НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ
Факультет інформаційних технологій

ПОГОДЖЕНО	**ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ**
Декан факультету	**Завідувач кафедри комп'ютерних наук**
_____ **Ігор БОЛБОТ**	_____ **Белла ГОЛУБ**
(підпис)	(підпис)
«___» _____ 2026 р.	«___» _____ 2026 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Програмне забезпечення системи аналізу користувацьких відгуків на основі рейтингу кіноглядачів»

Спеціальність «121 Інженерія програмного забезпечення»

Освітньо-професійна програма «Інженерія програмного забезпечення»

Гарант освітньої програми к.т.н., доцент	_____ (підпис)	**Ганна ВАЙГАНГ**
Керівник бакалаврської кваліфікаційної роботи	_____ (підпис)	**Юрій НАУРИНСЬКИЙ**
Консультант бакалаврської кваліфікаційної роботи к.т.н., доцент	_____ (підпис)	**Ганна ВАЙГАНГ**
Виконав	_____ (підпис)	**Олег САВОШ**

Київ-2026

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ
Факультет інформаційних технологій

ЗАТВЕРДЖУЮ
Завідувач кафедри комп'ютерних наук

к.т.н., доцент _____ Белла ГОЛУБ
(підпис)

«___» _____ 2025 р.

ЗАВДАННЯ
ДО ВИКОНАННЯ БАКАЛАВРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ
ЗДОБУВАЧУ

Савошу Олегу Ігоровичу
(прізвище, ім'я, по батькові)

Спеціальність «121 Інженерія програмного забезпечення»
Освітньо-професійна програма «Інженерія програмного забезпечення»
Тема бакалаврської кваліфікаційної роботи
«Програмне забезпечення системи аналізу користувацьких відгуків на основі рейтингу кіноглядачів»
затверджена наказом від «19» грудня 2025 р. № 3204 «С»

Термін подання завершеної роботи на кафедру «_22__» _травня_____ 2026 р.
Вихідні дані до роботи: опис інформаційної системи аналізу користувацьких відгуків і формування рейтингів для кіноглядачів; дані про користувачів, фільми, текстові відгуки та рейтингові оцінки; технології Python, FastAPI, SQLite, SQLAlchemy, HTML, CSS, JavaScript і REST API; методи аналізу текстової інформації; наукові та технічні джерела з розробки веборієнтованих інформаційних систем..

Перелік питань що розглядаються: Аналіз предметної області та постановка задачі, проєктування інформаційного забезпечення системи, розробка програмного забезпечення системи; тестування та впровадження програмної системи.

Перелік графічних документів (за необхідності).

Дата видачі завдання «__22__» ___грудня_____ 2025 р.

Керівник бакалаврської кваліфікаційної роботи _____ (підпис) **Юрій НАУРИНСЬКИЙ**

Консультант бакалаврської кваліфікаційної роботи _____ (підпис) **Ганна ВАЙГАНГ**

Завдання взяв до виконання _____ (підпис) **Олег САВОШ**

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API – Application Programming Interface (інтерфейс програмування застосунків)

CRUD – Create, Read, Update, Delete (створення, читання, оновлення та видалення даних)

CSS – Cascading Style Sheets (каскадні таблиці стилів)

ER-модель – Entity Relationship Model (модель «сутність–зв’язок»)

HTML – HyperText Markup Language (мова розмітки гіпертексту)

HTTP – HyperText Transfer Protocol (протокол передачі гіпертексту)

JSON – JavaScript Object Notation (текстовий формат обміну даними)

JWT – JSON Web Token (токен вебавтентифікації JSON)

NLP – Natural Language Processing (обробка природної мови)

ORM – Object Relational Mapping (об’єктно-реляційне відображення)

REST – Representational State Transfer (архітектурний стиль взаємодії вебсервісів)

SPA – Single Page Application (односторінковий вебзастосунок)

SQL – Structured Query Language (мова структурованих запитів)

SSL – Secure Sockets Layer (протокол захищеної передачі даних)

TMDb – The Movie Database (онлайн-база даних фільмів)

UML – Unified Modeling Language (уніфікована мова моделювання)

URL – Uniform Resource Locator (уніфікований локатор ресурсу)

UX – User Experience (досвід взаємодії користувача)

UI – User Interface (користувацький інтерфейс)

ВСТУП

Сучасний розвиток інформаційних технологій та стрімке поширення цифрових платформ суттєво змінили способи взаємодії користувачів із мультимедійним контентом. Онлайн-сервіси перегляду фільмів і тематичні вебплатформи забезпечують не лише доступ до великої кількості кінострічок, але й створюють середовище для формування користувацьких оцінок, текстових відгуків та рейтингів. У таких умовах особливої актуальності набувають програмні системи, здатні автоматизовано аналізувати великі обсяги користувацького контенту, визначати емоційне забарвлення текстових повідомлень та формувати узагальнену аналітичну інформацію [14, 18, 22].

Актуальність теми зумовлена необхідністю створення програмного забезпечення для автоматизованого аналізу користувацьких відгуків у сфері кіноіндустрії [1, 19]. Зростання кількості текстових коментарів і оцінок унеможливлює їх ефективне ручне опрацювання, що потребує використання сучасних методів аналізу текстової інформації, засобів веброзробки та технологій інтеграції із зовнішніми інформаційними сервісами. Додаткової актуальності набуває задача формування інтерактивної рейтингової системи, яка дозволяє оцінювати не лише числові показники, але й емоційний зміст користувацьких рецензій.

Об’єктом дослідження є процес проєктування та розробки веборієнтованих програмних систем аналізу користувацького контенту.

Предметом дослідження є методи, моделі та програмні засоби реалізації системи аналізу користувацьких відгуків з інтерактивною рейтинговою платформою для кіноглядачів.

Метою кваліфікаційної роботи є проєктування та розробка програмного забезпечення аналізу користувацьких відгуків з інтерактивною системою рейтингів для кіноглядачів, що забезпечує автоматизоване опрацювання текстових рецензій, визначення тональності відгуків, формування рейтингових показників та надання засобів взаємодії між користувачами системи.

- провести аналіз предметної області, існуючих аналогів та сформулювати вимоги до програмного забезпечення;
- розробити архітектуру програмної системи та спроєктувати структуру бази даних;
- реалізувати серверну частину програмного забезпечення із використанням REST API та інтеграцією зовнішнього сервісу TMDb;
- розробити клієнтську частину вебзастосунку та реалізувати модуль аналізу текстових відгуків;
- провести тестування та оцінювання працездатності програмної системи;
- здійснити розгортання програмного забезпечення у хмарному середовищі та сформувати рекомендації щодо його використання.

У процесі виконання роботи використано методи системного аналізу, об’єктно-орієнтованого проєктування, моделювання програмних систем за допомогою UML-діаграм, технології клієнт-серверної взаємодії, REST-архітектури, методи аналізу текстової інформації та засоби вебпрограмування .

Практичне значення одержаних результатів полягає у створенні працездатного веборієнтованого програмного продукту, який дозволяє автоматизувати процес збору, обробки та аналізу користувацьких відгуків про фільми, забезпечує формування інтерактивних рейтингів та надає інструменти модерації контенту. Розроблене програмне забезпечення може бути використане як основа для подальшого розвитку інформаційних сервісів у сфері мультимедійного контенту та рекомендаційних систем.

Апробація результатів кваліфікаційної роботи здійснювалася шляхом представлення основних результатів дослідження на VII Всеукраїнській науково-практичній конференції студентів, аспірантів та молодих учених «Теоретичні та прикладні аспекти розробки комп’ютерних систем», що проводилася Національним університетом біоресурсів і природокористування України 22–23 квітня 2026 року. У межах конференції було представлено тези на тему: «Інформаційна система аналізу користувацьких відгуків із реалізацією інтерактивного механізму формування рейтингів для кіноглядачів». Під час

апробації висвітлено питання проєктування веборієнтованої системи аналізу користувацьких відгуків, використання REST-архітектури, інтеграції зовнішніх API та реалізації модулів аналізу текстової інформації. Результати дослідження опубліковані у матеріалах конференції.

