювати власні добірки фільмів, додавати стрічки до категорій «Улюблене» або «Подивитися пізніше»;	Виставлення оцінок за 10-бальною шкалою, написання розгорнутих текстових відгуків	Участь у загальних чатах, коментування рецензій інших учасників спільноти	Отримання індивідуальних рекомендацій на основі історії переглядів та виставлених раніше оцінок
--	---	---	---

Критик це спеціально призначена роль для професійних авторів чи кіно експертів. Цей статус видається адміністрацією після того як користувач пройде підтвердження кваліфікації. Критик має всі можливості звичайного користувача, але на додачу володіє унікальними інструментами:

Таблиця 2.4.

Функціонал для критика

Відображення спеціального бейджа біля імені, що підвищує довіру до його матеріалів	Публікації мають вищий пріоритет у стрічці відгуків і відображаються в окремому блоці на картці фільму	Оцінки критиків формують окремий «професійний рейтинг» фільму, що дозволяє звичайним глядачам порівнювати думку експертів і публіки
--	--	---

Головна вимога до звичайного користувацького інтерфейсу це мінімалізм та висока швидкодія для завантаження сторінок та інтуїтивно зрозуміла навігація.

Для забезпечення життєдіяльності вебплатформи, контролю якості кіноконтенту та підтримки порядку в спільноті було передбачено адміністративну панель з чітким розділенням на обов'язки:

Таблиця 2.5.

Різниця між модератором та адміністратором

Модератори	Адміністратори
Відповідають за текстове наповнення сайту. Вони перевіряють користувацькі відгуки, коментарі та повідомлення в чатах, очищають платформу від спаму, образ чи погрози, а також обробляють скарги від глядачів	Вони мають повний доступ до керування системою. Наповнення бази даних, керування обліковими записами (переглядають активність користувачів, видають права модераторам, блокують порушників правил).

Робоче місце для адміністратора спроектовано дуже просто, але його простий вигляд має в собі потужний дашборд. Він взагалі немає перевантажених корпоративних таблиць, для керуючого це дуже простий інтерфейс який допоможе фокусуватися на швидкому доступі до статистики, логів помилок чи інструментів швидкого редагування даних також це дозволить оперативно реагувати на будь-які технічні збої чи запити від користувачів.

Характеристика об'єкта комп'ютеризації

Об'єктом автоматизації у межах випускної кваліфікаційної роботи бакалавра є процеси оцінювання та рецензування медіа, а також соціальні взаємодії користувачів навколо кіно та серіалів. На сьогоднішній день ці процеси є розбитими на детальну інформацію про релізи, акторський склад, відгуки критиків, новини чи майданчики для обговорення розміщені на різних модулях, які не пов'язані між собою на сайті. Така ізольованість ускладнює швидкий аналіз контенту та заважає створенню цілісного й зручного простору для кіно любителів [8].

		Денисенко Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		27

Вебплатформа Kovix вирішує цю проблему завдяки централізації всіх процесів у межах єдиної системи. Вона об'єднує каталог фільмів та серіалів, механізми оцінювання й рецензування, інструменти модерації, соціальні функції та систему індивідуальних рекомендацій.

Основними сутностями предметної області, які моделюються в базі даних проєкту для забезпечення взаємодії користувачів було виділено чотири основних: “Користувач”, “Скарга”, “Апеляція” та “Досягнення”. Функціонування цих сутностей підтримується роботою окремих підсистем, зокрема модулями сповіщень у реальному часі, інтерактивних чатів та генерації персональних рекомендацій.

Таблиця 2.6.

Функціональні та нефункціональні вимоги

Функціональні вимоги	
Користувачі та соціальна взаємодія	Реєстрація, вхід, відновлення пароля, підтвердження email, редагування профілю
	Вхід через Google
	Додавання у друзі, підписки, блокування користувачів, перегляд публічних профілів та рейтингової активності
	Загальний чат і приватні повідомлення в реальному часі з підтримкою редагування, видалення, реакцій, закріплення та відповідей
	Реальні сповіщення про нові події, повідомлення, заявки, апеляції та нагороди
Робота з контентом	Пошук, фільтрація, сортування та перегляд детальної інформації про фільми і серіали
	Перегляд статистики, трейлерів, постерів, галерей кадрів, франшиз, схожого та рекомендованого контенту
	Інформація про акторів, персонажів та акторів озвучення

продовж. табл.2.6.

Функціональні вимоги	
	Приховування контенту за певними жанрами або акторами
	Оцінювання контенту, написання та голосування за рецензії
	Ведення списку перегляду, статусів перегляду, позначок «улюблене» та персональних добірок
	Автоматичний імпорт даних із TMDb та MyAnimeList для заповнення касту й пов'язаних сутностей
Адміністрування та модерація	Додавання, редагування та видалення фільмів, акторів і новин
	Керування рецензіями критиків, заявками рецензентів, ролями користувачів та нагородами
	Обробка скарг, блокування порушників, розгляд апеляцій і розблокування акаунтів
	Модерація форуму, добірок і контенту спільноти
	Підтримка користувацьких звернень через систему тікетів
Додаткові можливості	Форум для тематичних обговорень
	Рейтингові таблиці користувачів
	Тести/вікторини по фільмах із накопиченням досягнень
	Система новин, банерів та інформаційних повідомлень
	Платні підписки та платіжний сценарій
Нефункціональні вимоги	
Архітектура та технології	Backend реалізовано на ASP.NET Core як REST API
	Frontend побудовано на React

		Денисенко Д.О.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Сфремов Ю.М.				29
Змн.	Арк.	№ докум.	Підпис	Дата		

Нефункціональні вимоги	
	Інфраструктура підтримує Docker, Docker Compose, Nginx і автоматичне розгортання
	База даних: Microsoft SQL Server і Entity Framework Core
	Для реального часу використано SignalR і WebSockets
Продуктивність та надійність	API працює асинхронно та використовує пагінацію для великих наборів даних
	Під час запуску виконуються автоматичні міграції бази даних
	Для критичних операцій використовується централізоване збереження змін і журналювання помилок
	Передбачено базове обмеження частоти запитів для захисту авторизаційних сценаріїв
Безпека	Авторизація реалізована на основі JWT
	Паролі зберігаються у хешованому вигляді
	Підтримується вхід через Google
	Для реєстрації використовується захист через Google reCAPTCHA
	Налаштовано CORS, HTTPS, безпечні заголовки відповіді та обмеження доступу за ролями
Інтерфейс та API	Інтерфейс є україномовним і підтримує Unicode
	Обмін даними виконується у форматі JSON
	API побудовано за RESTful-принципами зі стандартними HTTP-статусами
	Для документації використовується Swagger

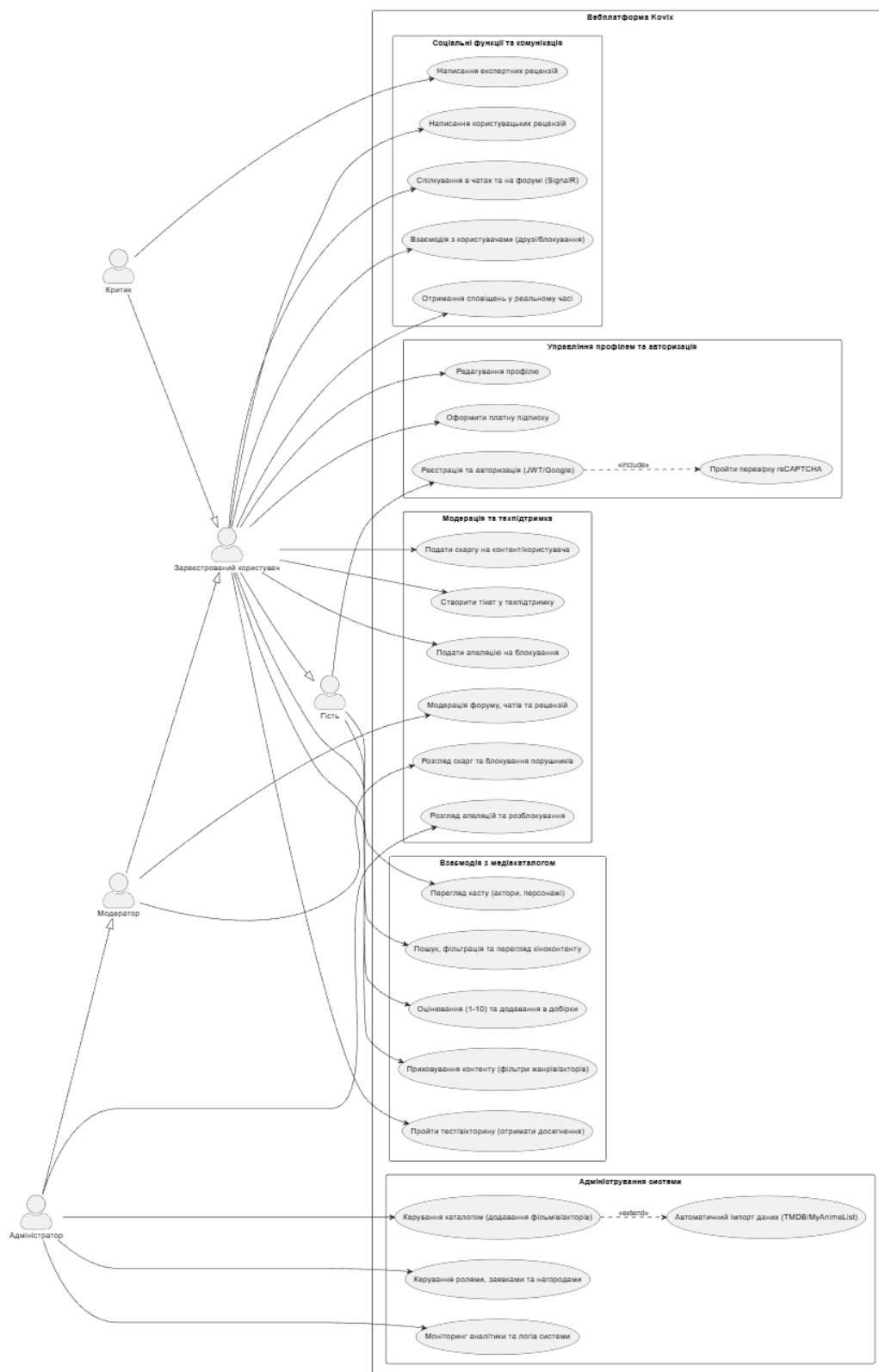


Рис. 2.1. Діаграма варіантів використання системи

Реєстрація та налаштування профілю. Новий користувач може створити акаунт через пошту або швидко увійти за допомогою Google [17]. Вхід у систему відкриває доступ до всіх функцій сайту. У власному кабінеті

користувач може змінювати нікнейм, аватар або пароль. Також користувач налаштовує фільтри контенту, щоб сховати певні жанри, також відображається особиста статистика, така як скільки кіно переглянуто та скільки оцінок поставлено, для більш детального ознайомлення можете переглянути детальні діаграми використання (див.рис.додаток.А) .

Користувач може шукати фільми чи серіали через звичайний пошуковий рядок або користуватися системою фільтрів. Знайдене кіно можна додавати до списку “Переглядаю” або позначати як “Улюблене”, також є можливість сортувати по власних добірках та тір-лістах.

Глядачі ставлять оцінки фільмам або окремим серіям, формуючи цим загальний рейтинг на сайті. Також можна писати розгорнуті текстові відгуки, створювати власні теми на форумі й спілкуватися з іншими кіно любителям у приватних або загальних чатах.

Функціонал для контролю за матеріалами, які створюють користувачі. Модератори перевіряють відгуки, коментарі, тір-листи та повідомлення на форумах, щоб забезпечити дотримання правил платформи. Вони також опрацьовують скарги від користувачів на неприйнятну поведінку інших учасників.

Комплект інструментів для наповнення та підтримки бази даних платформи. Адміністратор створює або редагує сторінки фільмів, акторів та новин, керує статистикою, а також може автоматизовано імпортувати дані з надійних зовнішніх джерел.

Надає адміністраторам повний контроль над акаунтами на платформі. Вони можуть змінювати ролі користувачів, відстежувати статистику їхньої активності, блокувати порушників та розглядати їхні зустрічні апеляції у разі незгоди з баном.

Наступним логічним кроком, після визначення ролей та сценаріїв використання, є побудова об'єктно-орієнтованої моделі платформи Kovix. Вона описується за допомогою діаграми класів, яка наочно демонструє, з яких

		Денисенко Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремюв Ю.М.				32
Змн.	Арк.	№ докум.	Підпис	Дата		

базових елементів складається система та як саме вони взаємодіють між собою.

База даних вебплатформи містить чотири основні блоки для інформації. Перший відповідає за користувачів та комунікацію у ньому забезпечується робота профілів, списків друзів, приватних і групових чатів та системи сповіщень. Другий блок формує каталог медіа й зберігає дані про фільм, серіал, новин та інших. Третя категорія описує акторський склад, включаючи сторінки акторів, персонажів та акторів дубляжу. Завершальним блоком призначено оцінювання та модерації, він містить текстові рецензії з оцінками, користувацькі списки перегляду та інструменти для подачі скарг на порушення й апеляцій у разі якщо у користувача було заблоковано обліковий запис [9].

Усі ці дані пов'язані між собою трьома способами. Композиція або жорсткий зв'язок означає, що залежна частина не може існувати без цілого. Тому при видаленні фільму з бази автоматично зникнуть усі його серії та увесь його вміст. Агрегація чи м'який зв'язок об'єднує незалежні об'єкти, якщо видалити актора, сам фільм не зникне. Так само видалення фільму ніяк не вплине на користувацькі списки перегляду. Останній тип зв'язку це асоціація, яка вказує на авторство. Вона визначає чи користувач є творцем повідомлень, власником списків, автором відгуків.

Усі ролі та статуси на сайті суттєво відрізняються. Користувачі діляться на чотири групи, гості, звичайні глядачі, модератори та адміни. Кожне відео в каталозі має мітку, чи це фільм чи серіал. У своїх списках користувачі можуть міняти статус стрічки на “Переглядаю”, “Заплановано”, “Переглянуто” або “Покинуто”. Списки та тір-лісти можна зробити приватними, відкритими для всіх або показувати лише за посиланням відповідним користувачам. Для заявок у друзі діють статуси “Очікує підтвердження”, “Прийнято” та “Заблоковано”. Схожий принцип працює для скарг з апеляціями, вони можуть бути в очікуванні на розгляд, схваленням або відхиленням.

		Денисенко Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		33

Функціонал вебплатформи розширено за допомогою кількох важливих модулів. Таких як онлайн чат на базі технології SignalR яка дозволяє миттєво обмінюватися повідомленнями, ставити реакції або закріплювати важливі повідомлення. Повноцінний форум розбитий на категорії і теми для детального обговорення.

Інструмент тір-лістів допомагає користувачам створювати візуальні таблиці лідерства для контенту, також модератори перевіряють на відповідність правилам. Фінансовий модуль відповідає за створення рахунків та підтвердження оплати для платних підписок. Увесь цей комплекс закриває зручна панель адміністрування, де адміністратори оперативно переглядають звіти про порушення, видають користувачам нагороди та керують всім текстовим контентом сайту.

Така структура бази даних забезпечує гнучкість системи. Вона дозволяє легко масштабувати вебплатформу для прикладу було без проблем інтегровано модуль для розширення тір-лістів, форумів, не порушуючи базову логіку роботи з контентом.

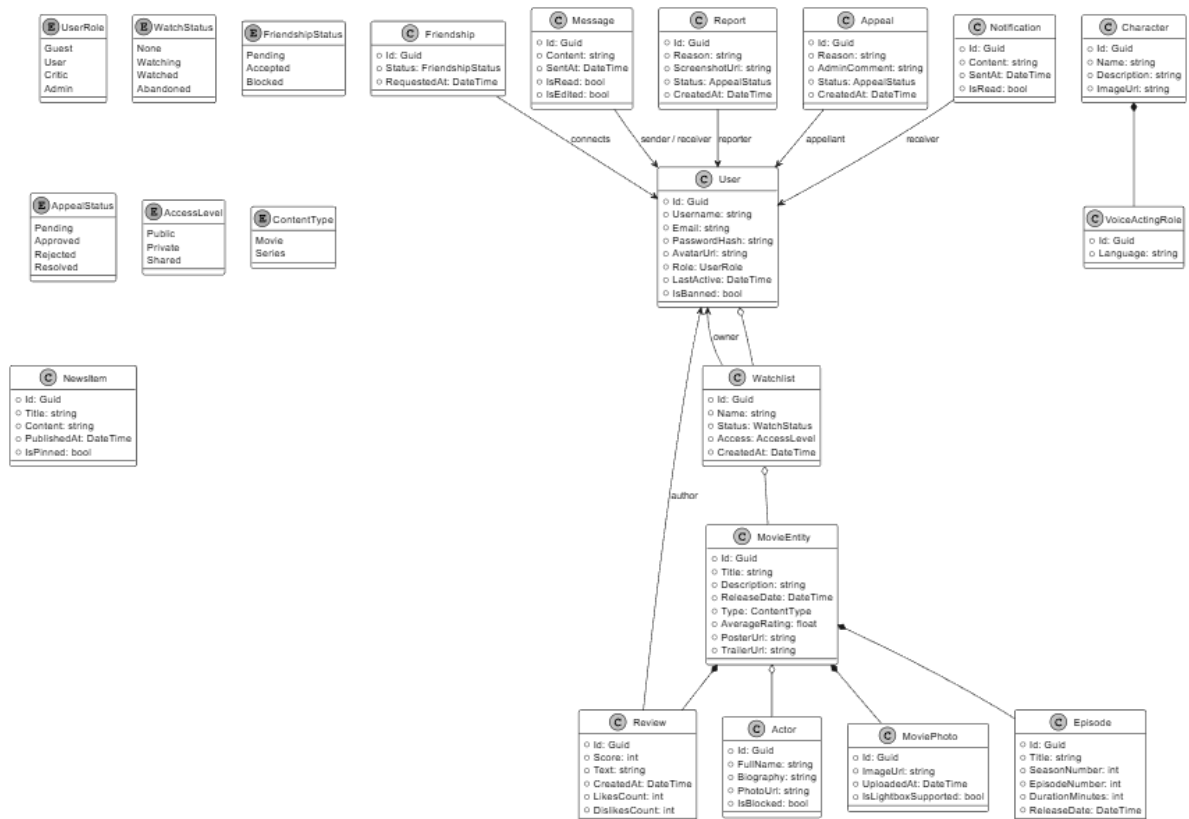


Рис. 2.2. Діаграма класів доменних моделей

Для детального відображення структури даних та архітектури розроблюваної системи було побудовано об'єктно-орієнтовану модель. На рис. 2.2 наведено повну діаграму класів доменних моделей платформи Kovix. Вона містить вичерпний перелік усіх ключових сутностей, їхні логічні поля, типи даних, перелічення, а також деталізовані зв'язки між ними. З метою спрощення візуального сприйняття загальної концепції взаємодії базових об'єктів, скорочена версія цієї діаграми (без деталізації полів класів) винесена до додатку (див. рис.додаток.А).

2.2. Розробка бази даних системи

З метою забезпечення надійного збереження, цілісності та ефективної обробки інформації в межах вебплатформи “Kovix”, було спроектовано структуру реляційної бази даних. Архітектура БД логічно відображає всі ключові процеси системи та базується на виділених раніше доменних моделях.

Особливу увагу при проектуванні структури приділено налаштуванням зв'язкам між таблицями за допомогою первинних і зовнішніх ключів. Щоб уникнути появи чорнових записів у системі, налаштовано правила каскадного видалення для композиційних залежностей між сутністю фільму та його епізодами чи фотоматеріалами. Водночас, для слабких посилань таких як агрегації чи зв'язку актора з фільмом, застосовується безпечне обмеження, яке забезпечує збереження історичних даних у каталозі [10].

База даних забезпечує централізоване зберігання облікових записів користувачів, розширеного каталогу контенту, а також матеріалів, згенерованих аудиторією (рецензій, оцінок, повідомлень). Крім того, структура БД містить необхідні відношення для підтримки механізмів соціальної взаємодії, процесів безпечної автентифікації, розмежування прав доступу на основі рольової моделі та функціонування комплексної підсистеми модерації контенту. На рисунку нижче представлена повна структура розробленої бази даних. Оскільки схема містить значний масив даних та відзначається високою деталізацією зв'язків, для більш зручного та наочного

		Денисенко Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		35

розуміння загальної архітектури проєкту її повну версію наведено в додатку (див.рис.додаток.А).

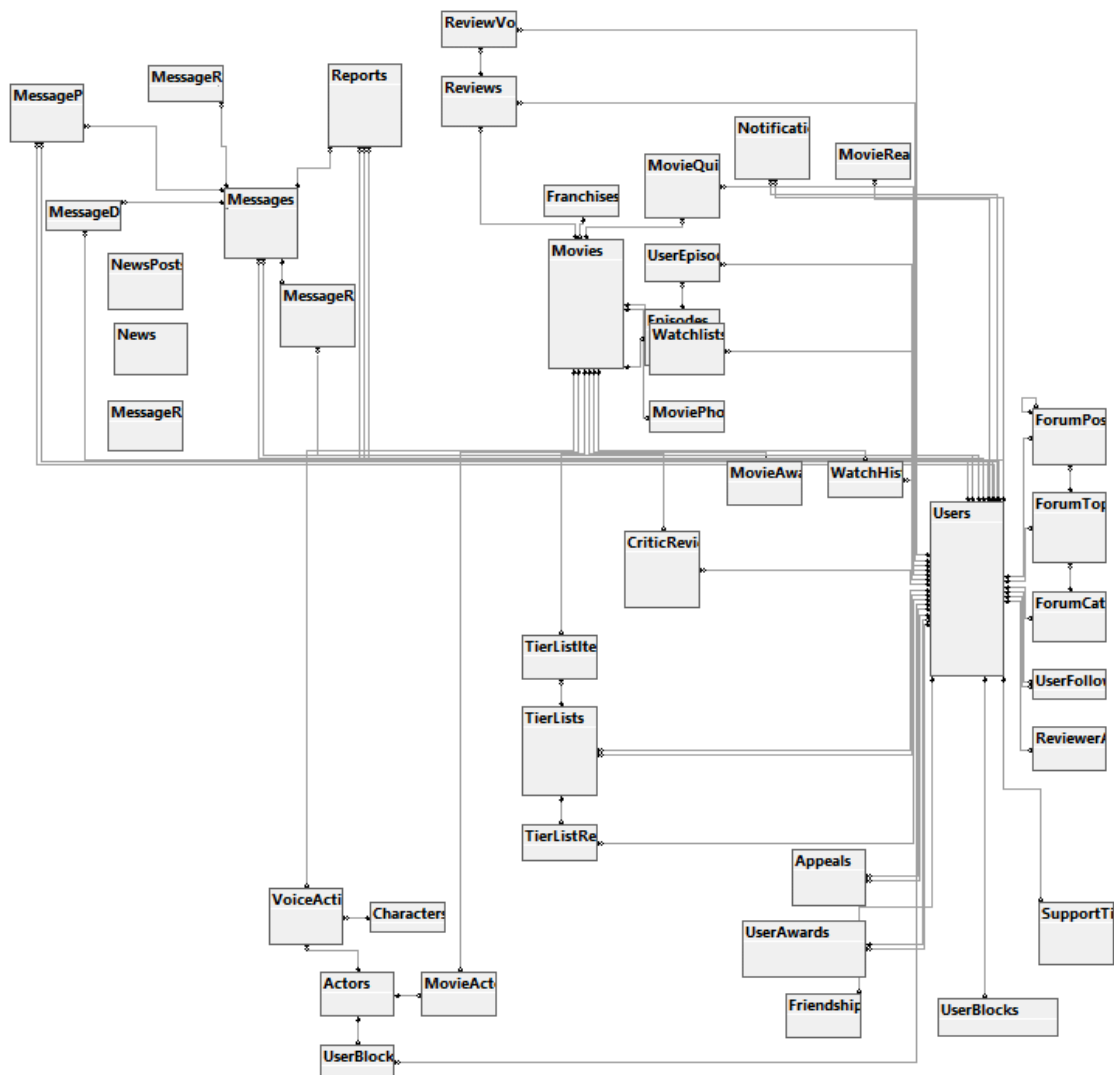


Рис. 2.3. Структура таблиць бази даних

Основною таблицею в базі даних є Users. Він зберігає облікові дані для входу, паролі та всі основні дані профілю. У ньому є поля для перевірки статусу користувача, наприклад IsOnline, LastActive, IsBlocked, AvatarUrl і Role для перевірки ролі користувача. Тут також додані відмітки про преміум-підписку, підтвердження пошти або особливості налаштування. Зберігання всього в одній таблиці дозволяє сайту швидко перевіряти користувача без постійного об'єднання з іншими таблицями. Доступи реалізовані через поле Роль, а взаємодія між людьми - через додаткові таблиці підписок, друзів, чорного списку, повідомлення, сповіщення, а також скарги з апеляціями. Усі

ці таблиці пов'язані з юзерами через зовнішні ключі, тому система завжди чітко знає, хто саме зробив ту чи іншу дію, також можете переглянути діаграму використання патернів (див.рис.додаток.А).

Увесь контент зосереджено навколо таблиці MovieEntity. У таблиці є просте логічне поле IsSeries, тому при використанні її не потрібно створювати окремі великі таблиці для фільмів та серіалів. Якщо це серіал, інформація про його епізоди береться з таблиці Episode. Таблиці для збереження кадрів та постерів, акторського складу, акторів дубляжу, франшиз, професійних рецензій, нагород та історії переглядів користувачами працюють паралельно.

Для оглядів, голосування за відповідний фільм чи серіал, використовуються таблиці реакцій та оцінок оглядів, оцінок та відповідей. Персональні списки та оцінки працюють через таблиці списків спостереження та реакцій на них. Звичайний зв'язок забезпечується модулем обміну повідомленнями, де можна реагувати, відповідати, видаляти або закріплювати текст. Все це прив'язано до користувачів та фільмів, що дозволяють системі збирати точну статистику для майбутніх рекомендацій [12].

Окремий блок таблиць з скаргами, апеляціями на блокування, заявками на статус критика та заявками підтримки для адміністрування та модерації. Тут також розташовані нагороди сайту, новини та вся структура форуму з його категоріями, темами та публікаціями. Усі модулі бази даних добре пов'язані один з одним за допомогою зовнішніх ключів з налаштовуваним каскадним видаленням, тому легко додавати нові функції до бази даних у майбутньому, не порушуючи стару логіку.

Таблиця 2.7

Опис таблиці User

Назва	Тип даних	PK	FK	Опис
Id	INT	+		Унікальний ідентифікатор користувача.
Username	NVARCHAR			Нікнейм користувача для відображення на сайті.

продовж. табл.2.7

Email	NVARCHAR			Електронна пошта для входу та сповіщень.
PasswordHash	NVARCHAR			Хеш пароля.
Role	INT (Enum)			Роль у системі.
IsBlocked	BIT			Прапорець, що вказує, чи заблоковано акаунт.
CreatedAt	DATETIME2			Дата створення акаунту.

Таблиця 2.8

Опис таблиці MovieEntity

Назва	Тип даних	PK	FK	Опис
Id	INT	+		Унікальний ідентифікатор контенту.
Title	NVARCHAR			Назва фільму або серіалу.
Description	NVARCHAR			Короткий опис сюжету.
Year	INT			Рік випуску.
IsSeries	BIT			Прапорець типу контенту.
AverageRating	FLOAT			Кешоване значення середнього рейтингу.
FranchiseId	INT		+	Посилання на франшизу.

Таблиця 2.9

Опис таблиці Episode

Назва	Тип даних	PK	FK	Опис
Id	INT	+		Ідентифікатор контенту.
MovieId	INT		+	Ідентифікатор серіалу.
SeasonNumber	INT			Порядковий номер сезону.
EpisodeNumber	INT			Порядковий номер епізоду в межах сезону.

продовж. табл.2.9

Назва	Тип даних	PK	FK	Опис
Title	NVARCHAR			Назва конкретного епізоду.
AverageRating	FLOAT			Середня оцінка епізоду від користувачів.

Таблиця 2.10

Опис таблиці Review

Назва	Тип даних	PK	FK	Опис
Id	INT	+		Унікальний ідентифікатор рецензії.
UserId	INT		+	Ідентифікатор автора рецензії.
MovietId	INT		+	Ідентифікатор оціненого контенту.
Rating	INT			Числова оцінка фільму/серіалу.
Comment	NVARCHAR			Текстовий зміст розгорнутої рецензії.
createdAt	DATETIME2			Дата та час публікації відгуку.

Таблиця 2.11

Опис таблиці Watchlist

Назва	Тип даних	PK	FK	Опис
Id	INT	+		Унікальний ідентифікатор оцінки.
UserId	INT		+	Ідентифікатор власника списку.
Name	NVARCHAR			Назва персональної добірки контенту.
AccessLevel	INT			Рівень доступу до списку.
IsFavorite	BIT			Прапорець, що позначає список улюбленого контенту.
AddedAt	DATETIME2			Дата та час створення списку.

Таблиця 2.12

Опис таблиці Collection

Назва	Тип даних	PK	FK	Опис
Id	INT	+		Унікальний ідентифікатор колекції.
Name	NVARCHAR			Повне ім'я та прізвище особи.
Bio	NVARCHAR			Текстова біографія персони.
BirthDate	DATETIME2			Дата народження.
PhotoUrl	NVARCHAR			URL-посилання на фотографію актора.

Таблиця 2.13

Опис таблиці Message

Назва	Тип даних	PK	FK	Опис
Id	INT	+		Унікальний ідентифікатор жанру.
SenderId	INT		+	ID відправника.
ReceiverId	INT		+	ID отримувача.
Content	NVACHAR			Текст надісланого повідомлення.
IsRead	BIT			Статус прочитання повідомлення отримувачем.
Timestamp	DATETIME2			Точний час відправлення.

Таблиця 2.14

Опис таблиці Report

Назва	Тип даних	PK	FK	Опис
Id	INT	+		Унікальний ідентифікатор персони.
SenderId	INT		+	ID ініціатора скарги.
ReportedUserId	INT		+	ID користувача-порушника.
Reason	NVARCHAR			Текстова причина подання скарги.
Resolution	INT			Статус вирішення проблеми.

		Денисенко Д.О.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Сфремов Ю.М.				40
Змн.	Арк.	№ докум.	Підпис	Дата		

Таблиця 2.15

Опис таблиці ReviewVote

Назва поля	Тип даних	PK	FK	Опис
Id	INT	+		Унікальний ідентифікатор голосу.
ReviewId	INT		+	Посилання на таблицю рецензій.
UserId	INT		+	Посилання на таблицю користувачів.
IsUpvote	BIT			Тип голосу: лайк (true) або дизлайк (false).
CreatedAt	DATETIME2			Дата та час голосування.

Таблиця 2.16

Опис таблиці MovieReaction

Назва поля	Тип даних	PK	FK	Опис
Id	INT	+		Унікальний ідентифікатор реакції.
MovieId	INT		+	Посилання на каталог контенту.
UserId	INT		+	Посилання на таблицю користувачів.
Type	INT (Enum)			Тип реакції (Like, Love, Funny тощо).
CreatedAt	DATETIME2			Дата та час залишення реакції.

Таблиця 2.17

Опис таблиці MoviePhoto

Назва поля	Тип даних	PK	FK	Опис
Id	INT	+		Унікальний ідентифікатор фото.
MovieId	INT		+	Посилання на медіафайл.
Url	NVARCHAR			Вебпосилання на завантажене зображення.
UploadedBy	INT		+	Авторизований користувач, що завантажив фото.

продовж. табл.2.17

Назва поля	Тип даних	PK	FK	Опис
CreatedAt	DATETIME2			Дата та час завантаження.
IsPrimary	BIT			Прапорець головного постера або обкладинки.

Таблиця 2.18

Опис таблиці MovieActor

Назва поля	Тип даних	PK	FK	Опис
MovieId	INT	+	+	Посилання на кіноконент.
ActorId	INT	+	+	Посилання на картку актора.
CharacterName	NVARCHAR			Сценічне ім'я персонажа або роль у фільмі.
Order	INT			Порядок відображення актора в титрах.

Таблиця 2.19

Опис таблиці VoiceActingRole

Назва поля	Тип даних	PK	FK	Опис
Id	INT	+		Унікальний ідентифікатор ролі озвучення.
MovieId	INT		+	Посилання на медіапроект.
CharacterId	INT		+	Посилання на конкретного персонажа.
ActorId	INT		+	Посилання на актора дубляжу.
Language	NVARCHAR			Мова виконання дубляжу або озвучки.
IsOriginal	BIT			Ознака оригінальної доріжки озвучення

Таблиця 2.20

Опис таблиці UserFollow

Назва поля	Тип даних	PK	FK	Опис
ObserverId	INT	+	+	Ініціатор підписки.
TargetId	INT	+	+	Користувач, на якого підписалися.
CreatedAt	DATETIME2			Дата та час оформлення підписки.

Таблиця 2.21

Опис таблиці Friendship

Назва поля	Тип даних	PK	FK	Опис
Id	INT	+		Унікальний ідентифікатор запиту дружби.
RequesterId	INT		+	Користувач, що надіслав запит.
ReceiverId	INT		+	Користувач, який отримав запит.
Status	INT (Enum)			Поточний статус (Pending, Accepted, Blocked).
CreatedAt	DATETIME2			Дата створення запиту.

Таблиця 2.22

Опис таблиці UserBlock

Назва поля	Тип даних	PK	FK	Опис
Id	INT	+		Унікальний ідентифікатор блокування.
BlockerId	INT		+	Користувач, який додає до чорного списку.
BlockedId	INT		+	Користувач, якого заблокували.
CreatedAt	DATETIME2			Дата та час внесення у чорний список.
Reason	NVARCHAR			Причина блокування користувачем.

Таблиця 2.23

Опис таблиці Notification

Назва поля	Тип даних	PK	FK	Опис
Id	INT	+		Унікальний ідентифікатор сповіщення.
UserId	INT		+	Користувач, який отримує сповіщення.

продовж. табл.2.23

Назва поля	Тип даних	PK	FK	Опис
SenderId	INT		+	Опціональне посилання на ініціатора події.
Message	NVARCHAR			Текстовий вміст системного сповіщення.
Url	NVARCHAR			Вебпосилання для переходу при кліку.
IsRead	BIT			Статус прочитання (true/false).
CreatedAt	DATETIME2			Дата та час генерації сповіщення.

Таблиця 2.24

Опис таблиці MessageReaction

Назва поля	Тип даних	PK	FK	Опис
Id	INT	+		Унікальний ідентифікатор реакції.
MessageId	INT		+	Посилання на повідомлення в чаті.
UserId	INT		+	Авторизований користувач, що залишив емодзі.
ReactionEmoji	NVARCHAR			Юнікод-значення або текстовий код емодзі.
CreatedAt	DATETIME2			Дата та час реакції.

Таблиця 2.25

Опис таблиці MessagePin

Назва поля	Тип даних	PK	FK	Опис
Id	INT	+		Унікальний ідентифікатор закріплення.
MessageId	INT		+	Повідомлення, що підлягає закріпленню.

продовж. табл.2.25

Назва поля	Тип даних	PK	FK	Опис
PinnedBy	INT		+	Модератор або учасник чату, що зробив пін.
PinForEveryone	BIT			Ознака закріплення повідомлення для всіх учасників.
CreatedAt	DATETIME2			Дата та час закріплення.

Таблиця 2.26

Опис таблиці MessageDelete

Назва поля	Тип даних	PK	FK	Опис
Id	INT	+		Унікальний ідентифікатор логу видалення.
MessageId	INT		+	Видалене повідомлення.
UserId	INT		+	Користувач, для інтерфейсу якого приховано чат.
DeletedAt	DATETIME2			Дата та час видалення з боку користувача.

Таблиця 2.27

Опис таблиці MessageReply

Назва поля	Тип даних	PK	FK	Опис
Id	INT	+		Унікальний ідентифікатор відповіді.
MessageId	INT		+	Посилання на вихідне повідомлення.
ReplyMessageId	INT		+	Посилання на нове повідомлення-відповідь.
CreatedAt	DATETIME2			Дата та час зв'язування повідомлень.

Таблиця 2.28

Опис таблиці MessageReadStatus

Назва поля	Тип даних	PK	FK	Опис
Id	INT	+		Унікальний ідентифікатор статусу.
MessageId	INT		+	Прочитане повідомлення.
UserId	INT		+	Користувач, що відкрив діалог.
ReadAt	DATETIME2			Дата та час фіксації прочитання.

Таблиця 2.29

Опис таблиці WatchHistoryItem

Назва поля	Тип даних	PK	FK	Опис
Id	INT	+		Унікальний ідентифікатор логу перегляду.
UserId	INT		+	Глядач.
MoviedId	INT		+	Переглянутий фільм/серіал.
ViewedAt	DATETIME2			Дата останньої активності перегляду.
Progress	DECIMAL			Відсоток або хронометраж пройденного плеєра.

Таблиця 2.30

Опис таблиці UserEpisodeRating

Назва поля	Тип даних	PK	FK	Опис
Id	INT	+		Унікальний ідентифікатор оцінки серії.
UserId	INT		+	Користувач, який виставив оцінку.
EpisodeId	INT		+	Конкретна серія сезону.
Rating	INT			Числова оцінка користувача.
CreatedAt	DATETIME2			Дата та час оцінювання серії.

Таблиця 2.31

Опис таблиці TierList

Назва поля	Тип даних	PK	FK	Опис
Id	INT	+		Унікальний ідентифікатор тір-лісту.
UserId	INT		+	Власник та автор списку.
Title	NVARCHAR			Публічний заголовок списку.
Description	NVARCHAR			Додатковий опис та правила добірки.
IsPublic	BIT			Рівень приватності (public/private).
Status	INT (Enum)			Статус перевірки модератором.
CreatedAt	DATETIME2			Дата створення рейтингової таблиці.

Таблиця 2.32

Опис таблиці TierListItem

Назва поля	Тип даних	PK	FK	Опис
Id	INT	+		Унікальний ідентифікатор елемента.
TierListId	INT		+	Посилання на батьківський список.
MovieId	INT		+	Об'єкт кінокаталогу всередині списку.
Tier	NVARCHAR			Буквена назва рівня ранжування.
Position	INT			Порядковий номер елемента всередині рядка.

Таблиця 2.33

Опис таблиці TierListReaction

Назва поля	Тип даних	PK	FK	Опис
Id	INT	+		Унікальний ідентифікатор реакції.
TierListId	INT		+	Посилання на оцінений тір-ліст.
UserId	INT		+	Користувач, що залишив фідбек.
ReactionType	NVARCHAR			Строковий ідентифікатор типу залишеної реакції.
CreatedAt	DATETIME2			Дата та час створення реакції.

		Денисенко Д.О.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Сфремов Ю.М.				47
Змн.	Арк.	№ докум.	Підпис	Дата		

Таблиця 2.34

Опис таблиці ForumCategory

Назва поля	Тип даних	PK	FK	Опис
Id	INT	+		Унікальний ідентифікатор категорії.
Title	NVARCHAR			Назва гілки категорій форуму.
Description	NVARCHAR			Тематичний опис розділу.
ProposedByUserId	INT		+	Користувач, який ініціював створення.
IsApproved	BIT			Ознака схвалення категорії адміністрацією.

Таблиця 2.35

Опис таблиці ForumTopic

Назва поля	Тип даних	PK	FK	Опис
Id	INT	+		Унікальний ідентифікатор теми обговорення.
CategoryId	INT		+	Родовий розділ форуму.
UserId	INT		+	Автор нової гілки обговорення.
Title	NVARCHAR			Текстова назва теми.
CreatedAt	DATETIME2			Дата та час відкриття теми.

Таблиця 2.36

Опис таблиці ForumPost

Назва поля	Тип даних	PK	FK	Опис
Id	INT	+		Унікальний ідентифікатор повідомлення.
TopicId	INT		+	Тема, всередині якої розміщено пост.
UserId	INT		+	Користувач-автор тексту.
ParentPostId	INT		+	Посилання на вихідне повідомлення при відповіді.

продовж. табл.2.36

Назва поля	Тип даних	PK	FK	Опис
Content	NVARCHAR			Текстове наповнення публікації на форумі.
CreatedAt	DATETIME2			Дата та час відправлення.

Таблиця 2.37

Опис таблиці CriticReview

Назва поля	Тип даних	PK	FK	Опис
Id	INT	+		Унікальний ідентифікатор рецензії критиків.
UserId	INT		+	Профіль сертифікованого критика.
MovieId	INT		+	Медіаконтент, що оцінюється.
Rating	INT			Професійна бальна оцінка.
Content	NVARCHAR			Текст розгорнутої рецензії.
CreatedAt	DATETIME2			Дата та час оприлюднення відгуку.

Таблиця 2.38

Опис таблиці ReviewerApplication

Назва поля	Тип даних	PK	FK	Опис
Id	INT	+		Унікальний ідентифікатор заявки.
UserId	INT		+	Кандидат на отримання ролі критика.
MotivationText	NVARCHAR			Портфоліо та супровідний опис кандидата.
Status	INT (Enum)			Поточний статус заявки.
CreatedAt	DATETIME2			Дата та час надсилання заявки.

Таблиця 2.39

Опис таблиці SupportTicket

Назва поля	Тип даних	PK	FK	Опис
Id	INT	+		Унікальний ідентифікатор тікета.
UserId	INT		+	Користувач, який надіслав запит у підтримку.
Subject	NVARCHAR			Короткий заголовок проблеми.
Message	NVARCHAR			Розгорнутий опис технічної проблеми або питання.
IsResolved	BIT			Ознака остаточного вирішення тікета.
AdminReply	NVARCHAR			Офіційна відповідь від служби підтримки.
CreatedAt	DATETIME2			Дата відкриття звернення.

Таблиця 2.40

Опис таблиці UserAward

Назва поля	Тип даних	PK	FK	Опис
Id	INT	+		Унікальний ідентифікатор виданої нагороди.
UserId	INT		+	Користувач, який отримав досягнення.
Name	NVARCHAR			Офіційна назва отриманої нагороди.
Icon	NVARCHAR			Посилання на графічну іконку досягнення.
IssuedAt	DATETIME2			Дата та час автоматичного чи ручного вручення.

Таблиця 2.41

Опис таблиці MovieAward

Назва поля	Тип даних	PK	FK	Опис
Id	INT	+		Унікальний ідентифікатор нагороди проекту.
MovieId	INT		+	Фільм чи серіал, що отримав нагороду.

		Денисенко Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремов Ю.М.				50
Змн.	Арк.	№ докум.	Підпис	Дата		

продовж. табл.2.41

Назва поля	Тип даних	PK	FK	Опис
Name	NVARCHAR			Назва кінопремії або відзнаки.
IssuedAt	DATETIME2			Рік та дата нагородження кінострічки.

Таблиця 2.42

Опис таблиці NewsPost

Назва поля	Тип даних	PK	FK	Опис
Id	INT	+		Унікальний ідентифікатор новини.
Title	NVARCHAR			Публічний заголовок новини.
ShortDescription	NVARCHAR			Текст для картки прев'ю на головній сторінці.
Content	NVARCHAR			Розгорнуте текстове тіло новини.
PublishedAt	DATETIME2			Дата та час публікації на сайті.
AuthorId	INT		+	Посилання на аккаунт автора або редактора.

2.3. Проектування та реалізація алгоритмів роботи системи

Діаграма активності на рис.2.4 демонструє, алгоритм роботи системи для пошуку та додавання відгуку. Процес починається з ініціалізації пошукового запиту. Система фільтрує дані та показує результат. Момент прийняття рішення (розгалуження) є важливим. Усі користувачі мають інформацію про відгуки, але для оцінки та написання відгуку потрібна перевірка прав доступу. Система перенаправляє користувача до скрипта входу/реєстрації, якщо користувач не авторизований і лише після успішної авторизації дозволяє зберегти відгук у базі даних, нижче описано пояснення більш детально [11].

Процес починається з головної сторінки, яку відкриває відвідувач вебплатформи. Потім він вводить необхідні критерії пошуку, назву, рік

		Денисенко Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		51

випуску або жанр фільму, після чого система робить запит до бази даних та відображає на екрані список знайдених фільмів чи серіалів. Користувач вибирає певний елемент і система завантажує його сторінку з детальним описом, поточним рейтингом та вже існуючими відгуками інших людей. Якщо користувач вирішує натиснути кнопку, щоб написати власний відгук, система переходить до перевірки його статусу. У цей момент алгоритм розгалужується. Якщо відвідувач не авторизований у своєму обліковому записі, сайт показує йому форму для входу/реєстрації і лише після успішного входу повертається назад. Якщо користувач вже авторизований, він одразу переходить до наступного кроку. На екрані відкривається форма, де можна поставити оцінку від одного до десяти зірок та залишити текст відгуку. Користувач заповнює ці поля і натискає кнопку збереження.

Перед записом в базу даних система перевіряє отриману інформацію на наявність порожніх полів або спаму, а потім зберігає відгук в базі. автоматично перераховує новий середній рейтинг цього фільму та оновлює сторінку. Далі алгоритм дій виконано успішно.

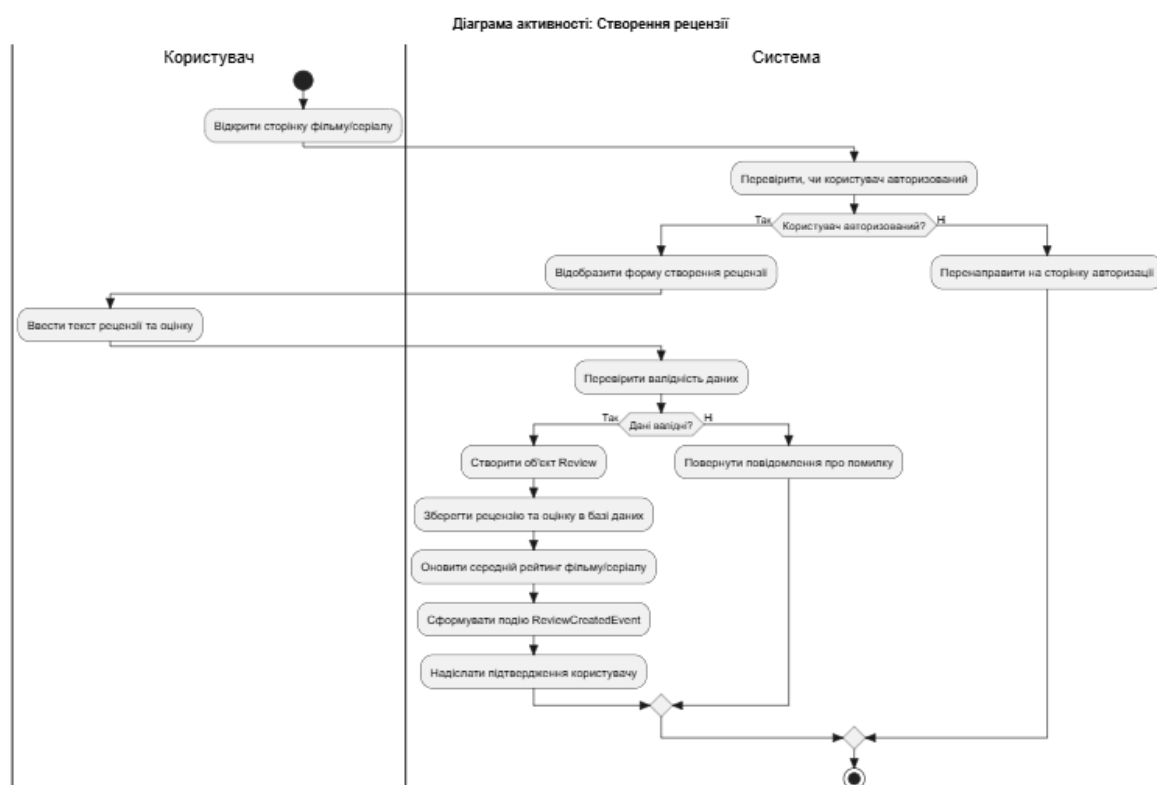


Рис. 2.4. Діаграма активності для створення рецензій

На рис.2.5 показано діаграму активності того, як користувач взаємодіє з особистими колекціями. Алгоритм роботи з колекціями починається в профілі авторизованого користувача. Система пропонує створити новий набір або відкрити існуючий. Користувач вводить назву й опис нової колекції та рівні, а сервер створює запис у таблиці TierList із першими елементами в TierListItem. За умовчанням нова колекція набуває статусу приватної і маркується як така, що очікує на перевірку. Якщо користувач відкриває вже наявну добірку, система надсилає запит GET для завантаження списку елементів з їх метаданими. На цьому етапі права чітко розподілені, зовнішні відвідувачі бачать лише останні затверджені публічні списки, приватизовані або ще не перевірені добірки доступні лише власнику, модераторам чи адміністраторам вебплатформи.