Під час виконання роботи використано сучасні технології веброзробки та програмної інженерії, зокрема Python, FastAPI, SQLAlchemy, SQLite, HTML, CSS, JavaScript, REST API та хмарні сервіси розгортання вебзастосунків [1, 2, 23, 24].

Пояснювальна записка складається зі вступу, чотирьох розділів, висновків, списку використаних джерел та додатків.

У першому розділі проведено аналіз предметної області, досліджено особливості сучасних платформ аналізу користувацьких відгуків, сформульовано вимоги до програмної системи та виконано постановку задачі.

У другому розділі описано процес проєктування програмного забезпечення, розроблено архітектуру системи, UML-діаграми, ER-модель бази даних та визначено взаємодію компонентів системи.

Третій розділ присвячений реалізації програмного забезпечення, опису серверної та клієнтської частин застосунку, модулю аналізу тексту, структурі бази даних та реалізації REST API.

У четвертому розділі наведено результати тестування програмної системи, описано процес розгортання застосунку, вимоги до програмного та апаратного забезпечення, а також рекомендації щодо використання та подальшого розвитку системи.

Загальний обсяг пояснювальної записки становить 70 сторінок. Робота містить 10 рисунків, 18 таблиць, 3 додатків та список використаних джерел із 36 найменувань.

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Аналіз предметної області

Сучасна кіноіндустрія характеризується значними обсягами цифрового мультимедійного контенту та високою динамікою його оновлення. Щоденно на стримінгових платформах, вебресурсах і сервісах онлайн-кінотеатрів з’являються нові фільми, серіали, документальні стрічки та анімаційні проєкти. У таких умовах користувач стикається з проблемою вибору контенту, що відповідає його індивідуальним уподобанням. Одним із ключових факторів, які впливають на прийняття рішення щодо перегляду фільму, є рейтинги та текстові відгуки інших користувачів. Саме користувацький контент формує суспільне сприйняття кінострічок і значною мірою впливає на популярність медіапродукту [19, 20].

У сучасному інформаційному середовищі існує велика кількість платформ, орієнтованих на зберігання та публікацію відгуків про фільми. Найбільш поширеними серед них є IMDb, Rotten Tomatoes, Metacritic та інші подібні сервіси [15, 19]. Основний функціонал таких платформ полягає у формуванні числових рейтингів, зборі текстових рецензій і наданні базової інформації про кінострічки. Проте аналіз існуючих рішень показав, що більшість із них використовують переважно статистичний підхід до оцінювання контенту, де головним критерієм виступає середнє арифметичне користувацьких оцінок. При цьому зміст текстових коментарів здебільшого не піддається автоматизованому аналізу, а емоційне забарвлення відгуків не враховується під час формування загального рейтингу.

Водночас текстові рецензії містять значний обсяг прихованої інформації про сприйняття користувачами окремих аспектів фільму: сюжетної лінії, акторської гри, візуальних ефектів, музичного супроводу або режисерської роботи. Використання методів аналізу природної мови (Natural Language

Processing, NLP) дозволяє автоматизувати процес обробки таких текстових даних, визначати тональність повідомлень, виявляти ключові слова та формувати узагальнену аналітичну інформацію [16, 17]. Це створює передумови для побудови більш інтелектуальних систем рейтингування, які враховують не лише числові оцінки, але й зміст користувацьких відгуків.

Особливого значення задача аналізу текстових рецензій набуває в умовах постійного зростання обсягів інформації. Ручна модерація та аналіз великої кількості коментарів є малоефективними та потребують значних часових витрат. Автоматизований підхід дозволяє значно підвищити швидкість обробки інформації, забезпечити оперативне формування статистики та покращити якість рекомендаційних механізмів [18, 20]. Крім цього, результати аналізу можуть бути використані для виявлення негативних тенденцій, визначення популярних жанрів і покращення взаємодії між користувачами та адміністраторами системи.

Предметна область розроблюваної програмної системи охоплює процеси збору, зберігання, аналізу та візуалізації інформації про фільми й користувацькі відгуки. Основними інформаційними сутностями системи є користувачі, фільми, рейтинги, текстові рецензії та результати аналізу тональності. Користувач взаємодіє із системою через вебінтерфейс, отримує доступ до каталогу фільмів, переглядає інформацію про кінострічки, залишає оцінки та текстові коментарі.

Після збереження відгуку система автоматично виконує аналіз тексту та визначає його емоційне забарвлення. Додатково система забезпечує можливість формування статистичних показників, пов’язаних із популярністю фільмів, активністю користувачів та співвідношенням позитивних і негативних рецензій. Важливим аспектом предметної області є використання методів аналізу природної мови для автоматизованої обробки текстових даних, що дозволяє не лише зберігати відгуки, але й отримувати на їх основі аналітичну інформацію щодо сприйняття контенту аудиторією. Це забезпечує більш високий рівень інформативності системи порівняно з традиційними платформами рейтингування, які використовують переважно числові оцінки без детального аналізу змісту рецензій.

Для кращого розуміння особливостей предметної області доцільно узагальнити основні функціональні процеси системи у вигляді табл. 1.1.

Таблиця 1.1

Основні процеси предметної області

Процес	Характеристика
Формування каталогу фільмів	Отримання та збереження інформації про фільми із зовнішніх API
Реєстрація користувачів	Створення облікових записів та авторизація в системі
Формування відгуків	Створення текстових рецензій та виставлення оцінок
Аналіз тексту	Визначення тональності та емоційного забарвлення відгуків
Модерація контенту	Перевірка відгуків та контроль дотримання правил
Формування рейтингів	Автоматичний розрахунок рейтингових показників
Візуалізація статистики	Побудова графіків та аналітичних представлень

Як показано у табл. 1.1, предметна область поєднує класичні механізми веборієнтованих інформаційних систем із елементами інтелектуального аналізу текстових даних. Це дозволяє віднести розроблювану систему до класу інформаційно-аналітичних вебзастосунків, які забезпечують автоматизовану обробку користувацького контенту.

Важливим аспектом предметної області є інтеграція із зовнішніми інформаційними сервісами. Для отримання актуальних даних про фільми доцільним є використання відкритих API, зокрема TMDb API [15]. Такий підхід дозволяє уникнути ручного наповнення каталогу, забезпечує актуальність інформації та підвищує масштабованість системи.

Аналіз предметної області також показав, що сучасні веборієнтовані системи повинні відповідати вимогам швидкодії, масштабованості, безпеки та зручності використання [1, 2]. Оскільки користувачі взаємодіють із платформою в режимі реального часу, особливого значення набуває ефективна організація клієнт-серверної взаємодії, оптимізація обробки HTTP-запитів та забезпечення стабільної роботи програмного забезпечення [5, 13].

Таким чином, предметна область дослідження охоплює комплекс взаємопов'язаних процесів аналізу користувацьких відгуків, формування інтерактивних рейтингів, модерації контенту та візуалізації аналітичної інформації. Проведений аналіз підтверджує актуальність розробки програмного забезпечення, яке поєднує функції веборієнтованої інформаційної системи та засоби автоматизованого аналізу текстових даних, що дозволяє підвищити інформативність рейтингів і покращити якість взаємодії користувачів із платформою.

1.2 Сучасні аналоги

У сучасному цифровому середовищі значного поширення набули веборієнтовані платформи, призначені для пошуку, оцінювання та аналізу кіноконтенту. Такі системи забезпечують користувачам доступ до інформації про фільми, рейтинги, рецензії, трейлери та рекомендації. Основною метою подібних сервісів є спрощення вибору медіаконтенту та формування спільноти користувачів, які взаємодіють між собою через систему оцінок і текстових коментарів [10, 19].

Однією з найвідоміших платформ є IMDb (Internet Movie Database), яка містить велику базу даних фільмів, серіалів, акторів і режисерів. Система дозволяє користувачам залишати оцінки та текстові рецензії, формуючи загальний рейтинг кінострічок. Основною перевагою IMDb є масштабність інформаційної бази та висока популярність серед користувачів у всьому світі.

Водночас сервіс переважно орієнтований на статистичне усереднення оцінок і не забезпечує глибокого автоматизованого аналізу змісту текстових відгуків [19].

Ще одним популярним рішенням є Rotten Tomatoes, особливістю якого є поділ оцінювання на професійні рецензії критиків і відгуки звичайних користувачів. Платформа формує так званий показник «Tomatometer», який відображає відсоток позитивних оцінок. Незважаючи на розвинену систему категоризації відгуків, аналіз текстових рецензій виконується переважно вручну або у спрощеному вигляді, без використання повноцінних механізмів аналізу тональності [10].

Платформа Metacritic використовує інший підхід до оцінювання контенту, об’єднуючи рецензії з різних джерел у єдиний числовий показник – Metascore. Основною перевагою сервісу є агрегування професійних оцінок і побудова інтегрованого рейтингу. Проте система також має обмежені можливості щодо автоматичного аналізу текстових коментарів користувачів та виділення ключових аспектів рецензій [18].

Окрім міжнародних платформ, існують і спеціалізовані вебресурси для аналізу користувацького контенту, які використовують елементи штучного інтелекту та методи обробки природної мови. Такі системи дозволяють визначати емоційне забарвлення текстів, виявляти ключові слова та автоматично класифікувати відгуки за категоріями. Проте більшість подібних рішень орієнтована на загальний аналіз текстів і не адаптована безпосередньо до специфіки кінематографічної предметної області [14, 17].

Суттєвим напрямом розвитку сучасних інформаційних платформ є інтеграція рекомендаційних механізмів та аналітичних модулів, побудованих на основі алгоритмів машинного навчання. Такі технології дозволяють персоналізувати взаємодію користувача із системою, прогнозувати його вподобання та формувати рекомендації на основі історії переглядів, оцінок і текстових рецензій. Крім цього, використання алгоритмів аналізу тональності забезпечує можливість автоматичного виявлення негативних тенденцій у відгуках та оперативного реагування на зміну суспільного сприйняття

кіноконтенту. Однак у більшості існуючих систем аналітичні можливості реалізовані як додатковий функціонал і не становлять основу програмної архітектури платформи, що обмежує глибину аналізу користувацьких даних та точність формування рекомендацій [16, 18].

Для порівняння основних функціональних можливостей сучасних платформ доцільно узагальнити їх характеристики у табл. 1.2.

Таблиця 1.2

Порівняння сучасних платформ аналізу кіноконтенту

Платформа	Основний функціонал	Переваги	Недоліки
IMDb	Каталог фільмів, рейтинги, рецензії	Велика база даних, популярність	Відсутність глибокого аналізу тексту
Rotten Tomatoes	Рейтинги критиків і користувачів	Поділ професійних та користувацьких оцінок	Обмежений аналіз тональності
Metacritic	Інтегрований рейтинг Metascore	Агрегація рецензій з різних джерел	Спрощена аналітика текстових відгуків
Letterboxd	Соціальна взаємодія користувачів	Зручний інтерфейс, списки перегляду	Відсутність автоматизованого NLP-аналізу
Розроблювана система	Аналіз відгуків та рейтингів	Аналіз тональності, ключових слів, статистики	Потребує навчання моделей аналізу

Як показано у табл. 1.2, більшість існуючих платформ забезпечують базові можливості роботи з рейтингами та текстовими рецензіями, проте не реалізують комплексний автоматизований аналіз користувацьких відгуків. Саме тому актуальною є розробка програмної системи, яка поєднуватиме функції веборієнтованої платформи для роботи з кіноконтентом із засобами інтелектуального аналізу текстових даних. Такий підхід дозволить підвищити інформативність рейтингів, автоматизувати аналіз емоційного забарвлення рецензій та забезпечити користувачів більш детальною аналітичною інформацією щодо сприйняття фільмів аудиторією.

1.3 Опис функціональних вимог

Одним із ключових етапів проєктування програмної системи є формування функціональних вимог, які визначають основні можливості вебзастосунку, сценарії взаємодії користувачів із системою та перелік автоматизованих процесів. Функціональні вимоги описують, які саме операції повинна виконувати система для забезпечення повноцінної роботи із каталогом фільмів, користувацькими рецензіями та модулем аналізу текстових даних. Формування таких вимог дозволяє визначити межі функціонування програмного забезпечення, структуру майбутньої системи та особливості реалізації окремих програмних модулів [3, 7].

Під час проєктування системи було виділено три основні категорії функціональних можливостей: функції звичайного користувача, функції адміністратора та автоматизовані системні функції. Такий підхід забезпечує розмежування прав доступу, підвищує рівень безпеки та дозволяє реалізувати рольову модель взаємодії із програмним забезпеченням.

Для звичайного користувача (глядача) система повинна забезпечувати можливість перегляду каталогу фільмів із базовою інформацією, включаючи назву, жанри, рік випуску, короткий опис та постер. Користувач повинен мати змогу здійснювати пошук фільмів через інтеграцію із зовнішнім кіно-API, що

дозволяє швидко знаходити потрібний контент та отримувати актуальні дані про кінострічки. Крім цього, система повинна підтримувати відкриття детальної сторінки фільму, на якій відображаються розширені характеристики: опис, список акторів, рейтинги та користувацькі рецензії.

Важливою функціональною можливістю є створення власних відгуків, які містять текстовий коментар і числову оцінку за визначеною шкалою. Користувач також повинен мати змогу редагувати або видаляти власні рецензії, а також переглядати історію своєї активності через особистий профіль. Наявність персонального профілю дозволяє зберігати інформацію про залишені оцінки, переглянуті фільми та статистику активності користувача в системі.

Для адміністратора система повинна надавати розширений набір функцій, пов’язаних із керуванням інформаційним наповненням і модерацією контенту. Адміністратор має можливість переглядати всі відгуки незалежно від їх статусу, перевіряти коректність коментарів та блокувати небажаний контент, який містить спам, нецензурну лексику або образливі висловлювання. Також адміністративна панель повинна забезпечувати перегляд узагальненої статистики щодо кількості користувачів, фільмів, відгуків та результатів аналізу тональності текстів.

Окремою групою функціональних вимог є автоматизовані системні процеси, які виконуються без безпосереднього втручання користувача. Після збереження відгуку система повинна автоматично запускати модуль аналізу тексту, який визначає емоційне забарвлення рецензії, обчислює рівень позитивності повідомлення та виділяє ключові слова або фрази. Отримані результати використовуються для побудови аналітики, фільтрації контенту та формування інтегрованого рейтингу фільму. При додаванні нової оцінки загальний рейтинг кінострічки має автоматично перераховуватися з урахуванням як числових оцінок користувачів, так і результатів аналізу тональності текстових рецензій.

Для узагальнення функціональних можливостей системи доцільно представити основні вимоги у вигляді табл. 1.3.

Таблиця 1.3

Основні функціональні вимоги системи

Категорія користувачів	Основні функції
Користувач	Перегляд каталогу, пошук фільмів, створення та редагування відгуків, виставлення оцінок
Адміністратор	Модерація відгуків, керування каталогом, перегляд статистики
Система	Аналіз тональності, виділення ключових слів, автоматичний перерахунок рейтингів

Таким чином, сформовані функціональні вимоги визначають основні принципи роботи програмної системи та створюють основу для подальшого етапу проєктування архітектури, бази даних і програмних модулів вебзастосунку.

1.4 Нефункціональні вимоги

Окрім функціональних можливостей, важливим етапом проєктування програмної системи є визначення нефункціональних вимог, які характеризують якість роботи програмного забезпечення, особливості його експлуатації, рівень безпеки, продуктивності та надійності. Нефункціональні вимоги визначають обмеження та критерії, яким повинна відповідати система під час функціонування у реальному середовищі використання [4, 6]. Саме ці вимоги значною мірою впливають на вибір архітектури застосунку, технологічного стеку та способів організації клієнт-серверної взаємодії.

Однією з основних нефункціональних характеристик системи є продуктивність. Вебзастосунок повинен забезпечувати швидке завантаження сторінок, оперативне виконання запитів до бази даних та мінімальний час обробки користувацьких дій. Особливого значення ця вимога набуває у випадках значного навантаження на систему, наприклад під час виходу популярних

фільмів, коли кількість одночасних запитів і нових відгуків може суттєво зростати. Для забезпечення належного рівня продуктивності необхідно використовувати оптимізовані механізми роботи з даними, кешування та асинхронну обробку запитів [5].