Після успішного завантаження інтерфейсу користувача можна виконати декілька незалежних операцій над елементами добірки, які одночасно можна надсилати на сервер, наприклад, додавання нового фільму через окрему кінцеву точку, повне видалення вмісту картки або оновлення параметрів елемента, де можливо змінити сам фільм, його рівень у системі розташування або порядкову позицію. Усі ці операції виконуються паралельно і обробляються сервером незалежно від однієї.

Сервер повинен перевірити авторизацію користувача та порівняти його ідентифікатор з власником колекції перед будь-яким збереженням, видаленням або оновленням. Сервер знову блокує спробу редагування чужого списку, якщо немає прав адміністратора або модератора, з повертанням помилки доступу. Будь-які зміни, збережені в загальнодоступних зразках, автоматично повертаються до стадії розгляду, що очікує на розгляд і вимагатимуть повторного затвердження адміністрацією сайту. За таким же принципом працює команда розповсюдження списку, яка робить колекцію загальнодоступною та надсилає її на модерацію. Окремо користувач може одночасно взаємодіяти із загальним модулем списку спостереження, щоб

		Денисенко Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		53

```

graph TD
    subgraph UI [Користувач UI]
        Start(( )) --> Login[Перехід у розділ «Користувачі»]
        Login --> CheckAuth{Статус користувача}
        CheckAuth --> Logout[Вихід з системи, повернення на головну]
        CheckAuth --> GetPlaylist[Завантаження GET /api/playlist/{id}]
        Logout --> EndUI(( ))
    end

    subgraph Backend [Сервер Backend API]
        GetPlaylist --> GetPlaylistData[Встановлення {id}/public = false, Status = Pending]
        GetPlaylistData --> GetPlaylistItems[Завантаження списку з /api/playlist/{id}]
        GetPlaylistItems --> GetPlaylistDetails[Завантаження деталей списку]
        GetPlaylistDetails --> CheckUserAdmin{Користувач == Власник або Admin?}
        CheckUserAdmin -- Yes --> UpdatePlaylist[Завантаження playlist з параметрами]
        CheckUserAdmin -- No --> AddPlaylistItem[Додавання елементу до playlist]
        AddPlaylistItem --> CheckItemExists{Елемент існує?}
        CheckItemExists -- Yes --> UpdateItem[Оновлення елементу]
        CheckItemExists -- No --> AddNewItem[Додавання нового елементу]
        AddNewItem --> CheckNewItemExists{Елемент існує?}
        CheckNewItemExists -- Yes --> UpdateNewItem[Оновлення елементу]
        CheckNewItemExists -- No --> AddNewItem
        AddNewItem --> EndBackend(( ))
    end

    GetPlaylistData --> UI
    GetPlaylistItems --> UI
    GetPlaylistDetails --> UI
    UpdatePlaylist --> UI
    AddPlaylistItem --> UI
    UpdateItem --> UI
    AddNewItem --> UI
    UpdateNewItem --> UI
    EndBackend --> EndUI
  
```

Процес додавання медіаконтенту з двома треками: Користувач та Адміністратор показано на рис. 2.6. На даний момент немає механізму для звичайного користувача додавання запису безпосередньо до медіакаталогу, створення запису в таблиці MovieEntity виконується лише адміністратором через захищену кінцеву точку. Користувачі не можуть безпосередньо додавати нові фільми чи серіали. У разі запитів вони використовують служби підтримки або інші канали зв'язку, але офіційне додавання контенту відбувається лише після ручної перевірки та схвалення адміністрацією [13]. Під час створення адміністратор перевіряє унікальність, за назвою та роком, додає метадані, кешує автора постера та створює пов'язані записи такі як акторський склад, франшиза тощо. Якщо виявляються проблеми, запит відхиляється, а користувачеві повідомляється про причину.

Користувач звертається до служби підтримки, але фактичне додавання медіаконтенту виконується адміністратором, лише після перевірки дублювання запис з'являється в каталозі.

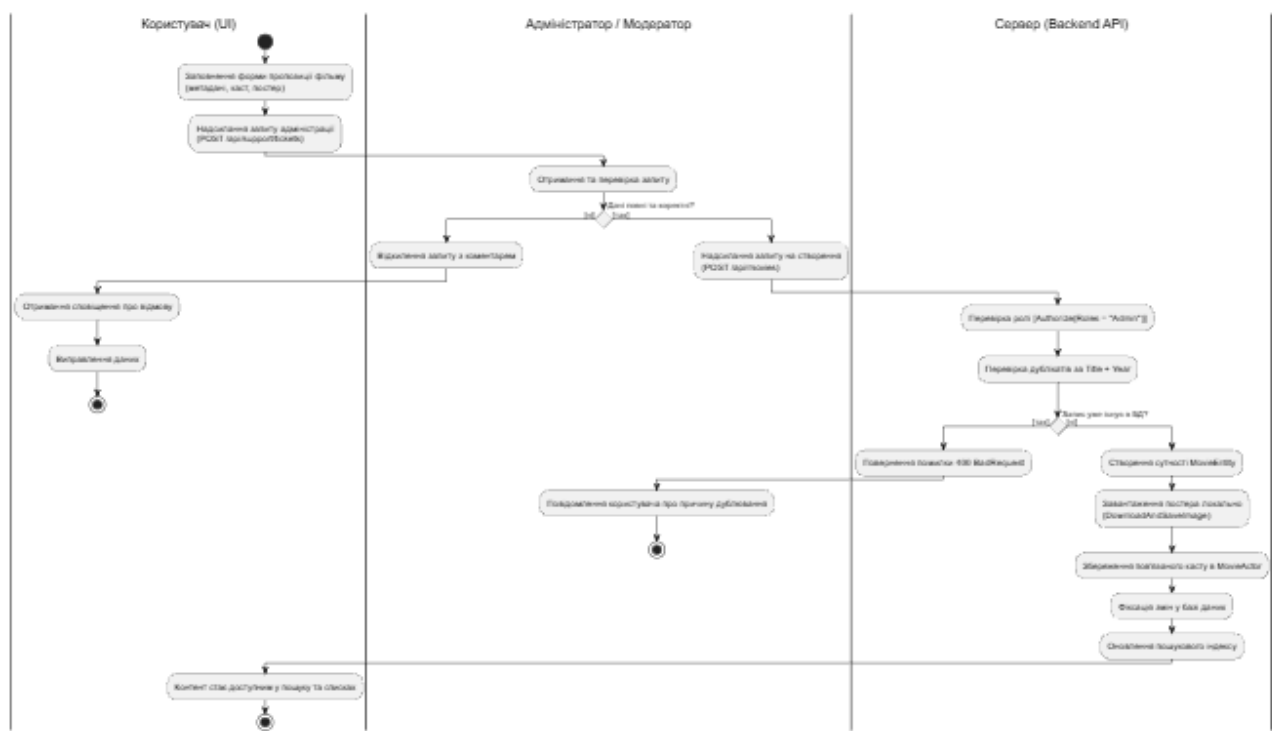


Рис. 2.6. Діаграма активності для медіаконтенту

На рис. 2.7 показана діаграма послідовності, яка детально описує взаємодію користувачів та критиків з каталогом контенту, при пошуку та оцінюванні фільмів. Все починається з того, що користувач надсилає запит на пошук або відкриває конкретну сторінку фільму в інтерфейсі. Клієнтська частина додатка звертається до веб-API, яке у свою чергу робить запит до бази даних, щоб отримати повну інформацію про обраний медіаконтент. Сервер відповідає метаданими фільму до клієнтського інтерфейсу, середньою оцінкою та всіма доступними відгуками. Після завантаження сторінки авторизований глядач може написати звичайний відгук, вказавши текстовий коментар та оцінку від одного до десяти. Згодом сервер перевіряє сеанс користувача, переконується у відсутності блокувань чи дублікатів відгуків з цього облікового запису на той самий фільм, а потім зберігає запис у таблиці відгуків. Процедура перерахунку середнього рейтингу стрічки запускається автоматично, статистика оновлюється в базі даних і на екрані з'являється

останній відгук. Користувачі можуть взаємодіяти з іншими відгуками, оцінюючи щоб визначити оцінку корисності відгуків.

Окремим захищеним ланцюжком у системі є професійні критичні відгуки. Розширені аналітичні матеріали можуть бути створені користувачем зі спеціальною роллю критика або адміністратора. На відміну від звичайного відгуку, тут інтерфейс надсилається на сервер у вигляді детальної оцінки за окремими критеріями, якістю сюжету, акторською грою, видимими ефектами та звуковим супроводом, а потім додається загальний текстовий вердикт. Внутрішній контролер перевіряє, чи має користувач необхідну роль у маркері авторизації і контролює, щоб один рецензент не міг залишити кілька відгуків для одного фільму. У разі успішної перевірки професійний відгук миттєво записується в базу без додаткових кроків ручної модерації та стає доступним у спеціальному блоці на сторінці фільму. У разі відсутності прав або спроби перерейтингу сервер миттєво відхиляє запит і повертає помилку доступу [24].

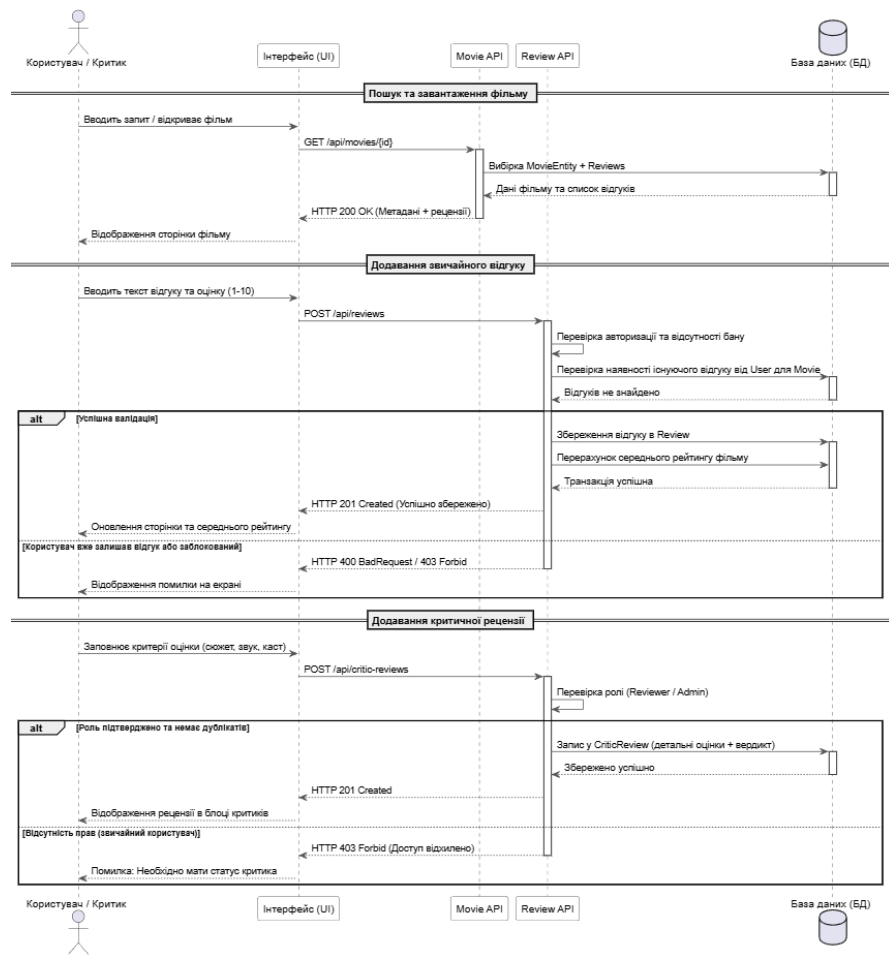


Рис. 2.7. Діаграма послідовності для додавання оцінки

Наступна діаграма - це діаграма комунікації (або кооперації), зображена на рис. 2.8. На відміну від діаграми послідовності, яка акцентує увагу на часі, діаграма комунікації підкреслює структурні зв'язки між об'єктами, що взаємодіють.

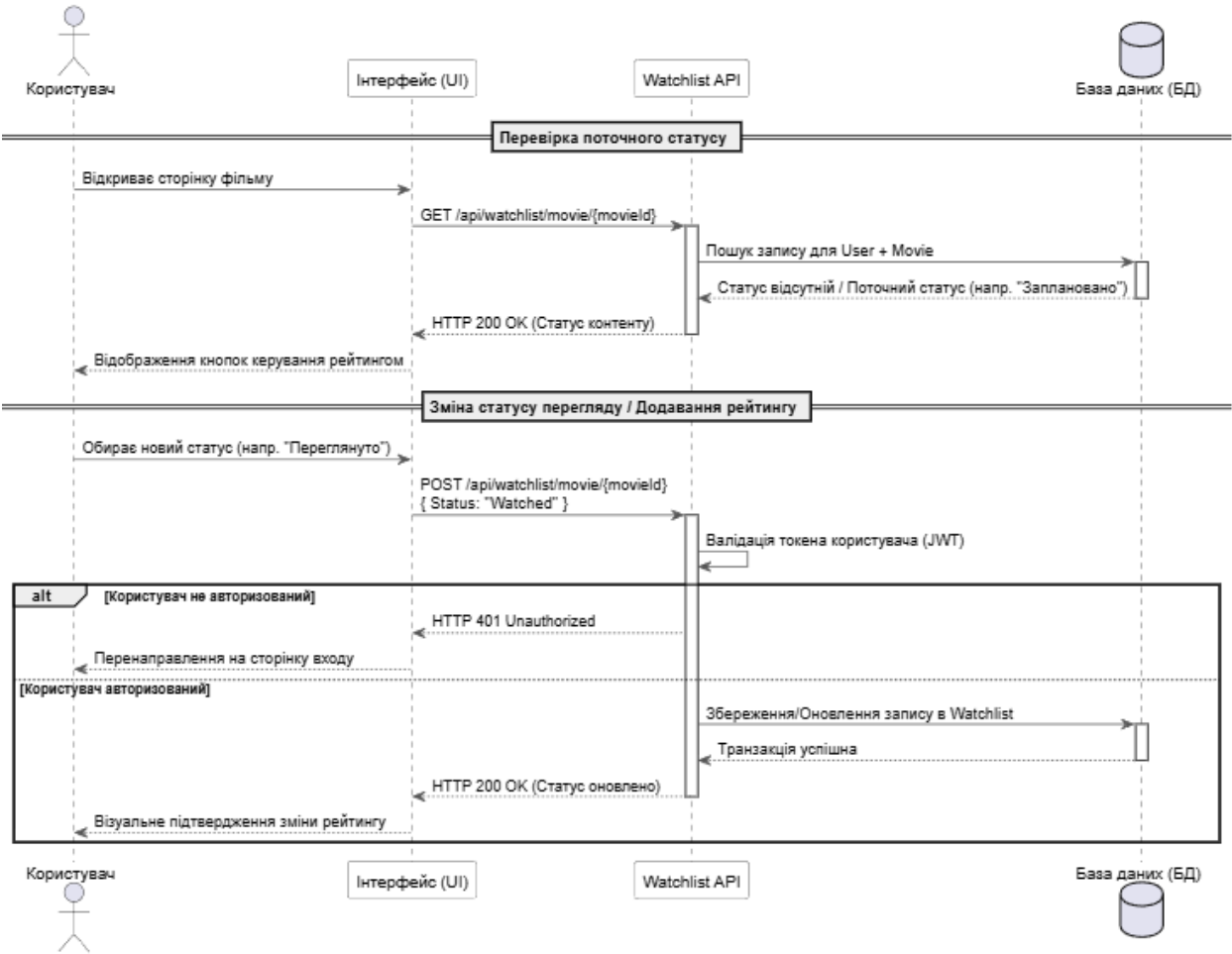


Рис. 2.8. Діаграма послідовності для додавання рейтингу

На рис. 2.8. продемонстровано діаграму послідовності взаємодії користувачів, контролерів та бази даних під час пошуку фільмів та роботи з рецензіями. На діаграмі можна побачити, що користувач шукає фільм або відкриває сторінку. Запит обробляється контролером, який отримує дані з бази даних та передає їх разом з метаданими клієнтській частині застосунку. Після отримання цієї інформації уповноважений рецензент може одразу залишити коментар та оцінку від 1 до 10.

Після того, як сервер збереже новий відгук у базі даних без використання зовнішніх сервісів модерації, система автоматично оновить середній рейтинг та кількість відгуків для цього фільму чи телешоу. В рамках цього процесу

було створено механізм голосування, який дозволяє іншим користувачам оцінювати корисність залишених відгуків.

На цій діаграмі було продемонстровано логіку для додавання професійних рецензій користувачам з ролями критик або адміністратор. Після перевірки прав доступу система додає до бази експертний аналіз в окрему таблицю, ця дія також не використовує модерацію. Адміністратор керує каталогом фільмів, в цьому блоці а не перечитує тексти чи редагує їх.

На діаграмі також показано роботу модуля сповіщень у режимі реального часу через хаб SignalR, який надсилає користувачам сповіщення в синхронному режимі про нові нагороди або системні події. Це підтверджує, що оновлення рейтингів та збереження рецензій повністю автоматизовані та відбуваються на рівні контролерів додатків.

Діаграма взаємодії на рис. 2.9 описує структуру компонентів системи в процесі пошуку медіа та публікації рецензій, зосереджуючись на структурних зв'язках між об'єктами та точній нумерації повідомлень. У цьому сценарії реалізується взаємодія профілів користувачів, контролерів фільмів, звичайних коментарів та рецензій критиків, моделей предметної області контенту та рецензій, контексту бази даних та центру сповіщень. Все починається з повідомлення користувача до контролера фільмів, який через контекст даних витягує необхідні фільми з бази даних та повертає всю інформацію з актуальними рейтингами до інтерфейсу.

Коли публікується відгук контролер перевіряє сесію та зберігає коментар у базі даних, після чого система починає перераховувати середній рейтинг фільму. Інші користувачі можуть голосувати за цей відгук.

Для критиків діє окремий контролер який допомагає та спрощує написання професійної рецензії до фільму також він займається реєстрацією професійної рецензії без участі модераторів.

		Денисенко Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		58

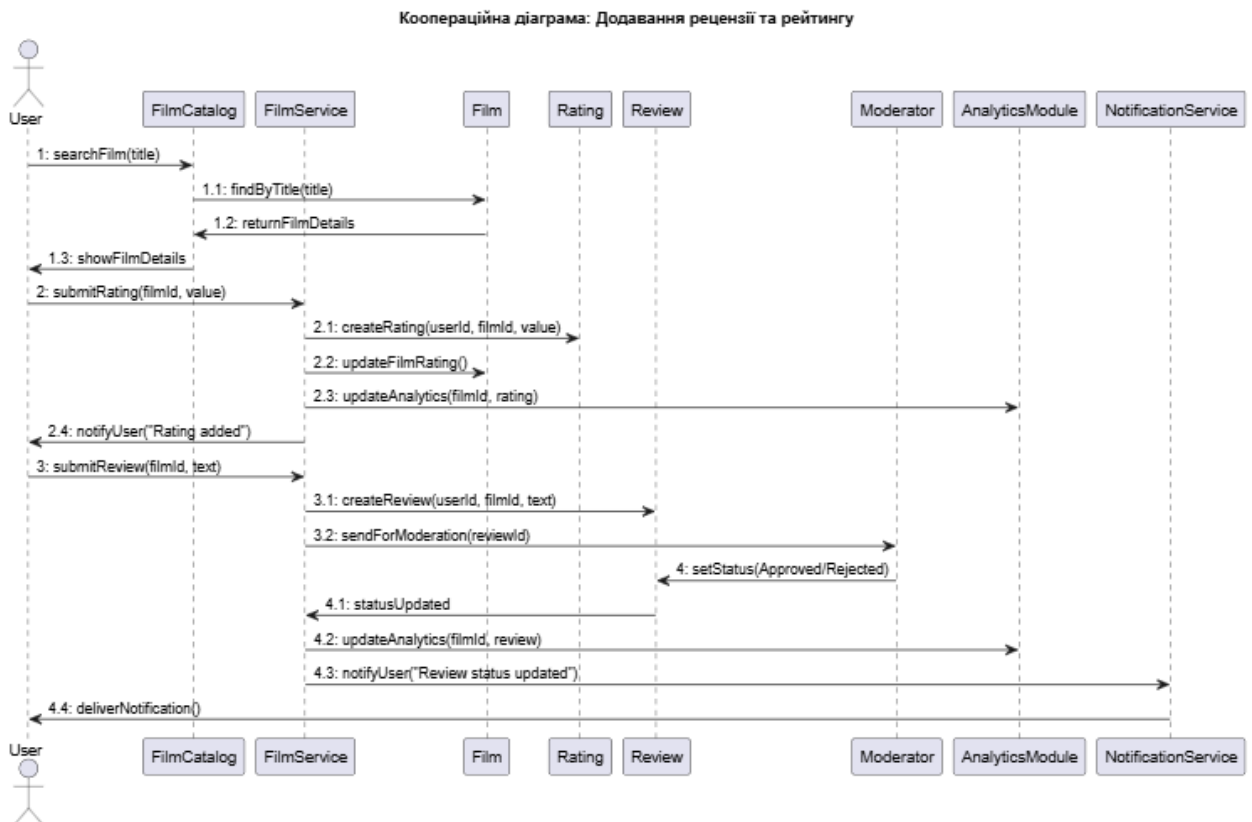


Рис. 2.9. Діаграма кооперацій для рецензій та рейтингу

2.4. Реалізація функціоналу системи рецензування фільмів та серіалів

Головним елементом у соціальній взаємодії платформи “Kovix” є створення та публікації відгуку чи рецензії користувача. Цей відгук є сполучною артерією між основними компонентом системи, людиною, яка створює, матеріалом для перегляду та механізмом оцінювання. Це пов’язано з тим що корисно розглянути логіку обробки даних та оновлення статистичних показників на прикладі процедури додавання нового відгуку.

На рівні інтерфейсу прикладного програмування зв'язок з клієнтським додатком здійснюється контролерами, побудованими на ASP.NET Core [16]. Існує спеціальний метод під назвою “Create”, який призначений для отримання HTTP-запиту, перевірки особи користувача, перевірки виконання бізнес-вимог та постійного зберігання отриманих даних у реляційній базі даних.

Ця кінцева точка інкапсулює логіку роботи з відгуками. Окрім збереження об'єкта, метод перераховує середній рейтинг фільму та кількість відгуків на кожен об'єкт відгуку в режимі реального часу. Атрибут [Authorize]

гарантує, що функціональність доступна лише зареєстрованим користувачам, а додаткові перевірки в коді запобігають діям заблокованих користувачів та дублюванню контенту. Дані, що надходять для операції, упаковуються в клас з назвою "CreateReviewDto", який є сутністю переміщення даних. Цей клас має лише необхідні атрибути, тобто унікальний ідентифікатор фільму, рейтинг і вміст відгуку. Такий підхід забезпечує чітке розділення між внутрішньою моделлю бази даних та зовнішнім інтерфейсом програми це допомагає запобігти передачі непотрібної інформації.

Валідація вхідних даних дозволяє не тільки обробляти некоректну інформацію до етапу взаємодії з базою даних, а й підвищує надійність системи та уніфікує процес обробки помилок. Основна логіка формування зворотнього зв'язку знаходиться безпосередньо в методі контролера.

На цьому рівні перевіряються конкретні бізнес обмеження, перевіряється чи заблоковано користувача, перевірка на заборону повторення відгуку для того самого медіа-продукту і ініціалізується сутність домену Review, після чого оновлюються сукупні показники фільму.

Після того як відбудеться збереження відгуку від звичайного користувача чи рецензії критика, система автоматично почне перераховувати, оновлює середню оцінку та загальну кількість голосів для відповідного фільму. Таке об'єднання даних забезпечує швидке відображення актуальних параметрів у каталозі та прибирає потребу в складних обчисленнях під час перегляду сторінки кожного разу.

Даний метод обробки даних зменшує навантаження на базу даних під час активного перегляду фільмів та серіалів. Щоб не виконувати важкі загальні функції для будь-якого запиту, система отримує готові обчислені значення з вже готової бази даних. Такий підхід перетворює важку операцію на просте зчитування даних з таблиці фільмів, це дуже важливо для швидкості роботи інтерфейсу, який може використовувати велика кількість користувачів.

Наявність окремого класу "DTO" для передачі даних додає додатковий рівень безпеки на бекенді, також це дуже спрощує структуру та розуміння для

		Денисенко Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		60

чого потрібен той чи інший клас. Це дозволяє явно контролювати поля в які можна записувати з боку клієнта, а в які ні. Це допомагає уникнути випадкового перезапису системних полів, таких як внутрішні службові теги. Через це зменшується ймовірність атаки та спрощується налагодження завдяки тому, що вхідні дані завжди мають передбачувану структуру.

Однією з головних переваг на сайті є вплив на клієнта. Сервер повертає добре структуровану відповідь, яку можна миттєво оновлювати на сторінках без перезавантаження в асинхронному режимі. Завдяки такому підходу користувач бачить свій опублікований відгук та оновлений рейтинг фільму миттєво, забезпечуючи приємне відчуття без помітних затримок.

З точки зору випускої кваліфікаційної роботи бакалавра, такий рівень деталізації бізнес логіки демонструє дуже глибоке розуміння того, як архітектурні рішення впливають на взаємодію з кінцевим користувачем та стабільність програмного продукту в цілому.

Такий підхід дозволяє забезпечити актуальність пошукових даних без втручання в основну бізнес-логіку програми та спрощує пошук відповідної інформації, що допомагає не тільки самому розробнику, а й людині яка буде в майбутньому займатися цієї чи іншою частиною програми.

Ще потрібно розповісти про архітектурне рішення масштабованості системи. Винесення логіки перерахунку статистики у метод створення відгуку дозволяє уникнути проблем з синхронізацією даних, які часто виникають у розподілених системах. Всі необхідні обчислення відбуваються в межах однієї транзакції, це гарантує що користувач ніколи не побачить ситуацію, де відгук уже додано, а загальний рейтинг фільму ще не змінився. Це підвищує довіру користувачів до вебплатформи та робить поведінку інтерфейсу повністю передбачуваною.

Крім того даний підхід полегшує процес тестування. Оскільки бізнес логіка не розкинута по всьому проєкту, а тримається в конкретних методах контролерів та сервісах, це дає змогу легко писати модульні тести для кожної частини системи.

		Денисенко Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		61

Висновки до другого розділу

Було проведено аналіз предметної сфери з позицій об'єктно-орієнтованого підходу. Визначено найважливіші вимоги до функціонала системи, а також виявлено основних учасників взаємодії та їхні типові шляхи роботи з платформою.

Для візуалізації структури додатка було розроблено діаграми кооперацій та класів. Це допомогло чітко розділити основні об'єкти системи на користувачів, медіаконтент, відгуки та оцінки та інших і налаштувати зв'язки між ними, асоціацію, агрегацію та композицію.

На основі відомих сутностей було спроектовано реляційну базу даних, яка зберігає інформацію про користувачів, контент, відгуки та оцінки. Приведення таблиць до норми дозволило уникнути дублювання даних, захистити від помилок та пришвидшити виконання запитів до бази даних.

Здійснено моделювання динаміки роботи системи. Використовуючи діаграми діяльності, послідовності та співпраці, було деталізовано низку ключових алгоритмів бізнесових процесів: механізм пошуку потрібного контенту, процедуру надсилання рецензій, методи управління наборами даних та процеси модерації. Це дозволило виявити й усунути дисфункції в логічних ланцюжках ще на етапі проєктної роботи. Відбулася імплементація програмної складової.

Створено програмну частину через реалізацію як RESTful API на базі ASP.NET Core. Створено контролери для обробки запитів, впроваджено DTO для безпечного обміну даними між клієнтом і сервером, а також налаштував валідацію вхідних параметрів.

Щоб система працювала швидше, було додано механізм автоматичного оновлення й кешування агрегованих показників за допомогою середньої оцінки та кількості коментарів.

		Денисенко Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		62

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ВЕБПЛАТФОРМОЮ ДЛЯ РЕЦЕНЗУВАННЯ ФІЛЬМІВ ТА СЕРІАЛІВ

3.1. Визначення варіантів використання та об'єктно-орієнтованої структури системи

Система Kovix, що була розроблена для випускної кваліфікаційної роботи бакалавра, використовує мікросервісну архітектуру, тому кожен з етапів може бути розміщений на окремих середовищах. Це робить розгортання більш гнучким та полегшує масштабування окремих компонентів системи та не залежить від інших мікросервісів, це допомагає ефективніше керувати та масштабувати систему в майбутньому [18].

Для кращого розуміння процесу розгортання було візуалізовано представлення діаграми розгортання на рис. 3.1

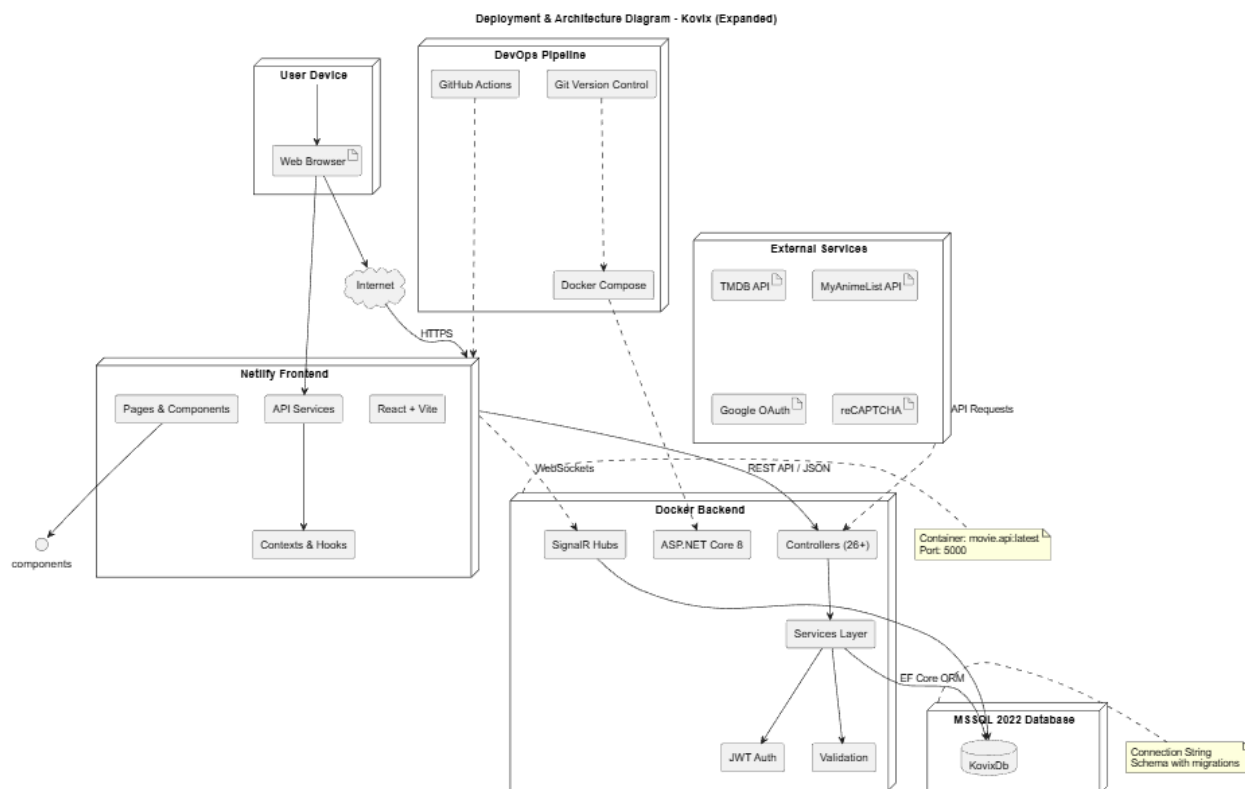


Рис.3.1. Діаграма розгортання

Вебплатформа “Kovix” має модульну архітектуру, розділену на три основні компоненти, Frontend який складається з React + Vite [27]. Backend

		Денисенко Д.О.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Сфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		63

який включає в себе ASP.NET Core Web API які були контейнеризовані у Docker. Також було використано MS SQL Server. Цей підхід забезпечує відокремлення відмов, незалежну розробку та оновлення сервісів, гнучке масштабування ресурсів під кожен компонент.



Схема 3.1. Взаємодія користувача

Схема взаємодії наведено на діаграмі розгортання (рис. 3.1).

Рекомендовані системні вимоги до сервера:

Процесор	Оперативна пам'ять	Дисковий простір
• 4 ядра • тактова частота від 3 ГГц	• мінімум 8 ГБ	• мінімум 16 ГБ вільного місця

Схема 3.2. Системні вимоги до сервера

У системі передбачено три етапи запуску проєкту

Таблиця 3.1

Етапи збірки проєкта

Розгортання через Docker Compose	Ручне розгортання .NET	Інфраструктурні компоненти
Спочатку потрібно підготувати сервер, для цього встановіть робочому пристрої Docker та Docker Compose, після чого клонуйте репозиторій проєкту.	Збірка релізу на робочому місці або CI/CD	Якщо БД розгортається окремо від основного файлу конфігурації, потрібно виконати

Далі потрібно створити файл .env у корені проєкту та задати необхідні змінні.	Далі потрібно перенести файли Скопіюйте вміст папки out на сервер.	спеціальну команду, щоб мати доступ до неї.
Потім перейдіть у каталог проєкту та виконайте збірку та запустіть контейнер у фоновому режимі	Також потрібно виконати налаштування змінних оточення на сервері, рядок підключення до MS SQL, порт для запуску.	
І фінальною перевіркою стану буде Bash. Потрібно перевірити статус контейнера.	І вкінці запустіть як systemd на Linux або через CLI	

Важливе уточнення так як проєкт був розміщений на сайті у відкритому доступі і вам доведеться теж задеплоїти, при хостингу обов'язково вкажіть змінну оточення VITE_API_BASE_URL у панелі керування та .env файлі, щоб клієнтський застосунок знав адресу API.

Рекомендації з безпеки та експлуатації

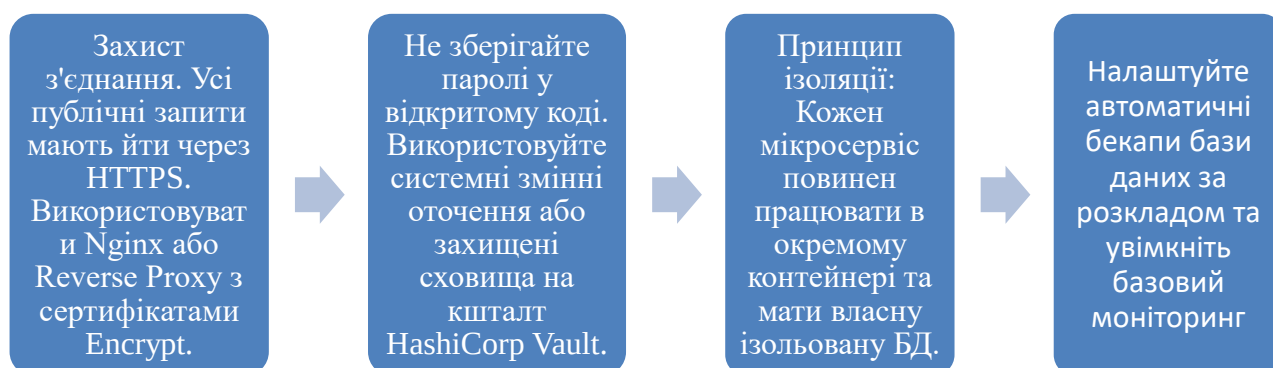


Схема 3.3. Рекомендації з безпеки та експлуатації

Клієнтська частина системи є кросбраузерною та сумісною з усіма сучасними браузерами. Коректне відображення та стабільна робота

інтерфейсу забезпечується в Google Chrome, Mozilla Firefox, Microsoft Edge, Safari та їх аналогах.

3.2. Структура інтерфейсу вебплатформи

Вебплатформа для рецензування фільмів та серіалів працює в браузері через локалхост та розміщено на сервері у відкритому доступі.

Після того як користувач потрапить на головну сторінку, він бачить слайдер з фільмами, навігаційну панель в якій в нього є “Каталог”, “Актори”, “Оголошення”, “Тір Лісти”, “KovixPRO”, “Рейтинги”. Над навігаційною панеллю знаходиться блок, з “Анонс Банером” де користувач може переглянути останній доданий фільм чи серіал на сайт, також цей блок користувач може закрити якщо він йому заважає. Під банером, знаходиться блок “Новин” де адміністрація сповіщує користувачів про можливі технічні роботи. Також користувач може змінити тему сайту на вбудовану або кастомну (рис.3.3-3.5). Далі користувач може зареєструватися/увійти (рис.3.6-3.8). Ще користувача при вході на сайт, поки він не відреагує на впливаюче меню в нижньому куті, буде вітати повідомлення про використання cookie [19].

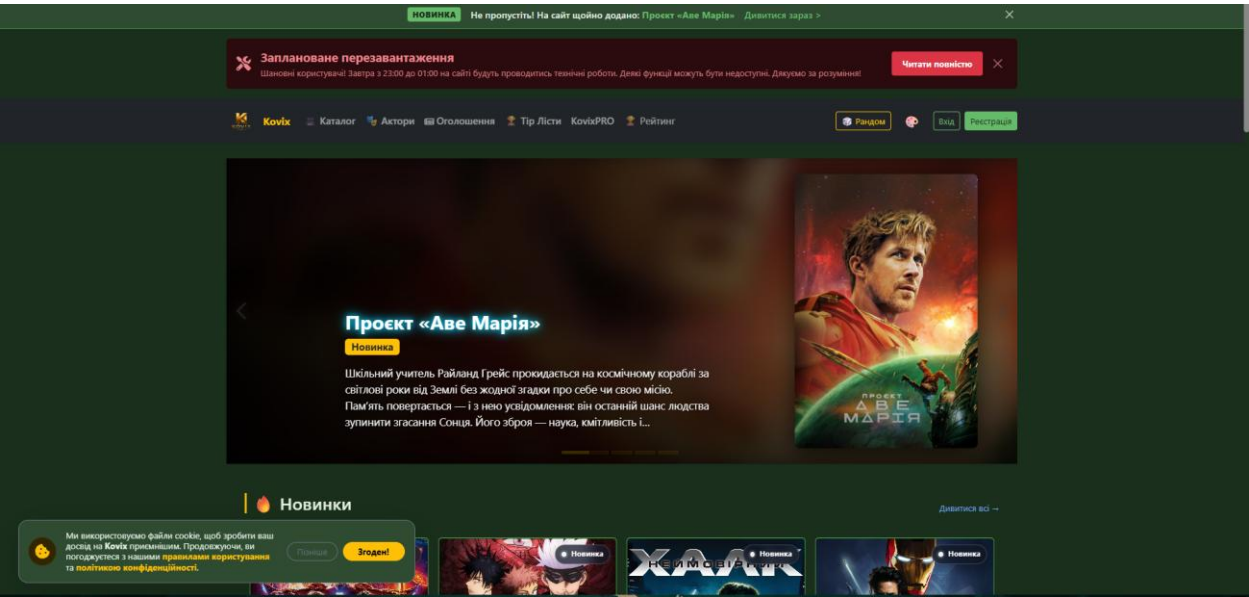


Рис.3.2. Головна сторінка вебплатформи Kovix

		Денисенко Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		66

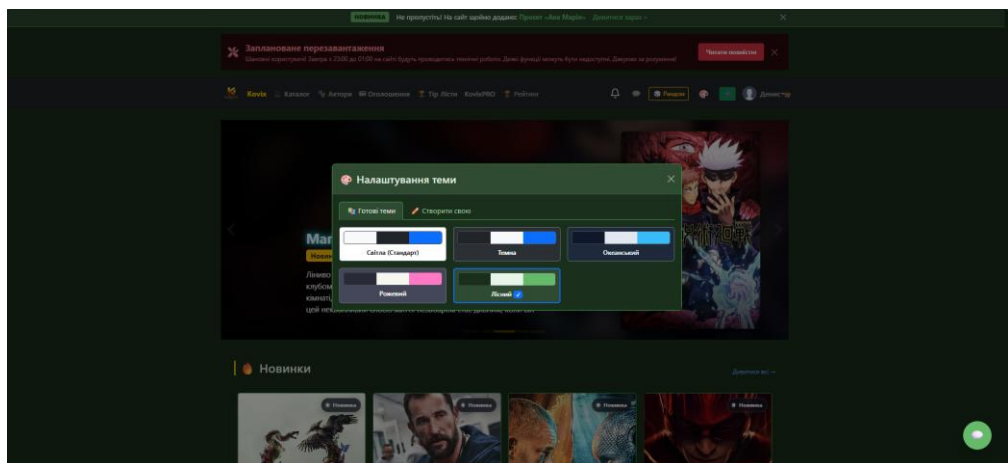


Рис.3.3. Налаштування теми

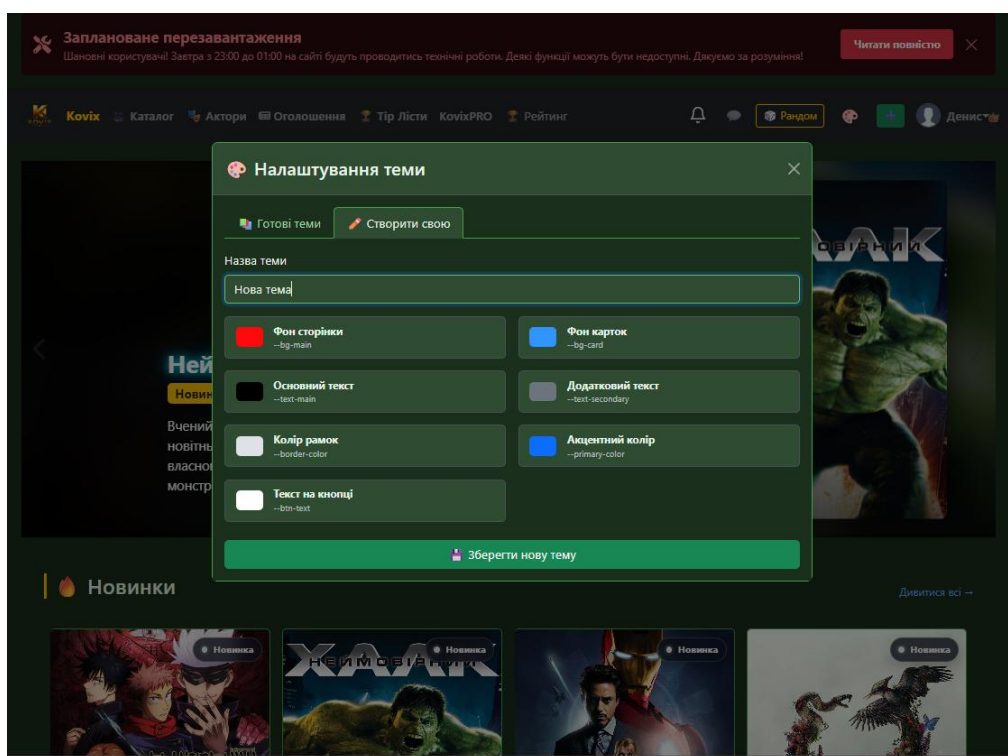


Рис.3.4. Створення кастомної теми

В користувача навіть не авторизованого є можливість кастомізації всього сайту (рис.3.3 - 3.5), також ці кастомні теми можна видаляти та створювати нові, в необмеженій кількості. Користувач у будь який момент може створити будь яку кольорову палітру на своєму профілі.

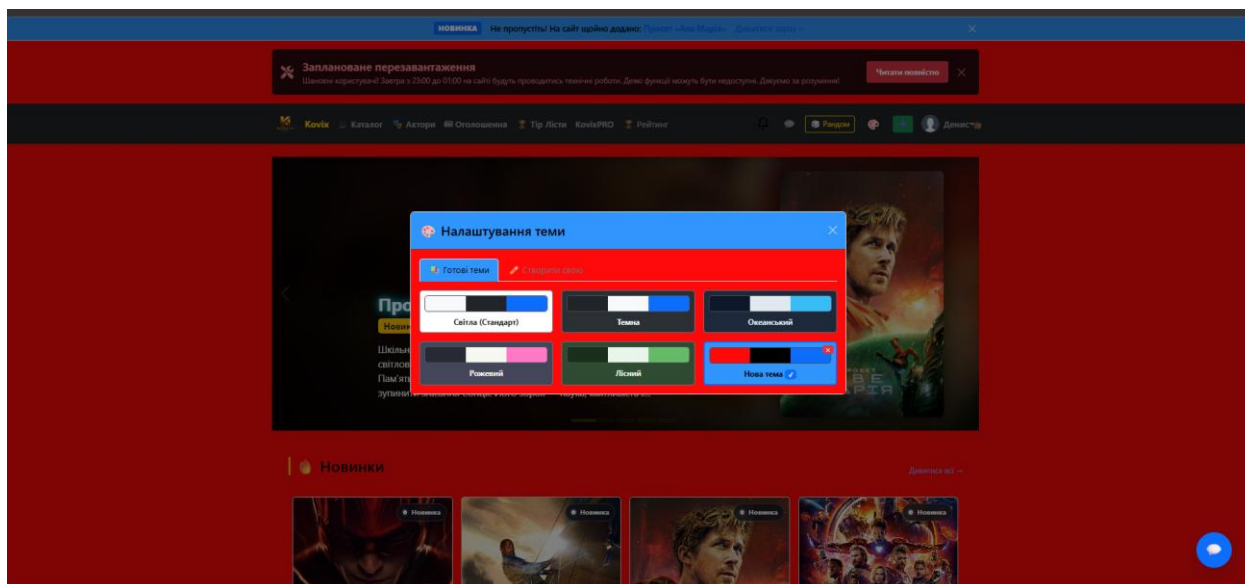


Рис.3.5. Відображення кастомної теми

Натиснувши кнопку “Зареєструватись”, користувачу побачить форму реєстрації (рис. 3.6). Після успішного введення всіх даних користувачу надсилається на електронну пошту лист з вітанням про успішну реєстрацію (рис.3.7). Якщо користувач зареєструвався він може зайти, але щоб потрапити на сайт йому ще треба підтвердити свою пошту (рис.3.8), після натиснення кнопки “Відправити на пошту” йому на пошту прийде код з 6 цифр які потрібно ввести щоб підтвердити особу, після успішного підтвердження користувач нарешті зареєстрований [20].