Важливою вимогою є масштабованість системи. Архітектура програмного забезпечення повинна забезпечувати можливість подальшого розширення функціоналу без необхідності повного перепроєктування системи. Зокрема, у майбутньому до вебзастосунку можуть бути додані рекомендаційні механізми, соціальні функції, система персоналізованих підписок або модулі машинного навчання для покращення точності аналізу текстів. Масштабованість також передбачає можливість збільшення кількості користувачів і обсягів даних без критичного зниження швидкодії системи.

Не менш важливою характеристикою є надійність програмного забезпечення. Система повинна функціонувати стабільно у режимі безперервної роботи, коректно обробляти помилки та забезпечувати цілісність даних навіть у випадку виникнення збоїв або перевантажень. Для цього необхідно передбачити механізми резервного копіювання даних, логування подій, обробки виняткових ситуацій та відновлення працездатності після помилок [8].

Окрему увагу необхідно приділити безпеці системи, оскільки вебзастосунок працює з персональними даними користувачів. Інформація про облікові записи, адреси електронної пошти та паролі повинна зберігатися у захищеному вигляді з використанням сучасних алгоритмів хешування та засобів автентифікації. Доступ до адміністративних функцій повинен надаватися лише авторизованим користувачам із відповідними правами доступу. Крім цього, під час розробки необхідно враховувати базові принципи захисту від типових вебзагроз, зокрема SQL-ін’єкцій, XSS-атак та несанкціонованого доступу до даних [9, 12].

Важливим фактором ефективності системи є зручність використання (usability). Інтерфейс вебзастосунку повинен бути інтуїтивно зрозумілим, логічно структурованим та адаптованим для користувачів із різним рівнем

технічної підготовки. Основні елементи навігації мають забезпечувати швидкий доступ до пошуку фільмів, перегляду рецензій та створення власних відгуків. Крім цього, інтерфейс повинен бути адаптивним і коректно відображатися як на персональних комп’ютерах, так і на мобільних пристроях.

Ще однією важливою нефункціональною вимогою є портативність системи. Програмне забезпечення повинно легко розгортатися у хмарному середовищі або на віддаленому сервері без прив’язки до конкретної апаратної платформи. Це дозволяє забезпечити гнучкість розгортання, спрощує адміністрування та створює можливість подальшого масштабування вебзастосунку відповідно до потреб користувачів [13].

Для узагальнення основних нефункціональних характеристик системи доцільно представити їх у вигляді табл. 1.4.

Таблиця 1.4

Основні нефункціональні вимоги системи

Нефункціональна вимога	Характеристика
Продуктивність	Швидка обробка запитів та мінімальний час відповіді системи
Масштабованість	Можливість розширення функціоналу та збільшення навантаження
Надійність	Стабільна робота та збереження цілісності даних
Безпека	Захист персональних даних і контроль доступу
Зручність використання	Інтуїтивний та адаптивний інтерфейс
Портативність	Можливість розгортання у хмарному середовищі

Таким чином, сформовані нефункціональні вимоги визначають критерії якості програмної системи та забезпечують основу для побудови стабільного, безпечного й масштабованого вебзастосунку, орієнтованого на аналіз користувацьких відгуків про фільми.

1.5 Постановка задачі

Метою бакалаврської кваліфікаційної роботи є розробка програмного забезпечення системи аналізу глядацьких відгуків із інтерактивною платформою рейтингування фільмів, яка забезпечує автоматизовану обробку текстових рецензій, визначення тональності відгуків та формування аналітичної інформації щодо сприйняття кіноконтенту користувачами. Розроблювана система повинна поєднувати функції веборієнтованої інформаційної платформи та інструментів аналізу текстових даних, забезпечуючи зручну взаємодію користувачів із каталогом фільмів і системою рецензування [3, 7].

Для досягнення поставленої мети у роботі необхідно вирішити комплекс взаємопов’язаних завдань, які охоплюють етапи аналізу, проєктування, програмної реалізації та тестування системи. Насамперед необхідно провести аналіз предметної області та сучасних платформ для роботи з кіноконтентом, визначити їх переваги, недоліки та особливості функціонування. Це дозволяє сформувати перелік актуальних проблем, пов’язаних із відсутністю автоматизованого аналізу текстових відгуків та недостатньою інформативністю традиційних рейтингових систем.

Наступним етапом є формування функціональних і нефункціональних вимог до програмного забезпечення. Система повинна підтримувати роботу декількох категорій користувачів, забезпечувати можливість перегляду каталогу фільмів, створення та модерації рецензій, автоматичного аналізу текстових повідомлень і формування статистичних показників. Крім цього, програмний продукт має відповідати вимогам продуктивності, безпеки, масштабованості, надійності та зручності використання.

У межах роботи необхідно спроєктувати архітектуру програмної системи, яка забезпечуватиме розподіл функціональності між клієнтською та серверною частинами, підтримуватиме взаємодію із зовнішніми API та забезпечуватиме можливість подальшого розширення функціоналу. Особлива увага повинна бути приділена розробці серверної частини застосунку, яка відповідає за реалізацію

бізнес-логіки, взаємодію з базою даних, автентифікацію користувачів та обробку результатів аналізу текстових рецензій.

Одним із ключових завдань є реалізація модуля аналізу текстових відгуків, який повинен автоматично визначати емоційне забарвлення рецензій, виділяти ключові слова та формувати аналітичні показники на основі текстових даних. Для забезпечення актуальності інформації про фільми необхідно реалізувати інтеграцію із зовнішнім кіно-API, що дозволить автоматично отримувати дані про фільми, жанри, постери та інші характеристики кіноконтенту.

Важливим етапом роботи є проєктування та реалізація структури бази даних, яка повинна забезпечувати надійне зберігання інформації про користувачів, фільми, рейтинги та результати аналізу рецензій. Також необхідно реалізувати систему модерації контенту, яка дозволить адміністраторам контролювати коректність користувацьких повідомлень та обмежувати поширення небажаного контенту.

Завершальним етапом виконання роботи є тестування програмного забезпечення, яке повинно підтвердити коректність функціонування окремих модулів системи, стабільність роботи вебзастосунку та відповідність отриманих результатів сформованим вимогам [8, 13].

Таким чином, у першому розділі було проведено аналіз предметної області, розглянуто сучасні аналоги програмних платформ для роботи з кіноконтентом, визначено функціональні та нефункціональні вимоги до системи, а також сформульовано основні завдання розробки програмного забезпечення. Отримані результати створюють основу для подальшого етапу проєктування архітектури системи, моделювання структури бази даних і реалізації програмних модулів вебзастосунку.

2. ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ СИСТЕМИ

2.1 Проєктування архітектури програмної системи

Проєктування архітектури програмної системи є одним із ключових етапів розробки веборієнтованих застосунків, оскільки саме архітектура визначає принципи взаємодії між компонентами системи, організацію обробки даних, масштабованість, безпеку та можливість подальшого розвитку програмного забезпечення [4, 6]. Для системи аналізу глядацьких відгуків було обрано клієнт-серверну архітектуру, яка забезпечує розподіл функціональних обов'язків між інтерфейсною частиною, серверною логікою та системою зберігання даних.

Основною перевагою клієнт-серверного підходу є можливість незалежного розвитку окремих компонентів системи. Клієнтська частина відповідає за взаємодію з користувачем, відображення інформації та обробку інтерфейсних подій, тоді як серверна частина забезпечує виконання бізнес-логіки, автентифікацію користувачів, взаємодію з базою даних та обробку результатів аналізу текстових рецензій. Такий підхід дозволяє спростити супровід програмного забезпечення та підвищити його масштабованість [5].

Для взаємодії між клієнтською та серверною частинами використовується REST API, який забезпечує обмін даними у форматі JSON через HTTP-запити. Використання REST-архітектури дозволяє реалізувати стандартизований механізм доступу до ресурсів системи, спрощує інтеграцію із зовнішніми сервісами та забезпечує незалежність між frontend і backend компонентами [7]. Завдяки цьому клієнтський застосунок може отримувати інформацію про фільми, користувачів та відгуки незалежно від внутрішньої реалізації серверної частини.

У структурі програмної системи можна виділити декілька основних компонентів:

- клієнтський вебінтерфейс;

- серверний API;
- модуль аналізу текстових відгуків;
- систему управління базою даних;
- зовнішній API для отримання інформації про фільми.

Клієнтський рівень системи забезпечує взаємодію користувача із вебзастосунком через браузер. Саме на цьому рівні реалізується навігація між сторінками, пошук фільмів, перегляд рейтингової інформації, створення рецензій та робота з особистим профілем користувача. Для покращення швидкодії та зручності використання інтерфейс повинен підтримувати асинхронний обмін даними із сервером без повного перезавантаження сторінки.

Серверний рівень відповідає за реалізацію бізнес-логіки системи. Основними завданнями серверної частини є обробка HTTP-запитів, перевірка прав доступу, взаємодія з базою даних, збереження відгуків та запуск модуля аналізу тональності текстів. Крім цього, сервер забезпечує інтеграцію із зовнішнім кіно-API, через яке автоматично отримуються актуальні дані про фільми, жанри, постери та інші характеристики кіноконтенту [10, 15].

Окремим компонентом архітектури є модуль аналізу текстових відгуків, який автоматично виконує обробку рецензій користувачів після їх збереження у системі. Модуль визначає емоційне забарвлення тексту, розраховує рівень позитивності повідомлення та виділяє ключові слова або фрази. Результати аналізу використовуються для побудови аналітики та формування інтегрованого рейтингу фільмів [14, 17].

Для зберігання інформації у системі використовується реляційна база даних, яка забезпечує цілісність даних, підтримку зв’язків між сутностями та можливість ефективного виконання запитів. У базі даних зберігається інформація про користувачів, фільми, рейтинги, рецензії та результати аналізу текстів. Організація даних у вигляді взаємопов’язаних таблиць дозволяє забезпечити структурованість інформаційного середовища та спростити подальшу аналітичну обробку даних [8].

Для кращого розуміння загальної структури програмної системи доцільно представити архітектуру застосунку у вигляді структурної схеми.

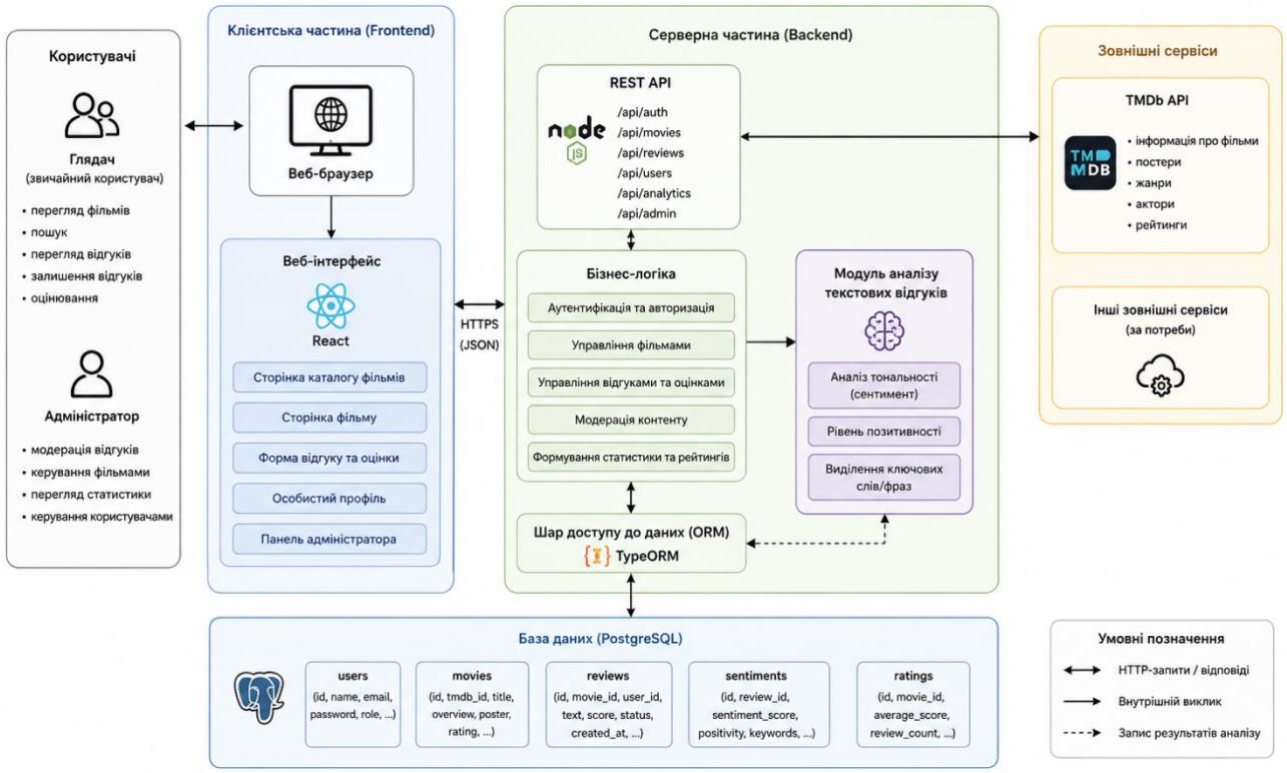

Рис. 2.1 Загальна архітектура програмної системи аналізу глядацьких відгуків

Після опису архітектури доцільно узагальнити функції основних компонентів системи у вигляді табл. 2.1.

Таблиця 2.1

Основні компоненти архітектури системи

Компонент	Призначення
Frontend	Взаємодія користувача із системою
Backend	Реалізація бізнес-логіки та API
REST API	Передача даних між компонентами
Модуль аналізу тексту	Аналіз тональності та ключових слів
База даних	Зберігання інформації системи
TMDb API	Отримання актуальної інформації про фільми

Як показано у табл. 2.1, кожен компонент системи виконує окремий набір функцій, що дозволяє забезпечити модульність архітектури та спрощує супровід програмного забезпечення. Розподіл функціональності між компонентами також позитивно впливає на масштабованість системи та створює можливість подальшого розширення її можливостей.

Для формалізації функціональної структури системи та визначення сценаріїв взаємодії користувачів із вебзастосунком було побудовано UML-діаграму прецедентів (Use Case Diagram). Діаграма відображає основні функціональні можливості програмного забезпечення та взаємозв’язки між користувачами і компонентами системи.

У межах розробленого вебзастосунку передбачено три основні ролі: кіноглядач (User), модератор та адміністратор. Користувач має можливість переглядати каталог фільмів, здійснювати пошук, залишати відгуки та оцінки. Модератор відповідає за перевірку й контроль користувацького контенту. Адміністратор виконує керування інформаційним наповненням системи, користувачами та окремими компонентами програмного забезпечення.

Побудована діаграма прецедентів дозволила визначити основні сценарії роботи із системою та деталізувати функціональні вимоги до програмного продукту на етапі проєктування архітектури вебзастосунку.