Рис.3.6. Форма реєстрації

		Денисенко Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		68

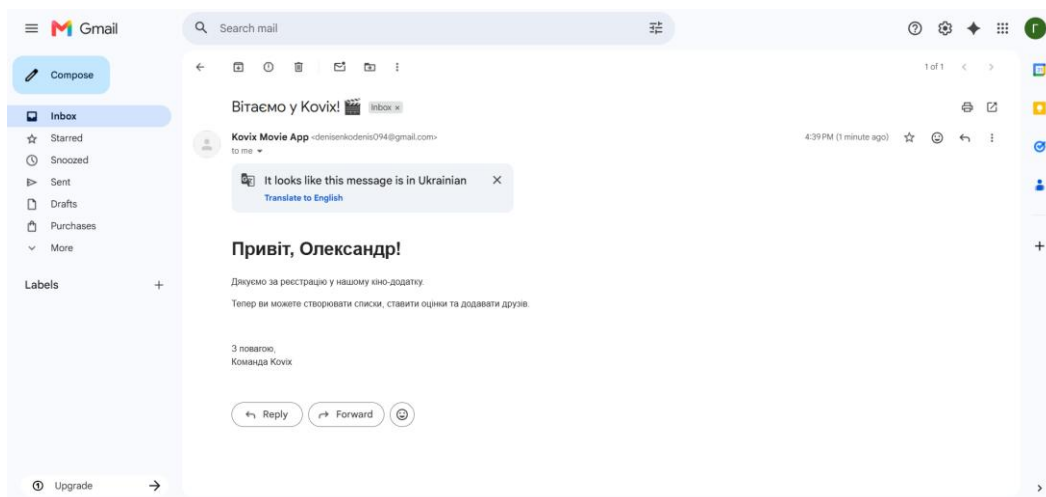


Рис.3.7. Лист з вітанням про успішну реєстрацію

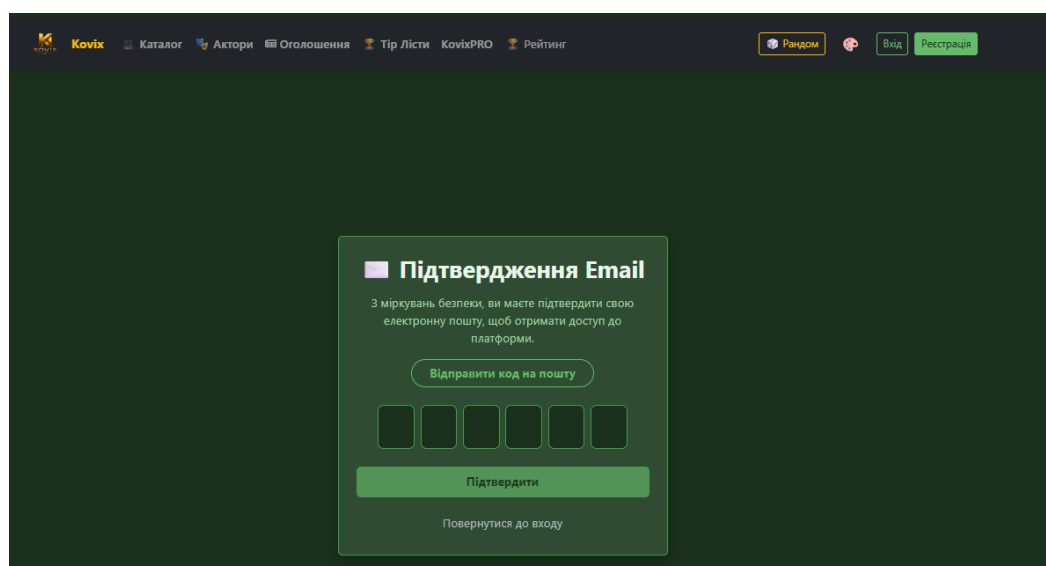


Рис.3.8. Підтвердження пошти

Наступним кроком буде вхід та декілька захисних механізмів для підвищення безпеки сайту. Спершу звичайний вхід (рис.3.9), тут користувач вводить свій емейл та пароль, пароль приховується на фронтенді та хешується на бекенді. Далі якщо користувач введе декілька разів не правильний пароль його буде заблоковано на 5 хвилин (рис.3.10 - 3.11). Через деякий час користувачу знову стане можливість увійти на сайт, після тимчасового блокування, якщо зайти з іншого аккаунту то користувач зайде без блокування, адже блокування відбувається не для браузера, а саме для користувача.

		Денисенко Д.О.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Сфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		69

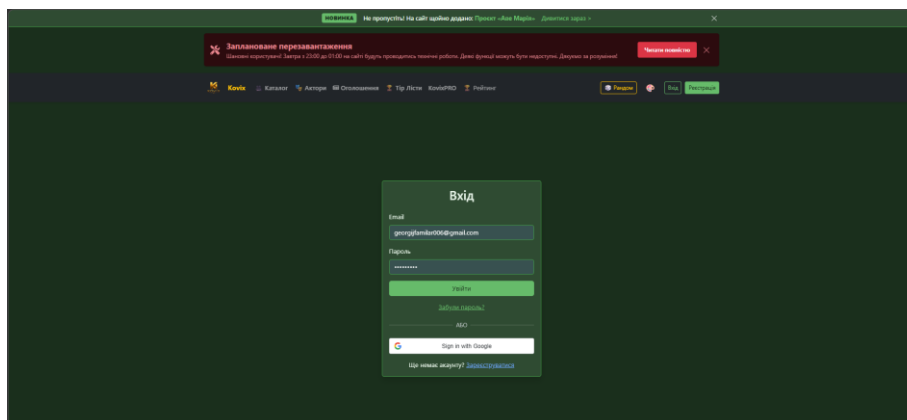


Рис.3.9. Вхід на сайт

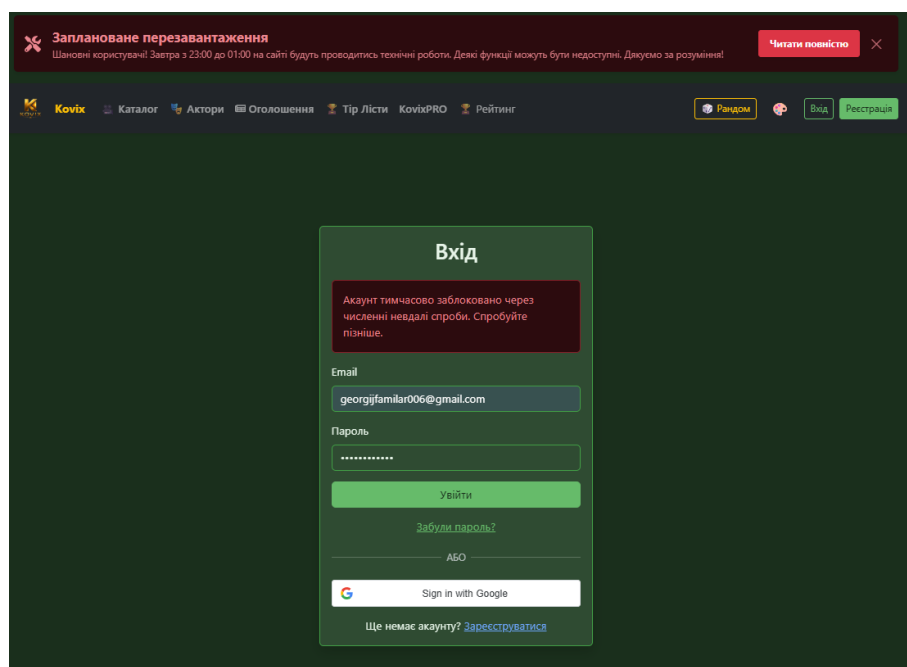


Рис.3.10. Блокування акаунта користувача

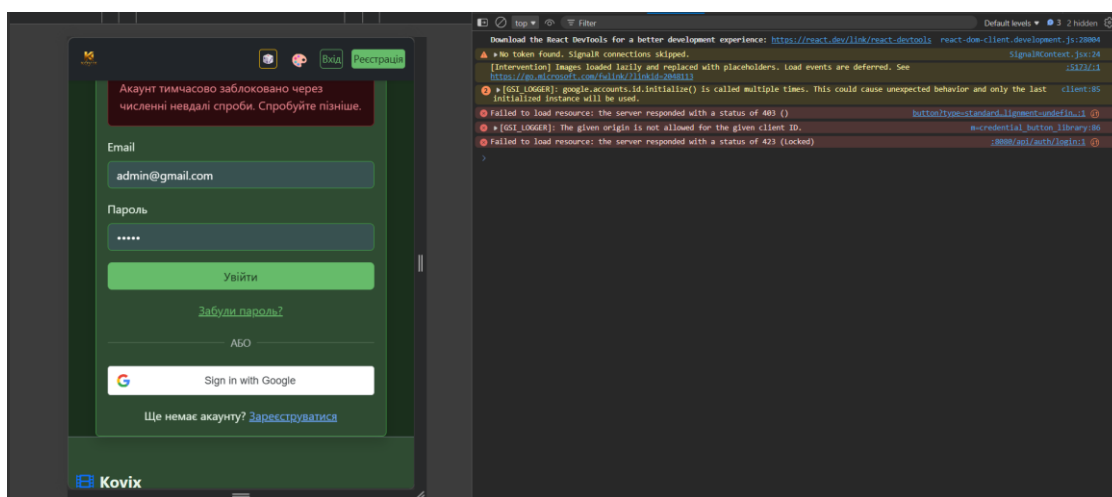


Рис.3.11. Заблокований стан з бекенду

		Денисенко Д.О.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Сфремов Ю.М.				70
Змн.	Арк.	№ докум.	Підпис	Дата		

Далі розберемо блок “Оголошення”, прикріплені повідомлення відображаються відразу на всіх сторінках їх можна детальніше переглянути якщо натиснути кнопку “Читати повністю” (рис.3.12), також якщо вони заважають їх можна закрити. Також користувач може перейти в блок з всім оголошеннями, для зручності продемонструю відображення зі сторони адміністратора (рис.3.13), адміністратор може виконувати всі базові CRUD операції над ними.

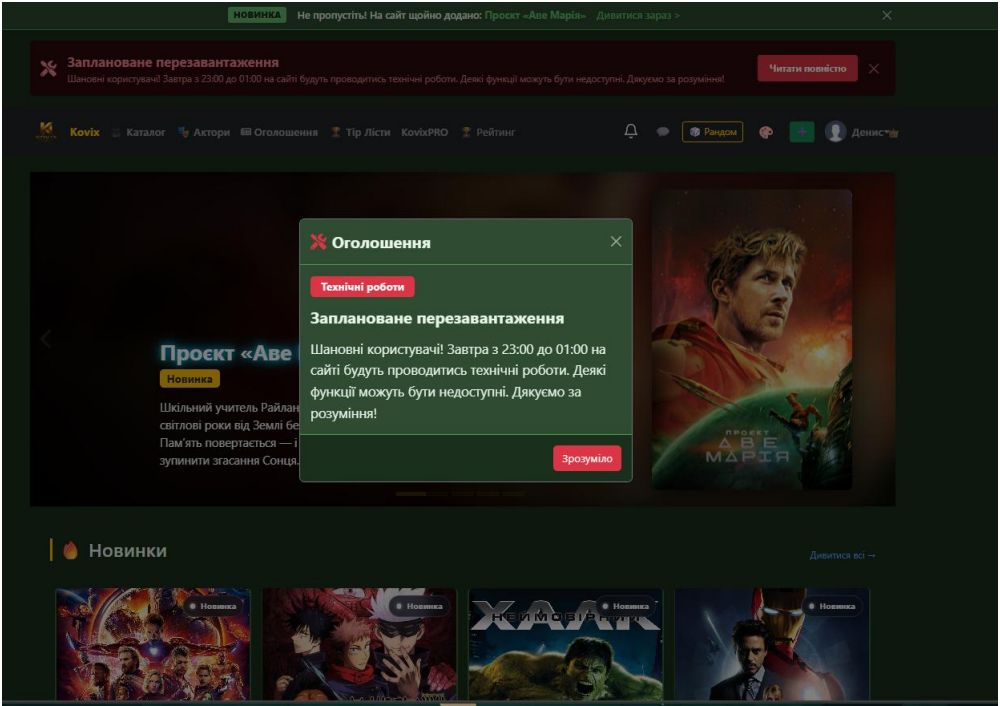


Рис.3.12. Детальний перегляд оголошення

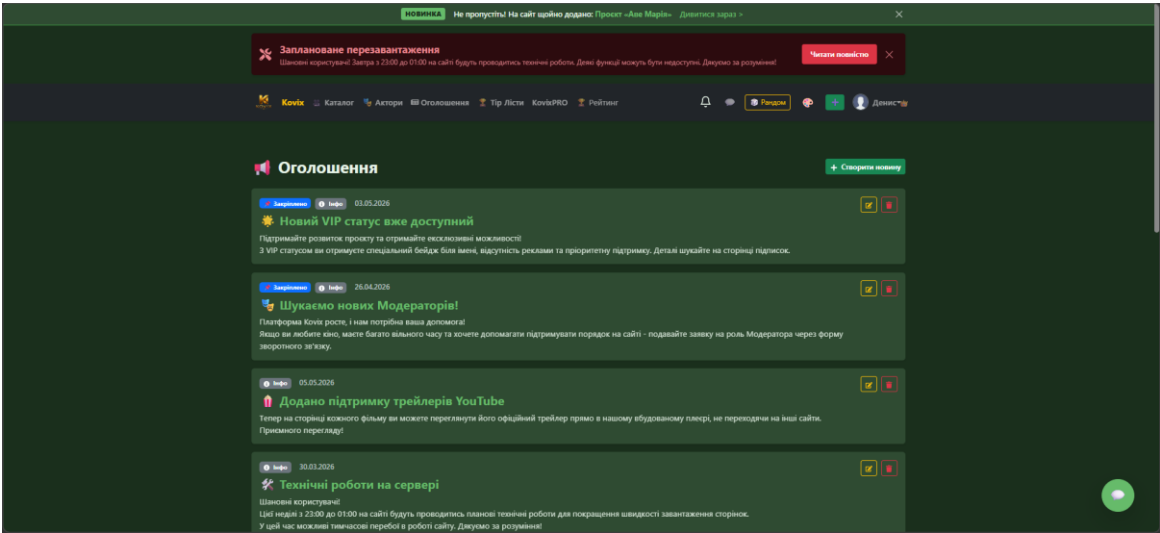


Рис.3.13. Сторінка з всіма оголошеннями

Наступним етапом буде блок “Каталог”, в ньому користувач може шукати фільм використовуючи пошук, фільтрацію по роках, жанрах або нагородах, також реалізовано сортування за такими типами, рейтинг, рік, датою додавання, популярністю та назвою (рис.3.15). Також користувач може змінити відображення фільмів використовуючи перемикач навпроти сортування (рис.3.14, 3.16), ще якщо в каталозі стає більше 8 фільмів то з’являється пагінація для комфортного перегляду сторінки (рис.3.17).

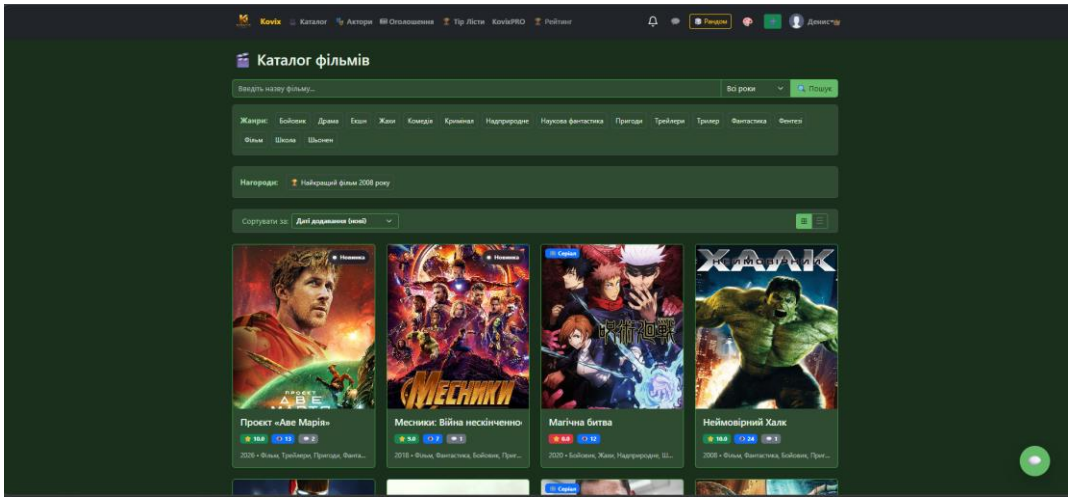


Рис.3.14. Каталог фільмів

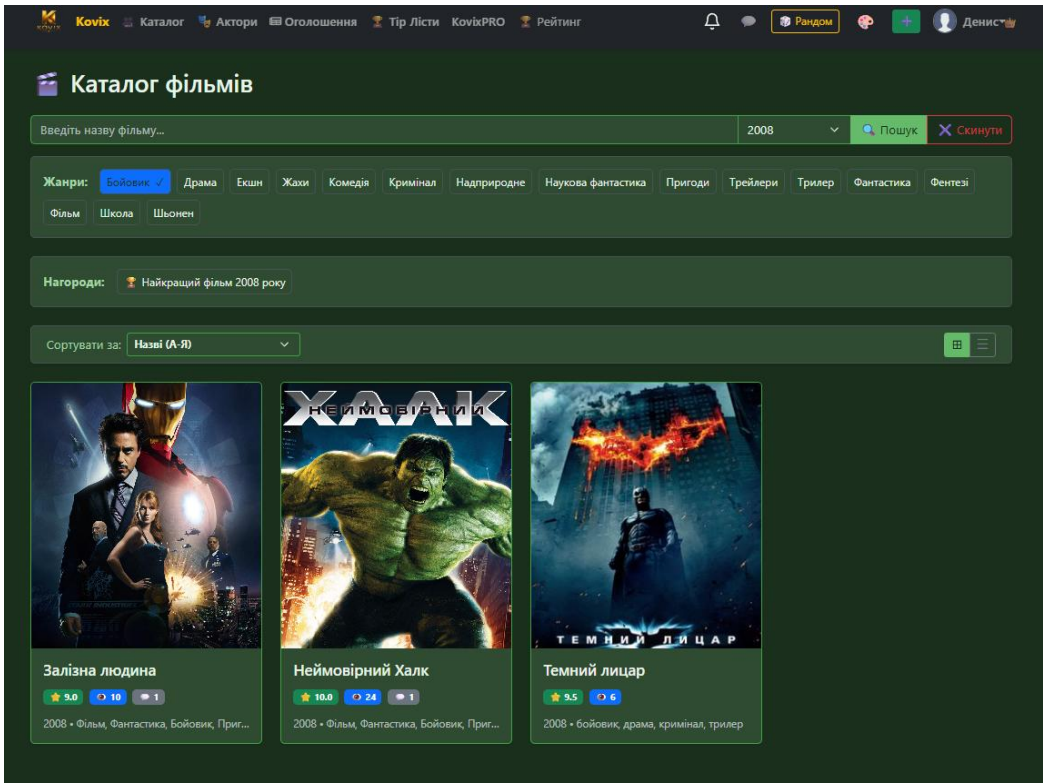


Рис.3.15. Фільтрація та сортування

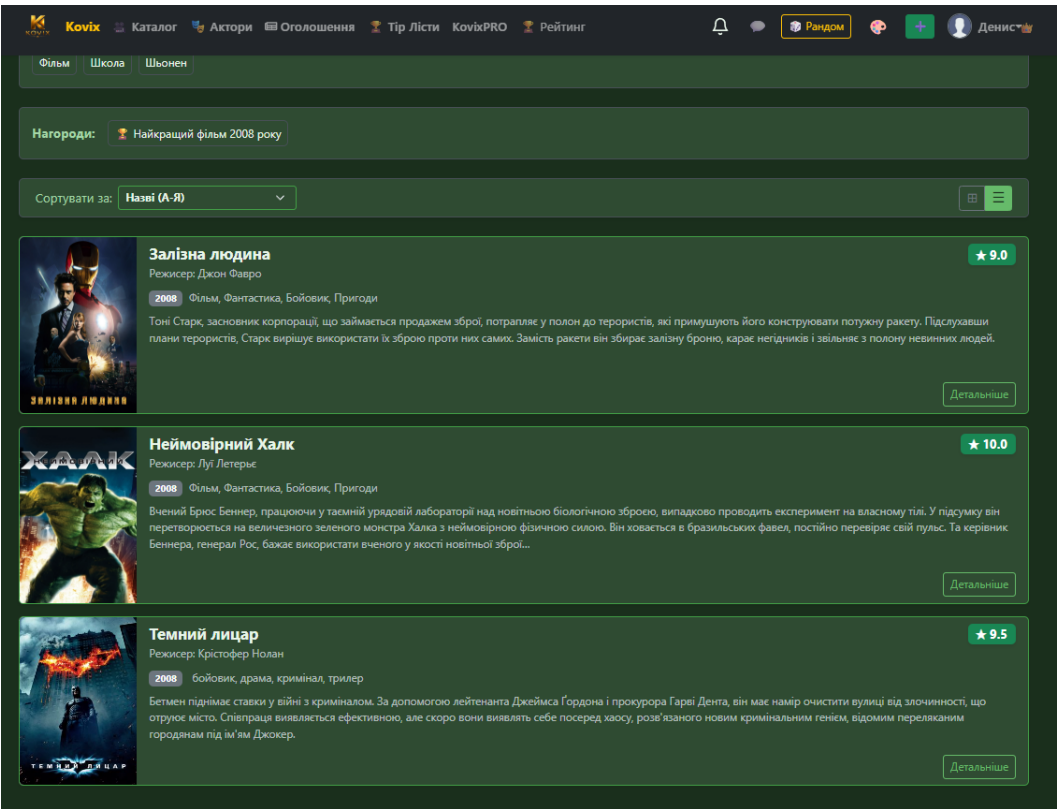


Рис.3.16. Зміна відображення

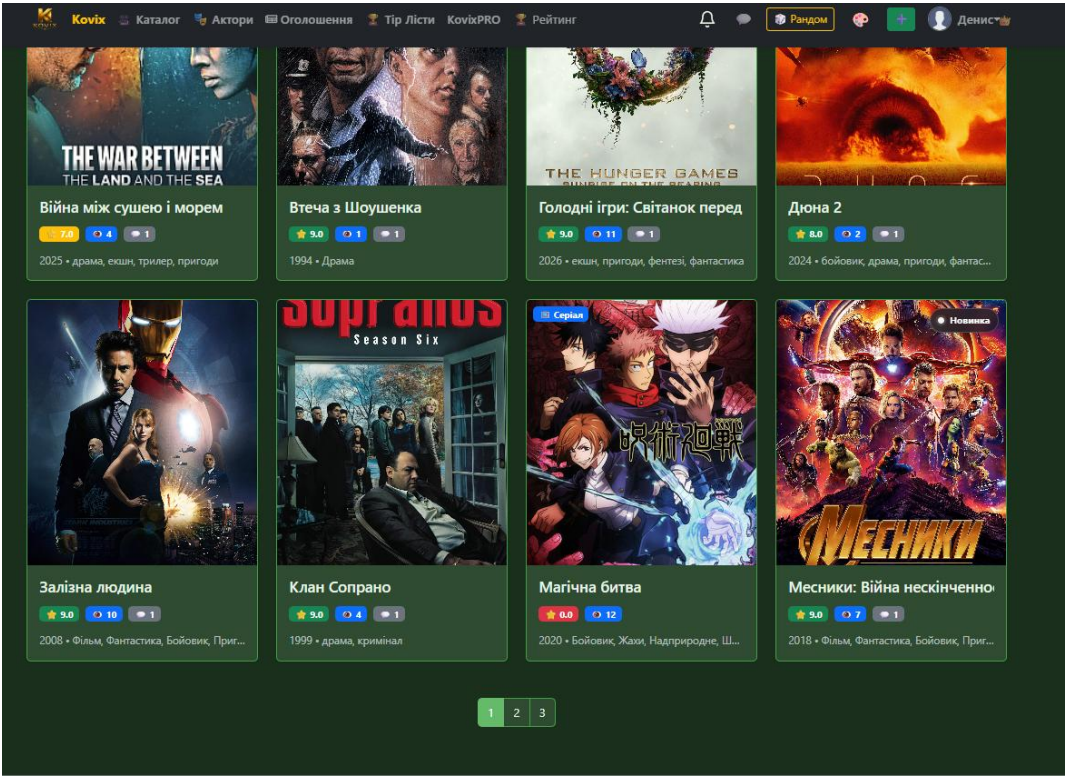


Рис.3.17. Пагінація

Далі йде блок з “Акторами”, тут також реалізовано базовий CRUD та пошук (рис.3.18), ще було створено пагінацію, на цій сторінці можна детально переглянути інформацію про актора чи додати його в “Чорний список”

		Денисенко Д.О.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Сфремов Ю.М.				73
Змн.	Арк.	№ докум.	Підпис	Дата		

(рис.3.19) натиснувши на кнопку “Заблокувати”, якщо заблокувати актора всі фільми з ним не будуть показуватися, до тих пір поки користувач його не розблокує.

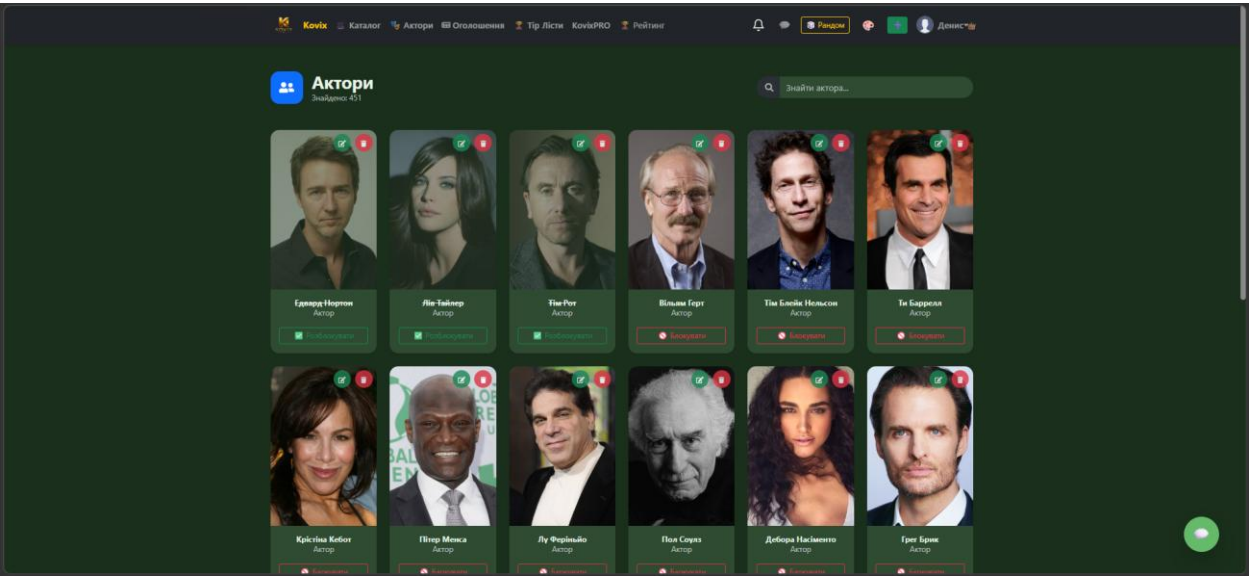


Рис.3.18. Сторінка акторів

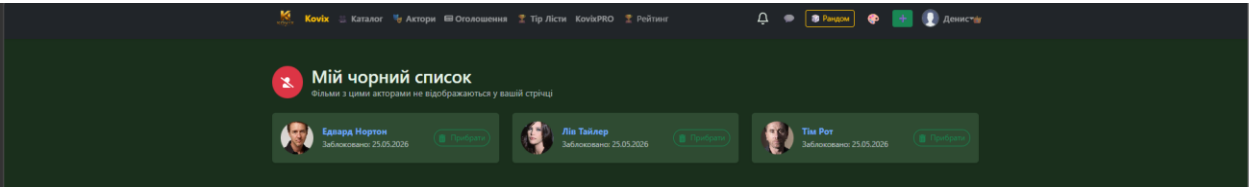


Рис.3.19. Чорний список

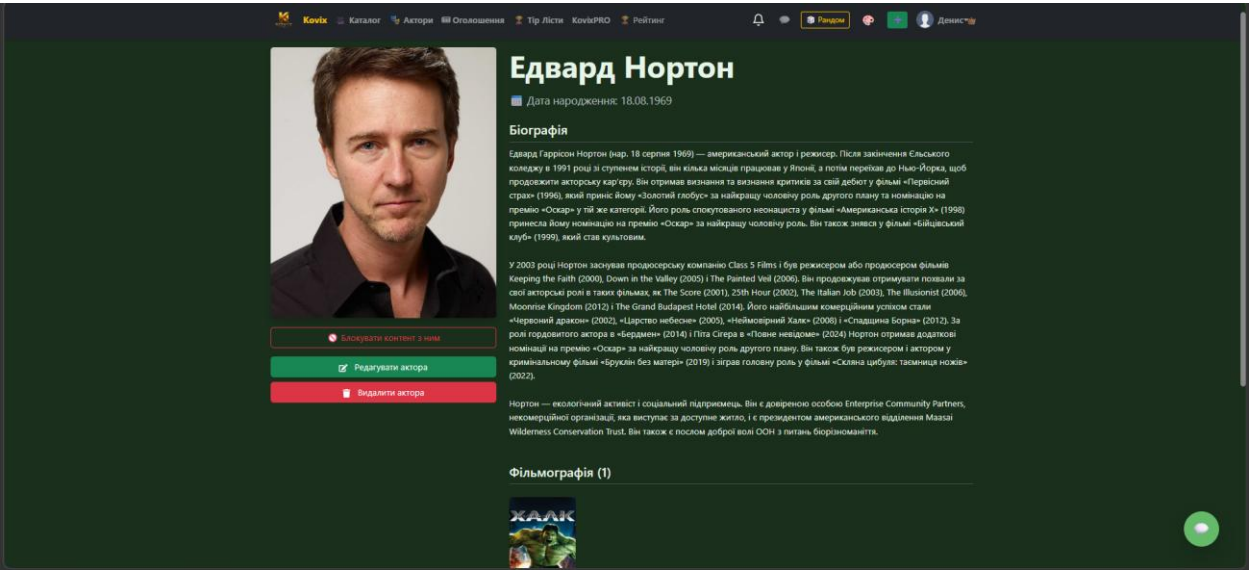


Рис.3.20. Детальна інформація про актора

Наступним етапом будуть тір лісти, тут також реалізовано всі CRUD операції, також є перемикування між різними типами лістів (рис.3.21).

Тір Лісти можуть створювати будь які зареєстровані користувачі (рис.3.22), але після створення тір ліст, в користувача є вибір залишити його приватним, його буде бачити тільки автор, якщо ж користувач вибере поділитися (рис.3.23), він відправляється модератору на погодження (рис.3.24), після того як буде погоджено заявку вона відразу стане доступною для всіх користувачів.

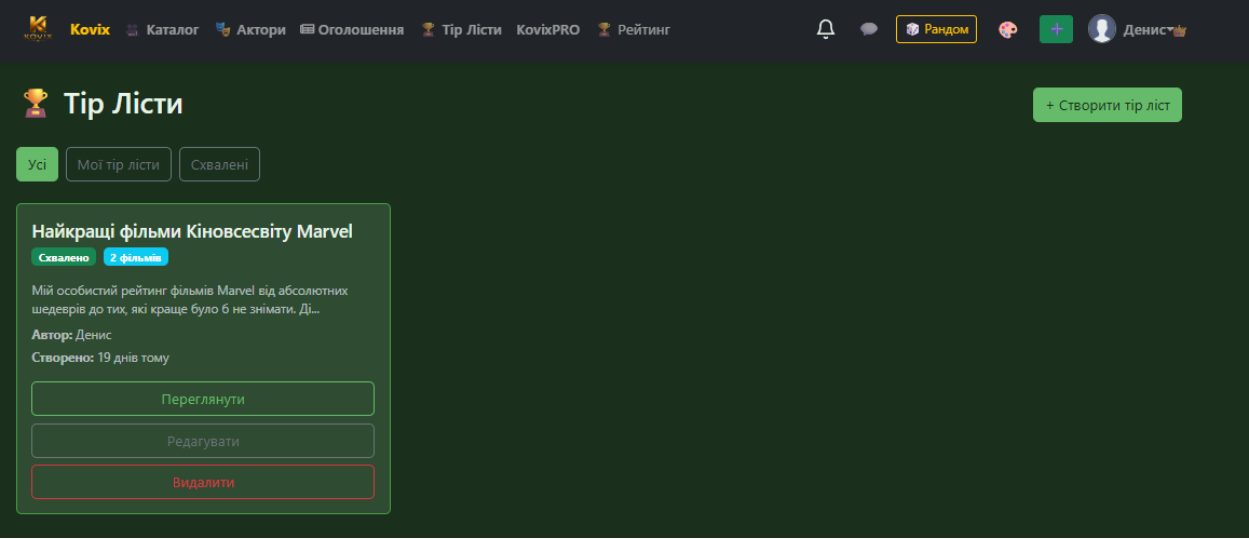


Рис.3.21. Сторінка Тір Лістів

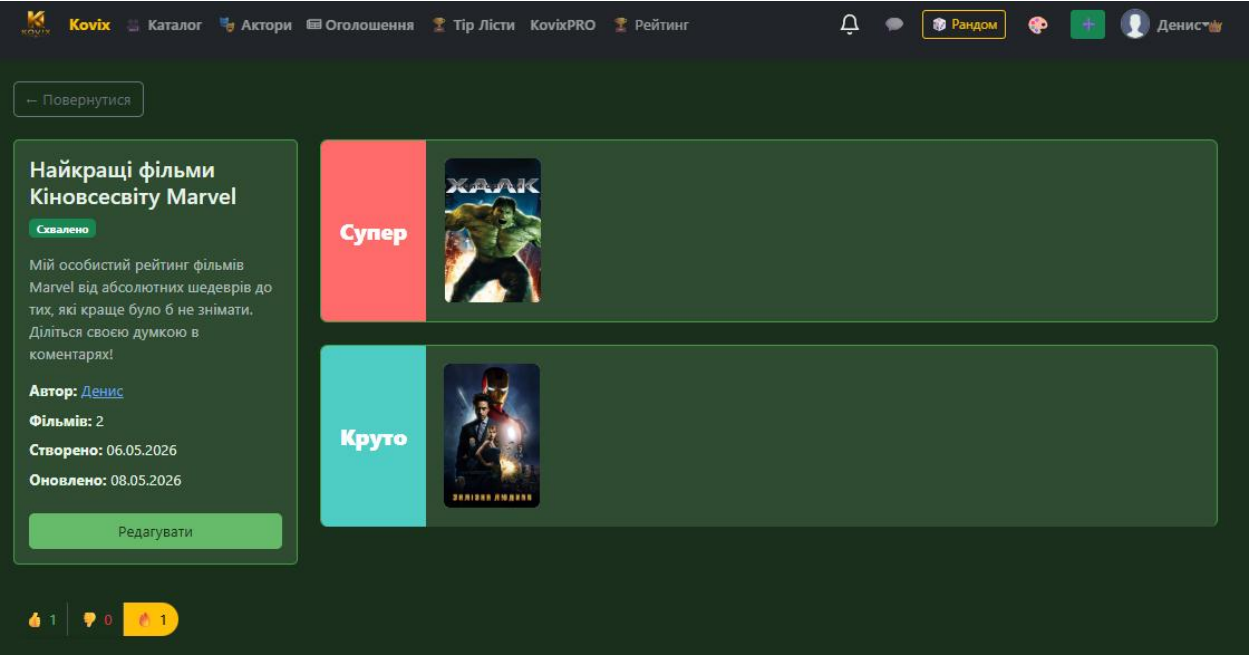


Рис.3.22. Перегляд Тір Ліста

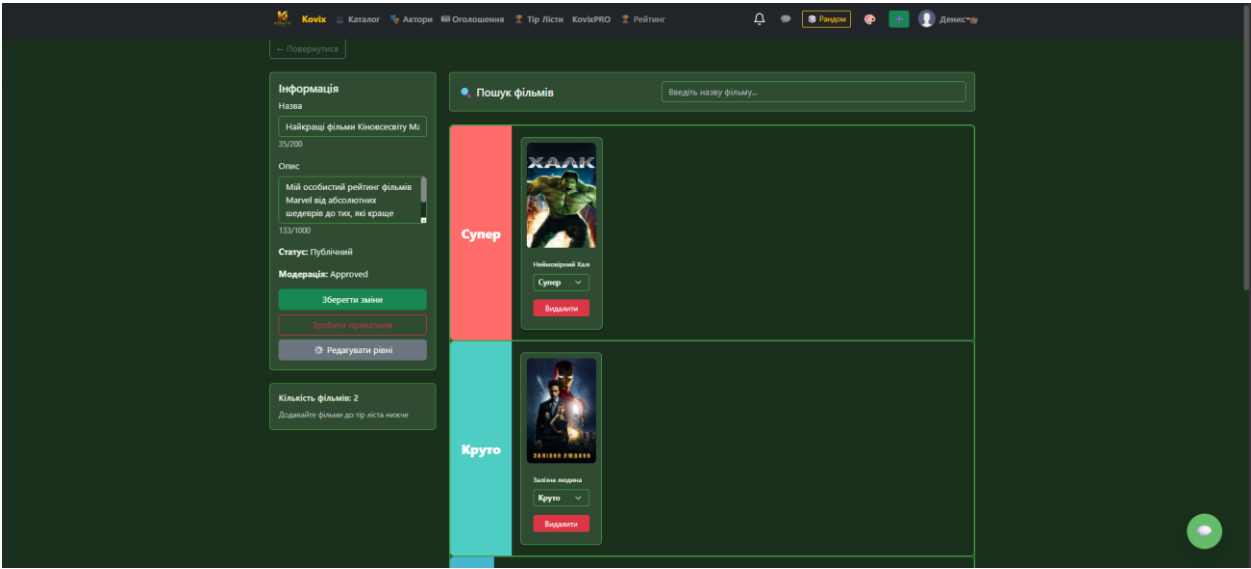


Рис.3.23. Редагування Тір Ліста

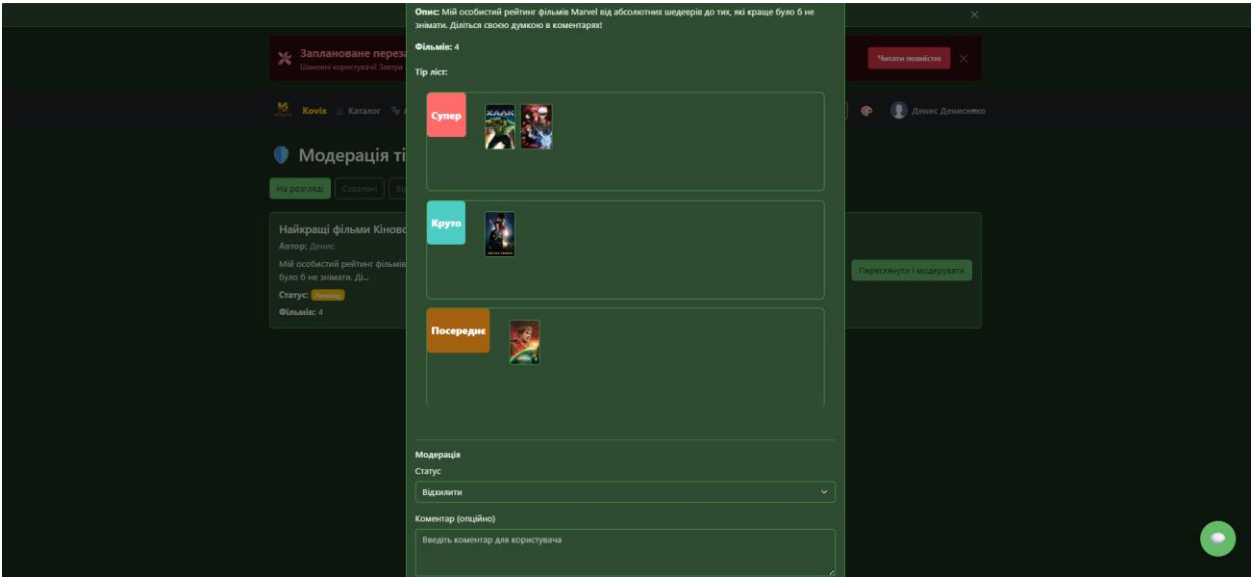


Рис.3.24. Відображення Тір Ліста на погодження модератором

На вебплатформі було реалізовано платні підписки (рис.3.25) через сервіс Stripe, якщо користувач вирішить придбати собі підписку на місяць чи рік, щоб отримувати бонуси від вебплатформи то він може перейти в блок “KovixPRO” та придбати собі відповідну підписку (рис.3.26), підписки реалізовані на даному етапі у вигляді “sandbox” через обмеження правил оплати на даній платформі, щоб візуалізувати можливість провести оплату цього безкоштовного режиму буде достатньо.

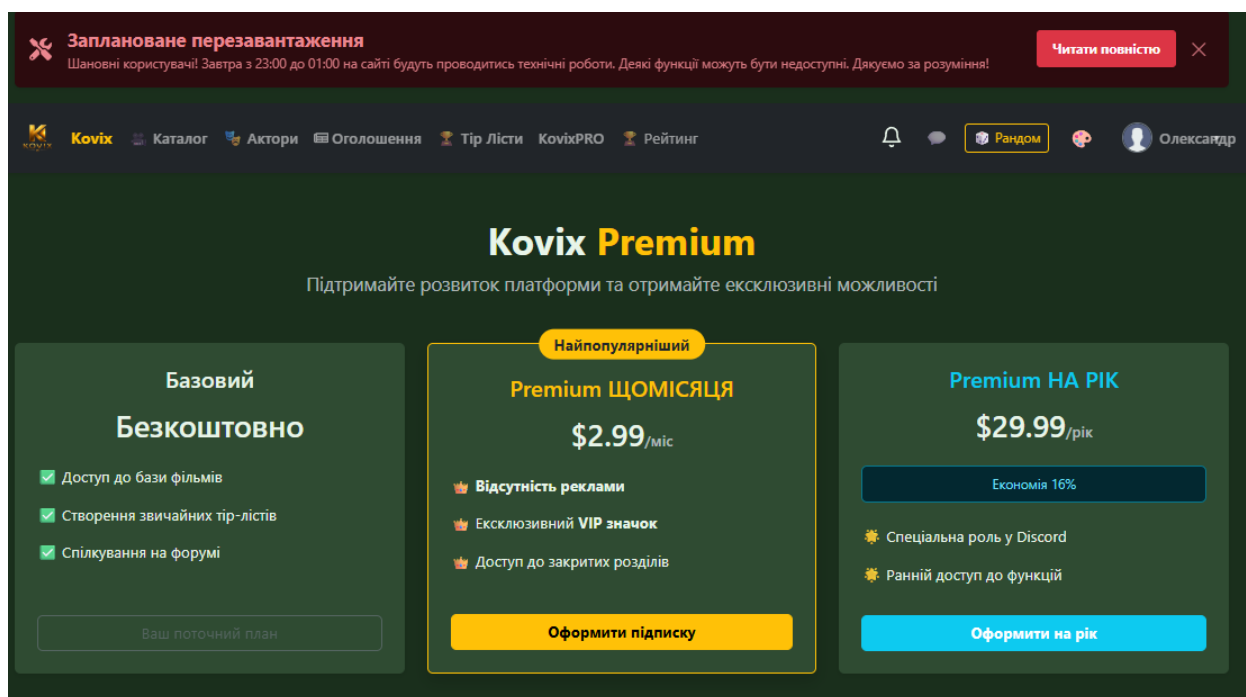


Рис.3.25. Сторінка купівлі

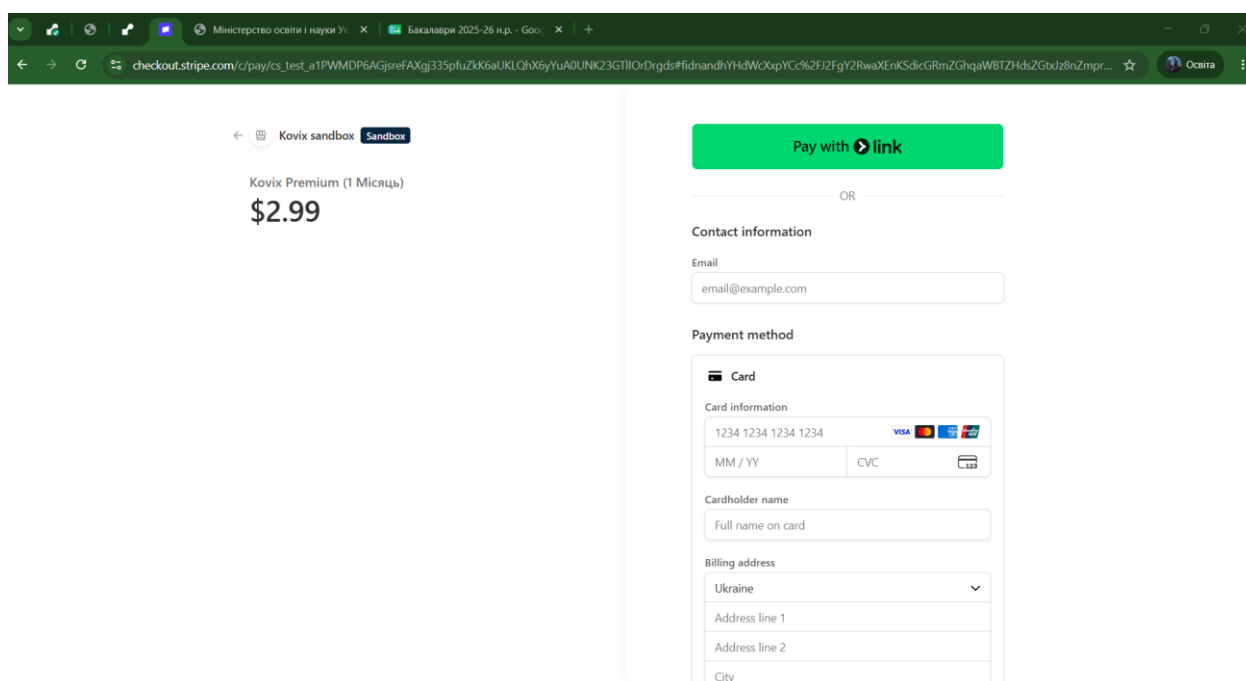


Рис.3.26. Оформлення купівлі

Наступний блок це “Рейтинг” тут відображаються найактивніші користувачі які пишуть коментарі (рис.3.27) чи проходять тести (рис.3.28), тести проходяться після фільму за бажанням користувача, він може перепроходити їх по декілька разів, щоб отримати ідеальний результат.

		Денисенко Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремов Ю.М.				77
Змн.	Арк.	№ докум.	Підпис	Дата		

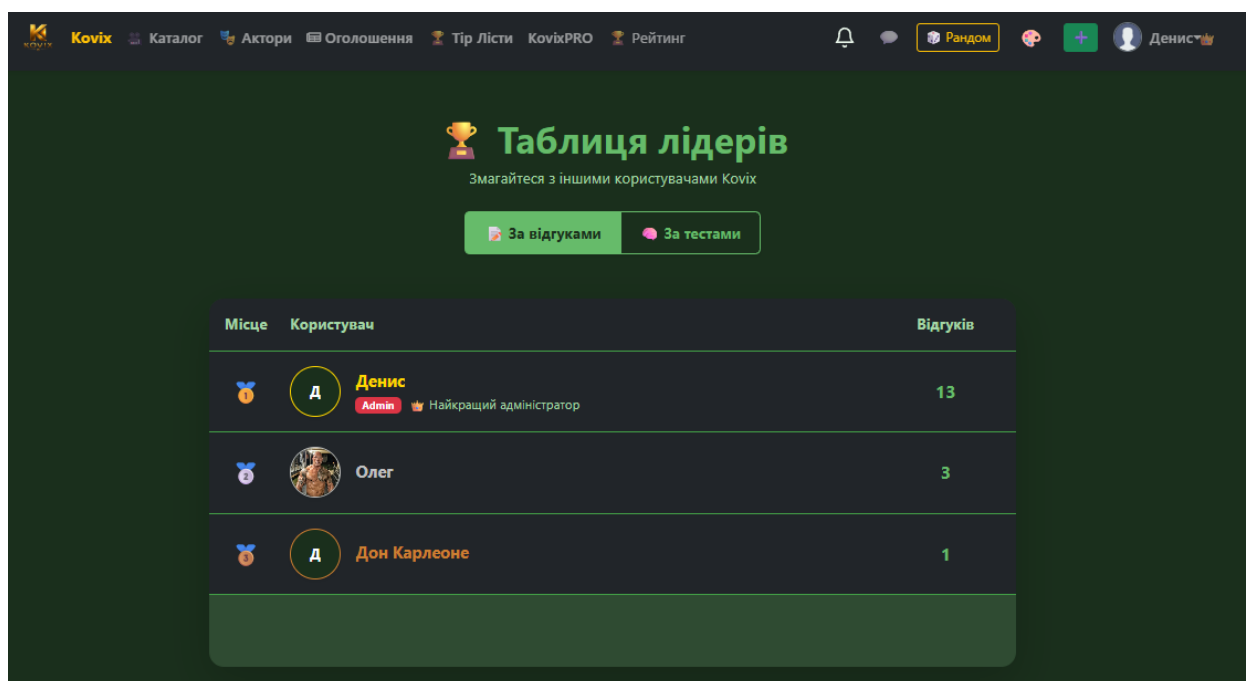


Рис.3.27. Таблиця лідерів за відгуками

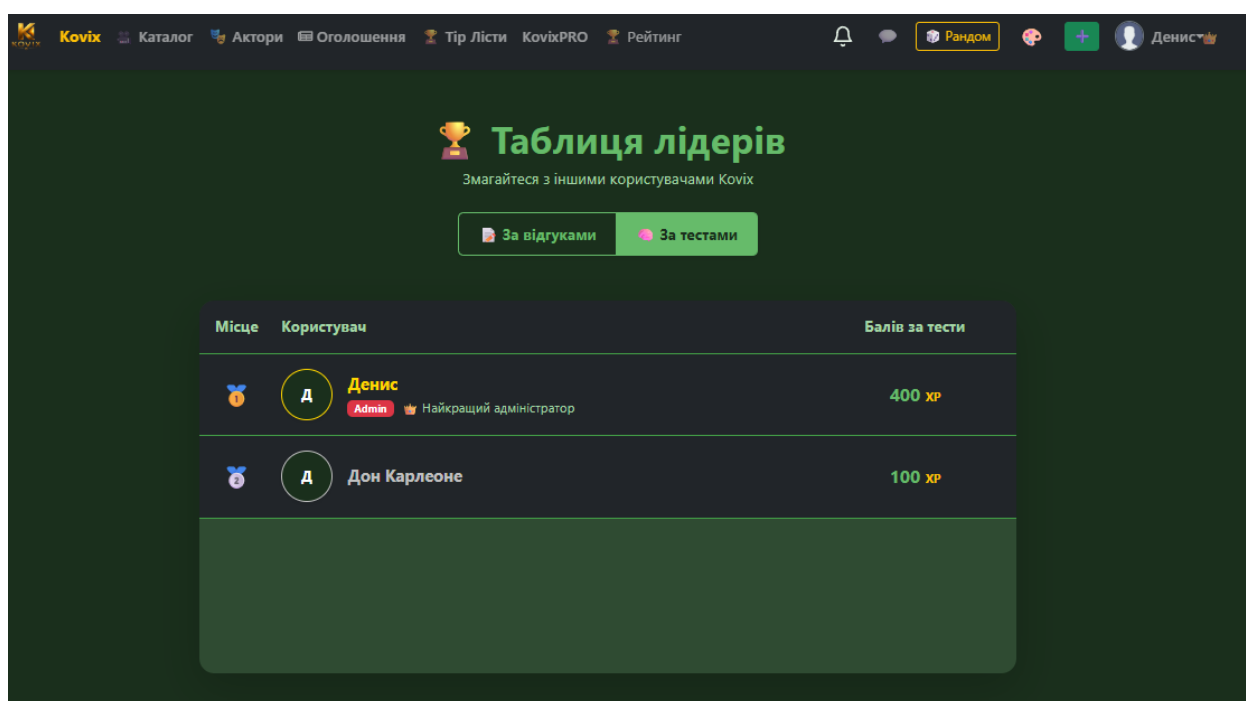


Рис.3.28. Таблиця лідерів за тестами

Також на сайті було реалізовано логіку сповіщення (рис.3.29), якщо користувачу відправили запит в друзі чи він отримав скаргу або блокування система сповістить його про це

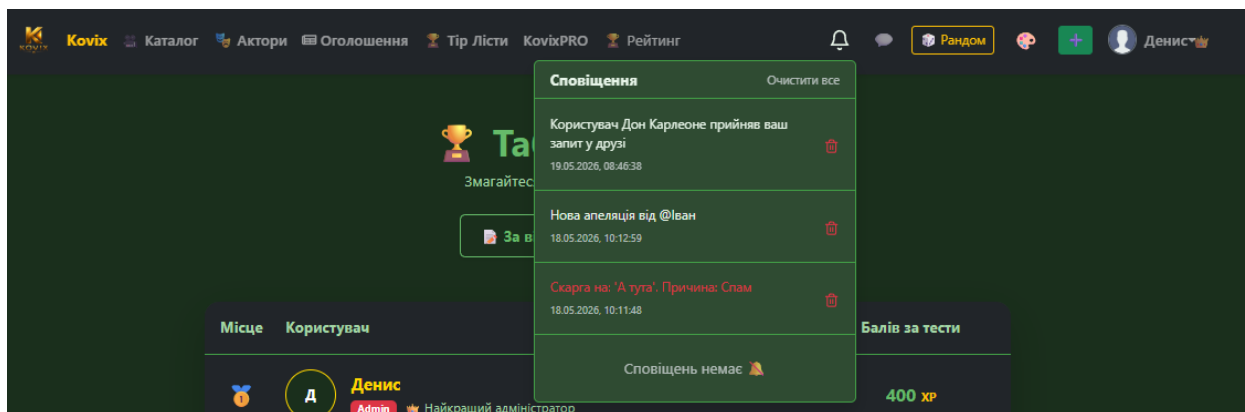


Рис.3.29. Система сповіщення

Наступним етапом будуть чати, в цьому блоці користувачі можуть переписуватися в загальному чи приватних чатах (рис.3.30), також було реалізовано систему закріплених повідомлень для себе та всіх, в загальному чаті це може робити тільки адміністратор чи модератор, а в приватних повідомленнях це можуть робити всі (рис.3.31), також на повідомлення можна кинути скаргу (рис.3.32) яка потім відправиться модератору на розгляд (рис.3.33).

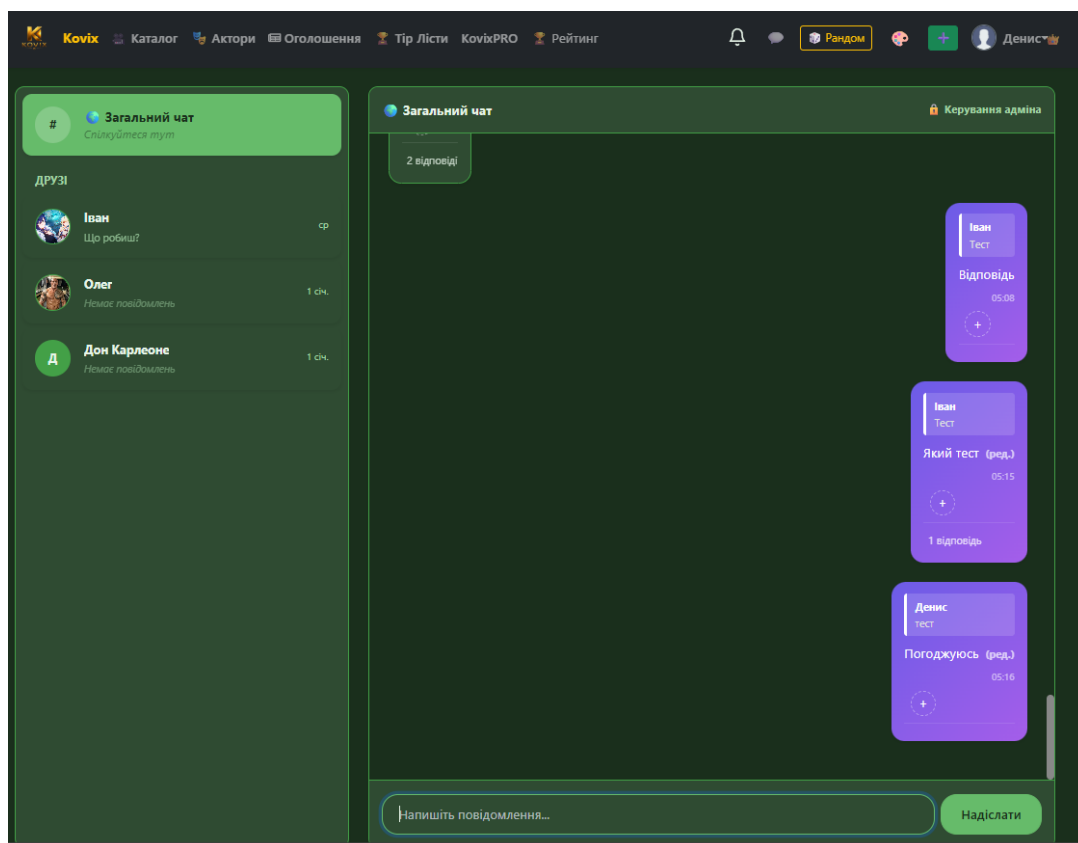


Рис.3.30. Приватні та загальні

		Денисенко Д.О.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Сфремов Ю.М.				79
Змн.	Арк.	№ докум.	Підпис	Дата		

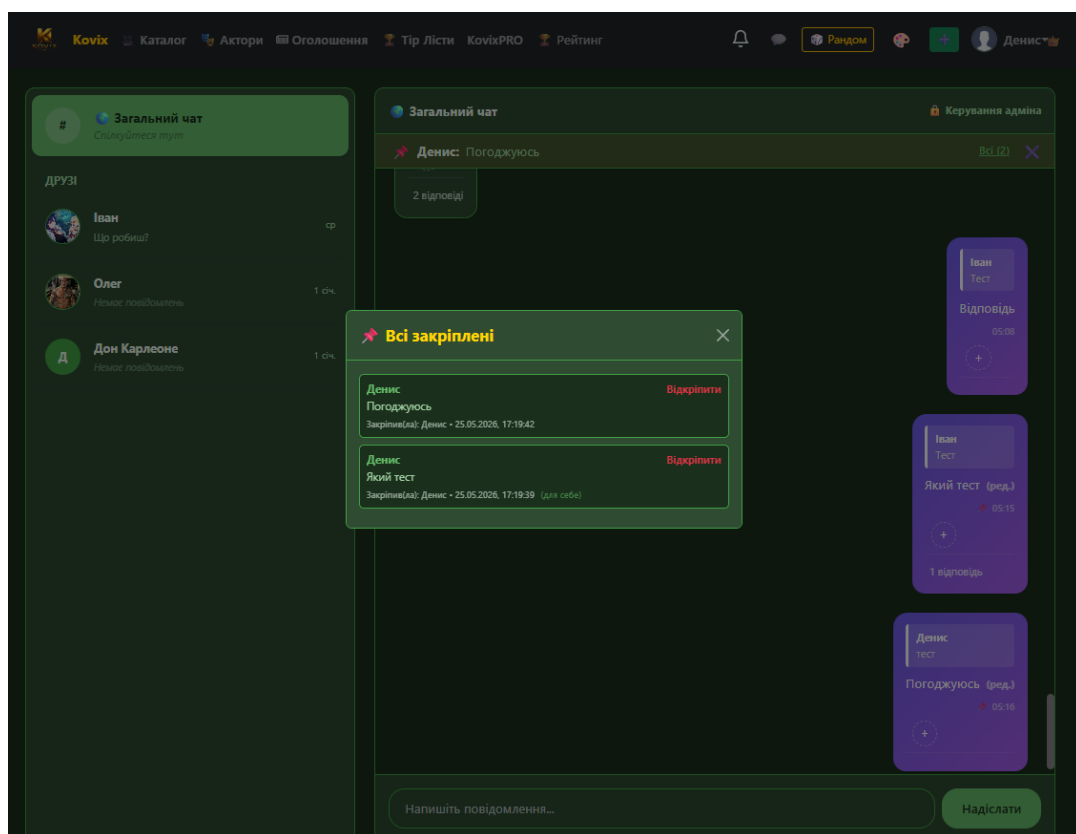


Рис.3.31. Закріплення та перегляд закріплених повідомлень

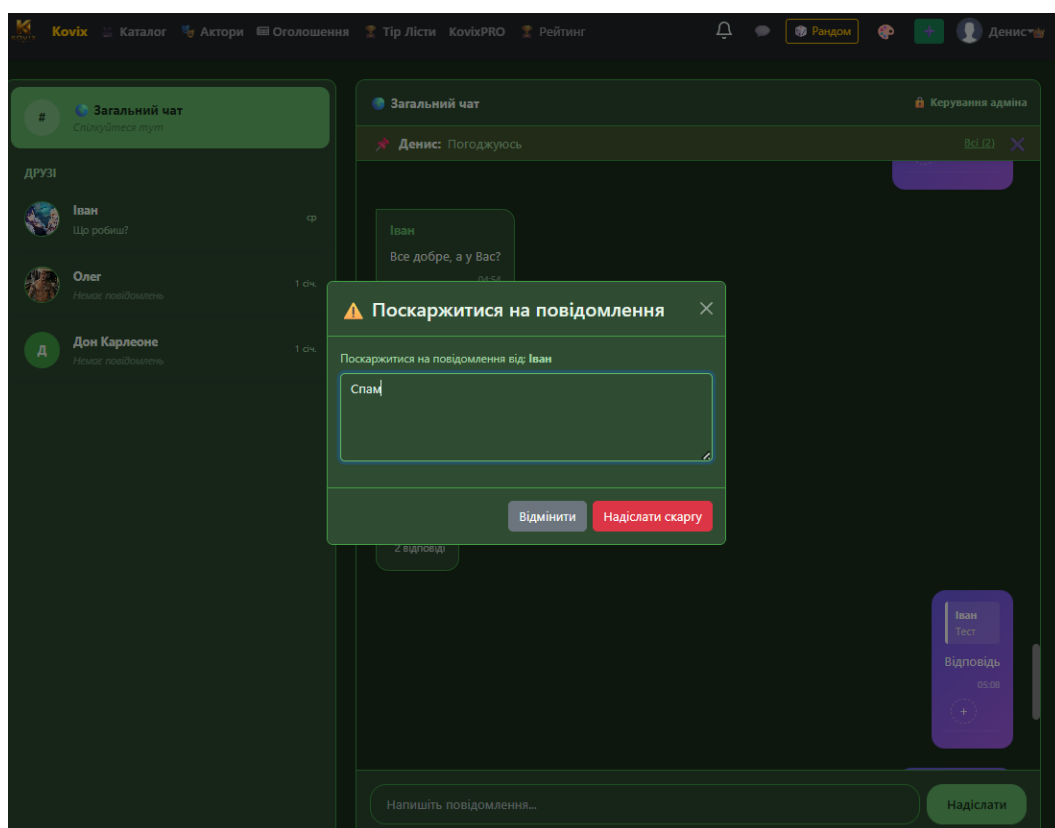


Рис.3.32. Скарги на користувачів

		Денисенко Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремов Ю.М.				80
Змн.	Арк.	№ докум.	Підпис	Дата		

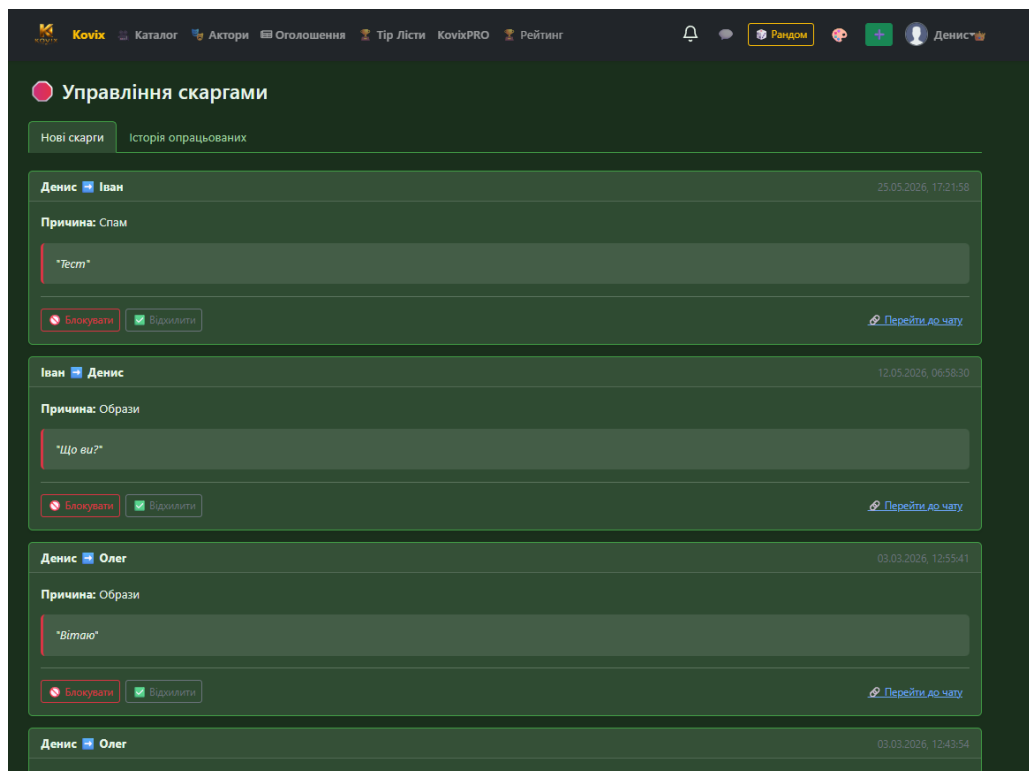


Рис.3.33. Перегляд скарги зі сторони адміністрації

Доповненням до попереднього блоку буде “Історія опрацювань” скарг з сторони адміністрації (рис.3.34), тут зберігається вся історія скарг.

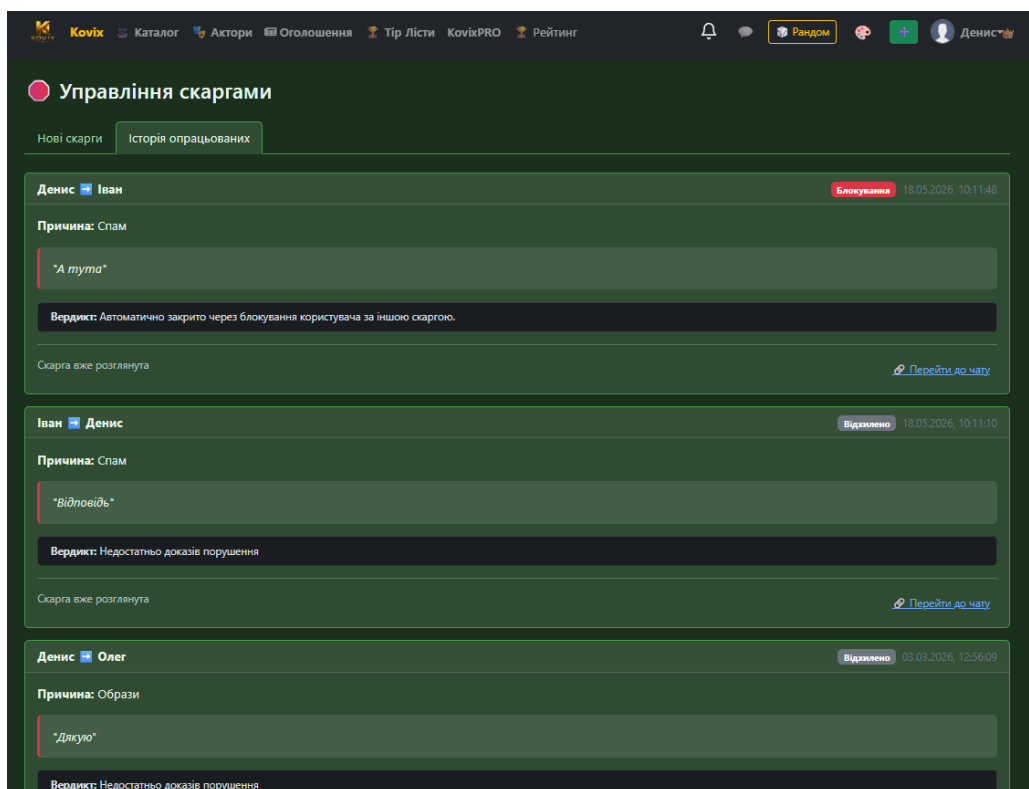


Рис.3.34. Історія опрацьованих скарг зі сторони адміністрації

		Денисенко Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремов Ю.М.				81
Змн.	Арк.	№ докум.	Підпис	Дата		

Найголовнішим етапом розробки був саме цей етап, адже він розбився на декілька модулів, які поєднуються в один механізм саме тут. Почати потрібно з сторінки фільму (рис.3.35), для адміністратора відразу над фільмом знаходиться “Адмін-панель”, яка допомагає йому виконувати базові CRUD операції. Далі можна побачити кількість переглядів фільму, він вираховується не просто клікнувши по проєкту, а перегляд зарахується тільки в тому випадку, якщо користувач побуде на сторінці фільма 5 секунд і більше не нарахується, адже відбувається перевірка на сесію та кукі. Потім видно рейтинг, який вираховується середнім значенням всіх коментарів та рецензій, та жанри які відповідають цьому фільму. Після короткого опису фільму йде трейлер, який відразу стоїть на “вбудованому”, але його можна змінити на кастомний.

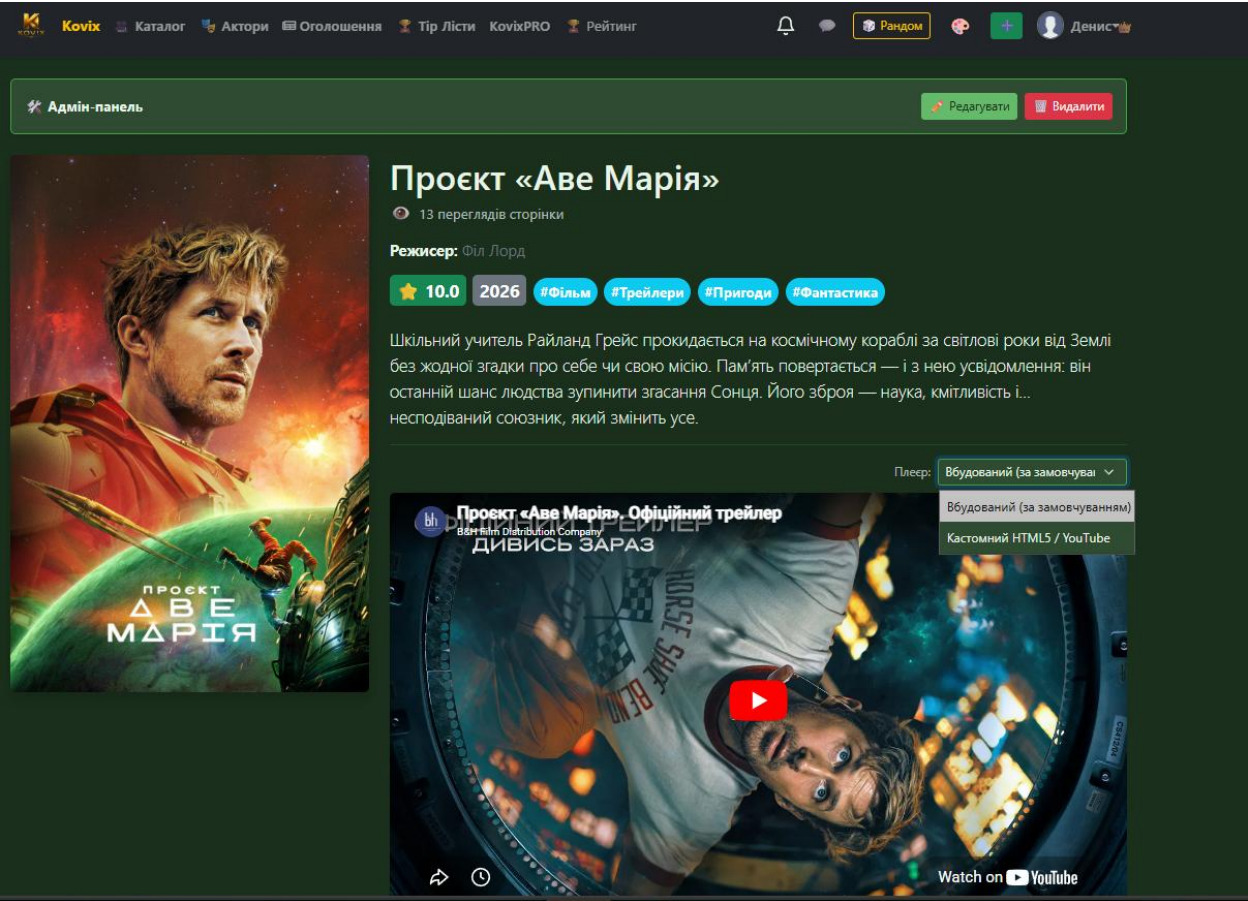


Рис.3.35. Сторінка фільму

Після трейлера йде модуль “Тест на знання” тут користувач може випробувати свої сили та пройти короткий тест (рис.3.36), після проходження тест відобразиться у профілі, та результат занесеться до таблиці лідерів.

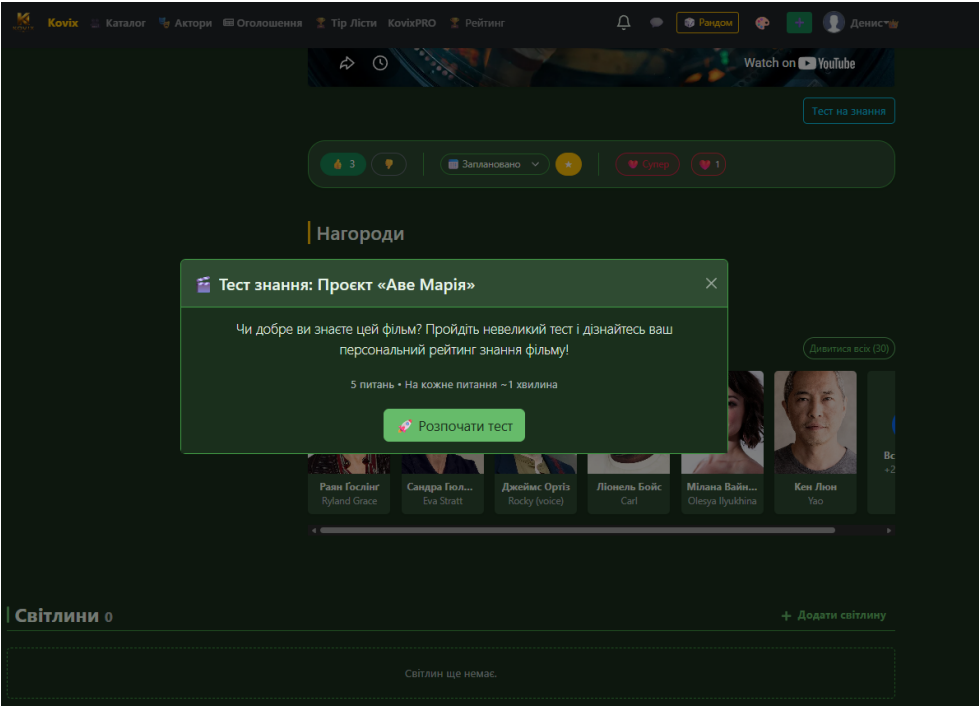


Рис.3.36. Тестування

Якщо фільм дуже сподобався аудиторії, введено функцію нагородження фільму, якоюсь нагородою по якій потім можна буде фільтрувати фільми (рис.3.37).

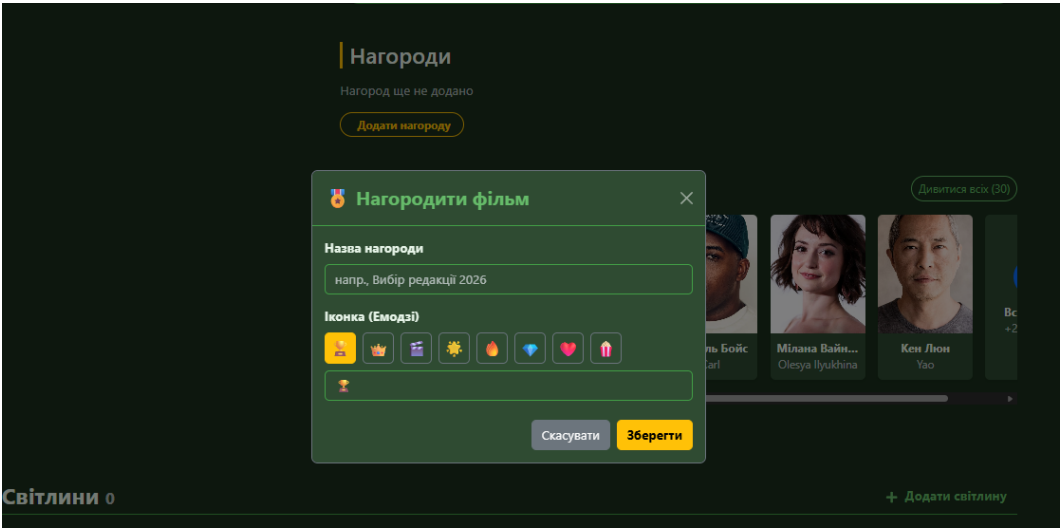


Рис.3.37. Нагородження

Потім йде блок з “Акторський склад” якщо перейти по кнопці “Дивитися всіх” буде відправлено на сторінку з героями фільму (рис.3.38)

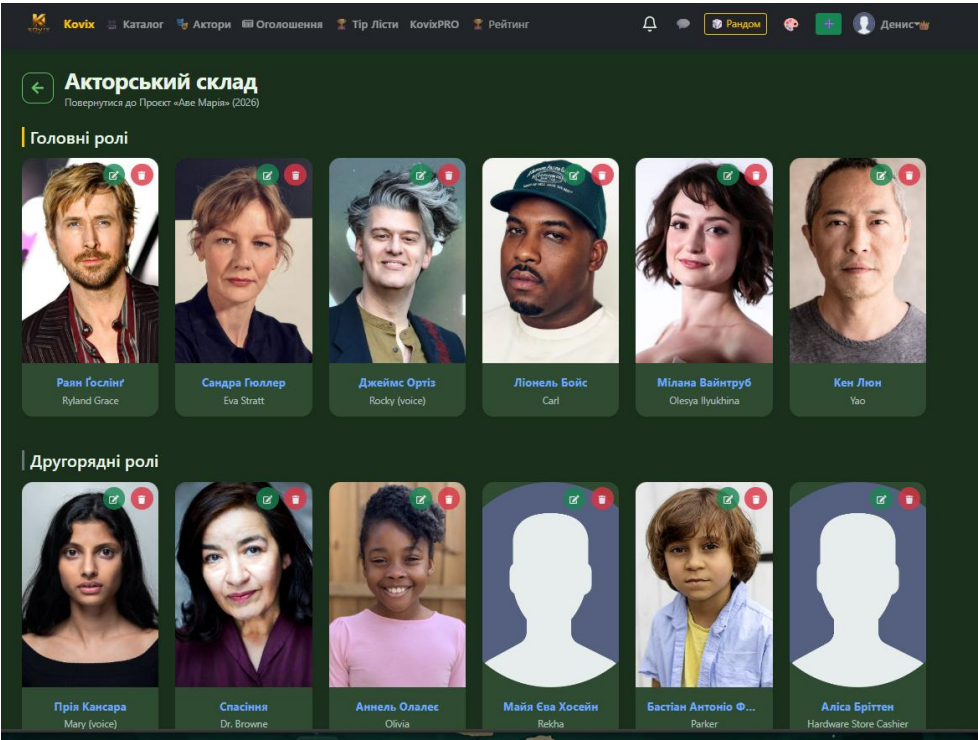


Рис.3.38. Акторський склад

Далі йде блок з “Світлинами”, тут адміністратор для кращого розуміння додає фото, якщо користувач не зрозумів по опису, трейлеру про що фільм чи серіал (рис.3.39), також світлини можна розкрити та подивитися більш детальніше клікнувши по фото

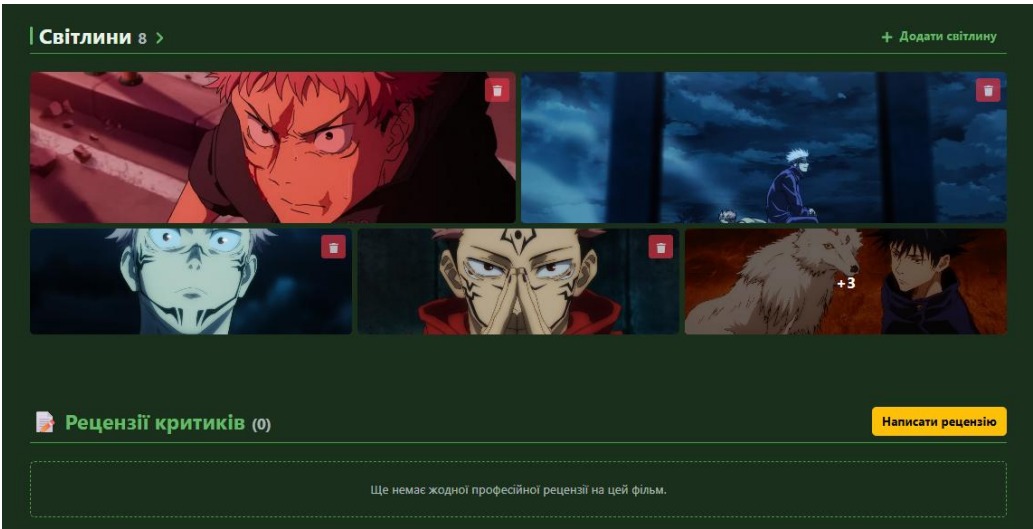


Рис.3.39. Світлини фільму

Після цього йде блок з “Рецензіями” та “Коментарі”, в цьому блоці можуть писати лише користувачі з відповідною роллю “Критик” та вся адміністрація (рис.3.40). Для меншого навантаження на користувача було винесено всі рецензії, які перевищують максимум (2 рецензії) для 1 фільму, всі інші занесені на іншу сторінку, якщо натиснути на кнопку “Всі рецензії” (рис.3.41).

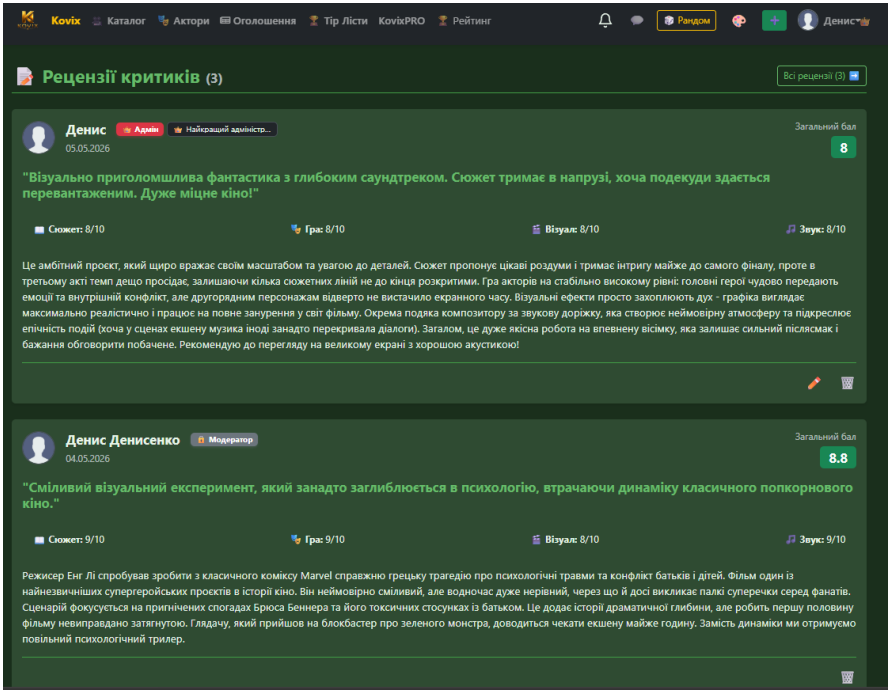


Рис.3.40. Рецензії на фільм

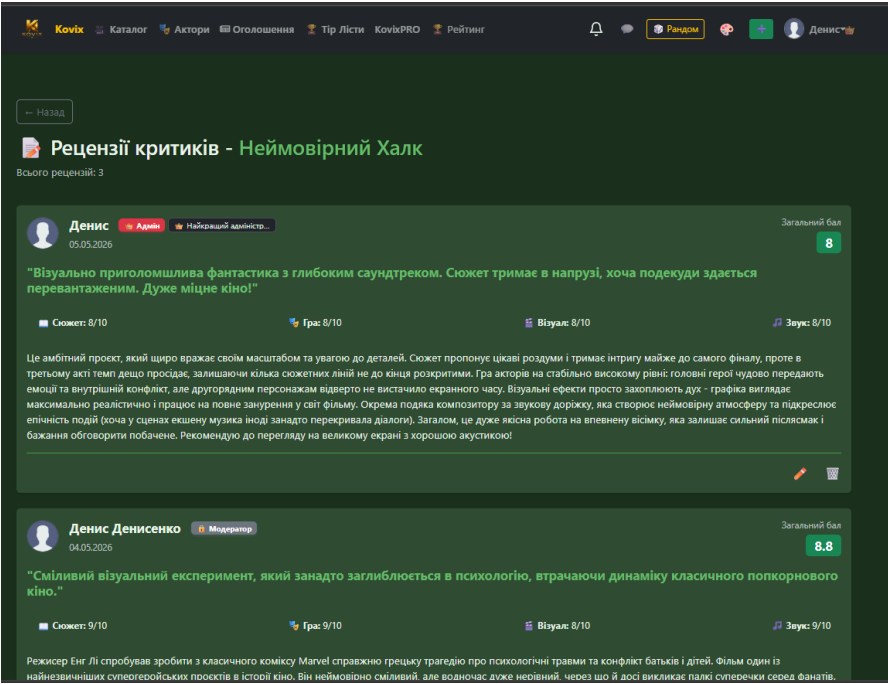


Рис.3.41. Всі рецензії фільму

Після рецензій йдуть вже звичайні коментарі, від користувачів, вони також впливають на фільм, але менше ніж рецензії, тут теж реалізовано базовий CRUD та систему реакцій.

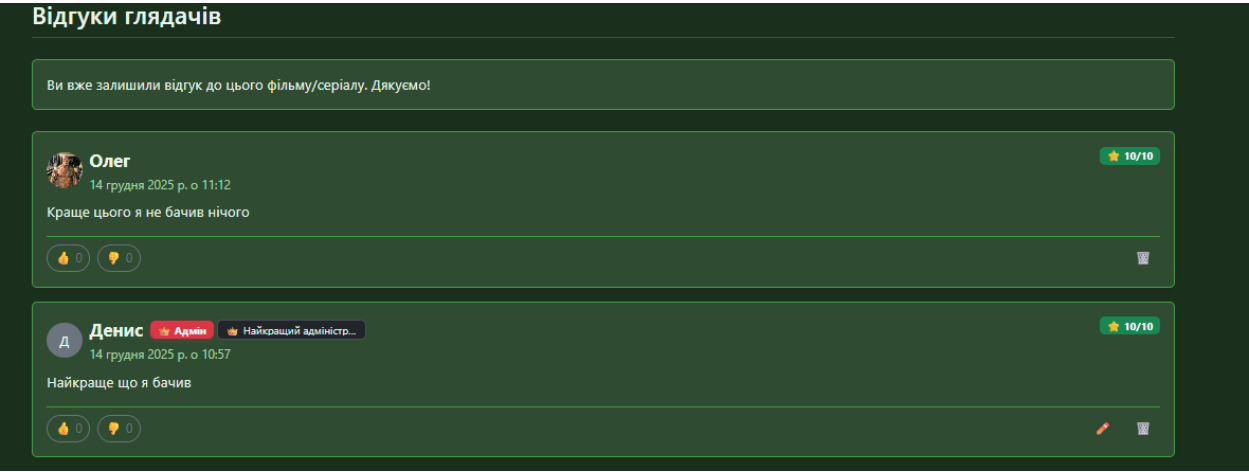


Рис.3.41. Коментарі до фільму

Ще було розроблено систему реакцій та списків під фільмом (рис.3.42). Користувач може ставити реакції на фільм, вони створюють топи в блоці “Найпопулярніші трейлера”, також користувач може додати фільм в список або додати його в улюблене, це виодиться у власну статистику у профілі, та на окрему сторінку на сайті (рис.3.43).

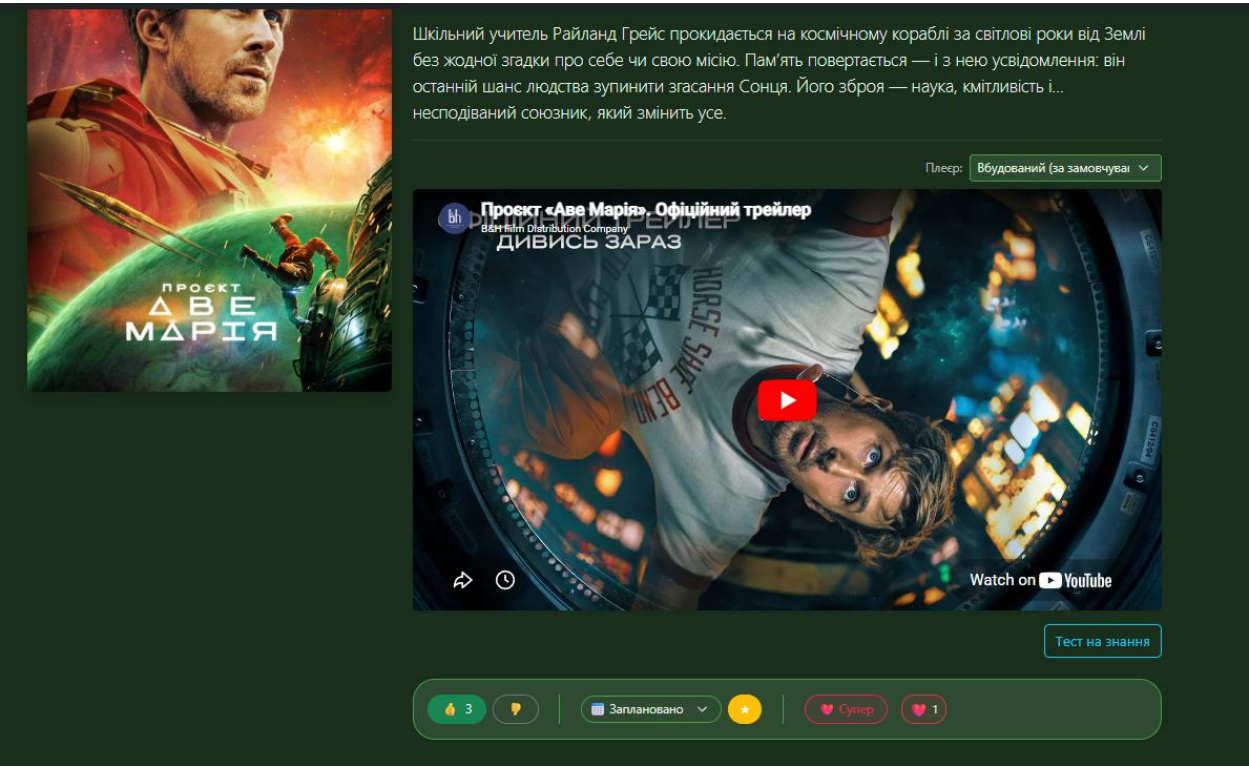


Рис.3.42. Реакції та списки фільму

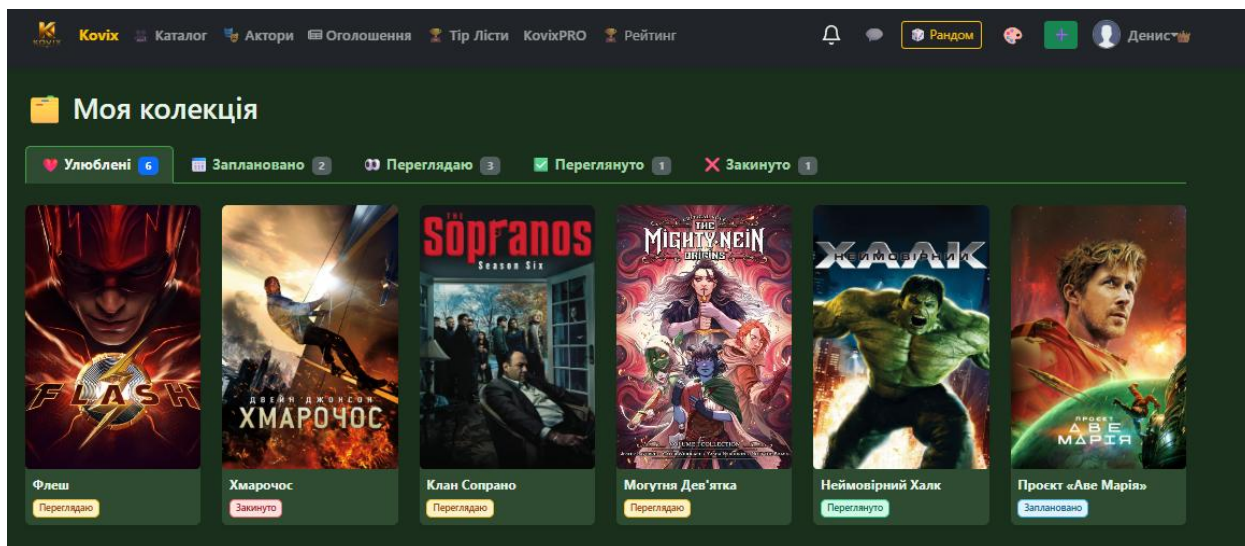


Рис.3.43.Списки користувача

І зафіналізувати етап з фільмами потрібно блоком “Схожі проєкти”, які формуються по жанрам фільму, який переглядає користувач на даний момент (рис.3.44)

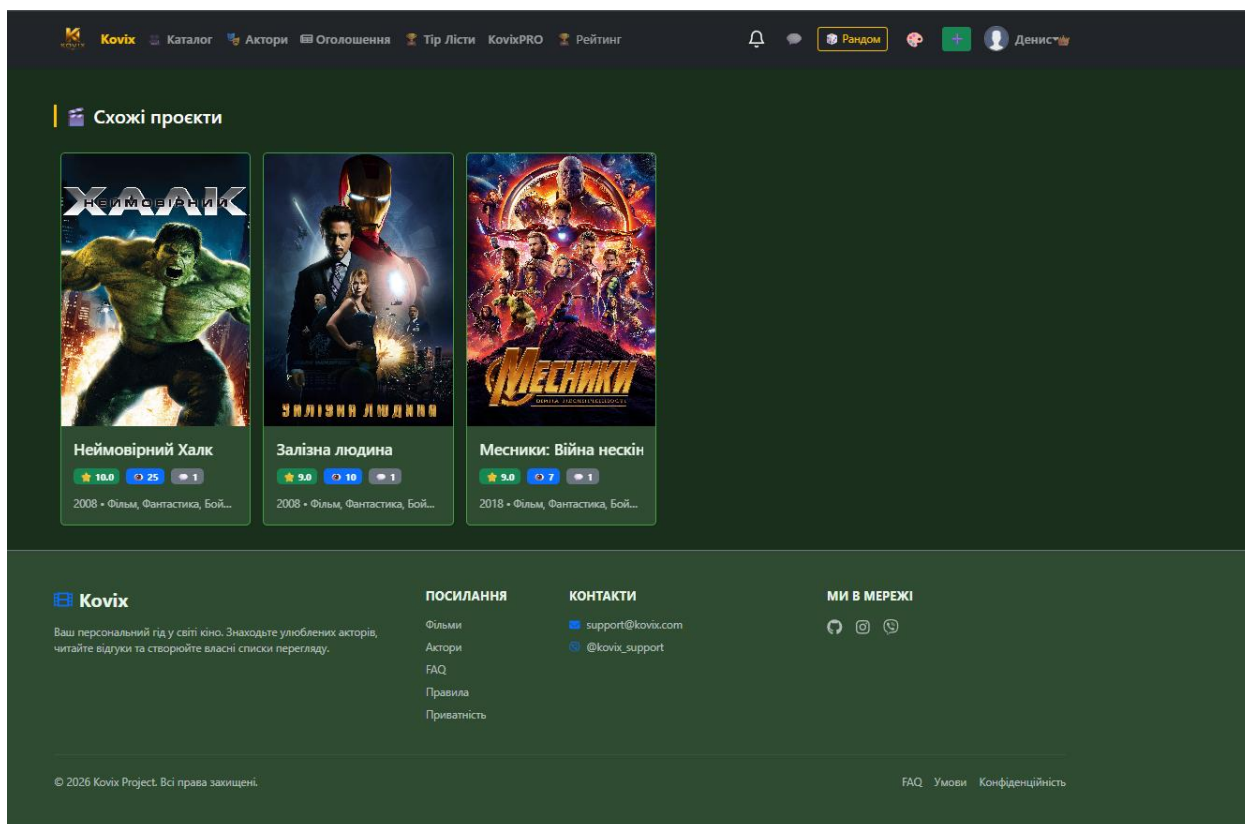


Рис.3.44.Схожі проєкти

Наступним етапом будуть профілі користувачів, зайшовши на профіль відразу видно статистику по списках користувача, його аватар, ім'я, пошта,

		Денисенко Д.О.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Сфремов Ю.М.				87
Змн.	Арк.	№ докум.	Підпис	Дата		

бейдж та його роль (рис.3.45), статистика користувача формується по кількості фільмів чи серіалів в “Моїх списках”.