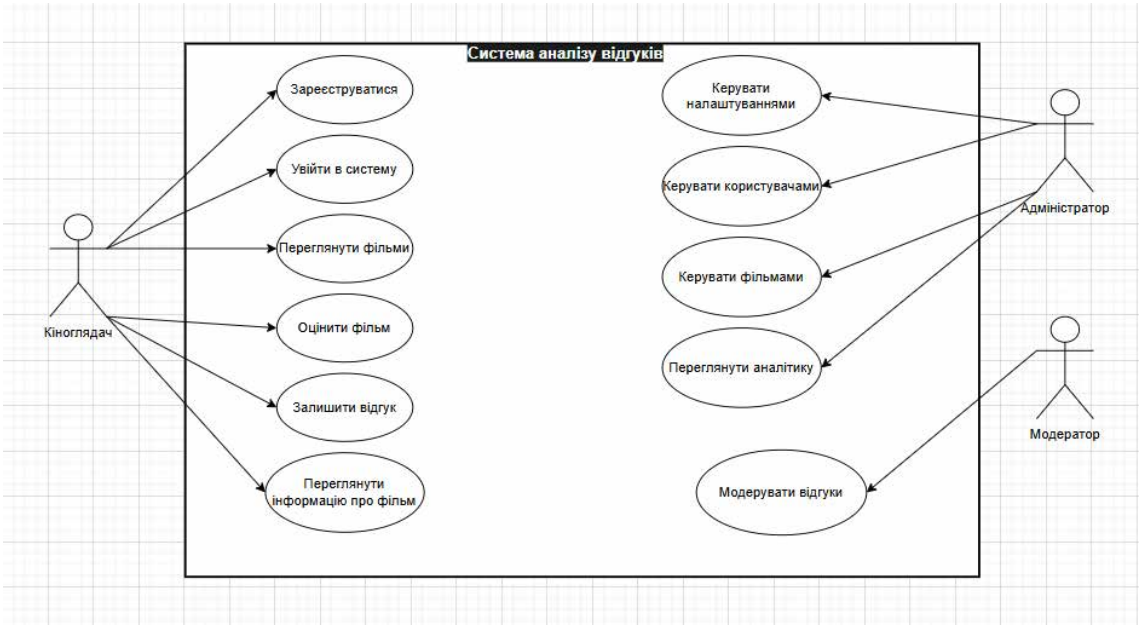

Рис. 2.2 Діаграма прецедентів програмної системи аналізу глядацьких відгуків

Як показано на рис. 2.2, функціональні можливості системи розподілені між користувачами відповідно до їх ролей та рівнів доступу. Використання UML-діаграми прецедентів дозволило структуризувати взаємодію між користувачами та програмними модулями системи, що спростило подальше проєктування серверної логіки та компонентів вебзастосунку.

Таким чином, обрана архітектура програмної системи забезпечує ефективну взаємодію між користувачем, серверною частиною та модулем аналізу текстових даних. Використання клієнт-серверного підходу, REST API та модульної структури дозволяє створити масштабоване, гнучке та придатне до подальшого розвитку програмне забезпечення для аналізу глядацьких відгуків і формування інтерактивної рейтингової платформи.

2.2 Моделювання інформаційного забезпечення системи

Одним із ключових етапів проєктування програмної системи є моделювання інформаційного забезпечення, оскільки саме структура даних визначає ефективність зберігання, обробки та подальшого аналізу інформації у вебзастосунку [8]. Для системи аналізу глядацьких відгуків інформаційне забезпечення повинно забезпечувати надійне зберігання даних про користувачів, фільми, текстові рецензії, рейтинги та результати аналізу тональності. Крім цього, структура бази даних повинна підтримувати масштабованість системи, можливість швидкого виконання запитів і забезпечення цілісності інформації.

На основі аналізу предметної області було визначено основні інформаційні сутності системи, між якими формуються логічні взаємозв’язки. До ключових сутностей належать:

- користувачі;
- фільми;
- відгуки;
- рейтинги;
- результати аналізу тональності.

Сутність «Користувач» містить інформацію про зареєстрованих користувачів системи, включаючи ім’я, адресу електронної пошти, пароль та роль доступу. Кожен користувач може залишати декілька відгуків та оцінок до різних фільмів, що формує зв’язок типу «один до багатьох» між таблицями users і reviews.

Сутність «Фільм» призначена для зберігання інформації про кіноконтент, отриманий із зовнішнього API. Для кожного фільму зберігаються назва, опис, постер, жанри, дата випуску та інші характеристики. Один фільм може мати велику кількість користувацьких рецензій і рейтингів, тому між сутностями movies і reviews також реалізується зв’язок типу «один до багатьох».

Сутність «Відгук» є центральним елементом інформаційної моделі системи, оскільки поєднує користувача, фільм, текст рецензії та числову оцінку. Саме після створення відгуку запускається модуль аналізу текстових даних, результати якого зберігаються у відповідній таблиці результатів аналізу тональності.

Сутність «Результат аналізу тональності» містить інформацію про емоційне забарвлення тексту, рівень позитивності та виділені ключові слова. Зв’язок між відгуком та результатом аналізу реалізується за принципом «один до одного», оскільки кожен відгук має власний результат автоматичного аналізу [14, 17].

Структура даних у системі організована так, щоб ефективно поєднувати інформацію про контент та активність користувачів. Для забезпечення коректної організації інформаційного середовища системи було побудовано ER-діаграму бази даних, яка відображає основні сутності та зв’язки між ними.

Розроблена ER-модель демонструє, як відгуки стають сполучною ланкою між глядачем та фільмом [20]. Важливою особливістю є те, що база даних зберігає не лише «сирий» текст рецензій, а й результати їхнього інтелектуального аналізу. Це включає цифрові індикатори емоційного фону та список ключових понять, що були виявлені в ході обробки тексту (рис. 2.3).

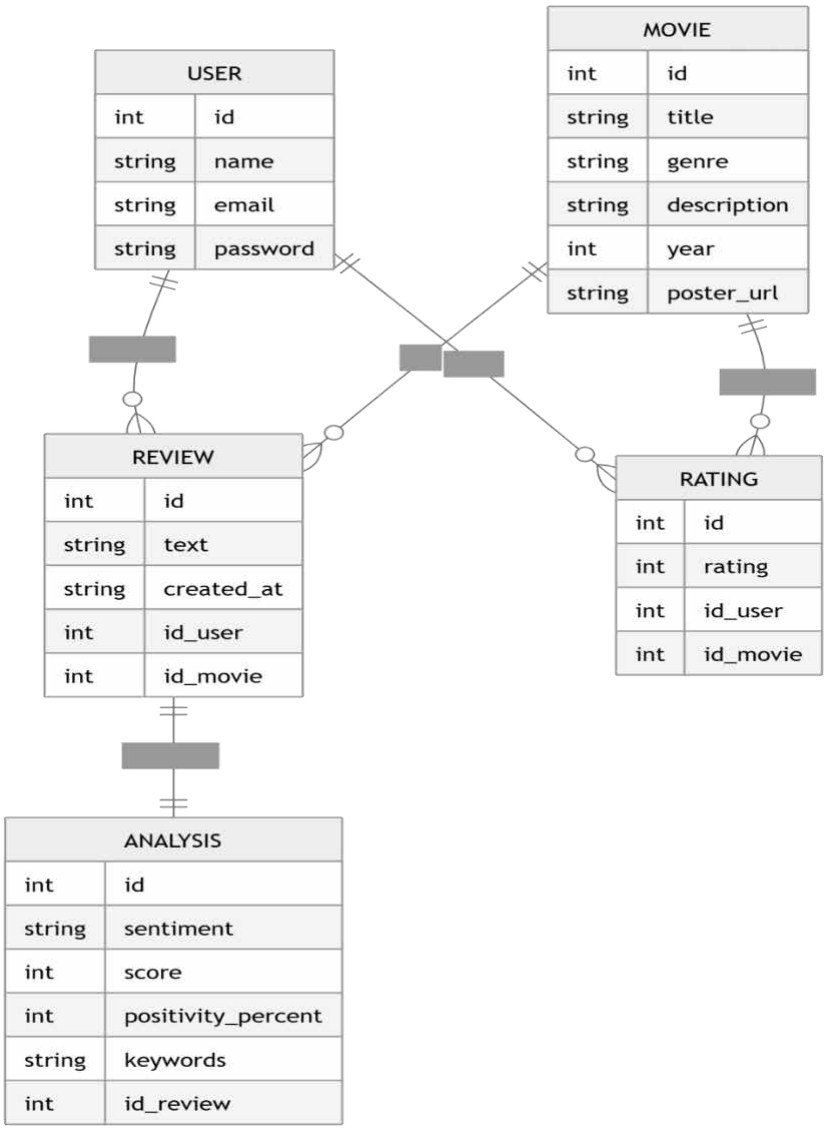

Рис. 2.3 ER-діаграма бази даних системи аналізу глядацьких відгуків

ER-діаграма дозволяє візуально представити структуру інформаційної системи, визначити основні зв’язки між таблицями та забезпечити цілісність даних під час подальшої реалізації бази даних. Використання такого підходу спрощує проєктування серверної логіки та дозволяє заздалегідь виявити можливі проблеми, пов’язані з дублюванням інформації або некоректною організацією зв’язків між сутностями. Крім цього, ER-модель є основою для подальшого створення фізичної структури бази даних і реалізації механізмів взаємодії між серверною частиною застосунку та системою управління базами даних [8].