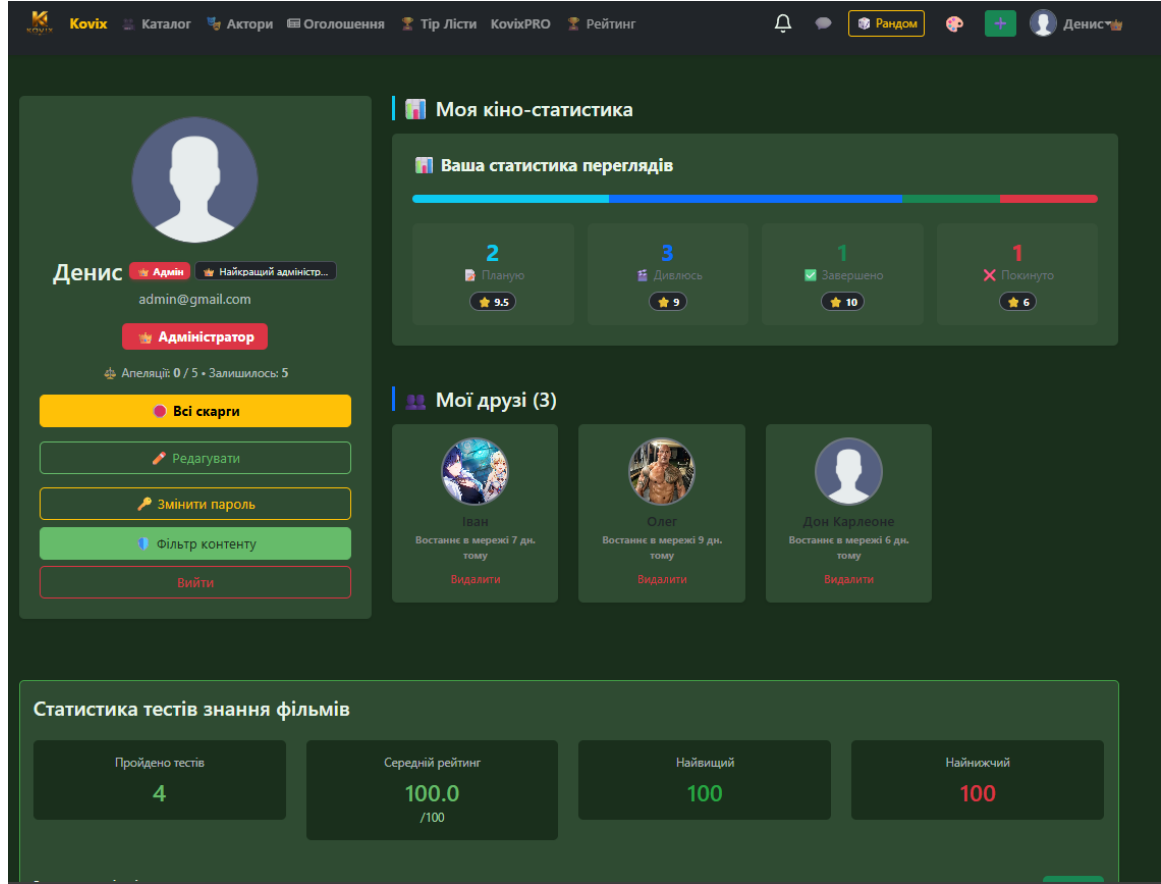


Рис.3.45.Профіль користувача

Потім йде блок “Друзі”, користувач може перейти на публічний профіль користувача (рис.3.46), де він може переглянути коли користувач востанє був в мережі, це допомагає зрозуміти чи він активний, також на нього можна підписатися, та додати в друзі, якщо користувач відправить запит в друзі іншому прийде повідомлення в дзвоник, після натиснення на сповіщення перекине публічну сторінку, де можна переглянути “Досягнення користувача” та всі його відгуки, щоб не навантажувати користувача зроблено пошук, фільтрацію та підгруження відгуків (рис.3.47), також у користувача є можливість відновити чи змінити пароль, якщо він забув його, ще є можливість невідображати відповідний жанр за допомогою “Фільтр контенту”[21].

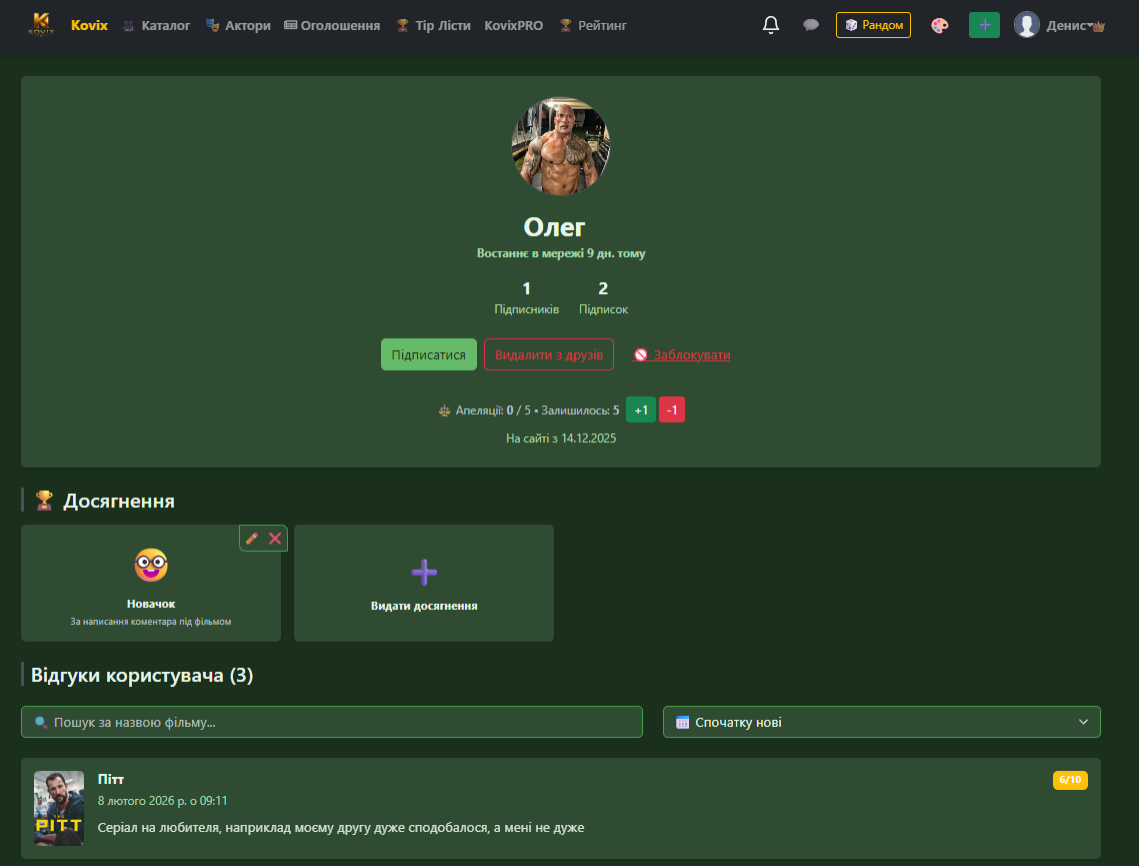


Рис.3.46. Публічна сторінка користувача

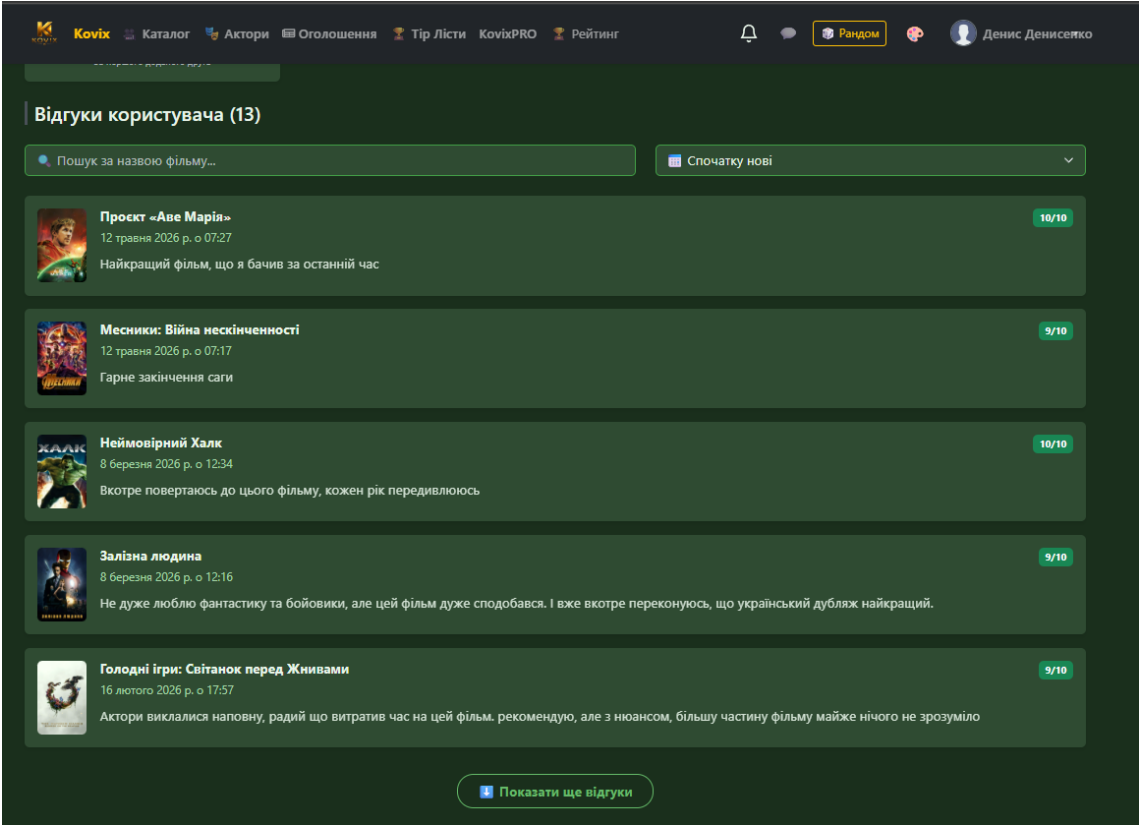


Рис.3.47. Відгуки користувача

		Денисенко Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремов Ю.М.				89
Змн.	Арк.	№ докум.	Підпис	Дата		

Далі йде блок “Статистика тестів” та “Досягнення”, про статистику більше детальніше буде описано в наступному підпункті, досягнення користувача зроблені в 2 форматах, перші це автоматичні які видаються, якщо користувач виконує відповідні вимоги, другий тип це ті які адміністратор видає кастомно через відповідні заслуги (рис.3.48), отримання автоматичного досягнення впливає кастомне повідомлення (рис.3.49) і відповідне повідомлення в дзвоник, також ці “Досягнення” тут потрібні не тільки для відображення в блоці на профілі, ці досягнення можна ставити собі біля імені у вигляді бейджі, які будуть відображатися повсюди де є користувач.

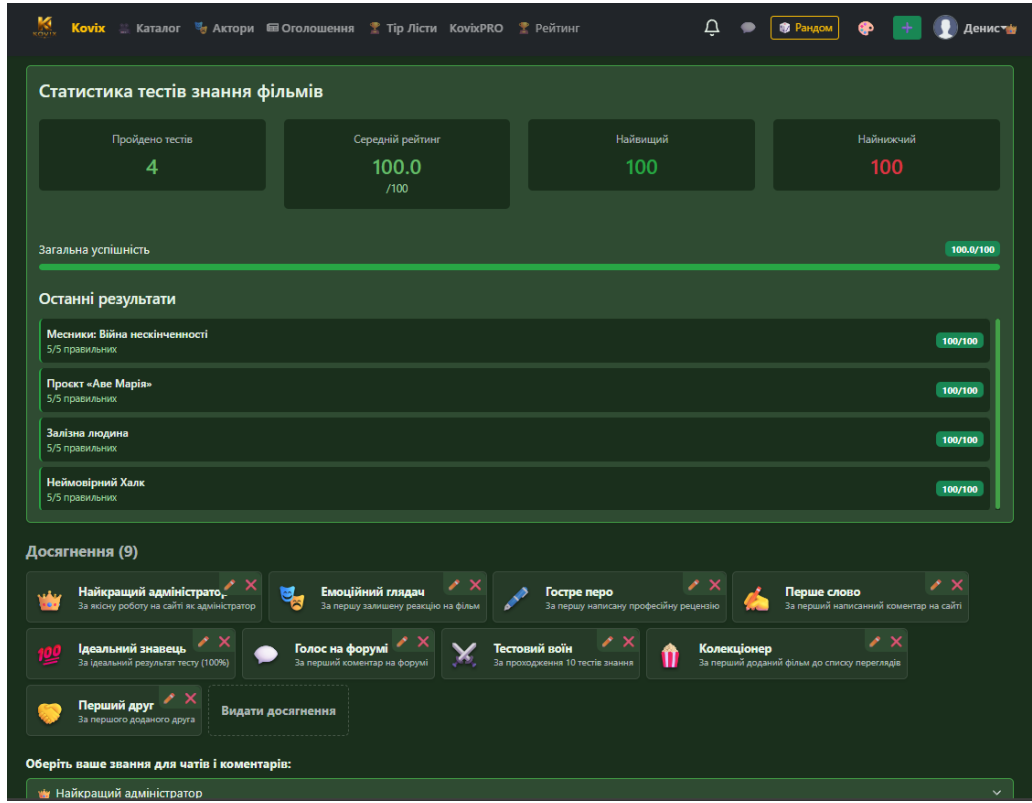


Рис.3.48. Досягнення користувача

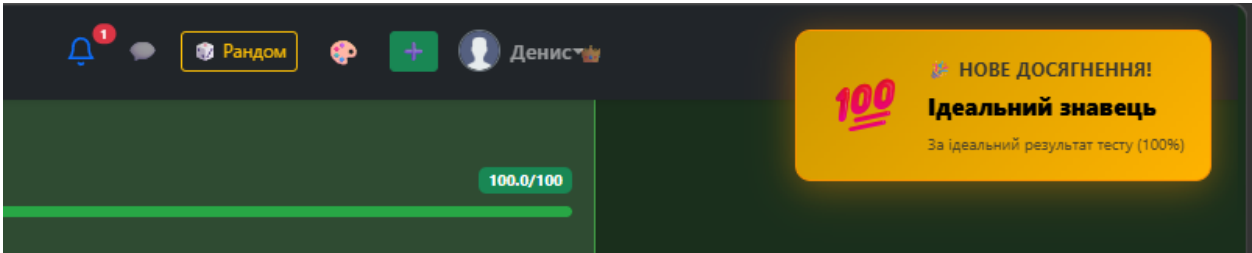


Рис.3.49. Отримання досягнення

		Денисенко Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремов Ю.М.				90
Змн.	Арк.	№ докум.	Підпис	Дата		

Наступним етапом буде блок “Історія переглядів” в її також можна змінювати відображення, вона формується якщо користувач 3 секунди побуде на сторінці фільму, також тут є швидкий перехід до каталогу.

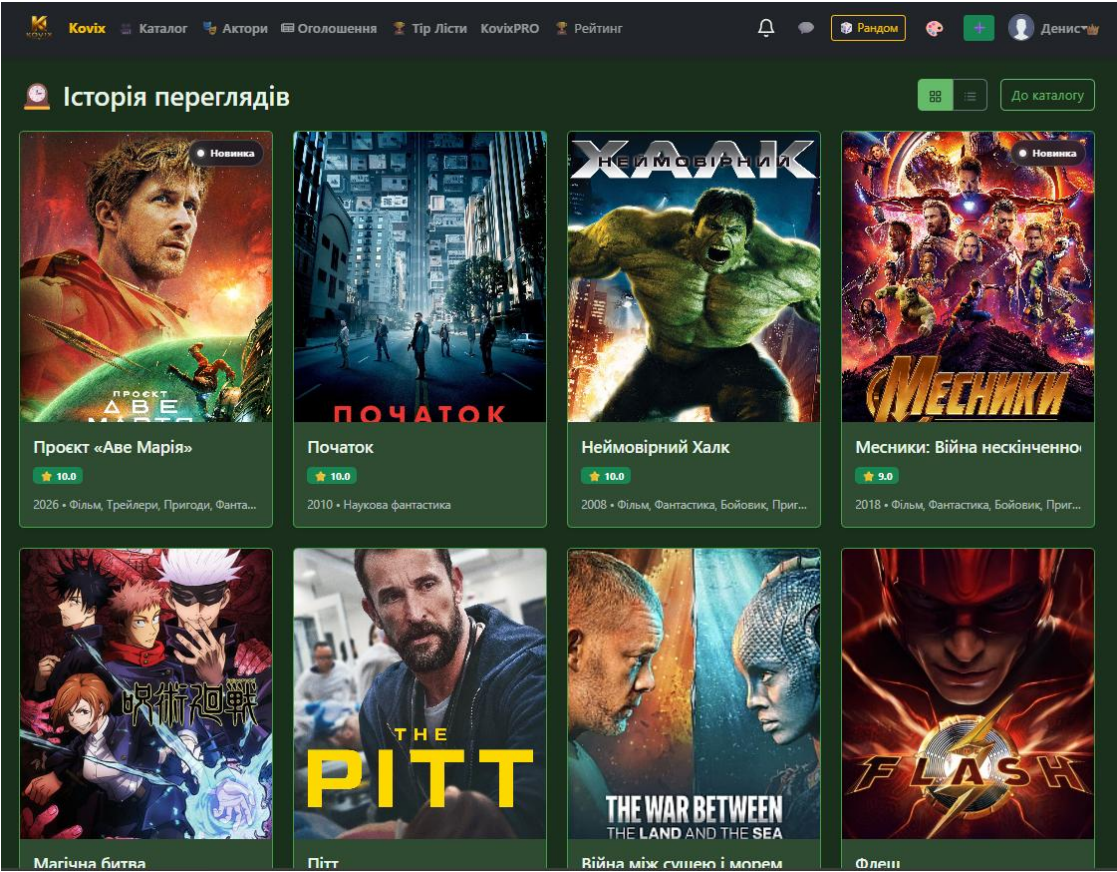


Рис.3.50. Історія переглядів

Наступним етап будуть “Новини”, до них можна отримати доступ з головної сторінки та з випадваючого списку, далі з цим блоком також реалізовано всі CRUD операції, також їх можна налаштовувати та переглядати всі “Новини”, цей етап розроблявся як блог новин тут можна буде прописувати фільми чи серіали які додадуться в майбутньому чи якась додаткова інформація щодо сайту, також було реалізовано логіку “Чернетки”, автор новини може написати новину, але не ставити галочку “Опублікувати”, поки автор не опублікує її на загал.

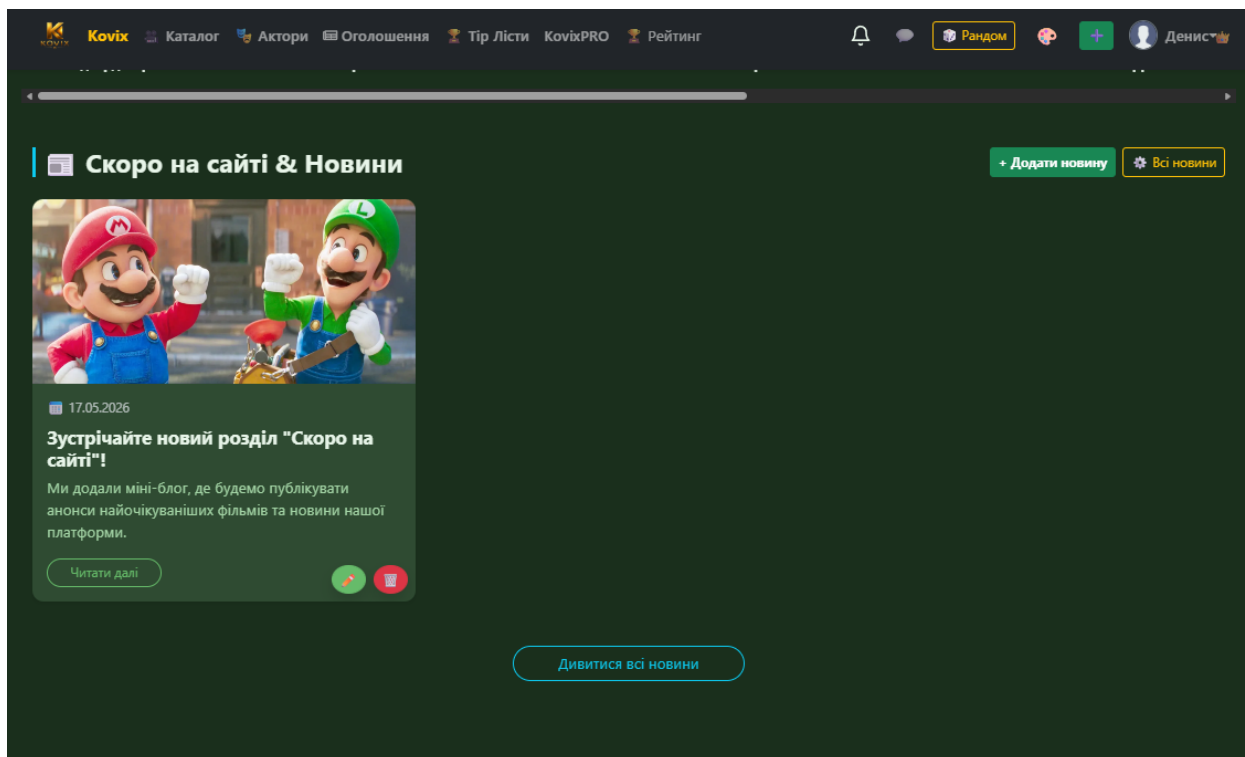


Рис.3.51. Новини сайту

Далі йде блок “Форум”, тут все починається з категорій, для адміністратора (рис.3.52), де реалізований весь CRUD, також для звичайних користувачів додано можливість пропонування категорій (рис.3.53), яка потім відправиться адміністрації на погодження.

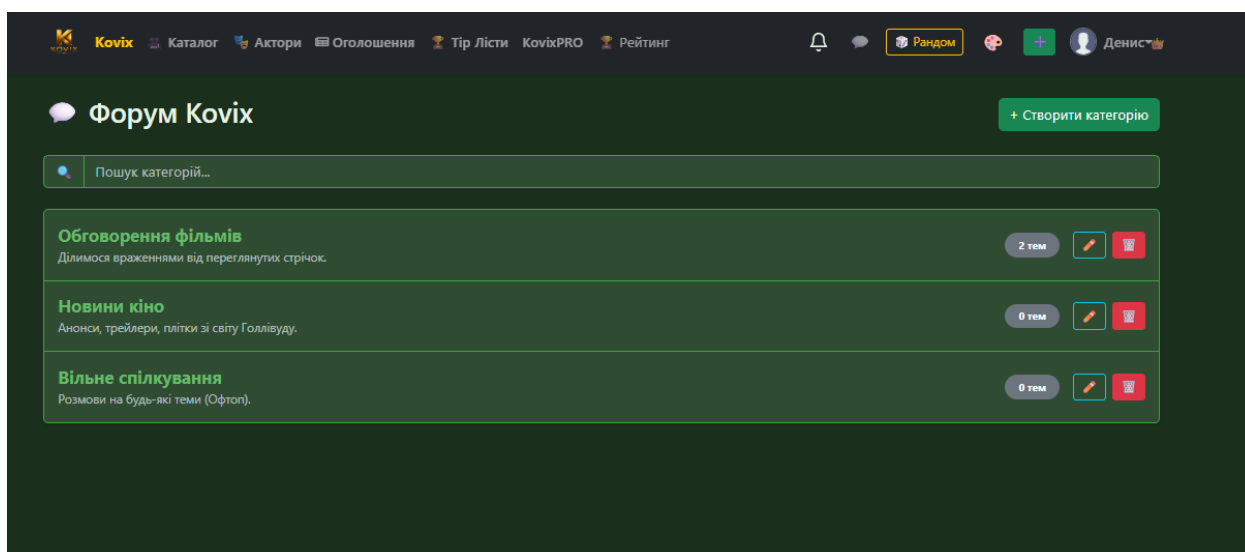


Рис.3.52. Категорії форуму для адміністратора

		Денисенко Д.О.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Сфремов Ю.М.				92
Змн.	Арк.	№ докум.	Підпис	Дата		

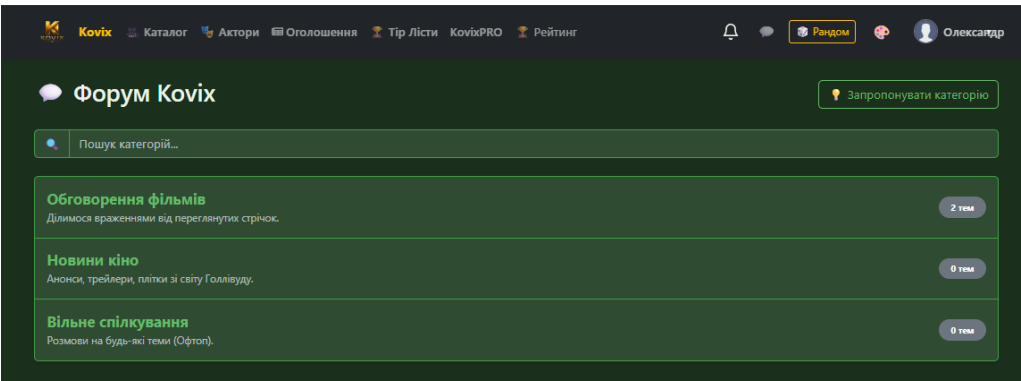


Рис.3.53. Категорії форуму для користувача

Якщо зайти в категорію можна шукати будь яку тему яка цікавить користувача та сортувати за різними категоріями (рис.3.54).

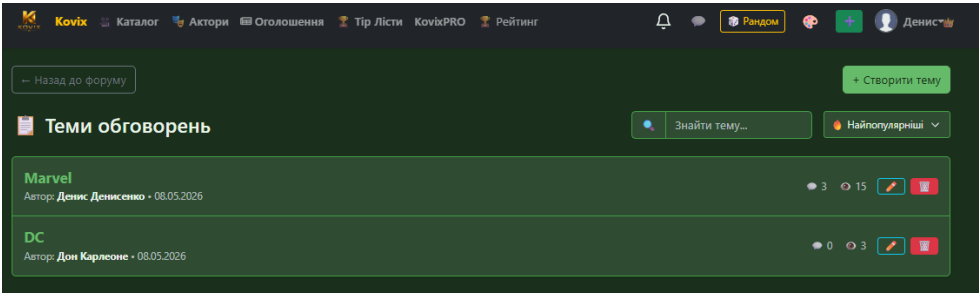


Рис.3.54. Теми форуму

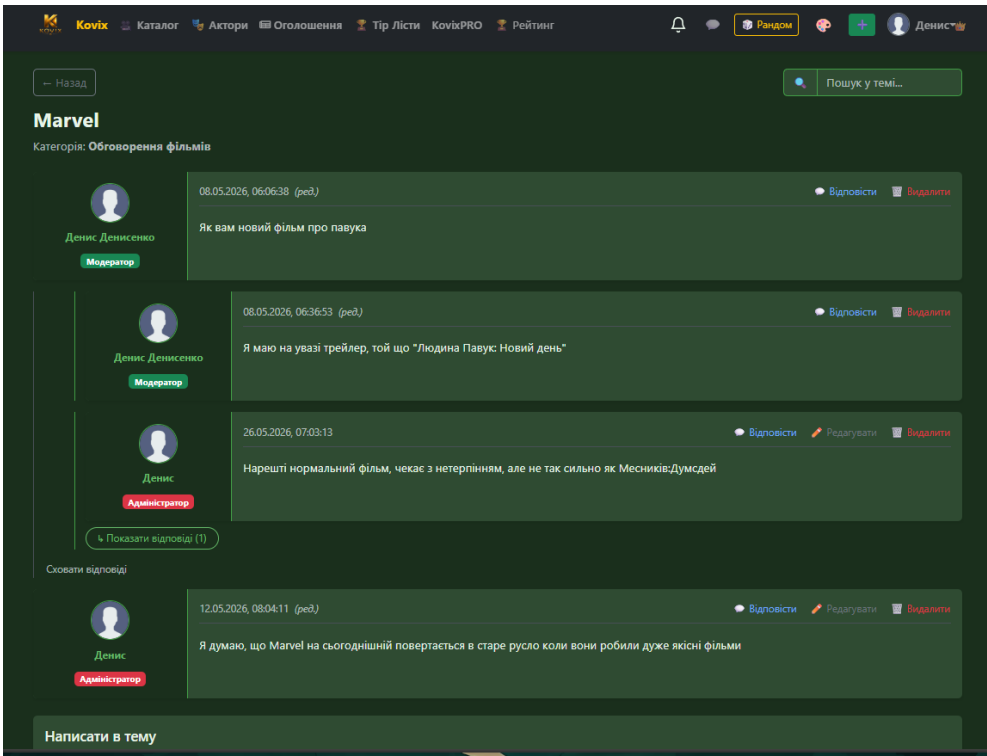


Рис.3.55. Обговорення форуму

Доповненням до теми формуму, буде етап обговорення тут також реалізовано базовий CRUD, пошук, вкладеність повідомлень та відповіді на повідомлення (рис.3.56).

Далі йде блок “Топ-100” тут формується список найкращих фільмів на сайті, тут також можна змінити відображення списку, рейтинг формується за рейтингом та кількістю коментарів під фільмом.

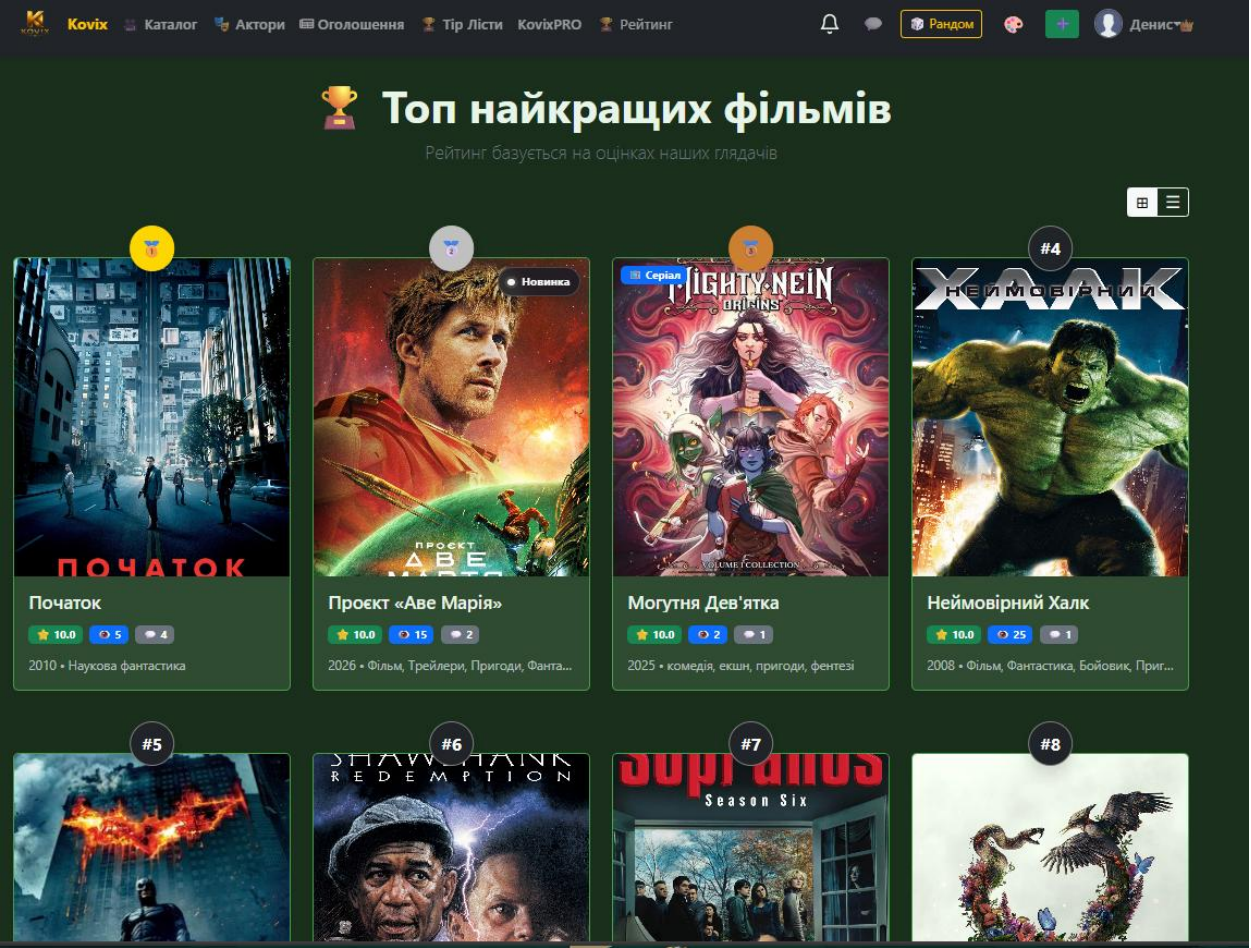


Рис.3.56. Топ-100

Наступним блоком буде подача заявки на становлення критиком, на профілі користувача, якщо в нього немає ролі “Критик” чи адміністрації він може подати заявку (рис.3.57), далі ця заявка відправляється на перевірку та погодження, де адміністратор переглядає мотивацію, переглядає профіль користувача та приймає рішення (рис.3.58).

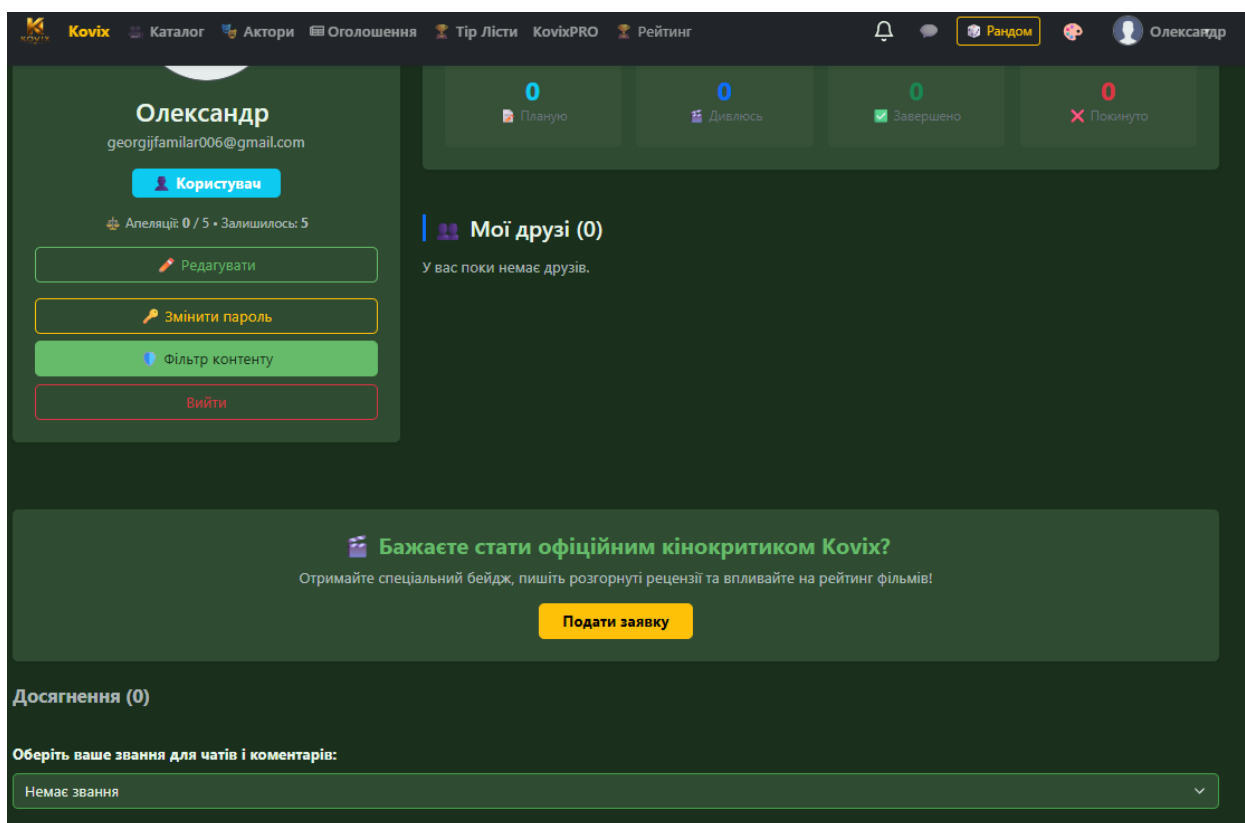


Рис.3.57. Подача заявки на сторінці користувача

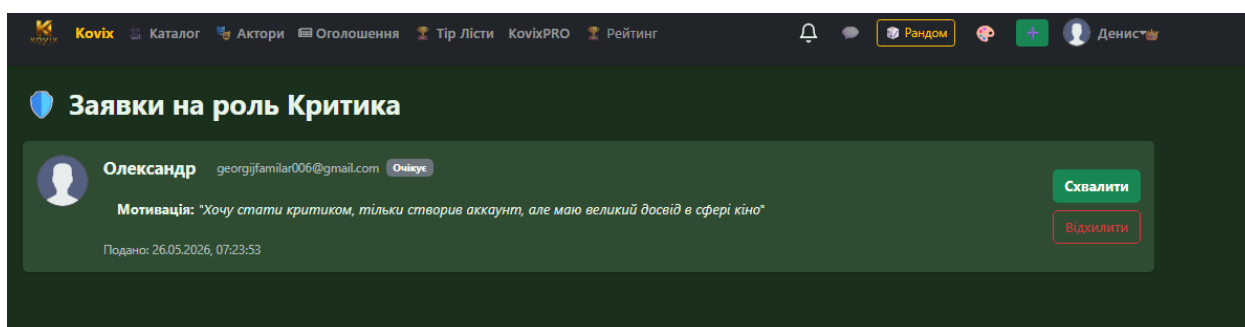


Рис.3.58. Заявки на сторінці адміністрації

Наступним етапом буде “Керування користувачами та ролями”, ця сторінка дуже спрощує роботу з користувачами, якщо адміністратор захоче змінити чи додати якусь роль він з легкістю може це зробити. Після зміни ролі також змінюється чи додається бейдж з його роллю (рис.3.59), також це спрощує пошук відповідного користувача, адміністратор може зайти на цю сторінку і переглянути список всіх користувачів на сайті та детальну про них інформацію.

		Денисенко Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремов Ю.М.				95
Змн.	Арк.	№ докум.	Підпис	Дата		

ID	Користувач	Email	Поточна роль	Дія (Змінити роль)
#1013	Олександр	georgijfamiliar006@gmail.com	User	Глядач (User)
#1012	Дон Карлоне	negauttedine4003@gmail.com	Reviewer	Критик (Reviewer)
#1005	Денис Денисенко	denisenkodenis094@gmail.com	Moderator	Модератор (Moderator)
#1004	Іван	ipz225_ddo@student-ztu.edu.ua	User	Глядач (User)
#4	Денис	admin@gmail.com	Admin	Адміністратор (Admin)
#3	Олег	ipz224_ddo@student-ztu.edu.ua	User	Глядач (User)

Рис.3.59. Керування ролями

Наступний етап це блокування та апеляції, якщо користувач буде заблокований йому на головній сторінці та у профілі буде про це повідомлено, також у нього буде заблокована можливість щось робити на сайті, його можливості прирівнюються гостю (рис.3.61), але якщо користувач був заблокований випадково він може подати “Апеляцію” (рис.3.60), але в користувача є обмежена кількість на апеляції в кількості 5, але якщо було випадково видано блокування, адміністрація завжди може на головній сторінці повернути кількість апеляцій, після успішної відправки адміністраторам, відбувається перевірка та прийняття рішення (рис.3.63), якщо користувацька апеляція була прийнята то користувач отримує повідомлення про успішне розблокування аккаунту, та може далі вести всі свої дії на сайті, як і раніше.

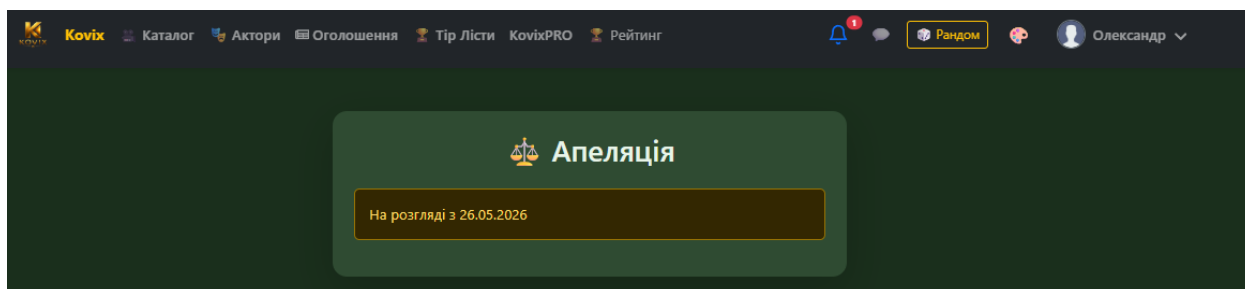


Рис.3.60 Відправка апеляції користувача

		Денисенко Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремов Ю.М.				96
Змн.	Арк.	№ докум.	Підпис	Дата		

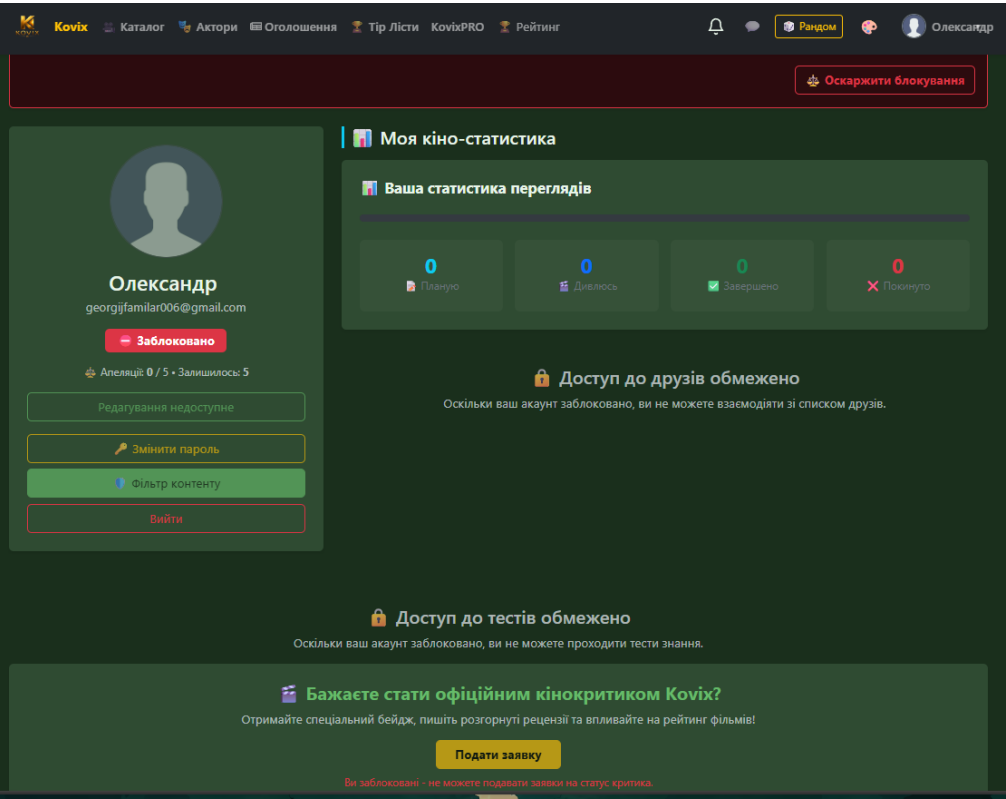


Рис.3.61. Блокування користувача

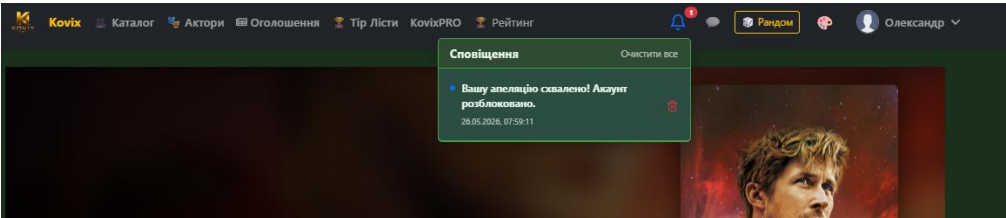


Рис.3.62. Сповіщення про прийняття апеляції

Історія апеляцій				
Користувач	Дата	Статус	Текст апеляції	Дії
@Олександр	26.05.2026	Очікує	Я не розумію, що я порушив, я нічого не писав і перед реєстрацією прочитав всі правила	<button>✓ Схвалити</button> <button>✗ Відхилити</button>
@Іван	18.05.2026	Схвалено	Мене випадково заблокували в загальному чаті	Опрацьовано
@Іван	23.03.2026	Схвалено	Я більше так не буду чесно	Опрацьовано
@Олег	03.03.2026	Схвалено	Я прошу вибачення був не правий	Опрацьовано
@Олег	03.03.2026	Схвалено	Я прошу вибачення більше такого не буде, був не правий	Опрацьовано
@Олег	03.03.2026	Відхилено	Я прошу вибачення був не правий	Опрацьовано
@Олег	03.03.2026	Схвалено	Я прошу вибачення був не правий	Опрацьовано
@Олег	03.03.2026	Схвалено	Я прошу вибачення був не правий	Опрацьовано
@Олег	03.03.2026	Схвалено	Я прошу вибачення був не правий	Опрацьовано
@Олег	03.03.2026	Схвалено	Я прошу вибачення був не правий	Опрацьовано
@Олег	03.03.2026	Відхилено	Я прошу вибачення був не правий	Опрацьовано

Рис.3.63. Сторінка апеляції для адміністратора

Також було реалізовано сторінку для зручного перегляду всіх своїх відгуків та рецензій на відповідні фільми та серіали (рис.3.64)

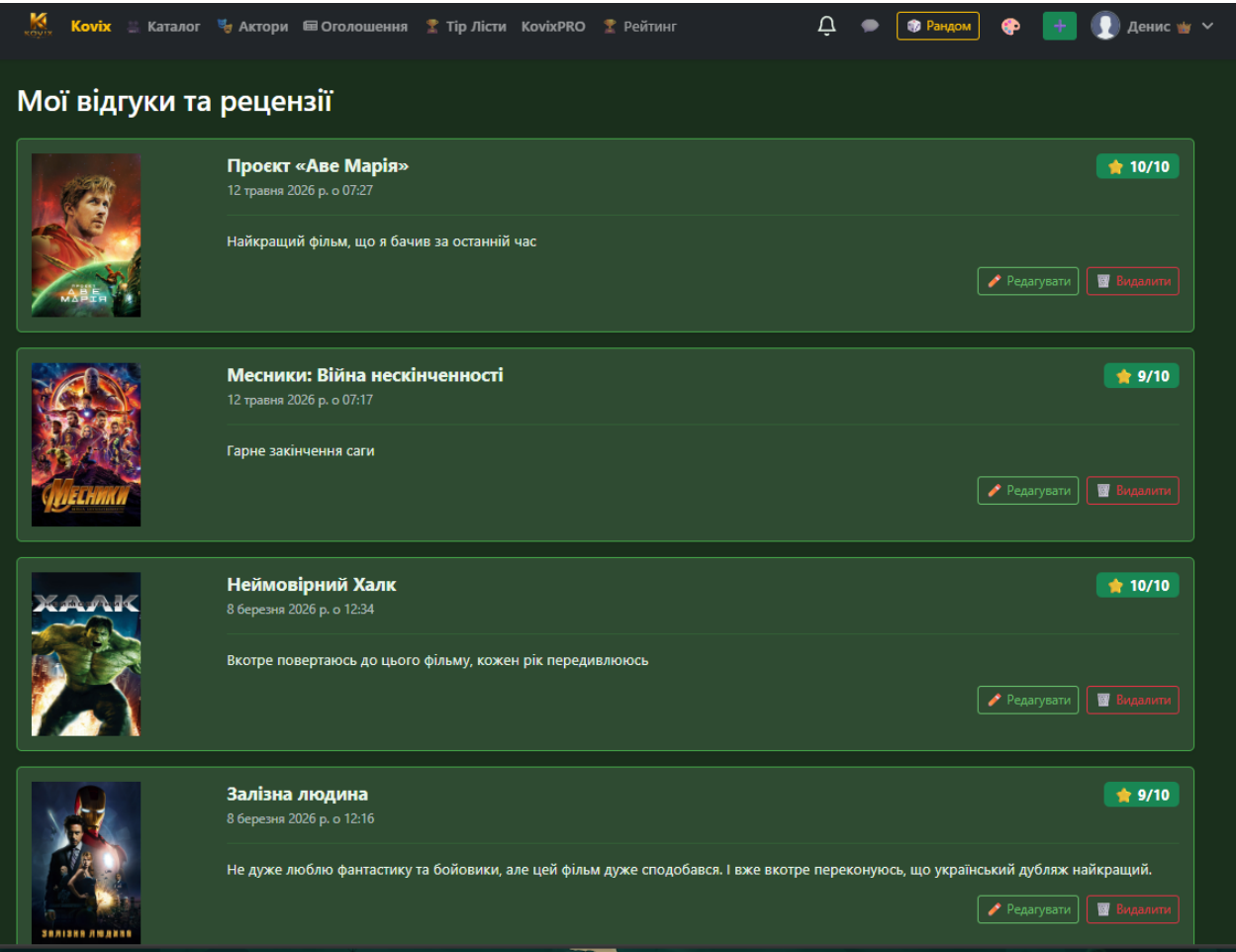


Рис.3.64. Сторінка всіх відгуків та рецензій

Також окрему увагу хотів би приділити на модерацію сайту, для модераторів було створено відповідну роль та відповідні вимоги до них, спершу модератор контролює всі скарги які відправляються в чатах (рис.3.65), також модератору дозволено заблокувати чи розблокувати користувача якщо він побачить відповідні загрози для сайту (рис.3.66), також модератор контролює тір-лісти, йому відправляють всі тір лісти які потрібно опублікувати, він вже приймає рішення чи цей тір лист не порушує правила платформи та чи тір лист відповідає вимогам (рис.3.67).

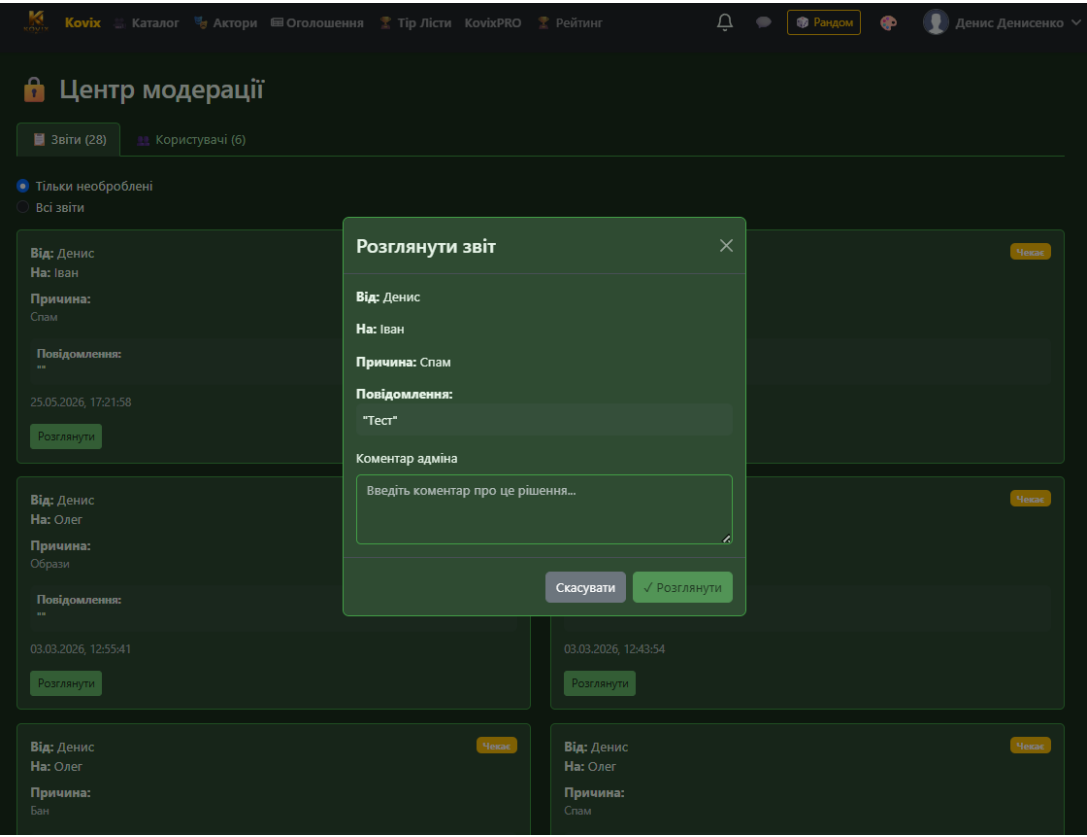


Рис.3.65. Модерація сайту

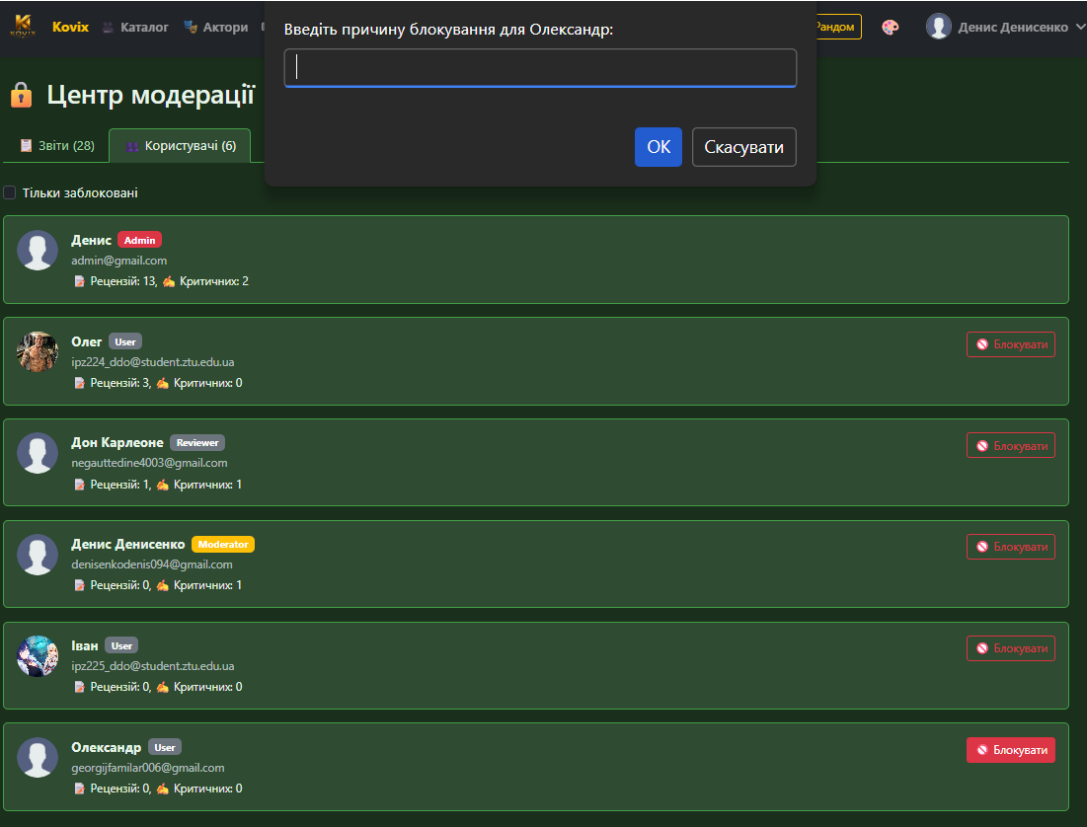


Рис.3.66. Блокування користувачів на сайті

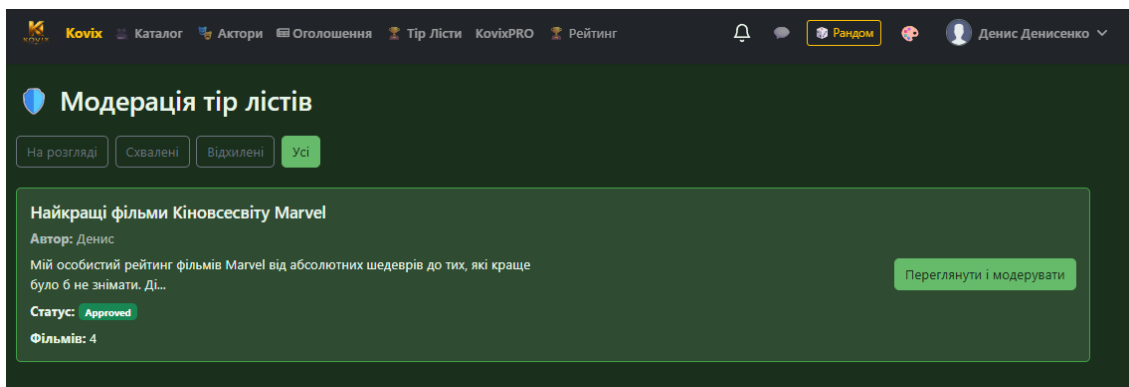


Рис.3.67. Модерація тір лістів

Останім блоком буде футер сайту, тут знаходиться коротка інформація про сайт, корисні посилання, контакти як можна зв'язатися з адміністрацією, та реклама власних соціальних мереж (рис.3.68).

З важливого тут можна виділити посилання перші два посилання це швидкі переходи на відповідні сторінки, які були пояснені раніше тут додалися 2 нові сторінки та один повноцінний функціонал.

Спершу по сторінках це “Правила користування” та “Конфіденційність” (рис.3.69-3.70), ці дві сторінки потрібні для того щоб повідомити користувача про те як себе потрібно вести та можливі наслідки.

І завершальним етапом буде блок “FAQ” (рис.3.71) [22].

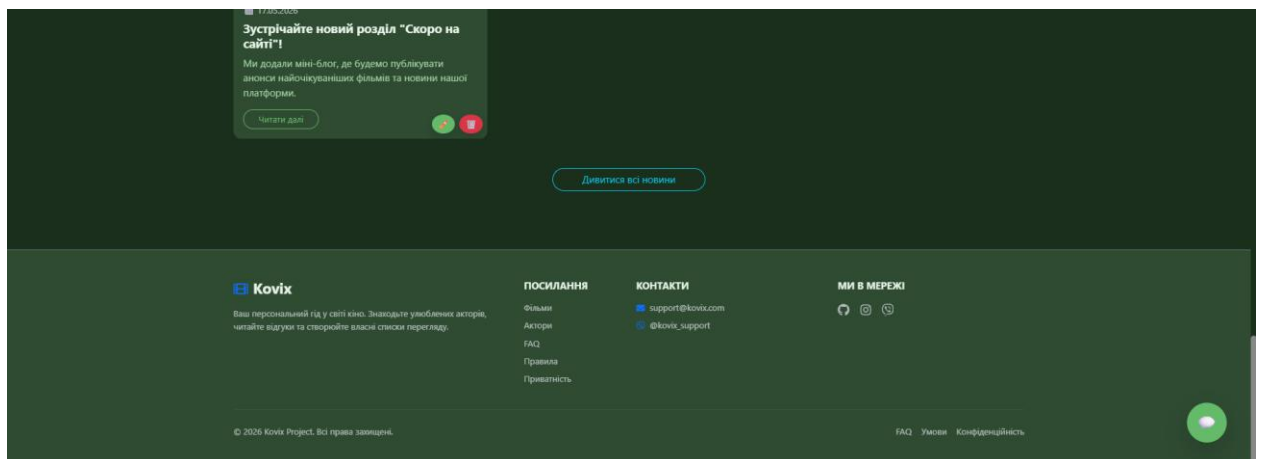


Рис.3.68. Футер

		Денисенко Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремов Ю.М.				100
Змн.	Арк.	№ докум.	Підпис	Дата		

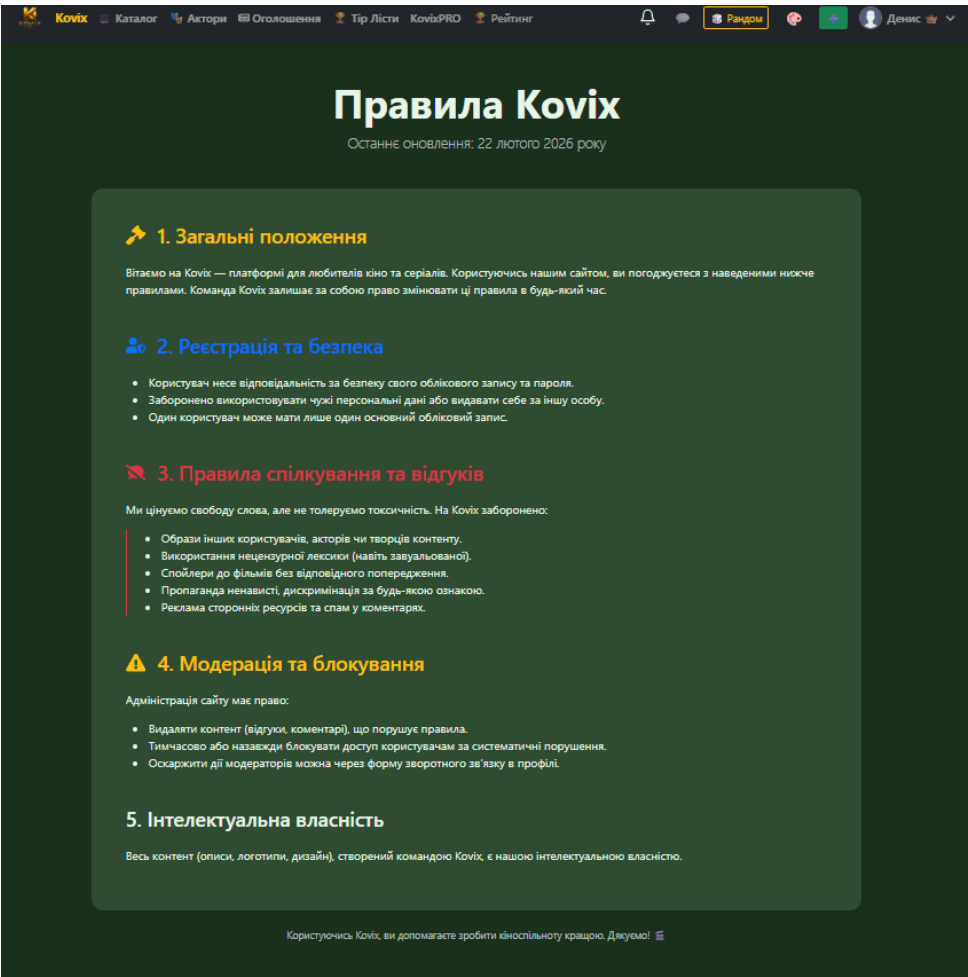


Рис.3.69. Правила користування на сайті

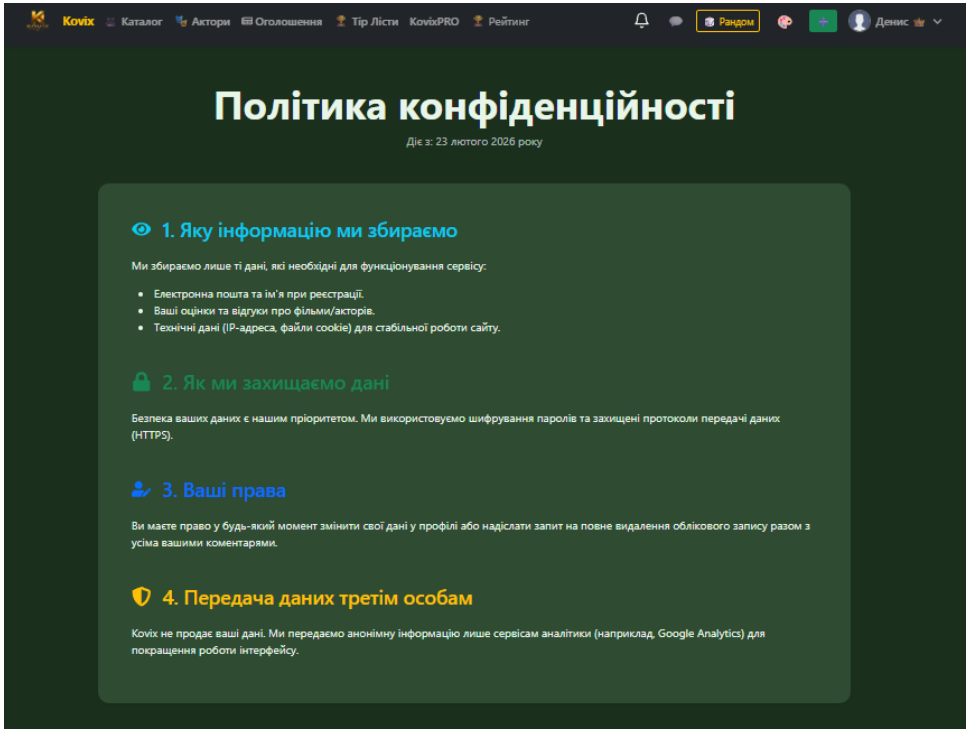


Рис.3.70. Політика конфіденційності на сайті

		Денисенко Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремов Ю.М.				101
Змн.	Арк.	№ докум.	Підпис	Дата		

Цей блок зібрав можливі запитання які можуть зацікавити користувача (рис.3.71), якщо ж користувач не знайде відповідь на запитання на будь якій сторінці його супроводжує маленька кнопка із зв'язком адміністрації (рис.3.72), користувач може написати будь яке запитання яке відправиться на розгляд (рис.3.73).

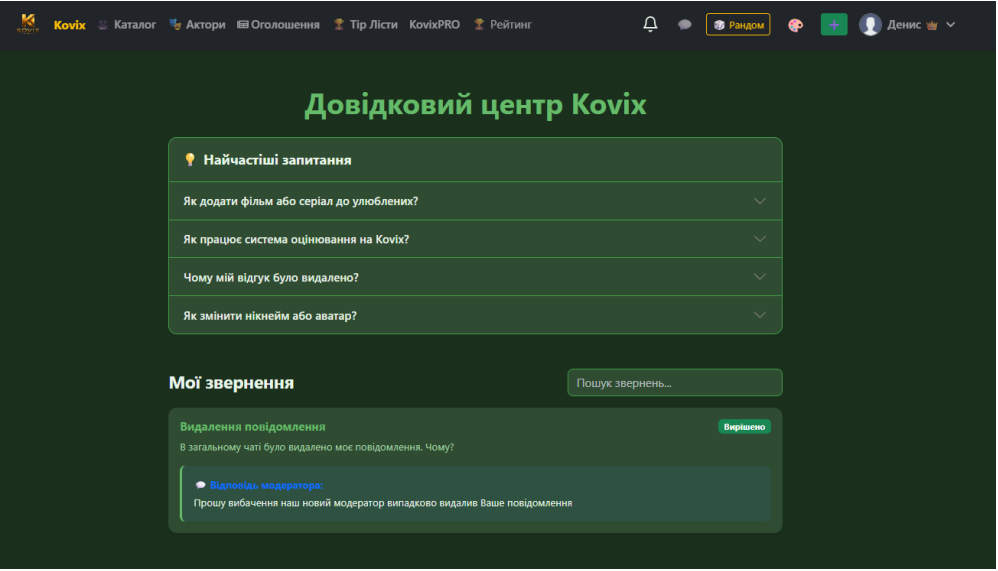


Рис.3.71. FAQ

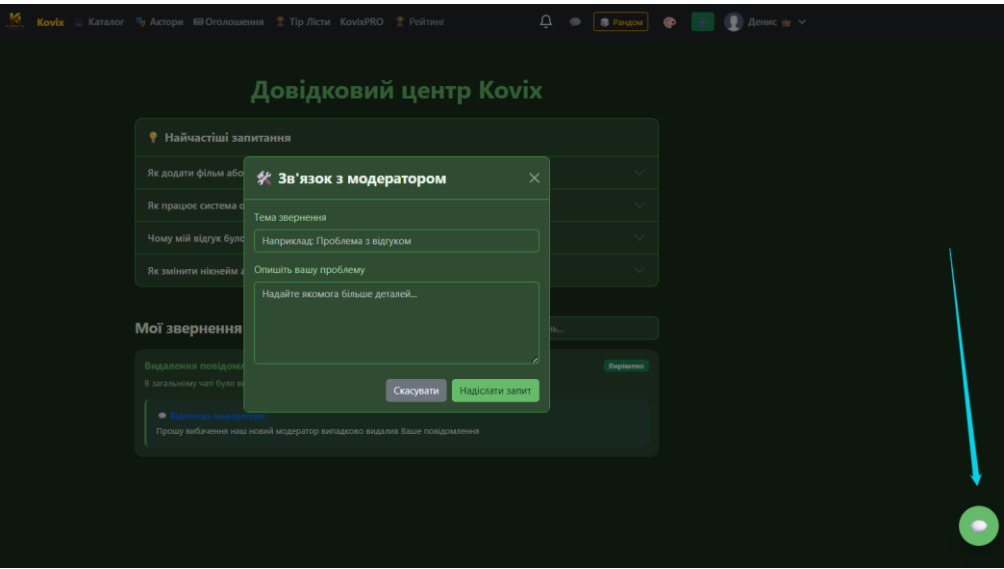


Рис.3.72. Зв'язок з адміністрацією

Панель Модератора: Звернення					
ID	Користувач	Тема / Повідомлення	Відповідь модератора	Статус	Дія
#1	Денис	Видалення повідомлення В загальному чаті було видалено моє повідомлення. ...	Проху вибачення наш новий модератор випадково видалив Ваше повідомлення	Вирішено	Відкрити знову

Рис.3.73. Розгляд звернення

		Денисенко Д.О.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Сфремов Ю.М.				102
Змн.	Арк.	№ докум.	Підпис	Дата		

3.3. Статистика користувача

На профілі користувача можна побачити блок статистики, за допомогою якого відслідковується активність переглянутих чи запланованих фільмів. В ньому відображається загальна кількість фільмів, розподілені за відповідним статусом. Для кожної категорії відображається кількість елементів та середній рейтинг фільмів категорії. Завдяки цим даних можна побудувати візуальна діаграма, яка відображає зв'язок між різними статусами (рис.3.74).

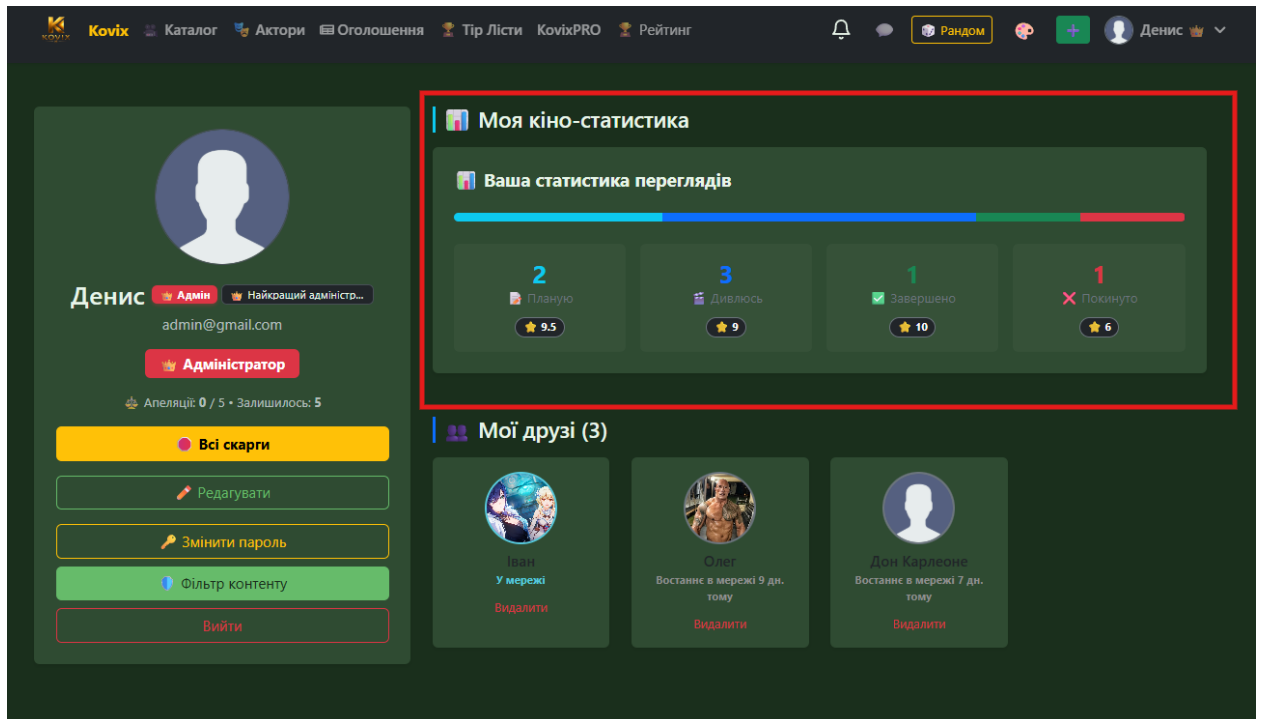


Рис.3.74. Статистика користувацьких списків

Також в користувача є можливість пройти тест перевіряючи свої знання та переглянути детально інформацію по тестах, ця статистика відображається у такому вигляді тільки для власника (рис.3.75), якщо ж користувач хоче побачити цю статистику десь поза межами свого профілю, аби була мотивація для проходження тестів, було створено сторінку з рейтингом найкращих за тестами (рис.3.28) [30]. Така реалізація допомагає тримати користувача завдяки введенню користувацької взаємодії з іншими учасниками вебплатформи.

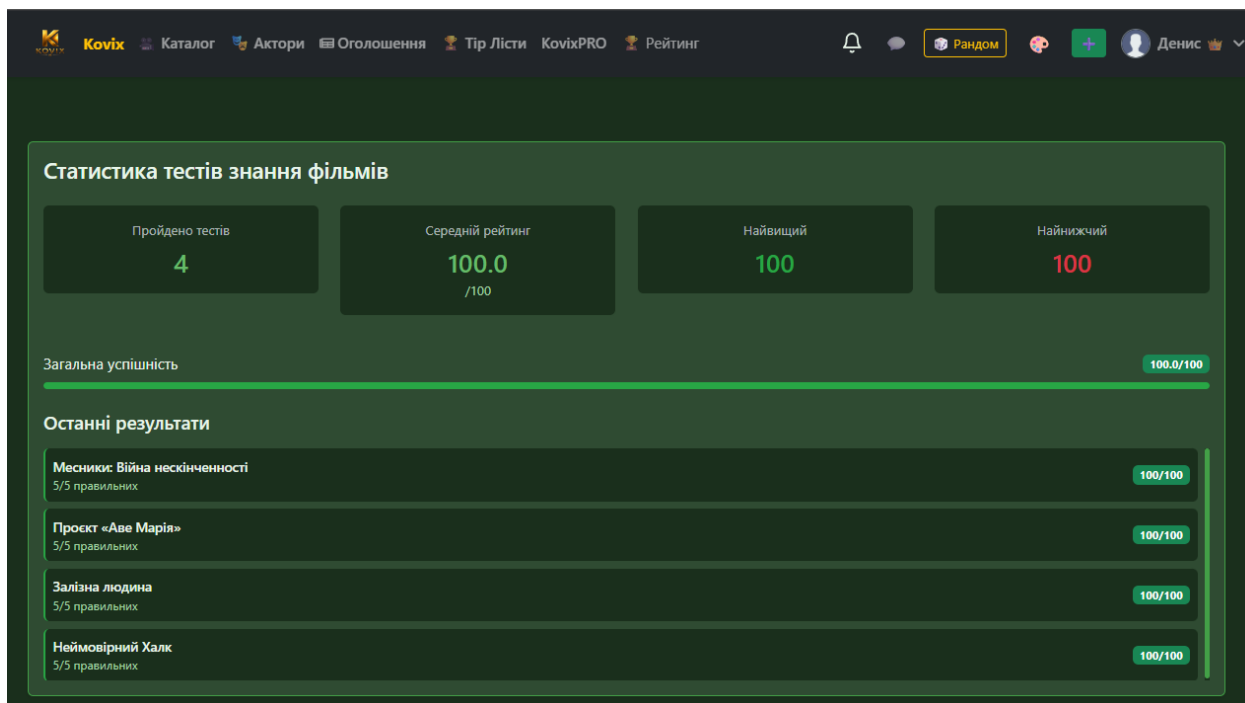


Рис.3.75. Статистика користувацьких тестів

Все це мотивує користувачів все більше заходити на вебплатформу, адже з'являється дух змагання. Особиста статистика дозволяє кожному стежити за своєю історією переглядів, а ще порівнювати свої досягнення з іншими учасниками спільноти. Завдяки діаграмам люди без проблем можуть бачити власні вподобання та які жанри вони люблять найбільше. Це дуже допомагає системі пропонувати фільми, які будуть цікавити відповідну людину.

Завдяки уведенню тестів та загального рейтингу найактивніших учасників мотивує людей частіше писати відгуки та рецензії, разом з проходженням тестів під фільмів (рис.3.76).

Такий підхід до ігрових елементів перетворює простий фільм на захопливе заняття, де хочеться постійно зростати та взаємодіяти з іншими користувачами спільноти.

Сервіс в цей момент перетворюється вже не на базу даних, а на місце де спілкування є основою користувацького досвіду.

Найбільшу увагу заслуговує особиста аналітика та елементи гейміфікації. Вебплатформа створює активну спільноту, де кожен користувач відчуває себе частиною інтерактивного простору.

		Денисенко Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремов Ю.М.				104
Змн.	Арк.	№ докум.	Підпис	Дата		

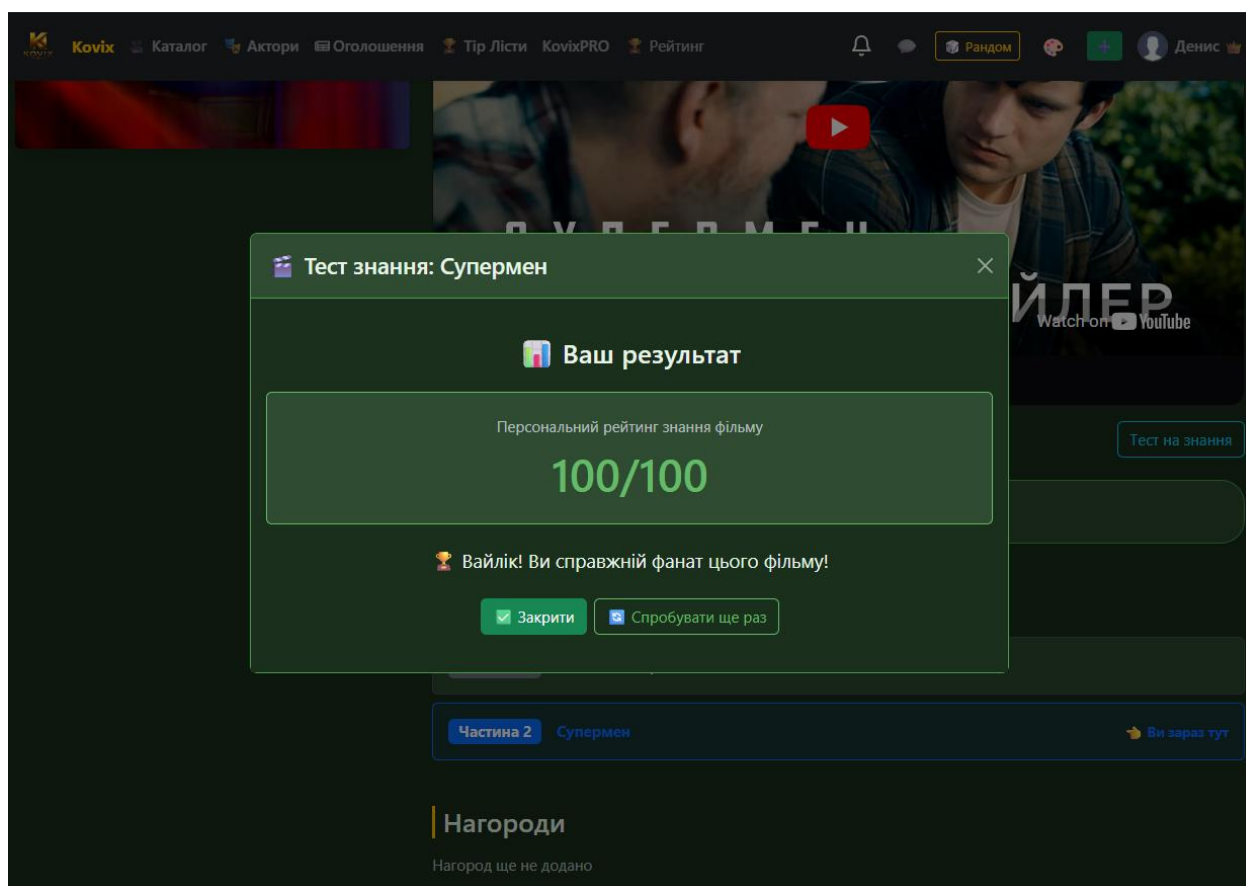


Рис.3.76. Результат проходження тесту

Завдяки поєднанню інформації про вподобання користувачів та їх активність в ігрових механіках, це дозволило системі змінити персоналізацію у вигляді особистої інформації, також створено правильність рекомендацій контенту.

Ця логіка створює унікальний персоналізований шлях для кожного кінолюбителя на свій лад, перетворюючи звичайне споживання інформації на постійну дискусію з користувачами на вебплатформі та формуючи тривалу вірність до неї.

Інтеграція аналітичних інструментів з використанням соціальних функцій допомогло забезпечити стабільне зростання активності через те, що користувачі отримують не лише доступ до контенту, але й можливість об'єктивно оцінювати свій прогрес, відстежувати зміни власних смакових уподобань, це допомагає відчувати себе частиною розвитку великої тематичної екосистеми, що робить кожну наступну взаємодію з вебплатформою змістовною та персоналізованою.

		Денисенко Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремов Ю.М.				105
Змн.	Арк.	№ докум.	Підпис	Дата		

3.4. Тестування вебплатформи

Тестування вебплатформи є невід’ємною частиною розробки будь якого великого програмного рішення, воно є в деяких сферах найважливішою частиною. На завершальному етапі розробки програмного продукту було проведено комплексний обхід по всіх компонентах програми. Головним завданням на цьому процесі було вирішення проблем у функціональності та впевненість, що програма відповідатиме всім вимогам які були їй поставлені.

Для того, щоб протестувати API серверну частину, було обрано інструмент Postman. Postman - це платформа, яка призначена для розробки та імітації запитів з клієнтською частиною вебсайтів. Ця платформа надає купу можливостей для тестування API серверів, включаючи налаштування параметрів запиту, роботу із заголовками та файлами cookies, також автоматизація тестування допомагає скриптам та генерувати тестові дані. Цей застосунок дуже ефективний для перевірки функціональності розроблених програмних рішень перед впровадженням у реальне середовище. Після тестування API сервера за допомогою інструмента Postman можна виявити критичні помилки та виправити їх, було протестовано надсилання запиту до серверу через інтерфейс інструмента Postman (рис.3.77) [23].

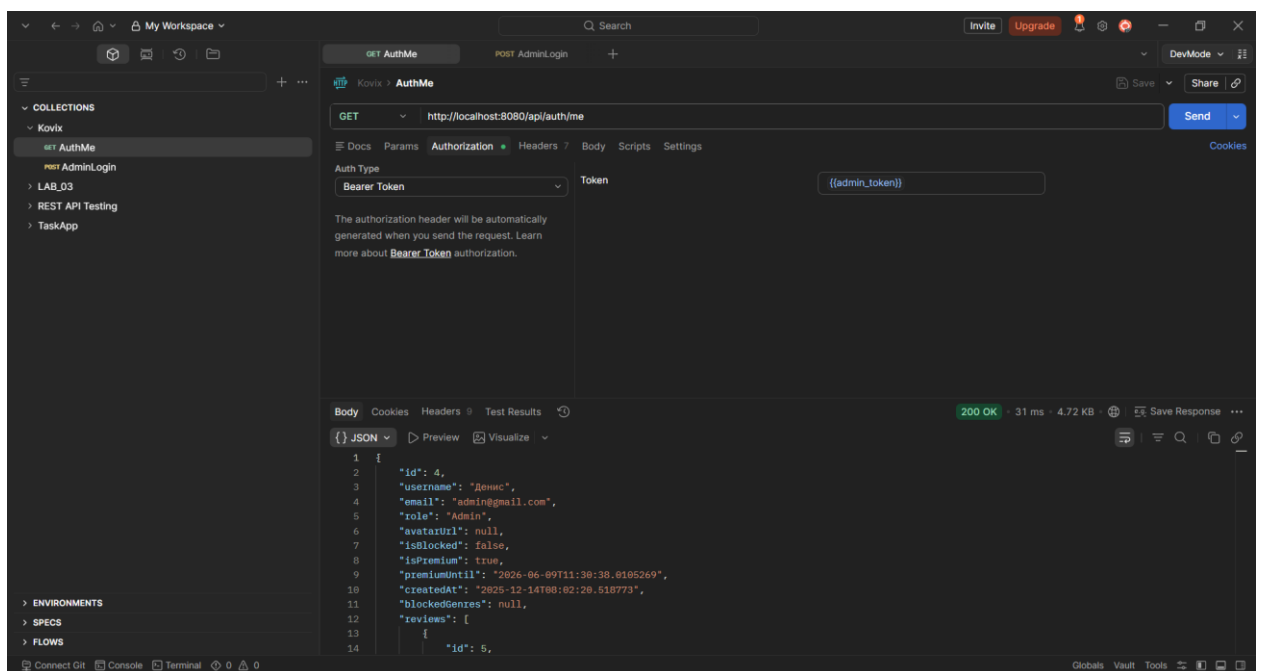


Рис.3.77. Приклад запиту для тесту API

		Денисенко Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		106

Далі потрібно перевірити наскільки швидко відповідає вебплатформа, для цього процесу було використано сайт PageSpeed, розроблений Google для тестування вебсторінок. PageSpeed демонструє як швидко прогрузиться сайт на тому чи іншому девайсі це допомагає виявити проблеми та недоліки які потім будуть виправлені для оптимізації роботи вебплатформи. Нижче подано результати тестування за допомогою PageSpeed (рис.3.77).

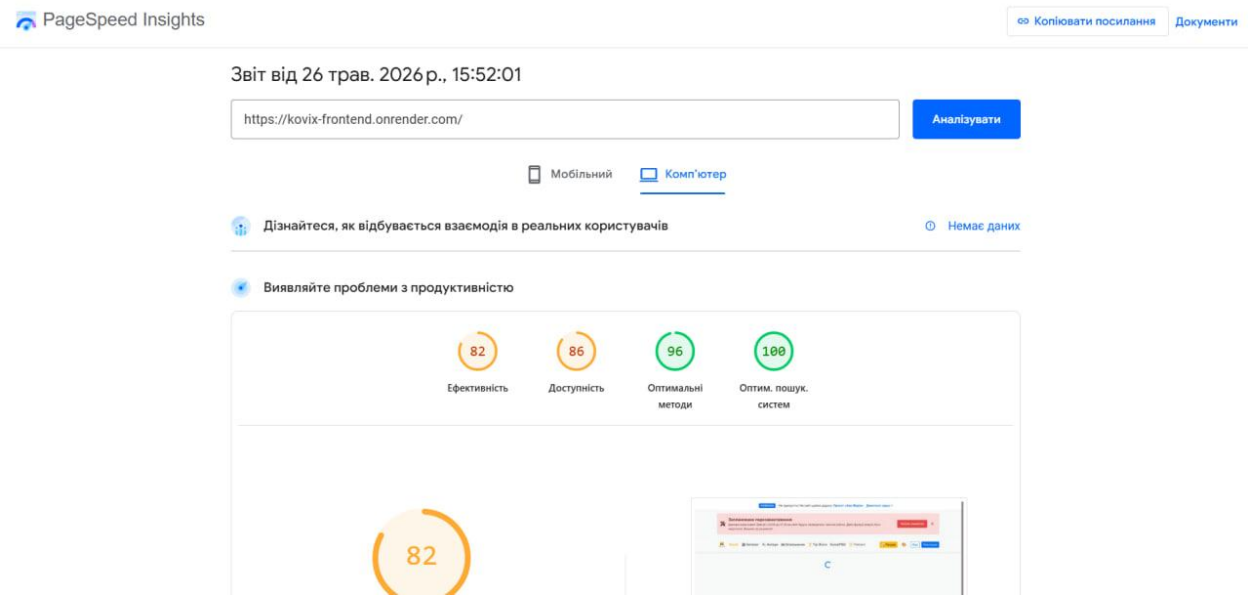


Рис.3.78. Результат виконання тестування PageSpeed

Для перевірки на чистоту коду, правильності побудови структури та уникнення дублювання було використано сайт SonarCloud (рис.3.78) [14].