Після ER-моделі доцільно представити опис основних сутностей системи у вигляді табл. 2.2.

Таблиця 2.2

Основні сутності інформаційної моделі системи

Сутність	Призначення
Users	Зберігання інформації про користувачів
Movies	Зберігання інформації про фільми
Reviews	Зберігання текстових відгуків та оцінок
Sentiments	Зберігання результатів аналізу тональності
Ratings	Зберігання агрегованих рейтингових показників

Як показано у табл. 2.2, структура інформаційної системи побудована за принципом логічного розділення даних відповідно до функціонального призначення окремих компонентів. Такий підхід забезпечує спрощення обробки інформації та підвищує ефективність виконання SQL-запитів.

Для забезпечення цілісності даних і мінімізації дублювання інформації структура бази даних була приведена до нормалізованого вигляду. Під час проєктування враховувалися вимоги третьої нормальної форми (3НФ), згідно з якою кожна таблиця повинна містити лише атрибути, що безпосередньо залежать від первинного ключа [8]. Це дозволило уникнути надлишковості даних та спростити механізми оновлення інформації у системі.

Важливим елементом інформаційного забезпечення є механізм взаємодії бази даних із серверною частиною застосунку. Для роботи з даними доцільно використовувати ORM-технологію, яка забезпечує об’єктно-реляційне відображення таблиць бази даних у структури програмного коду. Такий підхід спрощує виконання операцій створення, оновлення, пошуку та видалення даних, а також підвищує безпеку роботи із SQL-запитами [5].

Окрім ER-діаграми, для більш детального представлення структури інформаційної системи доцільно використати UML-діаграму класів, яка відображає логічні зв’язки між програмними сутностями та їх атрибутами (рис. 2.4).

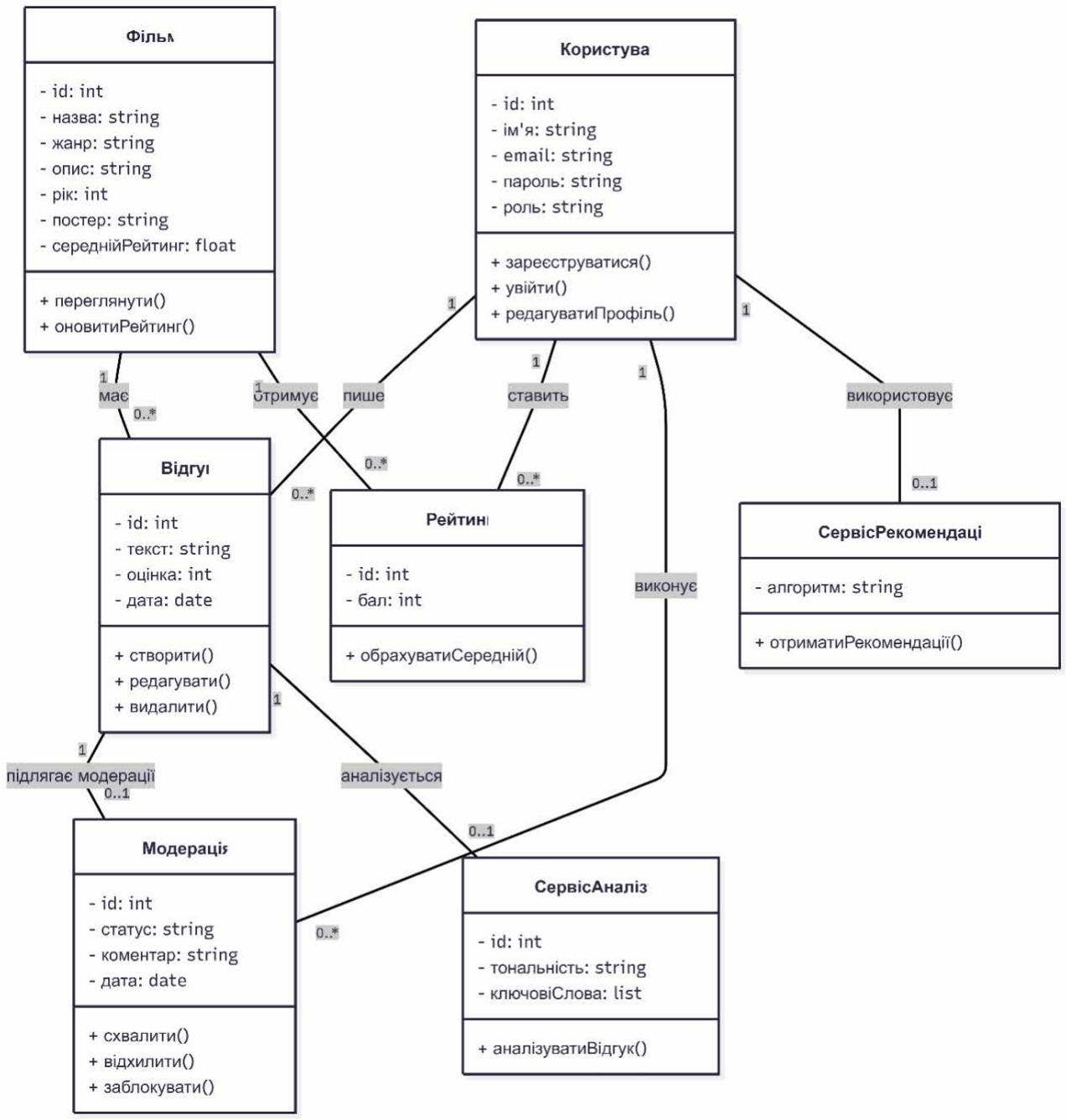

Рис. 2.4 UML-діаграма класів інформаційної системи

Таким чином, розроблена модель інформаційного забезпечення забезпечує структуроване зберігання даних, підтримує взаємозв’язки між основними сутностями системи та створює основу для подальшої реалізації серверної логіки й механізмів аналізу користувацьких відгуків. Використання нормалізованої реляційної структури та UML-моделювання дозволяє підвищити надійність, масштабованість і ефективність функціонування програмної системи.

2.3 Вибір технологій та засобів реалізації

Одним із важливих етапів проєктування програмної системи є вибір технологій та програмних засобів реалізації, оскільки саме технологічний стек визначає продуктивність, масштабованість, зручність супроводу та можливість подальшого розвитку вебзастосунку [5, 6]. Під час вибору програмних інструментів для реалізації системи аналізу глядацьких відгуків враховувалися такі критерії, як швидкість розробки, простота інтеграції компонентів, підтримка сучасних підходів до створення вебзастосунків, можливість роботи із REST API та підтримка хмарного розгортання.

Для реалізації серверної частини програмного забезпечення було обрано фреймворк FastAPI — сучасний засіб розробки REST-сервісів мовою Python. Мова програмування Python була обрана завдяки простоті реалізації серверної логіки, великій кількості бібліотек для веброзробки та підтримці сучасних підходів до створення інформаційних систем. Основними перевагами FastAPI є підтримка асинхронного програмування, автоматична генерація OpenAPI-документації та інтеграція із засобами валідації даних на основі бібліотеки Pydantic [7]. Використання FastAPI дозволяє спростити реалізацію серверної логіки та прискорити розробку API.

Для зберігання даних у системі використовується реляційна СУБД SQLite, яка характеризується компактністю, простотою інтеграції та відсутністю потреби у встановленні окремого серверного середовища [8]. Для взаємодії між серверною частиною та базою даних використовується ORM-технологія SQLAlchemy, що забезпечує об’єктно-реляційне відображення та спрощує виконання операцій роботи з даними.

Клієнтська частина системи реалізується із використанням HTML, CSS та JavaScript. Такий підхід дозволяє створити легкий вебінтерфейс без використання складних frontend-фреймворків. Для візуалізації статистичних даних використовується бібліотека Chart.js, яка забезпечує побудову інтерактивних графіків і діаграм [13].

Для автоматичного отримання інформації про фільми використовується зовнішній сервіс TMDb API [15]. Інтеграція із даним API дозволяє автоматизувати наповнення каталогу фільмів та підтримувати актуальність інформаційного наповнення системи.