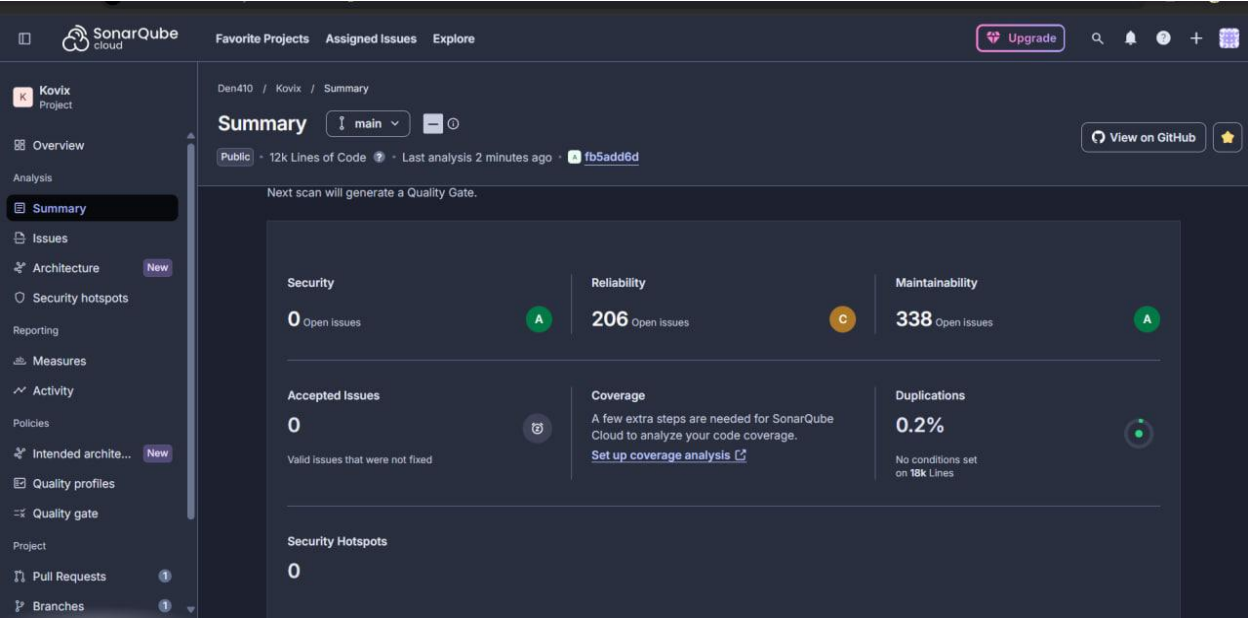


Рис.3.79. SonarCloud тест

Також для перевірки на адаптивність було використано вбудовану функцію браузера. Це дозволило оцінити як відображається сайт на пристроях. Нижче наведено скріншот як виглядає головна сторінка на телефоні [25].

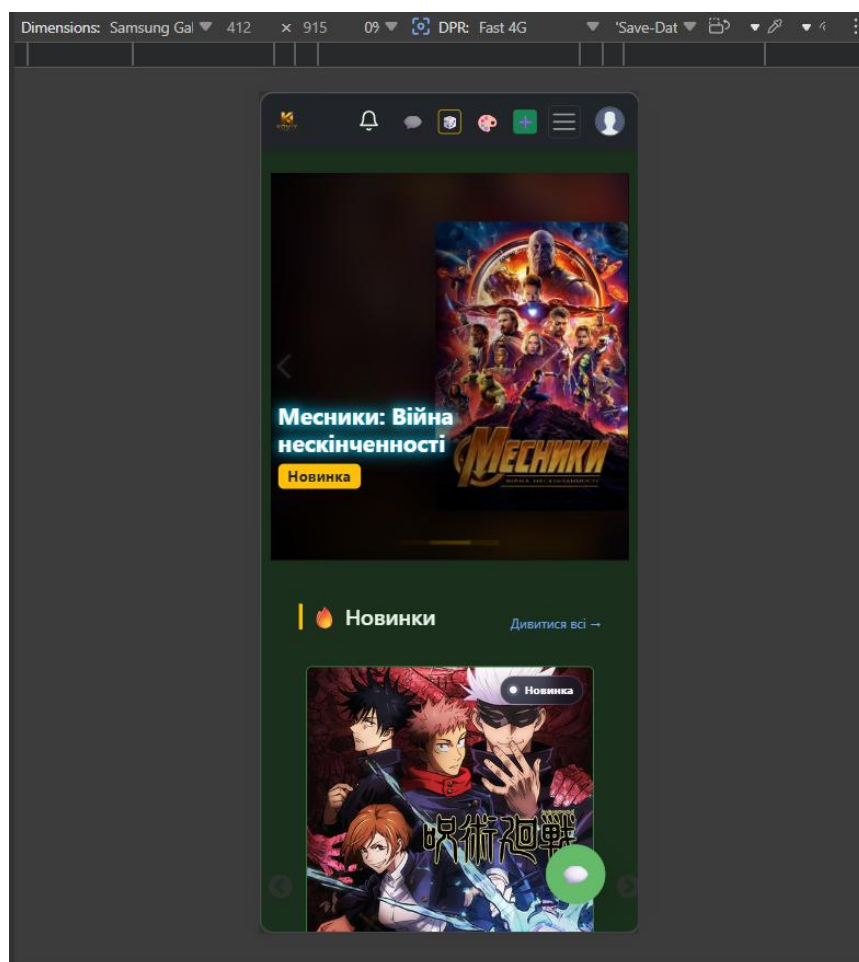


Рис.3.80. Головна сторінка на телефоні

Після завершення розробки програмного продукту було проведено тестування всіх компонентів. Особливістю даного процесу стала виявленість можливих проблем у функціоналі, це допомогло впевнитися у тому, що програма відповідає всім поставленим вимогам, що були перед нею з самого початку розробки.

Ще одним важливим кроком стала можливість забезпечення безпеки вебплатформи, окрім функціонального тестування. Було проведено аналіз вхідних даних та налаштувань заголовків для перевірки стійкості до найпоширеніших вразливостей, це дозволило мінімізувати ризики витоку інформації та неправомірний доступу до даних користувачів.

		Денисенко Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		108

Такий підхід до контролю якості від оптимізації коду та швидкості до тестування API гарантує стабільну роботу системи в реальних умовах експлуатації та забезпечує високий рівень довіри до розробленого продукту з боку кінцевих користувачів. Завдяки автоматизованим та ручним перевіркам вдалося усунути виявлені недоліки та закласти основу для подальшого масштабування та легкого обслуговування програмного продукту.

Висновки до третього розділу

У цьому розділі було продемонстровано та розглянуто процес розгортання, тестування розробленої вебплатформи “Kovix”. Також було змодельовано діаграму розгортання, яка описує взаємодію між клієнтською та серверною частиною з використанням бази даних.

Також було описано послідовність налаштування хостингу. Основний функціонал інтерфейсу користувача у вигляді графічних зображень, включаючи детальні модулі статистики у профілі користувача та сторінки тестів з рейтингом.

Далі було проведено глибоке ручне тестування серверної частини за допомогою інструменту Postman, це дозволило знайти та виправити помилки в маршрутизації по всьому сайті. Також клієнтська частина вебсайту була протестована за допомогою інструментів PageSpeed та SonarCloud, за допомогою цих двох інструментів було проведено аудит показників продуктивності, чистоти коду та оптимізації.

Повна адаптивність та коректність відображення інтерфейсу вебплатформи на різних типах пристроїв була підтверджена за допомогою вбудованих інструментів браузера.

		Денисенко Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		109

ВИСНОВКИ

У процесі виконання випускної кваліфікаційної роботи бакалавра було здійснено дослідження та проаналізовано існуючі сфери застосування та напрацьовання у проєкті “Веблатформа для рецензування фільмів та серіалів”. Головною метою першого розділу було проведення аналізу конкурентів та порівняння їхніх сильних та слабких сторін з моїм проєктом.

У межах виконання завдань було вивчено актуальні методики організації на різних сайтах для рецензування фільмів та серіалів. На підставі цього аналізу було сформульовано необхідні вимоги до функціональності системи, що розроблялася.

В роботі описано архітектурне проєктування системи, ключові сценарії використання, було розроблено модель на основі об'єктно-орієнтованого підходу з використанням сучасних мов програмування та логічну структуру сховища даних. Запропоновані рішення гарантують відповідність моделі предметної області до її вимог, формуючи фундамент для подальшого втілення програмного продукту. Також було поставлено мінімальні вимоги до сервера та розроблена ціла система у вигляді діаграм використання, розгортання та інших. Демонстрація інтерфейсу клієнтської частини стала основою функціоналу для системи, після реалізації програмного продукту було проведено тестування застосунку та усунення критичних помилок.

Крім того, було проаналізовано основні алгоритми функціонування системи. Продемонстровано застосування передових архітектурних концепцій, зокрема принципу розмежування обов'язків між рівнями системи, використання сервісних команд, впровадження доменної бізнес-логіки та механізмів, що базуються на подіях, для комунікації між складовими. Отримані підсумки засвідчують обґрунтованість вибраних технологічних та архітектурних рішень і можуть слугувати міцною базою для подальшої розробки та фіналізації випускної кваліфікаційної роботи бакалавра.

Практичне завдання - розроблено веблатформу для рецензування фільмів та серіалів.

		Денисенко Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремюв Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		110

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Денисенко Д.О., Єфремов Ю.М. ПРОЕКТУВАННЯ UX/UI-ІНТЕРФЕЙСУ ВЕБ-ПЛАТФОРМИ ДЛЯ МЕДІАКОНТЕНТУ. Тези доповідей XVI Міжнародної науково-технічної конференції Інформаційно-комп'ютерні технології - 2026, 02-03 квітня 2026 року. Житомир: «Житомирська політехніка», 2026.
2. Денисенко Д.О., Єфремов Ю.М. ВЕБПЛАТФОРМА ДЛЯ РЕЦЕНЗУВАННЯ ФІЛЬМІВ ТА СЕРІАЛІВ. Тези доповідей XVI Міжнародної науково-технічної конференції Інформаційно-комп'ютерні технології - 2026, 02-03 квітня 2026 року. Житомир: «Житомирська політехніка», 2026.
3. Фаулер М. Рефакторинг: поліпшення дизайну існуючого коду. 2-ге вид. Київ : Спадщина, 2019. 432 с.
4. Gamma E., Helm R., Johnson R., Vlissides J. *Design Patterns: Elements of Reusable Object-Oriented Software*. Boston: Addison-Wesley, 1994. 395 p.
5. Pressman R. S., Maxim B. R. *Software Engineering: A Practitioner's Approach*. 9th ed. New York: McGraw-Hill, 2019. 848 p.
6. OMG. *UML Profile for Software Architecture*. URL: <https://www.omg.org/spec/UML/> (дата звернення: 01.01.2026).
7. Booch G., Rumbaugh J., Jacobson I. *The Unified Modeling Language User Guide*. 2nd ed. Boston: Addison-Wesley, 2005. 560 p.
8. Pohl K., Böckle G., Van Der Linden F. *Software Product Line Engineering: Foundations, Principles, and Techniques*. Berlin: Springer, 2005. 585 p.
9. Microsoft Docs. *Introduction to Software Architecture* [Електронний ресурс]. - Режим доступу: <https://learn.microsoft.com/en-us/azure/architecture/> (дата звернення: 04.01.2026).
10. Lethbridge T. C., Laganière R. *Object-Oriented Software Engineering: Practical Software Development Using UML and Java*. 2nd ed. New York: McGraw-Hill, 2001. 512 p.

		Денисенко Д.О.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		111

11. Cockburn A. *Writing Effective Use Cases*. Boston: Addison-Wesley, 2000. 264 p.
12. StarUML Documentation [Електронний ресурс]. - Режим доступу: <https://docs.staruml.io> (дата звернення: 02.01.2026).
13. Проектування інформаційних систем управління медіа-контентом / за заг. наук. ред. В. М. Марценюка. Тернопіль : Укрмедкнига, 2023. 374 с.
14. SonarCloud Documentation. Exploring the Static Analysis. URL: <https://docs.sonarcloud.io> (дата звернення: 20.03.2026).
15. React Documentation. Quick Start. URL: <https://react.dev/learn> (дата звернення: 10.01.2026).
16. Комплексний підхід до тестування програмного забезпечення веб-платформ : матеріали Всеукр. наук.-практ. конф., м. Житомир, 14–15 квіт. 2025 р. Житомир : ДУ «Житомирська політехніка», 2025. 182 с.
17. Google Chrome Developers. Optimize Largest Contentful Paint (LCP). URL: <https://developer.chrome.com/docs/lighthouse/performance/lighthouse-total-blocking-time/> (дата звернення: 22.02.2026).
18. Ткаченко Д. В. Методи оптимізації рендерингу в клієнтських фреймворках // Юридичний науковий електронний журнал. 2024. № 4. С. 244–248. URL: http://lsei.org.ua/4_2024/62.pdf.
19. ДСТУ 8302:2015. Бібліографічний опис. Скорочення слів і словосполучень українською мовою. Загальні вимоги та правила. [Чинний від 2016-04-03]. Вид. офіц. Київ : Мінекономрозвитку України, 2016. 17 с.
20. Розенберг Дж., Скотт К. Застосування Use Case для проектування веб-інтерфейсів мультимедійних систем. Бостон : Addison-Wesley, 2019. 214 р.
21. Панасюк О. В. Використання паттерну Repository та Unit of Work в архітектурі сучасних веб-застосунків. Комп'ютерні технології та інновації. DOI: <https://doi.org/10.15407/cti2025.02.018>

		Денисенко Д.О.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Сфремов Ю.М.				112
Змн.	Арк.	№ докум.	Підпис	Дата		

- 22.Редмонд Е., Вілсон Дж. Сім баз даних за сім тижнів. Порівняльний аналіз реляційних та NoSQL СУБД. Даллас : Pragmatic Bookshelf, 2021. 312 р.
- 23.Райс Л. Безпека веб-застосунків. Розробка захищених систем на С# та .NET. Лондон : Manning, 2023. 280 р.
- 24.Чинний стандарт безпеки даних індустрії платіжних карток (PCI DSS) : версія 4.0. URL: <https://www.pcisecuritystandards.org> (дата звернення: 05.03.2026).
- 25.Хміль А. А. Проектування адаптивних веб-додатків на базі фреймворку Bootstrap 5 // Електронний науковий журнал «Сучасні ІТ-технології». 2025. № 3. С. 75–81. URL: http://sit-journal.org.ua/3_2025/12.pdf (дата звернення: 18.02.2026).
- 26.Docker Documentation. Containerizing an ASP.NET Core application. URL: <https://docs.docker.com/samples/dotnetcore/> (дата звернення: 11.02.2026).
- 27.React Slick Slider. Responsive Cloud Layouts and Slider Settings. URL: <https://react-slick.neostack.com> (дата звернення: 14.02.2026).
- 28.Сікерін Ю. О., Малафіїк Л. О. Паттерни проектування архітектури корпоративних веб-застосунків на базі .NET Core : навч. посіб. Одеса : ОНПУ, 2024. 156 с.
- 29.Шаллоуей А., Тротт Дж. Шаблони проектування. Новий підхід до об'єктно-орієнтованого аналізу та дизайну програмних систем. Київ : Діалектика, 2023. 288 с.
- 30.Оптимізація продуктивності клієнтської частини веб-платформ (Web Performance Optimization) : монографія / за ред.: П. О. Бедрія, О. О. Петренка. Львів : Львівська політехніка, 2025. 210 с.
- 31.Ковальчук К. І. Статичний аналіз безпеки та якості вихідного коду (SAST) за допомогою платформи SonarQube / SonarCloud. Наука, технології та інновації. 2025. Т. 14, № 1. С. 54–63. DOI: <https://doi.org/10.15407/sti2025.01.054>.

		Денисенко Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		113

ДОДАТКИ

		Денисенко Д.О.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Сфремов Ю.М.				114
Змн.	Арк.	№ докум.	Підпис	Дата		

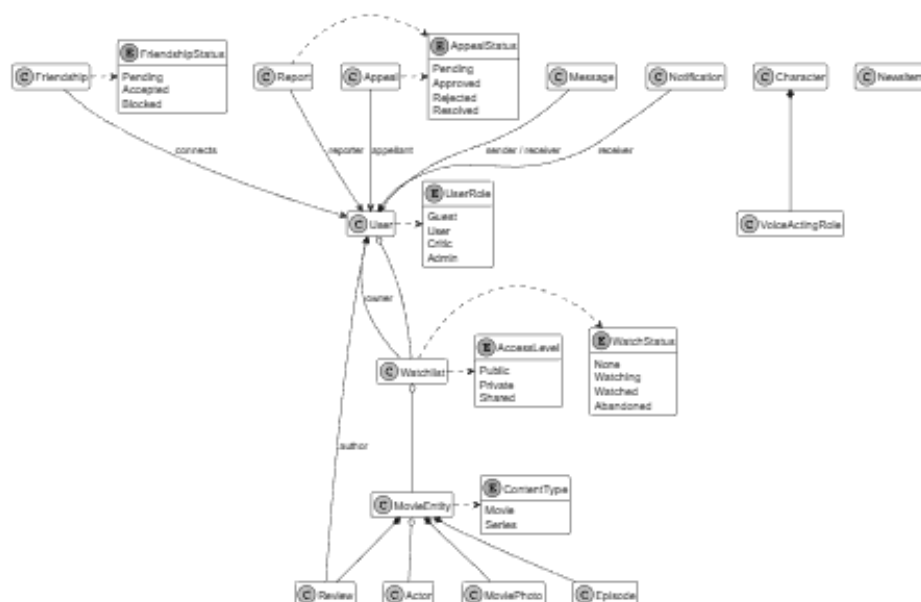


Рис.Скорочена діаграма класів доменних моделей

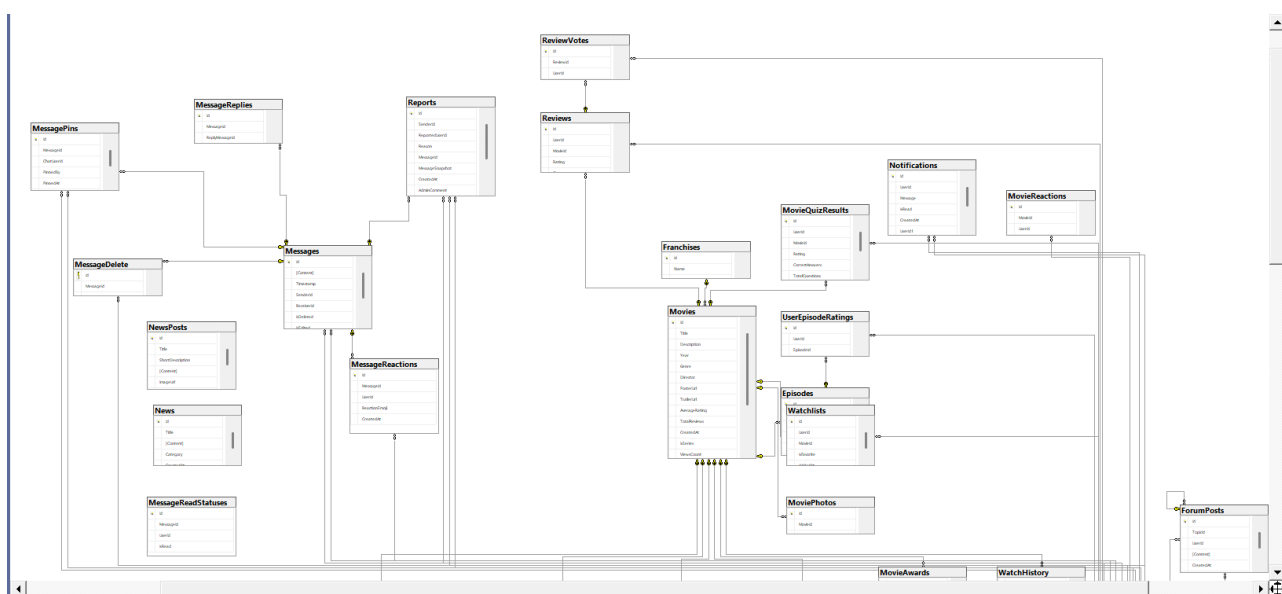


Рис.Детальна таблиця бази даних

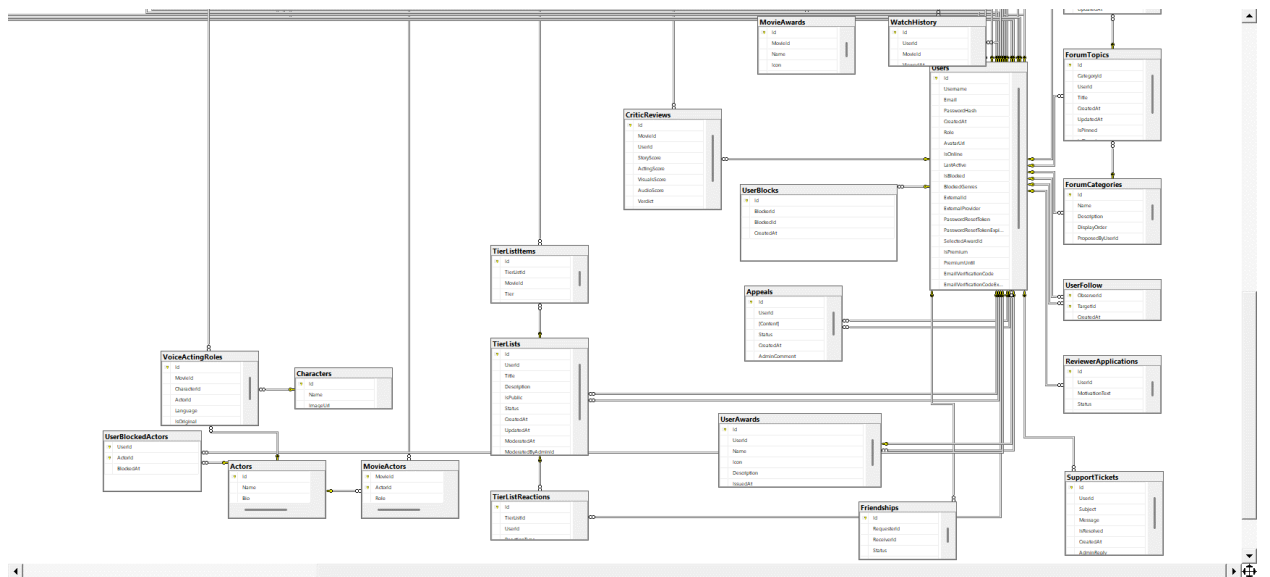


Рис. Детальна таблиця бази даних (Продовження)

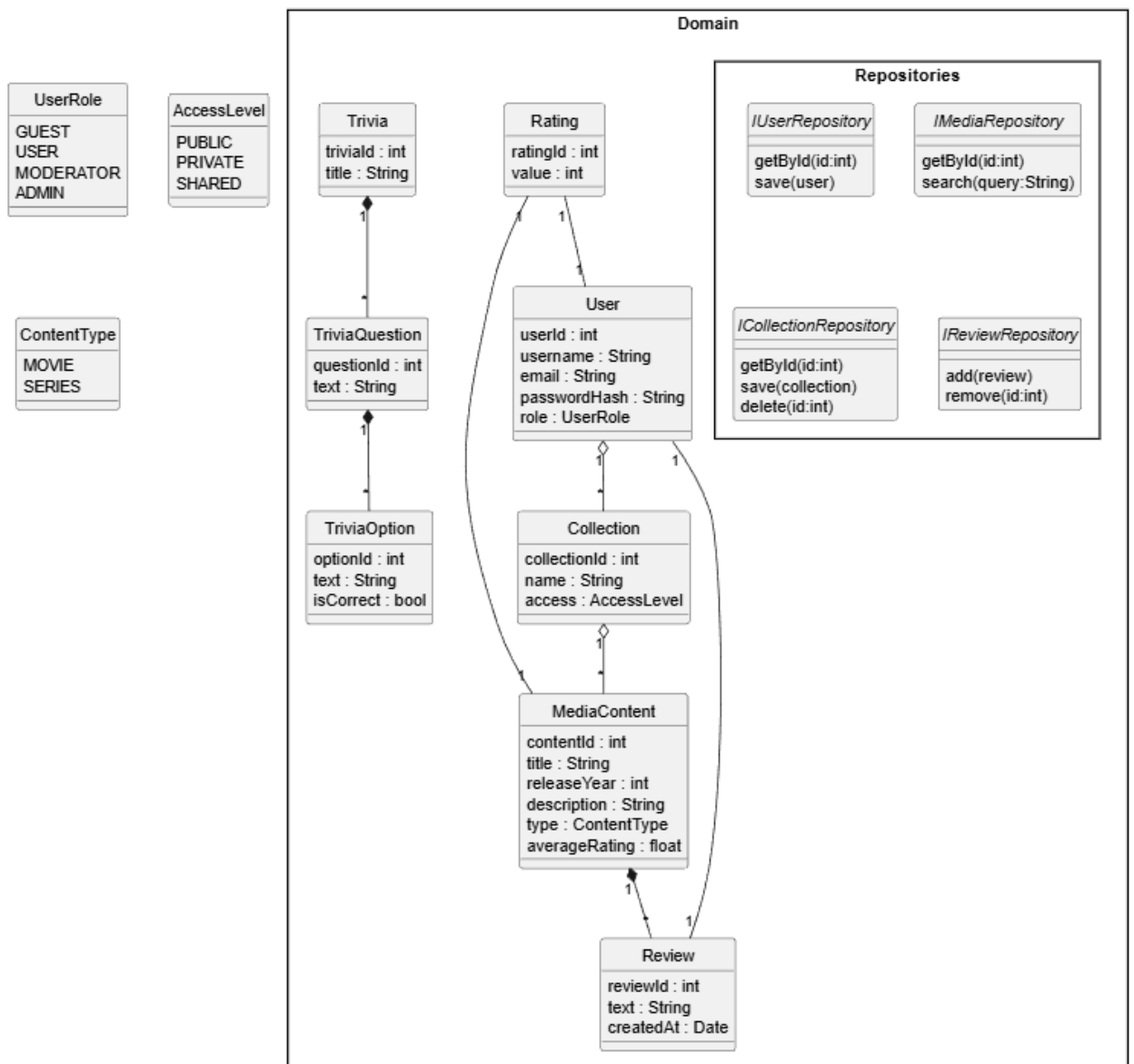


Рис. Діаграма класів при використанні Domain Model + Repositories

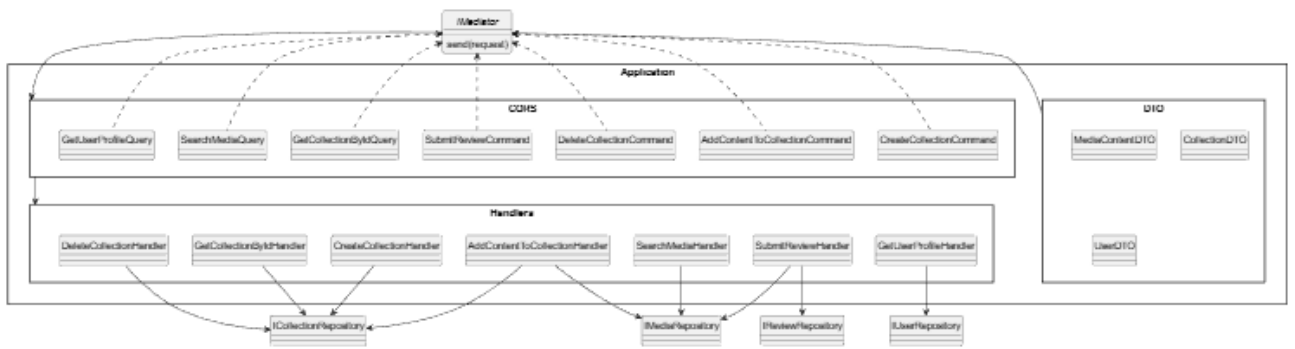


Рис. Діаграма класів при використанні CQRS та Mediator

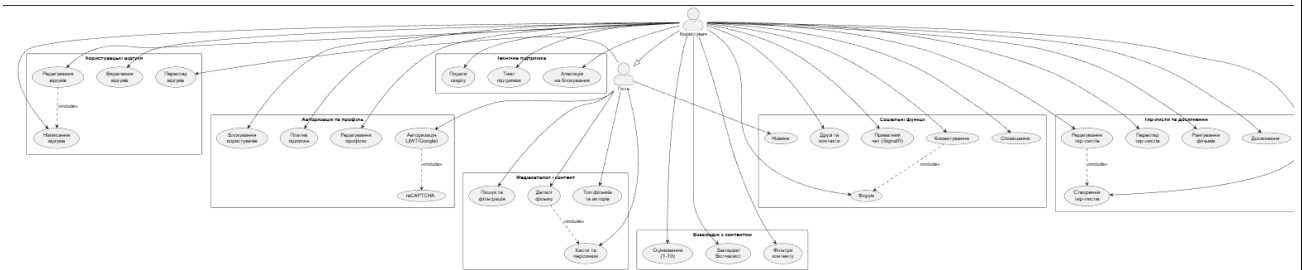


Рис. Діаграма юс кейсу для користувачів

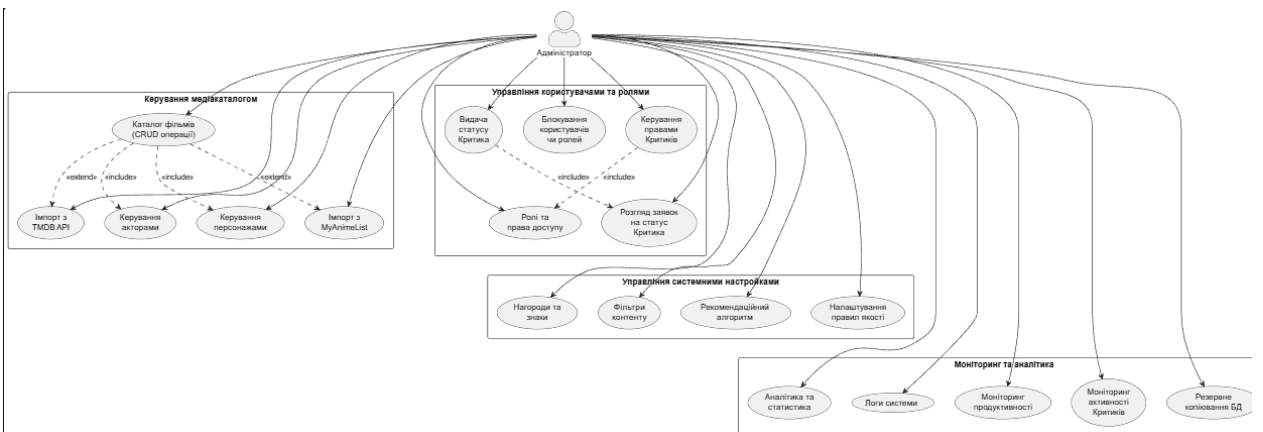


Рис. Діаграма юс кейсу для адміністрації

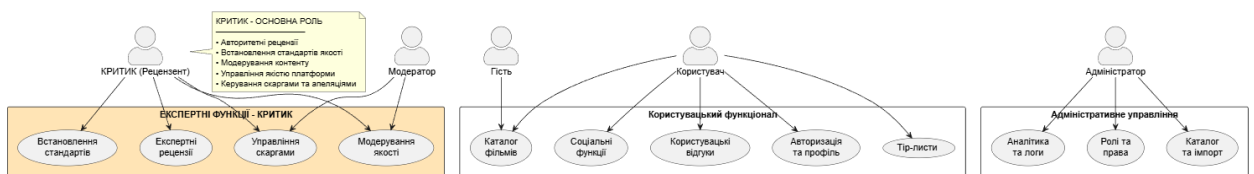


Рис. Коротка архітектура вебплатформи

Лістинг класу Content.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

public class Content {

    public Content() {
    }

    private contentId ;;

    private title ;;

    private releaseYear ;;

    private averageRating ;;

    /// <summary>
    /// @return
    /// </summary>
    public : String getDetails() {
        // TODO implement here
        return null;
    }

}

```

Лістинг класу ForumPost.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

public class ForumPost {

    public ForumPost() {
    }

}

```

		Денисенко Д.О.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Сфремов Ю.М.				119
Змн.	Арк.	№ докум.	Підпис	Дата		

```

private postId ;;

private content ;;

private postDate ;;

```

```

}

```

Лістинг класу ForumThread.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

public class ForumThread {

    public ForumThread() {
    }

    private threadId ;;

    private title ;;

    private isLocked ;;

}

```

Лістинг класу GlobalRating.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

public class GlobalRating {

    public GlobalRating() {
    }

    private ratingId ;;

```

		Денисенко Д.О.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Сфремов Ю.М.				120
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        private rank ;;

        private calculationDate ;;

    }

```

Лістинг класу ModerationRule.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

public class ModerationRule {

    public ModerationRule() {
    }

    private ruleId ;;

    private description ;;

    private penalty ;;

}

```

Лістинг класу Movie.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

public class Movie : Content {

    public Movie() {
    }

    private durationMinutes ;;

    /// <summary>
    /// @return
    /// </summary>

```

		Денисенко Д.О.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Сфремов Ю.М.				121
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        public : int getDuration() {
            // TODO implement here
            return null;
        }
    }
}

```

Лістинг класу Review.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

public class Review {

    public Review() {
    }

    private reviewId ;;

    private text ;;

    private rating ;;

    private reviewDate ;;

    /// <summary>
    /// @param :
    /// @return
    /// </summary>
    public : void updateRating(newRating :) {
        // TODO implement here
        return null;
    }

}

```

Лістинг класу Series.cs

```

using System;
using System.Collections.Generic;
using System.Linq;

```

		Денисенко Д.О.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Сфремов Ю.М.				122
Змн.	Арк.	№ докум.	Підпис	Дата		

```

using System.Text;

public class Series : Content {

    public Series() {
    }

    private numSeasons ;;

    /// <summary>
    /// @return
    /// </summary>
    public : String getCurrentEpisode() {
        // TODO implement here
        return null;
    }

}

```

Лістинг класу Statistics.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

public class Statistics {

    public Statistics() {
    }

    private statsId ;;

    private hoursWatched ;;

    private reviewsPosted ;;

    private lastLogin ;;

}

```

Лістинг класу Subscription.cs

```

using System;

```

		Денисенко Д.О.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Сфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		123

```

using System.Collections.Generic;
using System.Linq;
using System.Text;

public class Subscription {

    public Subscription() {
    }

    private subscriptionId ;;

    private subDate ;;

}

```

Лістинг класу TierList.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

public class TierList {

    public TierList() {
    }

    private listId ;;

    private name ;;

    private description ;;

    /// <summary>
    /// @param :
    /// @param :
    /// </summary>
    public void addItem(content :, tier :) {
        // TODO implement here
    }
}

```

		Денисенко Д.О.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Сфремов Ю.М.				124
Змн.	Арк.	№ докум.	Підпис	Дата		

```
}
```

Лістинг класу User.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

public class User {

    public User() {
    }

    private userId ;;

    private username ;;

    private role ;;

    private email ;;

    /// <summary>
    /// @return
    /// </summary>
    public : void register() {
        // TODO implement here
        return null;
    }

    /// <summary>
    /// @return
    /// </summary>
    public : boolean login() {
        // TODO implement here
        return null;
    }

    /// <summary>
    /// @param :
    /// @return
    /// </summary>
    public : void subscribe(user :) {
        // TODO implement here
    }
}
```

		Денисенко Д.О.			ІПЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
		Сфремов Ю.М.				125
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        return null;
    }
}

```

Лістинг класу UserProfile.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

public class UserProfile {

    public UserProfile() {
    }

    private bio ;;

    private avatarUrl ;;

    private registrationDate ;;

    /// <summary>
    /// @return
    /// </summary>
    public : Statistics viewStatistics() {
        // TODO implement here
        return null;
    }

}

```

Лістинг класу UserRole.cs

```

public enum UserRole {
    GUEST,
    REGISTERED,
    MODERATOR,
    ADMIN
}

```

Лістинг класу MovieControllers.cs

```

using GTranslate.Translators;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;

```

		Денисенко Д.О.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Сфремов Ю.М.				126
Змн.	Арк.	№ докум.	Підпис	Дата		

```

using Microsoft.AspNetCore.SignalR;
using Microsoft.EntityFrameworkCore;
using Movie.API.Data;
using Movie.API.DTOS;
using Movie.API.Hubs;
using Movie.API.Models;
using Movie.API.Models.Enums;
using Movie.API.Models.Tmdb;
using Movie.API.Services;
using System.Security.Claims;

namespace Movie.API.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class MoviesController : ControllerBase
    {
        private readonly ApplicationDbContext _context;
        private readonly IWebHostEnvironment _env;
        private readonly IHubContext<NotificationHub> _notificationHubContext;
        private const string TMDB_API_KEY = "f797261e58f7c171a30e466ce870cfbe";

        public MoviesController(ApplicationDbContext context, IWebHostEnvironment
env, IHubContext<NotificationHub> notificationHubContext)
        {
            _context = context;
            _env = env;
            _notificationHubContext = notificationHubContext;
        }

        [HttpGet]
        public async Task<ActionResult<PagedResult<MovieEntity>>> GetAll(
            [FromQuery] string? search,
            [FromQuery] string? genres,
            [FromQuery] string? sort,
            [FromQuery] int? year,
            [FromQuery] string? awards,
            [FromQuery] int page = 1,
            [FromQuery] int pageSize = 8)
        {
            var query = _context.Movies
                .Include(m => m.MovieActors)
                .Include(m => m.Awards)
                .AsQueryable();

            var userIdClaim = User.FindFirst(ClaimTypes.NameIdentifier);
            if (userIdClaim != null)
            {
                var userId = int.Parse(userIdClaim.Value);
                var user = await _context.Users.AsNoTracking().FirstOrDefaultAsync(u
=> u.Id == userId);

                if (user != null && !string.IsNullOrEmpty(user.BlockedGenres))
                {
                    var blockedList = user.BlockedGenres.Split(',',
StringSplitOptions.RemoveEmptyEntries);

                    foreach (var blocked in blockedList)
                    {
                        var b = blocked.Trim();
                        query = query.Where(m => m.Genre == null ||
!m.Genre.ToLower().Contains(b));
                    }
                }
            }
        }
    }
}

```

		Денисенко Д.О.			ІІЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
		Сфремов ІО.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		127

```

        var blockedActorIds = await _context.UserBlockedActors
            .Where(uba => uba.UserId == userId)
            .Select(uba => uba.ActorId)
            .ToListAsync();

        if (blockedActorIds.Any())
        {
            query = query.Where(m => !m.MovieActors.Any(ma =>
blockedActorIds.Contains(ma.ActorId)));
        }

        if (!string.IsNullOrWhiteSpace(search))
        {
            var searchLower = search.ToLower();
            query = query.Where(m => m.Title.ToLower().Contains(searchLower));
        }

        if (year.HasValue)
        {
            query = query.Where(m => m.Year == year.Value);
        }

        if (!string.IsNullOrWhiteSpace(genres))
        {
            var genreList = genres.ToLower().Split(',',
StringSplitOptions.RemoveEmptyEntries);
            foreach (var genre in genreList)
            {
                var g = genre.Trim();
                query = query.Where(m => m.Genre != null &&
m.Genre.ToLower().Contains(g));
            }
        }

        if (!string.IsNullOrWhiteSpace(awards))
        {
            var awardIds = awards.Split(',',
StringSplitOptions.RemoveEmptyEntries)
                .Select(a => int.TryParse(a.Trim(), out var id) ? (int?)id :
null)
                .Where(a => a.HasValue)
                .Select(a => a.Value)
                .ToList();

            if (awardIds.Any())
            {
                query = query.Where(m => m.Awards.Any(ma =>
awardIds.Contains(ma.Id)));
            }
        }

        switch (sort)
        {
            case "dateAsc":
                query = query.OrderBy(m => m.CreatedAt);
                break;
            case "ratingDesc":
                query = query.OrderByDescending(m => m.AverageRating +
(double)m.TotalReviews / (m.TotalReviews + 10));
                break;
            case "viewsDesc":
                query = query.OrderByDescending(m => m.ViewsCount);
                break;
            case "yearDesc":

```

		Денисенко Д.О.			ІІЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
		Сфремов І.О.М.				128
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        query = query.OrderByDescending(m => m.Year);
        break;
    case "titleAsc":
        query = query.OrderBy(m => m.Title);
        break;
    case "dateDesc":
    default:
        query = query.OrderByDescending(m => m.CreatedAt);
        break;
    }

    int totalCount = await query.CountAsync();

    var items = await query
        .Skip((page - 1) * pageSize)
        .Take(pageSize)
        .ToListAsync();

    var result = new PagedResult<MovieEntity>
    {
        Items = items,
        TotalCount = totalCount,
        PageNumber = page,
        PageSize = pageSize
    };

    return Ok(result);
}

[HttpGet("{id:int}")]
public async Task<ActionResult<MovieDetailDto>> GetById(int id)
{
    int? currentUserId = null;
    var userIdClaim = User.FindFirst(ClaimTypes.NameIdentifier);
    if (userIdClaim != null) currentUserId = int.Parse(userIdClaim.Value);

    var movie = await _context.Movies
        .Include(m => m.Reviews)
        .Include(m => m.Episodes)
        .Include(m => m.MovieActors)
        .ThenInclude(ma => ma.Actor)
        .Include(m => m.Franchise)
        .ThenInclude(f => f.Movies)
        .Include(m => m.Awards)
        .FirstOrDefaultAsync(m => m.Id == id);

    if (movie == null) return NotFound();

    var reactions = await _context.MovieReactions
        .Where(r => r.MovieId == id)
        .ToListAsync();

    var myReactions = currentUserId.HasValue
        ? reactions.Where(r => r.UserId == currentUserId.Value).ToList()
        : new List<MovieReaction>();

    var episodeDtos = new List<EpisodeDto>();
    if (movie.Episodes != null && movie.Episodes.Any())
    {
        var userRatings = currentUserId.HasValue
            ? await _context.UserEpisodeRatings
                .Where(r => r.UserId == currentUserId && r.Episode.MovieId
                    == id)
                .ToDictionaryAsync(r => r.EpisodeId, r => r.Rating)
            : new Dictionary<int, int>();
    }

```

		Денисенко Д.О.			ІІЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
		Сфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		129

```

        episodeDtos = movie.Episodes.Select(e => new EpisodeDto
        {
            Id = e.Id,
            SeasonNumber = e.SeasonNumber,
            EpisodeNumber = e.EpisodeNumber,
            Title = e.Title,
            AverageRating = e.AverageRating,
            CurrentUserRating = userRatings.ContainsKey(e.Id) ?
userRatings[e.Id] : null
        })
        .OrderBy(e => e.SeasonNumber).ThenBy(e => e.EpisodeNumber)
        .ToList();
    }

    var dto = new MovieDetailDto
    {
        Id = movie.Id,
        Title = movie.Title,
        Description = movie.Description,
        Year = movie.Year,
        Genre = movie.Genre,
        Director = movie.Director,
        PosterUrl = movie.PosterUrl,
        TrailerUrl = movie.TrailerUrl,
        AverageRating = movie.AverageRating,
        TotalReviews = movie.Reviews?.Count ?? 0,
        ViewsCount = movie.ViewsCount,

        ReactionCounts = reactions
            .GroupBy(r => r.Type)
            .ToDictionary(g => g.Key.ToString(), g => g.Count()),

        CurrentUserVote = (int?)myReactions
            .FirstOrDefault(r => r.Type == ReactionType.Like || r.Type ==
ReactionType.Dislike)?.Type,

        CurrentUserEmotion = (int?)myReactions
            .FirstOrDefault(r => r.Type != ReactionType.Like && r.Type !=
ReactionType.Dislike)?.Type,
        IsSeries = movie.IsSeries,
        Type = movie.IsSeries ? "Series" : "Movie",
        Episodes = episodeDtos,

        Cast = movie.MovieActors.Select(ma => new CastDto
        {
            ActorId = ma.ActorId,
            Name = ma.Actor.Name,
            Role = ma.Role,
            IsMainRole = ma.IsMainRole,
            PhotoUrl = ma.Actor.PhotoUrl
        }).ToList(),
        Awards = movie.Awards.Select(a => new AwardDto
        {
            Id = a.Id,
            Name = a.Name,
            Icon = a.Icon
        }).ToList(),
        MalId = movie.MalId,
        TmdbId = movie.TmdbId,
        CastImported = movie.CastImported
    };

    dto.FranchiseName = movie.Franchise?.Name;
    if (movie.Franchise != null)

```

		Денисенко Д.О.			ІІЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
		Сфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		130

```

        {
            dto.FranchiseMovies = movie.Franchise.Movies
                .OrderBy(fm => fm.OrderInFranchise)
                .Select(fm => new FranchiseMovieDto
                {
                    Id = fm.Id,
                    Title = fm.Title,
                    Order = fm.OrderInFranchise ?? 0,
                    IsCurrent = fm.Id == movie.Id
                }).ToList();
        }

        return Ok(dto);
    }

    [HttpPost("{id}/react")]
    [Authorize]
    public async Task<IActionResult> ReactToMovie(int id, [FromQuery]
        ReactionType type)
    {
        var userId =
            int.Parse(User.FindFirst(ClaimTypes.NameIdentifier)!.Value);

        bool isVoteGroup = (type == ReactionType.Like || type ==
            ReactionType.Dislike);

        var existingReaction = await _context.MovieReactions
            .FirstOrDefaultAsync(r =>
                r.MovieId == id &&
                r.UserId == userId &&
                (isVoteGroup
                    ? (r.Type == ReactionType.Like || r.Type ==
                        ReactionType.Dislike)
                    : (r.Type != ReactionType.Like && r.Type !=
                        ReactionType.Dislike))
            );

        if (existingReaction != null)
        {
            if (existingReaction.Type == type)
            {
                _context.MovieReactions.Remove(existingReaction);
            }
            else
            {
                existingReaction.Type = type;
            }
        }
        else
        {
            _context.MovieReactions.Add(new MovieReaction
            {
                MovieId = id,
                UserId = userId,
                Type = type
            });
        }

        bool hasReactionAward = await _context.UserAwards.AnyAsync(ua =>
            ua.UserId == userId && ua.Name == "Емоційний глядач");
        bool awardAdded = false;

        if (!hasReactionAward)
        {
            _context.UserAwards.Add(new UserAward

```

		Денисенко Д.О.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Сфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		131

```

        {
            UserId = userId,
            Name = "Емоційний глядач",
            Icon = "👍",
            Description = "За першу залишену реакцію на фільм"
        });
        awardAdded = true;
    }

    await _context.SaveChangesAsync();

    if (awardAdded)
    {
        await
            _notificationHubContext.Clients.User(userId.ToString()).SendAsync(
                "AchievementUnlocked",
                new { name = "Емоційний глядач", icon = "👍", description = "За
першу залишену реакцію на фільм" }
            );
    }

    return Ok();
}

[HttpGet("trending")]
public async Task<ActionResult<IEnumerable<MovieEntity>>> GetTrending()
{
    var blockedGenres = await GetUserBlockedGenres();
    var blockedActorIds = await GetUserBlockedActorIds();

    var moviesQuery = _context.Movies.Include(m =>
m.MovieActors).AsNoTracking().AsQueryable();

    foreach (var genre in blockedGenres)
    {
        moviesQuery = moviesQuery.Where(m => m.Genre == null ||
!m.Genre.ToLower().Contains(genre));
    }

    if (blockedActorIds.Any())
    {
        moviesQuery = moviesQuery.Where(m => !m.MovieActors.Any(ma =>
blockedActorIds.Contains(ma.ActorId)));
    }

    var trendingMovies = await _context.MovieReactions
        .Where(r => r.Type == ReactionType.Like)
        .GroupBy(r => r.MovieId)
        .Select(g => new { MovieId = g.Key, LikesCount = g.Count() })
        .OrderByDescending(x => x.LikesCount)
        .Take(10)
        .Join(moviesQuery, r => r.MovieId, m => m.Id, (r, m) => m)
        .Where(m => !string.IsNullOrEmpty(m.TrailerUrl))
        .ToListAsync();

    return Ok(trendingMovies);
}

[HttpGet("top-rated")]
public async Task<ActionResult<IEnumerable<MovieDetailDto>>> GetTopRated()
{
    var query = _context.Movies.Include(m =>
m.MovieActors).AsNoTracking().AsQueryable();

    var blockedGenres = await GetUserBlockedGenres();

```

		Денисенко Д.О.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Сфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		132

```

        foreach (var genre in blockedGenres)
        {
            query = query.Where(m => m.Genre == null ||
!m.Genre.ToLower().Contains(genre));
        }

        var blockedActorIds = await GetUserBlockedActorIds();
        if (blockedActorIds.Any())
        {
            query = query.Where(m => !m.MovieActors.Any(ma =>
blockedActorIds.Contains(ma.ActorId)));
        }

        var movies = await query
            // Apply same composite ordering as in GetAll for consistency
            .OrderByDescending(m => m.AverageRating + (double)m.TotalReviews /
(m.TotalReviews + 10))
            .Take(10)
            .Select(m => new MovieDetailDto
            {
                Id = m.Id,
                Title = m.Title,
                PosterUrl = m.PosterUrl,
                Year = m.Year,
                AverageRating = m.AverageRating,
                Genre = m.Genre,
                CreatedAt = m.CreatedAt,
                ViewsCount = m.ViewsCount,
                TotalReviews = m.TotalReviews
            })
            .ToListAsync();

        return Ok(movies);
    }
    private async Task<string?> DownloadAndSaveImage(string imageUrl)
    {
        try
        {
            if (string.IsNullOrEmpty(imageUrl))
                return null;

            if (imageUrl.StartsWith("/"))
                return imageUrl;

            using var handler = new HttpClientHandler
            {
                AllowAutoRedirect = true
            };

            using var client = new HttpClient(handler);

            client.DefaultRequestHeaders.Add("User-Agent",
                "Mozilla/5.0 (Windows NT 10.0; Win64; x64)");

            var response = await client.GetAsync(imageUrl);

            if (!response.IsSuccessStatusCode)
                return null;

            var imageBytes = await response.Content.ReadAsByteArrayAsync();

            var fileName = $"{Guid.NewGuid()}.jpg";

            var uploadsFolder = Path.Combine(_env.WebRootPath, "uploads");
            if (!Directory.Exists(uploadsFolder))

```

		Денисенко Д.О.			ІІЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
		Сфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		133

```

        Directory.CreateDirectory(uploadsFolder);

        var filePath = Path.Combine(uploadsFolder, fileName);

        await System.IO.File.WriteAllBytesAsync(filePath, imageBytes);

        return $"/uploads/{fileName}";
    }
    catch (Exception ex)
    {
        Console.WriteLine($"Не вдалося завантажити картинку: {ex.Message}");
        return null;
    }
}

[HttpPost]
[Authorize(Roles = "Admin")]
public async Task<IActionResult> Create(MovieDetailDto dto)
{
    try
    {
        var exists = await _context.Movies.AnyAsync(m =>
            m.Title.ToLower() == dto.Title.ToLower() &&
            m.Year == dto.Year);

        if (exists)
            return BadRequest($"Фільм '{dto.Title}' ({dto.Year}) вже існує в
базі даних!");

        var movie = new MovieEntity
        {
            Title = dto.Title,
            Description = dto.Description,
            Year = dto.Year,
            Genre = dto.Genre,
            Director = dto.Director,
            TrailerUrl = dto.TrailerUrl,
            IsSeries = dto.IsSeries,
            CreatedAt = DateTime.UtcNow,
            PosterUrl = (!string.IsNullOrEmpty(dto.PosterUrl) &&
dto.PosterUrl.StartsWith("http"))
                ? await DownloadAndSaveImage(dto.PosterUrl)
                : dto.PosterUrl,
            AverageRating = 0,
            TotalReviews = 0,
            MalId = dto.MalId,
            TmdbId = dto.TmdbId,
            CastImported = false,
            MovieActors = new List<MovieActor>()
        };

        if (dto.Cast != null)
        {
            var distinctCast = dto.Cast.GroupBy(c => c.ActorId).Select(g =>
g.First());

            foreach (var castMember in distinctCast)
            {
                if (castMember.ActorId > 0)
                {
                    movie.MovieActors.Add(new MovieActor
                    {
                        ActorId = castMember.ActorId,
                        IsMainRole = castMember.IsMainRole,
                        Role = castMember.Role
                    });
                }
            }
        }
    }
    catch { }
}

```

		Денисенко Д.О.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Сфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		134

```

        });
    }
}

if (!string.IsNullOrWhiteSpace(dto.NewFranchiseName))
{
    var newFranchise = new Franchise { Name = dto.NewFranchiseName };
    _context.Franchises.Add(newFranchise);
    await _context.SaveChangesAsync();
    movie.FranchiseId = newFranchise.Id;
}
else
{
    movie.FranchiseId = dto.FranchiseId > 0 ? dto.FranchiseId :
null;
}

movie.OrderInFranchise = dto.OrderInFranchise > 0 ?
dto.OrderInFranchise : null;

_context.Movies.Add(movie);
await _context.SaveChangesAsync();

return Ok(new { id = movie.Id });
}
catch (Exception ex)
{
    Console.WriteLine($"❌ ПОМИЛКА ЗБЕРЕЖЕННЯ: {ex.Message}");
    if (ex.InnerException != null)
        Console.WriteLine($"🔍 ДЕТАЛІ SQL:
{ex.InnerException.Message}");

    return StatusCode(500, new
    {
        message = "Помилка сервера",
        details = ex.InnerException?.Message ?? ex.Message
    });
}
}

[HttpPut("{id}")]
[Authorize(Roles = "Admin")]
public async Task<IActionResult> Update(int id, MovieDetailDto dto)
{
    try
    {
        var movie = await _context.Movies
            .Include(m => m.MovieActors)
            .FirstOrDefaultAsync(m => m.Id == id);

        if (movie == null) return NotFound();

        movie.Title = dto.Title;
        movie.Description = dto.Description;
        movie.Year = dto.Year;
        movie.Genre = dto.Genre;
        movie.Director = dto.Director;
        movie.TrailerUrl = dto.TrailerUrl;
        movie.IsSeries = dto.IsSeries;
        movie.MalId = dto.MalId;
        movie.TmdbId = dto.TmdbId;

        if (movie.PosteriorUrl != dto.PosteriorUrl)

```

		Денисенко Д.О.			ІІЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
		Сфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		135

```

        {
            movie.PosterUrl = (!string.IsNullOrEmpty(dto.PosterUrl) &&
dto.PosterUrl.StartsWith("http"))
                ? await DownloadAndSaveImage(dto.PosterUrl)
                : dto.PosterUrl;
        }

        if (!string.IsNullOrWhiteSpace(dto.NewFranchiseName))
        {
            var newFranchise = new Franchise { Name = dto.NewFranchiseName
};
            _context.Franchises.Add(newFranchise);
            await _context.SaveChangesAsync();
            movie.FranchiseId = newFranchise.Id;
        }
        else
        {
            movie.FranchiseId = dto.FranchiseId > 0 ? dto.FranchiseId :
null;
        }

        movie.OrderInFranchise = dto.OrderInFranchise > 0 ?
dto.OrderInFranchise : null;

        if (dto.Cast != null)
        {
            var distinctCast = dto.Cast.GroupBy(c => c.ActorId).Select(g =>
g.First()).ToList();
            var existingActorIds = movie.MovieActors.Select(ma =>
ma.ActorId).ToList();

            var actorsToRemove = movie.MovieActors.Where(ma =>
!distinctCast.Any(dc => dc.ActorId == ma.ActorId)).ToList();
            foreach (var remove in actorsToRemove)
                movie.MovieActors.Remove(remove);

            foreach (var castMember in distinctCast)
            {
                var existing = movie.MovieActors.FirstOrDefault(ma =>
ma.ActorId == castMember.ActorId);
                if (existing != null)
                {
                    existing.Role = castMember.Role;
                    existing.IsMainRole = castMember.IsMainRole;
                }
                else if (castMember.ActorId > 0)
                {
                    movie.MovieActors.Add(new MovieActor
                    {
                        ActorId = castMember.ActorId,
                        IsMainRole = castMember.IsMainRole,
                        Role = castMember.Role
                    });
                }
            }
        }

        await _context.SaveChangesAsync();
        return NoContent();
    }
    catch (Exception ex)
    {
        Console.WriteLine($"❌ ПОМИЛКА ОНОВЛЕННЯ ФІЛЬМУ: {ex.Message}");
        if (ex.InnerException != null) Console.WriteLine($"🔍 ДЕТАЛІ:
{ex.InnerException.Message}");
    }

```

		Денисенко Д.О.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Сфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		136

```

        return StatusCode(500, new { message = "Помилка при оновленні",
details = ex.Message });
    }
}
[HttpDelete("{id}")]
[Authorize(Roles = "Admin")]
public async Task<IActionResult> Delete(int id)
{
    var movie = await _context.Movies.FindAsync(id);
    if (movie == null) return NotFound();

    _context.Movies.Remove(movie);
    await _context.SaveChangesAsync();

    return NoContent();
}

[HttpGet("new")]
public async Task<ActionResult<IEnumerable<MovieDetailDto>>> GetNew()
{
    var query = _context.Movies.Include(m =>
m.MovieActors).AsNoTracking().AsQueryable();

    var blockedGenres = await GetUserBlockedGenres();
    foreach (var genre in blockedGenres)
    {
        query = query.Where(m => m.Genre == null ||
!m.Genre.ToLower().Contains(genre));
    }

    var blockedActorIds = await GetUserBlockedActorIds();
    if (blockedActorIds.Any())
    {
        query = query.Where(m => !m.MovieActors.Any(ma =>
blockedActorIds.Contains(ma.ActorId)));
    }

    var movies = await query
.OrderByDescending(m => m.CreatedAt)
.Take(10)
.Select(m => new MovieDetailDto
{
    Id = m.Id,
    Title = m.Title,
    PosterUrl = m.PosterUrl,
    Year = m.Year,
    AverageRating = m.AverageRating,
    Genre = m.Genre,
    CreatedAt = m.CreatedAt,
    ViewsCount = m.ViewsCount,
    TotalReviews = m.TotalReviews
})
.ToListAsync();

    return Ok(movies);
}

[HttpGet("filters")]
public async Task<ActionResult<MovieFiltersDto>> GetFilters()
{
    var rawData = await _context.Movies
.AsNoTracking()
.Select(m => new { m.Genre, m.Year })
.ToListAsync();

```

		Денисенко Д.О.			ІПЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
		Сфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		137

```

var genres = rawData
    .Where(m => !string.IsNullOrEmpty(m.Genre))
    .SelectMany(m => m.Genre.Split(','))
    .Select(g => g.Trim().ToLower())
    .Distinct()
    .OrderBy(g => g)
    .ToList();

var years = rawData
    .Select(m => m.Year)
    .Distinct()
    .OrderByDescending(y => y)
    .ToList();

var awards = await _context.MovieAwards
    .AsNoTracking()
    .Select(ma => new AwardDto
    {
        Id = ma.Id,
        Name = ma.Name,
        Icon = ma.Icon
    })
    .Distinct()
    .OrderBy(a => a.Name)
    .ToListAsync();

return Ok(new MovieFiltersDto { Genres = genres, Years = years, Awards =
awards });
}
private async Task<List<string>> GetUserBlockedGenres()
{
    var userIdClaim = User.FindFirst(ClaimTypes.NameIdentifier);
    if (userIdClaim == null) return new List<string>();

    var userId = int.Parse(userIdClaim.Value);

    var userSettings = await _context.Users
        .AsNoTracking()
        .Where(u => u.Id == userId)
        .Select(u => u.BlockedGenres)
        .FirstOrDefaultAsync();

    if (string.IsNullOrEmpty(userSettings)) return new List<string>();

    return userSettings.ToLower().Split(',',
StringSplitOptions.RemoveEmptyEntries)
        .Select(g => g.Trim())
        .ToList();
}

private async Task<List<int>> GetUserBlockedActorIds()
{
    var userIdClaim = User.FindFirst(ClaimTypes.NameIdentifier);
    if (userIdClaim == null) return new List<int>();

    var userId = int.Parse(userIdClaim.Value);

    return await _context.UserBlockedActors
        .Where(uba => uba.UserId == userId)
        .Select(uba => uba.ActorId)
        .ToListAsync();
}

[HttpGet("random")]

```

		Денисенко Д.О.			ІІЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
		Сфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		138

```

public async Task<ActionResult<object>> GetRandom()
{
    var blockedGenres = await GetUserBlockedGenres();
    var query = _context.Movies.AsNoTracking().AsQueryable();

    foreach (var genre in blockedGenres)
    {
        query = query.Where(m => m.Genre == null ||
!m.Genre.ToLower().Contains(genre));
    }

    var movieIds = await query.Select(m => m.Id).ToListAsync();

    if (!movieIds.Any())
    {
        return NotFound("Фільмів не знайдено (можливо, занадто суворі
фільтри)");
    }

    var random = new Random();
    var randomId = movieIds[random.Next(movieIds.Count)];

    return Ok(new { id = randomId });
}

[HttpPost("{id}/history")]
[Authorize]
public async Task<ActionResult> AddToHistory(int id)
{
    var userId =
int.Parse(User.FindFirst(ClaimTypes.NameIdentifier)!.Value);
    var existingItem = await _context.WatchHistory
        .FirstOrDefaultAsync(h => h.UserId == userId && h.MovieId == id);

    if (existingItem != null)
    {
        existingItem.ViewedAt = DateTime.UtcNow;
    }
    else
    {
        _context.WatchHistory.Add(new WatchHistoryItem
        {
            UserId = userId,
            MovieId = id,
            ViewedAt = DateTime.UtcNow
        });
    }

    await _context.SaveChangesAsync();
    return Ok();
}

[HttpGet("history")]
[Authorize]
public async Task<ActionResult<IEnumerable<MovieDetailDto>>> GetHistory(
    [FromQuery] int page = 1,
    [FromQuery] int pageSize = 20)
{
    var userId =
int.Parse(User.FindFirst(ClaimTypes.NameIdentifier)!.Value);

    var historyQuery = _context.WatchHistory
        .AsNoTracking()
        .Where(h => h.UserId == userId)
        .OrderByDescending(h => h.ViewedAt)

```

		Денисенко Д.О.			ІПЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
		Сфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		139

```

        .Include(h => h.Movie);

var movies = await historyQuery
    .Skip((page - 1) * pageSize)
    .Take(pageSize)
    .Select(h => new MovieDetailDto
    {
        Id = h.Movie.Id,
        Title = h.Movie.Title,
        PosterUrl = h.Movie.PosterUrl,
        Year = h.Movie.Year,
        Description = h.Movie.Description,
        AverageRating = h.Movie.AverageRating,
        Genre = h.Movie.Genre,
        CreatedAt = h.Movie.CreatedAt
    })
    .ToListAsync();

return Ok(movies);
}
[HttpPost("rate-episode/{episodeId}")]
[Authorize]
public async Task<IActionResult> RateEpisode(int episodeId, [FromQuery] int
rating)
{
    if (rating < 1 || rating > 10) return BadRequest("Оцінка має бути від 1
до 10");

    var userId =
int.Parse(User.FindFirst(ClaimTypes.NameIdentifier)!.Value);

    var episode = await _context.Episodes
        .Include(e => e.Movie)
        .FirstOrDefaultAsync(e => e.Id == episodeId);

    if (episode == null) return NotFound("Епізод не знайдено");

    var currentRating = await _context.UserEpisodeRatings
        .FirstOrDefaultAsync(r => r.UserId == userId && r.EpisodeId ==
episodeId);

    if (currentRating != null)
    {
        currentRating.Rating = rating;
    }
    else
    {
        currentRating = new UserEpisodeRating
        {
            UserId = userId,
            EpisodeId = episodeId,
            Rating = rating
        };
        _context.UserEpisodeRatings.Add(currentRating);
    }

    await _context.SaveChangesAsync();

    var episodeAvg = await _context.UserEpisodeRatings
        .Where(r => r.EpisodeId == episodeId)
        .AverageAsync(r => r.Rating);

    episode.AverageRating = episodeAvg;

```

		Денисенко Д.О.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Сфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		140

```

        var allEpisodesRatings = await _context.Episodes
            .Where(e => e.MovieId == episode.MovieId)
            .Select(e => e.Id == episodeId ? episodeAvg : e.AverageRating)
            .ToListAsync();

        var activeEpisodes = allEpisodesRatings.Where(r => r > 0).ToList();

        double seriesAvg = 0;
        if (activeEpisodes.Any())
        {
            seriesAvg = activeEpisodes.Average();
        }

        if (episode.Movie != null)
        {
            episode.Movie.AverageRating = seriesAvg;
        }

        await _context.SaveChangesAsync();

        return Ok(new { episodeAverage = episodeAvg, seriesAverage = seriesAvg
    });
    }

    [HttpPost("{movieId}/add-episode")]
    [Authorize(Roles = "Admin")]
    public async Task<IActionResult> AddEpisode(int movieId, [FromBody]
EpisodeDto dto)
    {
        var episode = new Episode
        {
            MovieId = movieId,
            SeasonNumber = dto.SeasonNumber,
            EpisodeNumber = dto.EpisodeNumber,
            Title = dto.Title,
            AverageRating = 0
        };
        _context.Episodes.Add(episode);
        await _context.SaveChangesAsync();
        return Ok();
    }

    [HttpPut("episodes/{id}")]
    [Authorize(Roles = "Admin")]
    public async Task<IActionResult> UpdateEpisode(int id, [FromBody] EpisodeDto
dto)
    {
        var episode = await _context.Episodes.FindAsync(id);
        if (episode == null) return NotFound("Епізод не знайдено");

        episode.SeasonNumber = dto.SeasonNumber;
        episode.EpisodeNumber = dto.EpisodeNumber;
        episode.Title = dto.Title ?? string.Empty;

        await _context.SaveChangesAsync();
        return NoContent();
    }

    [HttpDelete("episodes/{id}")]
    [Authorize(Roles = "Admin")]
    public async Task<IActionResult> DeleteEpisode(int id)
    {
        var episode = await _context.Episodes.FindAsync(id);
        if (episode == null) return NotFound("Епізод не знайдено");
    }

```

```

        var ratings = _context.UserEpisodeRatings.Where(r => r.EpisodeId == id);
        _context.UserEpisodeRatings.RemoveRange(ratings);

        _context.Episodes.Remove(episode);
        await _context.SaveChangesAsync();
        return NoContent();
    }

    [HttpGet("tmdb/search")]
    [Authorize(Roles = "Admin")]
    public async Task<ActionResult<List<TmdbSearchResultDto>>>
SearchTmdb([FromQuery] string query)
    {
        if (string.IsNullOrEmpty(query)) return BadRequest();

        using var client = new HttpClient();
        var searchUrl =
$"https://api.themoviedb.org/3/search/movie?api_key={TMDB_API_KEY}&query={Uri.Escape
DataString(query)}&language=uk-UA";

        var searchResponse = await
client.GetFromJsonAsync<TmdbSearchResult>(searchUrl);

        if (searchResponse?.Results == null) return Ok(new
List<TmdbSearchResultDto>());

        var results = searchResponse.Results.Select(m => new TmdbSearchResultDto
        {
            TmdbId = m.Id,
            Title = m.Title,
            ReleaseDate = m.Release_Date,
            PosterUrl = !string.IsNullOrEmpty(m.Poster_Path)
                ? $"https://image.tmdb.org/t/p/w92{m.Poster_Path}"
                : null
        }).ToList();

        return Ok(results);
    }

    [HttpGet("tmdb/details/{tmdbId}")]
    [Authorize(Roles = "Admin")]
    public async Task<ActionResult<MovieDetailDto>> GetTmdbDetails(int tmdbId)
    {
        using var client = new HttpClient();

        var url =
$"https://api.themoviedb.org/3/movie/{tmdbId}?api_key={TMDB_API_KEY}&append_to_respo
nse=credits&language=uk-UA";
        var response = await client.GetAsync(url);

        if (!response.IsSuccessStatusCode) return NotFound("Film not found in
TMDB");

        var json = await response.Content.ReadAsStringAsync();
        dynamic data = Newtonsoft.Json.JsonConvert.DeserializeObject(json);

        var movieDto = new MovieDetailDto
        {
            Title = data.title,
            Description = data.overview,
            PosterUrl = data.poster_path != null ?
$"https://image.tmdb.org/t/p/original{data.poster_path}" : null,
            Year = 0,
            TmdbId = tmdbId,
            Cast = new List<CastDto>()

```

		Денисенко Д.О.			ІІЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
		Сфремов ІО.М.				142
Змн.	Арк.	№ докум.	Підпис	Дата		

```

};

string releaseDate = data.release_date;
if (DateTime.TryParse(releaseDate, out DateTime date))
{
    movieDto.Year = date.Year;
}

if (data.credits != null && data.credits.cast != null)
{
    var castMembers =
((IEnumerable<dynamic>)data.credits.cast).Take(50);

    var translator = new AggregateTranslator();

    foreach (var person in castMembers)
    {
        await Task.Delay(30);
        string name = person.name;
        string role = person.character;
        string profilePath = person.profile_path;
        int personId = person.id;

        if (System.Text.RegularExpressions.Regex.IsMatch(name, @"[a-zA-
Z]"))
        {
            try
            {
                var nameResult = await translator.TranslateAsync(name,
"uk");

                name = nameResult.Translation;
            }
            catch (Exception ex)
            {
                Console.ForegroundColor = ConsoleColor.Yellow;
                Console.WriteLine($"⚠ Не вдалося перекласти біографію
для '{name}': {ex.Message}");
                Console.ResetColor();
            }
        }

        int order = person.order != null ? (int)person.order : 999;
        bool isMainRole = order < 6;

        var existingActor = await _context.Actors.FirstOrDefaultAsync(a
=> a.Name == name);

        string biography = "";
        DateTime? birthDate = null;

        if (existingActor == null)
        {
            var personUrlUk =
$"https://api.themoviedb.org/3/person/{personId}?api_key={TMDB_API_KEY}&language=uk-
UA";

            var personResponse = await client.GetAsync(personUrlUk);

            if (personResponse.IsSuccessStatusCode)
            {
                var personJson = await
personResponse.Content.ReadAsStringAsync();
                dynamic personData =
Newtonsoft.Json.JsonConvert.DeserializeObject(personJson);

                biography = personData.biography ?? "";

```

		Денисенко Д.О.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Сфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		143

```

        string bdayStr = personData.birthday;
        if (DateTime.TryParse(bdayStr, out DateTime parsedDate))
        {
            birthDate = parsedDate;
        }
        if (string.IsNullOrEmpty(biography))
        {
            var personUrlEn =
                $"https://api.themoviedb.org/3/person/{personId}?api_key={TMDB_API_KEY}&language=en-US";
            var enResponse = await client.GetAsync(personUrlEn);
            if (enResponse.IsSuccessStatusCode)
            {
                var enJson = await
                    enResponse.Content.ReadAsStringAsync();
                dynamic enData =
                    Newtonsoft.Json.JsonConvert.DeserializeObject(enJson);
                string enBio = enData.biography ?? "";

                if (!string.IsNullOrEmpty(enBio))
                {
                    try
                    {
                        var bioResult = await
                            translator.TranslateAsync(enBio, "uk");
                        biography = bioResult.Translation;
                    }
                    catch { biography = enBio; }
                }
            }
        }

        movieDto.Cast.Add(new CastDto
        {
            ActorId = existingActor?.Id ?? 0,
            Name = name,
            Role = role,
            Biography = existingActor?.Bio ?? biography,
            PhotoUrl = !string.IsNullOrEmpty(profilePath) ?
                $"https://image.tmdb.org/t/p/w500{profilePath}" : null,
            IsMainRole = isMainRole,
            BirthDate = existingActor?.BirthDate ?? birthDate
        });
    }

    return Ok(movieDto);
}

[HttpGet("latest")]
[AllowAnonymous]
public async Task<IActionResult> GetLatestMovie()
{
    var latestMovie = await _context.Movies
        .OrderByDescending(m => m.Id)
        .Select(m => new
        {
            m.Id,
            m.Title
        })
        .FirstOrDefaultAsync();
}

```

		Денисенко Д.О.			ІІЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
		Сфремов І.О.М.				144
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        if (latestMovie == null)
        {
            return NotFound();
        }

        return Ok(latestMovie);
    }

    [HttpPost("{id}/increment-view")]
    public async Task<IActionResult> IncrementViewCount(int id)
    {
        var movie = await _context.Movies.FindAsync(id);
        if (movie == null) return NotFound("Фільм не знайдено.");

        movie.ViewsCount += 1;

        await _context.SaveChangesAsync();

        return Ok(new { viewsCount = movie.ViewsCount });
    }

    [HttpGet("franchises")]
    public async Task<IActionResult> GetFranchises()
    {
        var franchises = await _context.Franchises
            .Select(f => new { f.Id, f.Name })
            .OrderBy(f => f.Name)
            .ToListAsync();

        return Ok(franchises);
    }

    [Authorize(Roles = "Admin")]
    [HttpPost("{movieId}/import-mal/{malAnimeId}")]
    public async Task<IActionResult> ImportFromMal(int movieId, int malAnimeId,
    [FromServices] MalIntegrationService malService)
    {
        var movieExists = await _context.Movies.AnyAsync(m => m.Id == movieId);
        if (!movieExists) return NotFound("Фільм/Аніме не знайдено в локальній
        бази");

        var success = await malService.ImportCharactersAsync(movieId,
        malAnimeId);

        if (success) return Ok(new { Message = "Персонажів успішно імпортовано з
        MyAnimeList!" });

        return BadRequest("Не вдалося завантажити дані з MAL.");
    }

    [AllowAnonymous]
    [HttpPost("{movieId}/auto-import-cast")]
    public async Task<IActionResult> AutoImportCast(int movieId, [FromServices]
    MalIntegrationService malService)
    {
        var movie = await _context.Movies
            .Include(m => m.MovieActors)
            .Include(m => m.VoiceActingRoles)
            .ThenInclude(v => v.Character)
            .FirstOrDefaultAsync(m => m.Id == movieId);

        if (movie == null) return NotFound("Фільм не знайдено");

        if (movie.CastImported)

```

		Денисенко Д.О.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Сфремов Ю.М.				145
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        return Ok(new { Message = "Касту вже імпортовано", AlreadyImported =
true });

        try
        {
            string importSource = "none";

            if (movie.IsSeries && movie.MalId.HasValue && movie.MalId > 0)
            {
                Console.WriteLine($"📺 Імпорт персонажів з MAL ({movie.MalId})
для аніме '{movie.Title}'");
                await malService.ImportCharactersAsync(movieId,
movie.MalId.Value);
                importSource = "mal-characters";
            }
            else if (!movie.IsSeries && movie.TmdbId.HasValue && movie.TmdbId >
0)
            {
                Console.WriteLine($"🎬 Імпорт акторів з TMDB ({movie.TmdbId})
для фільму '{movie.Title}'");
                await ImportActorsFromTmdb(movieId, movie.TmdbId.Value);
                importSource = "tmdb-actors";
            }
            else if (movie.MalId.HasValue && movie.MalId > 0)
            {
                Console.WriteLine($"📺 Резервний імпорт персонажів з MAL
({movie.MalId}) для '{movie.Title}'");
                await malService.ImportCharactersAsync(movieId,
movie.MalId.Value);
                importSource = "mal-fallback";
            }

            movie.CastImported = true;
            await _context.SaveChangesAsync();

            return Ok(new
            {
                Message = "Касту успішно автоімпортовано",
                ImportSource = importSource,
                IsAnime = movie.IsSeries,
                movie.MalId,
                movie.TmdbId
            });
        }
        catch (Exception ex)
        {
            Console.WriteLine($"❌ Помилка при автоімпорті касту:
{ex.Message}");
            return StatusCode(500, new { Message = "Помилка при імпорті касту",
Details = ex.Message });
        }
    }

    [HttpGet("latest-releases")]
    [AllowAnonymous]
    public async Task<ActionResult<IEnumerable<object>>> GetLatestReleases()
    {
        var latestMovies = await _context.Movies
            .Include(m => m.Episodes)
            .OrderByDescending(m => m.CreatedAt)
            .Take(5)
            .ToListAsync();

        var result = latestMovies.Select(m => new
        {

```

		Денисенко Д.О.			ІПЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
		Сфремов Ю.М.				146
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        m.Id,
        m.Title,
        m.Description,
        m.PosterUrl,
        BannerUrl = m.PosterUrl,
        Type = m.IsSeries ? "Series" : "Movie",

        LatestSeason = m.IsSeries && m.Episodes != null && m.Episodes.Any()
            ? m.Episodes.Max(e => e.SeasonNumber)
            : (int?)null,

        LatestEpisode = m.IsSeries && m.Episodes != null && m.Episodes.Any()
            ? m.Episodes
                .Where(e => e.SeasonNumber == m.Episodes.Max(s =>
s.SeasonNumber))
                .Max(e => e.EpisodeNumber)
            : (int?)null
    });

    return Ok(result);
}

[HttpGet("{id}/similar")]
[AllowAnonymous]
public async Task<ActionResult<IEnumerable<object>>> GetSimilarMovies(int
id)
{
    var currentMovie = await _context.Movies.FindAsync(id);
    if (currentMovie == null) return NotFound("Фільм не знайдено");

    if (string.IsNullOrEmpty(currentMovie.Genre))
        return Ok(new List<object>());

    var currentGenres = currentMovie.Genre
        .Split(',', StringSplitOptions.RemoveEmptyEntries)
        .Select(g => g.Trim().ToLower())
        .ToList();

    int requiredMatches = Math.Min(3, currentGenres.Count);

    var candidateMovies = await _context.Movies
        .Where(m => m.Id != id && !string.IsNullOrEmpty(m.Genre))
        .Select(m => new
        {
            m.Id,
            m.Title,
            m.PosterUrl,
            m.AverageRating,
            m.Year,
            m.Genre,
            m.IsSeries,
            m.ViewsCount,
            m.TotalReviews
        })
        .ToListAsync();

    var similarMovies = candidateMovies
        .Select(m =>
        {
            var movieGenres = m.Genre
                .Split(',', StringSplitOptions.RemoveEmptyEntries)
                .Select(g => g.Trim().ToLower())
                .ToList();

```

```

        int matchCount =
currentGenres.Intersect(movieGenres).Count();

        return new { Movie = m, MatchCount = matchCount };
    })
    .Where(x => x.MatchCount >= requiredMatches)
    .OrderByDescending(x => x.MatchCount)
    .ThenByDescending(x => x.Movie.AverageRating)
    .Take(10)
    .Select(x => new
    {
        id = x.Movie.Id,
        title = x.Movie.Title,
        posterUrl = x.Movie.PosterUrl,
        averageRating = x.Movie.AverageRating,
        year = x.Movie.Year,
        genre = x.Movie.Genre,
        type = x.Movie.IsSeries ? "Series" : "Movie",
        viewsCount = x.Movie.ViewsCount,
        totalReviews = x.Movie.TotalReviews
    })
    .ToList();

    return Ok(similarMovies);
}

private async Task ImportActorsFromTmdb(int movieId, int tmdbId)
{
    using var client = new HttpClient();

    var url =
    $"https://api.themoviedb.org/3/movie/{tmdbId}?api_key={TMDB_API_KEY}&append_to_response=credits&language=uk-UA";
    var response = await client.GetAsync(url);

    if (!response.IsSuccessStatusCode) return;

    var json = await response.Content.ReadAsStringAsync();
    dynamic data = Newtonsoft.Json.JsonConvert.DeserializeObject(json);

    var movie = await _context.Movies
        .Include(m => m.MovieActors)
        .Include(m => m.VoiceActingRoles)
        .FirstOrDefaultAsync(m => m.Id == movieId);

    if (movie == null) return;

    if (data.credits == null || data.credits.cast == null) return;

    var castMembers = ((IEnumerable<dynamic>)data.credits.cast).Take(30);
    var translator = new AggregateTranslator();

    foreach (var person in castMembers)
    {
        try
        {
            await Task.Delay(20);
            string actorName = person.name;
            string characterName = person.character;
            string profilePath = person.profile_path;
            int personId = person.id;

            if (string.IsNullOrEmpty(actorName) ||
                string.IsNullOrEmpty(characterName)) continue;

```

```

        if (System.Text.RegularExpressions.Regex.IsMatch(actorName,
@"[a-zA-Z]"))
        {
            try
            {
                var nameResult = await
translator.TranslateAsync(actorName, "uk");
                actorName = nameResult.Translation;
            }
            catch { }
        }

        if (System.Text.RegularExpressions.Regex.IsMatch(characterName,
@"[a-zA-Z]"))
        {
            try
            {
                var charResult = await
translator.TranslateAsync(characterName, "uk");
                characterName = charResult.Translation;
            }
            catch { }
        }

        int order = person.order != null ? (int)person.order : 999;
        bool isMainRole = order < 6;

        var character = await _context.Characters.FirstOrDefaultAsync(c
=> c.Name == characterName);

        if (character == null)
        {
            character = new Character
            {
                Name = characterName,
                ImageUrl = null
            };
            _context.Characters.Add(character);
            await _context.SaveChangesAsync();
        }

        var actor = await _context.Actors.FirstOrDefaultAsync(a =>
a.Name == actorName);

        if (actor == null)
        {
            string biography = "";
            DateTime? birthDate = null;

            var personUrl =
$"https://api.themoviedb.org/3/person/{personId}?api_key={TMDB_API_KEY}&language=uk-
UA";

            try
            {
                var personResponse = await client.GetAsync(personUrl);
                if (personResponse.IsSuccessStatusCode)
                {
                    var personJson = await
personResponse.Content.ReadAsStringAsync();
                    dynamic personData =
Newtonsoft.Json.JsonConvert.DeserializeObject(personJson);
                    biography = personData.biography ?? "";

                    if (personData.birthday != null &&
!string.IsNullOrEmpty(personData.birthday.ToString()))

```

		Денисенко Д.О.			ІІЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
		Сфремов І.О.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		149

```

        {
            if
(DateTime.TryParse(personData.birthday.ToString(), out DateTime bd))
                birthDate = bd;
        }
    }
}
catch { }

actor = new Actor
{
    Name = actorName,
    Bio = biography,
    PhotoUrl = !string.IsNullOrEmpty(profilePath) ?
$"https://image.tmdb.org/t/p/w500{profilePath}" : null,
    BirthDate = birthDate
};

_context.actors.Add(actor);
await _context.SaveChangesAsync();
}

var voiceRoleExists = await _context.VoiceActingRoles
.AnyAsync(v => v.CharacterId == character.Id && v.ActorId ==
actor.Id && v.MovieId == movieId);

if (!voiceRoleExists)
{
    _context.VoiceActingRoles.Add(new VoiceActingRole
    {
        CharacterId = character.Id,
        ActorId = actor.Id,
        MovieId = movieId,
        IsMainRole = isMainRole
    });
}
}
catch (Exception ex)
{
    Console.WriteLine($"⚠ Помилка при обробці персонажа:
{ex.Message}");
}

await _context.SaveChangesAsync();
}
[HttpGet("{movieId}/quiz-questions")]
[Authorize]
public async Task<ActionResult<List<QuizQuestionDto>>> GetQuizQuestions(int
movieId)
{
    var movie = await _context.Movies.FirstOrDefaultAsync(m => m.Id ==
movieId);
    if (movie == null)
        return NotFound("Фільм не знайдено");

    var questions = GenerateMovieQuizQuestions(movie);

    return Ok(questions);
}

[HttpPost("{movieId}/submit-quiz")]
[Authorize]
public async Task<ActionResult<object>> SubmitQuiz(int movieId, [FromBody]
QuizSubmissionDto submission)

```

		Денисенко Д.О.			ІІЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
		Сфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		150

```

        {
            var userId = int.Parse(User.FindFirst(ClaimTypes.NameIdentifier)?.Value
            ?? "0");
            if (userId == 0)
                return Unauthorized();

            var movie = await _context.Movies.FirstOrDefaultAsync(m => m.Id ==
            movieId);
            if (movie == null)
                return NotFound("Фільм не знайдено");

            var quizQuestions = GenerateMovieQuizQuestions(movie);
            int correctAnswers = 0;

            for (int i = 0; i < submission.Answers.Count; i++)
            {
                if (i < quizQuestions.Count &&
                    submission.Answers[i].SelectedOptionIndex ==
            quizQuestions[i].CorrectAnswerIndex)
                {
                    correctAnswers++;
                }
            }

            int rating = (int)Math.Round(((double)correctAnswers /
            quizQuestions.Count) * 100);

            var user = await _context.Users.FindAsync(userId);
            var existingResult = await _context.MovieQuizResults
            .FirstOrDefaultAsync(r => r.UserId == userId && r.MovieId ==
            movieId);

            if (existingResult != null)
            {
                if (rating > existingResult.Rating)
                {
                    int difference = rating - existingResult.Rating;
                    if (user != null) user.TestScore += difference;
                }
            }
            else
            {
                if (user != null) user.TestScore += rating;

                var newResult = new MovieQuizResult
                {
                    UserId = userId,
                    MovieId = movieId,
                    Rating = rating,
                    CorrectAnswers = correctAnswers,
                    TotalQuestions = quizQuestions.Count,
                    CompletedAt = DateTime.UtcNow
                };
                _context.MovieQuizResults.Add(newResult);
            }

            bool perfectAwardAdded = false;
            if (rating == 100)
            {
                bool hasPerfectAward = await _context.UserAwards.AnyAsync(ua =>
            ua.UserId == userId && ua.Name == "Ідеальний знавець");
                if (!hasPerfectAward)
                {
                    _context.UserAwards.Add(new UserAward

```

		Денисенко Д.О.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Сфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		151

```

        {
            UserId = userId,
            Name = "Ідеальний знавець",
            Icon = "100",
            Description = "За ідеальний результат тесту (100%)"
        });
        perfectAwardAdded = true;
    }
}

bool warriorAwardAdded = false;
int totalCompleted = await _context.MovieQuizResults
    .Where(r => r.UserId == userId)
    .Select(r => r.MovieId)
    .Distinct()
    .CountAsync();

if (existingResult == null) totalCompleted++;

if (totalCompleted == 10)
{
    bool hasWarriorAward = await _context.UserAwards.AnyAsync(ua =>
        ua.UserId == userId && ua.Name == "Тестовий воїн");
    if (!hasWarriorAward)
    {
        _context.UserAwards.Add(new UserAward
        {
            UserId = userId,
            Name = "Тестовий воїн",
            Icon = "X",
            Description = "За проходження 10 тестів знання"
        });
        warriorAwardAdded = true;
    }
}

await _context.SaveChangesAsync();

if (perfectAwardAdded)
{
    await
        _notificationHubContext.Clients.User(userId.ToString()).SendAsync(
            "AchievementUnlocked",
            new { name = "Ідеальний знавець", icon = "100", description =
                "За ідеальний результат тесту (100%)" }
        );
}

if (warriorAwardAdded)
{
    await
        _notificationHubContext.Clients.User(userId.ToString()).SendAsync(
            "AchievementUnlocked",
            new { name = "Тестовий воїн", icon = "X", description = "За
                проходження 10 тестів знання" }
        );
}

var userStats = await GetUserQuizStats(userId);

return Ok(new { rating, stats = userStats });
}

[HttpGet("quiz-stats")]
[Authorize]

```

		Денисенко Д.О.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Сфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		152

```

public async Task<ActionResult<UserQuizStatsDto>> GetQuizStats()
{
    var userId = int.Parse(User.FindFirst(ClaimTypes.NameIdentifier)?.Value
?? "0");
    if (userId == 0)
        return Unauthorized();

    var stats = await GetUserQuizStats(userId);
    return Ok(stats);
}

private async Task<UserQuizStatsDto> GetUserQuizStats(int userId)
{
    var results = await _context.MovieQuizResults
        .Where(r => r.UserId == userId)
        .Include(r => r.Movie)
        .OrderByDescending(r => r.CompletedAt)
        .ToListAsync();

    var stats = new UserQuizStatsDto
    {
        TotalQuizzesCompleted = results.Count,
        AverageRating = results.Count > 0 ? results.Average(r => r.Rating) :
0,

        HighestRating = results.Count > 0 ? results.Max(r => r.Rating) : 0,
        LowestRating = results.Count > 0 ? results.Min(r => r.Rating) : 0,
        RecentResults = results.Take(10).Select(r => new MovieQuizResultDto
        {
            Id = r.Id,
            MovieId = r.MovieId,
            MovieTitle = r.Movie.Title,
            Rating = r.Rating,
            CorrectAnswers = r.CorrectAnswers,
            TotalQuestions = r.TotalQuestions,
            CompletedAt = r.CompletedAt
        }).ToList()
    };

    return stats;
}

private List<QuizQuestionDto> GenerateMovieQuizQuestions(MovieEntity movie)
{
    var random = new Random(movie.Id);
    var questions = new List<QuizQuestionDto>();

    var yearOptions = new List<string> { movie.Year.ToString() };
    yearOptions.Add((movie.Year - 1).ToString());
    yearOptions.Add((movie.Year + 1).ToString());
    yearOptions.Add((movie.Year + 3).ToString());
    var yearShuffled = yearOptions.OrderBy(x => random.Next()).ToList();
    int yearCorrectIndex = yearShuffled.IndexOf(movie.Year.ToString());

    questions.Add(new QuizQuestionDto
    {
        Question = $"У якому році вийшов фільм \"{movie.Title}\"?",
        Options = yearShuffled,
        CorrectAnswerIndex = yearCorrectIndex
    });

    if (!string.IsNullOrEmpty(movie.Genre))
    {
        var genres = movie.Genre.Split(',',
StringSplitOptions.RemoveEmptyEntries)
            .Select(g => g.Trim())
            .Where(g => g.ToLower() != "фільм" && g.ToLower() != "серіал")
    }
}

```

		Денисенко Д.О.			ІПЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
		Сфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		153

```

        .ToArray();
        if (genres.Length > 0)
        {
            var genreOptions = new List<string> { genres[0] };
            var allGenres = new[] { "Комедія", "Драма", "Жахи", "Триллер",
"Фантастика", "Пригода", "Романтика", "Біографія", "Кримінал", "Історичні",
"Детектив" };
            var wrongGenres = allGenres.Where(g =>
!genres.Contains(g)).OrderBy(x => random.Next()).Take(3).ToList();
            genreOptions.AddRange(wrongGenres);
            var genreShuffled = genreOptions.OrderBy(x =>
random.Next()).ToList();
            int genreCorrectIndex = genreShuffled.IndexOf(genres[0]);

            questions.Add(new QuizQuestionDto
            {
                Question = $"Який із цих жанрів належить до
\"{movie.Title}\"?",
                Options = genreShuffled,
                CorrectAnswerIndex = genreCorrectIndex
            });
        }

        if (!string.IsNullOrEmpty(movie.Director))
        {
            var directorOptions = new List<string> { movie.Director };
            var directors = new[] { "Крістофер Нолан", "Квентін Тарантіно",
"Денис Вільнев", "Грета Гервіг", "Райан Кулер", "Пітер Джексон" };
            foreach (var dir in directors)
            {
                if (dir != movie.Director && directorOptions.Count < 4)
                    directorOptions.Add(dir);
            }
            var directorShuffled = directorOptions.OrderBy(x =>
random.Next()).ToList();
            int directorCorrectIndex = directorShuffled.IndexOf(movie.Director);

            questions.Add(new QuizQuestionDto
            {
                Question = $"Хто режисер фільму \"{movie.Title}\"?",
                Options = directorShuffled,
                CorrectAnswerIndex = directorCorrectIndex
            });
        }

        var typeOptions = new List<string>
        {
            movie.IsSeries ? "Серіал" : "Фільм",
            movie.IsSeries ? "Фільм" : "Серіал",
            "Мультфільм",
            "Аніме"
        };
        var typeShuffled = typeOptions.OrderBy(x => random.Next()).ToList();
        int typeCorrectIndex = typeShuffled.IndexOf(movie.IsSeries ? "Серіал" :
"Фільм");

        questions.Add(new QuizQuestionDto
        {
            Question = $"Це фільм чи серіал: \"{movie.Title}\"?",
            Options = typeShuffled,
            CorrectAnswerIndex = typeCorrectIndex
        });

        var ratingStr = $"\"{movie.AverageRating:F1}\"";

```

		Денисенко Д.О.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Сфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		154

```

var ratingOptions = new List<string> { ratingStr };
double rating = Math.Round(movie.AverageRating, 1);

var alternatives = new HashSet<string> { ratingStr };
var ratingVariants = new[]
{
    Math.Max(1, rating - 1.5),
    Math.Min(10, rating + 1.5),
    Math.Max(1, rating - 3),
    Math.Min(10, rating + 2),
    Math.Max(1, rating - 2)
};

foreach (var alt in ratingVariants)
{
    var altStr = $"{alt:F1}";
    if (!alternatives.Contains(altStr) && ratingOptions.Count < 4)
    {
        ratingOptions.Add(altStr);
        alternatives.Add(altStr);
    }
}

var ratingShuffled = ratingOptions.OrderBy(x => random.Next()).ToList();
int ratingCorrectIndex = ratingShuffled.IndexOf(ratingStr);

questions.Add(new QuizQuestionDto
{
    Question = $"Який рейтинг у фільму \"{movie.Title}\"?",
    Options = ratingShuffled,
    CorrectAnswerIndex = ratingCorrectIndex
});

return questions;
}

[HttpGet("recommended")]
[Authorize]
public async Task<ActionResult<IEnumerable<object>>> GetRecommended()
{
    var userId =
int.Parse(User.FindFirst(ClaimTypes.NameIdentifier)!.Value);

    var historyIds = await _context.WatchHistory
        .Where(h => h.UserId == userId)
        .Select(h => h.MovieId)
        .ToListAsync();

    var likedIds = await _context.MovieReactions
        .Where(r => r.UserId == userId && r.Type == ReactionType.Like)
        .Select(r => r.MovieId)
        .ToListAsync();

    var interactedIds = historyIds.Concat(likedIds).Distinct().ToList();

    if (!interactedIds.Any())
    {
        return Ok(new List<object>());
    }

    var interactedGenresRaw = await _context.Movies
        .AsNoTracking()
        .Where(m => interactedIds.Contains(m.Id) &&
!string.IsNullOrEmpty(m.Genre))
        .Select(m => m.Genre)

```

		Денисенко Д.О.			ІІЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
		Сфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		155

```

        .ToListAsync();

        var favoriteGenres = interactedGenresRaw
            .SelectMany(g => g.Split(',',
StringSplitOptions.RemoveEmptyEntries))
            .Select(g => g.Trim().ToLower())
            .GroupBy(g => g)
            .OrderByDescending(g => g.Count())
            .Take(3)
            .Select(g => g.Key)
            .ToList();

        if (!favoriteGenres.Any())
        {
            return Ok(new List<object>());
        }

        var blockedGenres = await GetUserBlockedGenres();
        var blockedActorIds = await GetUserBlockedActorIds();

        var query = _context.Movies
            .Include(m => m.MovieActors)
            .AsNoTracking()
            .Where(m => !interactedIds.Contains(m.Id) &&
!string.IsNullOrEmpty(m.Genre));

        foreach (var genre in blockedGenres)
        {
            query = query.Where(m => !m.Genre.ToLower().Contains(genre));
        }

        if (blockedActorIds.Any())
        {
            query = query.Where(m => !m.MovieActors.Any(ma =>
blockedActorIds.Contains(ma.ActorId)));
        }

        var candidates = await query
            .OrderByDescending(m => m.AverageRating)
            .Take(100)
            .ToListAsync();

        var recommended = candidates
            .Where(m =>
            {
                var mGenres = m.Genre.ToLower();
                return favoriteGenres.Any(fg => mGenres.Contains(fg));
            })
            .Take(10)
            .Select(m => new
            {
                id = m.Id,
                title = m.Title,
                posterUrl = m.PosterUrl,
                averageRating = m.AverageRating,
                year = m.Year,
                genre = m.Genre,
                type = m.IsSeries ? "Series" : "Movie",
                viewsCount = m.ViewsCount
            })
            .ToListAsync();
        return Ok(recommended);}}

```