Для розгортання програмного забезпечення використовуються хмарні сервіси Render і Netlify. Платформа Render застосовується для хостингу серверної частини вебзастосунку, тоді як Netlify використовується для розміщення frontend-компонентів системи. Обидва сервіси підтримують автоматичне розгортання проєкту та HTTPS-з’єднання, що спрощує адміністрування та супровід вебзастосунку.

Для узагальнення обраного технологічного стеку доцільно представити основні програмні засоби у вигляді табл. 2.3.

Таблиця 2.3

Технології та засоби реалізації програмної системи

Технологія	Призначення	Основні переваги
FastAPI	Розробка REST API та backend-логіки	Асинхронність, автоматична документація, висока продуктивність
SQLite	Зберігання даних	Простота використання, компактність
SQLAlchemy	ORM для роботи з БД	Об’єктно-реляційне відображення
HTML/ CSS/ JavaScript	Реалізація frontend	Простота інтеграції та швидкодія
Chart.js	Візуалізація статистики	Інтерактивні графіки та діаграми
TMDb API	Отримання інформації про фільми	Актуальність та автоматизація наповнення
Render	Хостинг backend	Автоматичне розгортання та HTTPS
Netlify	Хостинг frontend	Швидке розгортання статичних ресурсів

Як показано у табл. 2.3, обраний технологічний стек забезпечує поєднання простоти реалізації, достатньої продуктивності та можливості подальшого розвитку програмного забезпечення. Використання сучасних вебтехнологій і засобів хмарного розгортання дозволяє створити гнучку та масштабовану програмну систему, придатну для подальшого вдосконалення та інтеграції нових функціональних модулів.

Таким чином, обраний технологічний стек забезпечує ефективну взаємодію між компонентами програмної системи, підтримує стабільну роботу вебзастосунку та створює основу для подальшого функціонального розширення системи. Використання сучасних вебтехнологій, REST API та хмарних платформ розгортання дозволяє забезпечити зручність супроводу, продуктивність і надійність програмного забезпечення.

2.4 Проєктування функціональних модулів системи

Проєктування функціональних модулів системи є одним із ключових етапів розроблення програмного забезпечення соціальної медіа-платформи для аналізу та оцінювання кінофільмів. На цьому етапі визначається структура окремих компонентів системи, їх функціональне призначення, принципи взаємодії між модулями, а також логіка обробки даних і реалізації користувацьких сценаріїв. Основною метою проєктування є створення гнучкої, масштабованої та зрозумілої архітектури застосунку, яка забезпечуватиме стабільність роботи системи, підтримку подальшого розширення функціональності та ефективну взаємодію між клієнтською і серверною частинами [12], [18].

Функціональна структура розроблюваної системи побудована за модульним принципом, що дозволяє розділити систему на окремі незалежні компоненти відповідно до їх функціонального призначення. Такий підхід забезпечує спрощення процесу супроводу програмного забезпечення, підвищує повторне використання програмного коду та зменшує залежність між окремими

частинами застосунку. У системі виділено декілька основних функціональних модулів: модуль автентифікації та авторизації користувачів, модуль управління профілем користувача, модуль роботи з фільмами та серіалами, модуль створення та аналізу відгуків, модуль рейтингової оцінки контенту, адміністративний модуль, а також модуль взаємодії з базою даних.

Для узагальнення функціонального призначення модулів системи доцільно представити їх у вигляді табл. 2.4.

Таблиця 2.4

Основні функціональні модулі системи

Модуль	Призначення
Модуль авторизації	Реєстрація та автентифікація користувачів
Модуль роботи з фільмами	Перегляд, пошук і отримання інформації про фільми
Модуль відгуків	Створення та обробка текстових рецензій
Модуль аналізу	Аналіз тональності та ключових слів
Рейтинговий модуль	Формування рейтингу фільмів
Адміністративний модуль	Модерація контенту та управління системою

Модуль автентифікації та авторизації забезпечує реєстрацію користувачів, вхід до системи та контроль прав доступу до функціональних можливостей вебзастосунку. Після успішної авторизації користувач отримує доступ до персоналізованого інтерфейсу та можливість взаємодії з іншими модулями системи.

Модуль роботи з фільмами реалізує функції пошуку, перегляду та отримання інформації про кіноконтент. Для автоматизації наповнення каталогу система взаємодіє із зовнішнім TMDb API, що дозволяє отримувати актуальні дані про фільми, жанри, постери та рейтинги [15].

Одним із ключових компонентів системи є модуль створення та аналізу відгуків. Після введення користувачем тексту рецензії система передає дані до серверної частини, де виконується аналіз тексту та визначення емоційного

забарвлення повідомлення. Результати аналізу зберігаються у базі даних та використовуються для формування аналітичних показників і рейтингів фільмів [14, 17].

Для візуалізації процесу взаємодії між компонентами системи під час створення відгуку доцільно використати UML-діаграму послідовності (рис. 2.5).

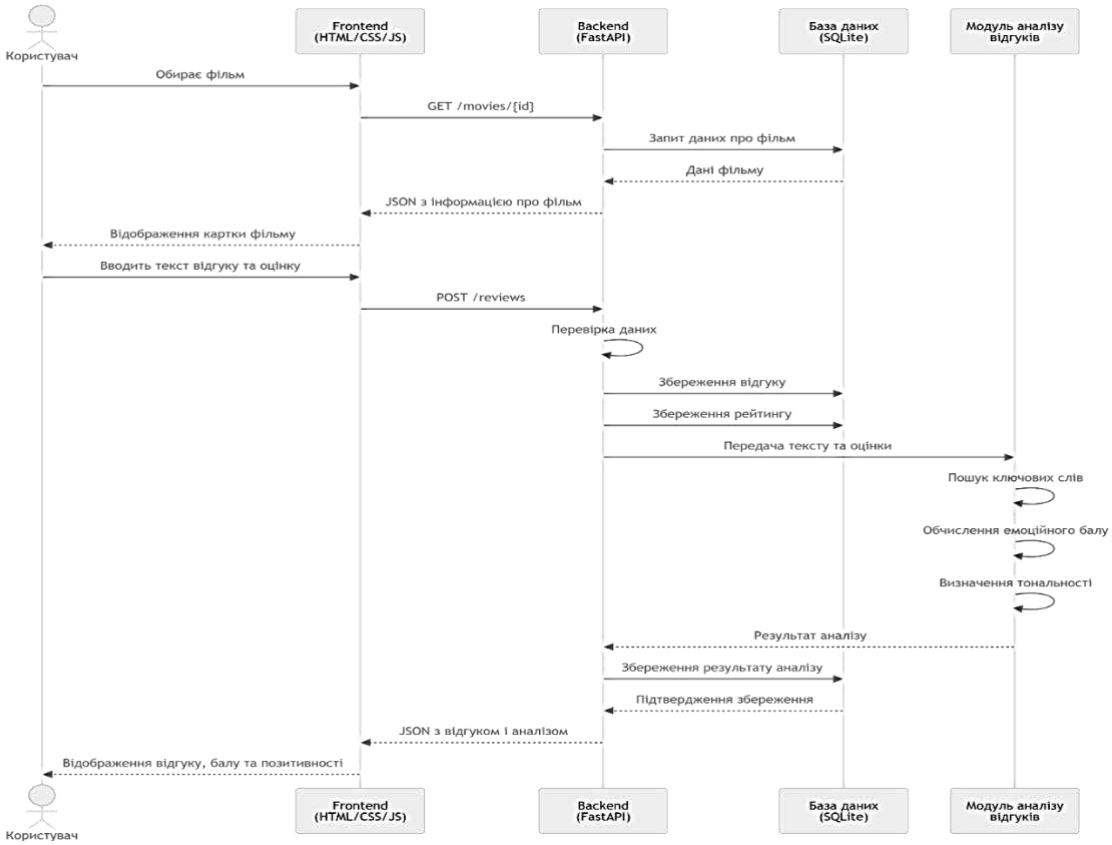

Рис. 2.5 Діаграма послідовності процесу додавання відгуку користувачем

Як показано на рис. 2.6, процес створення відгуку включає взаємодію між клієнтською та серверною частинами системи, передавання текстових даних до модуля аналізу та подальше збереження результатів у базі даних. Такий підхід забезпечує автоматизацію аналізу користувацьких рецензій та централізовану обробку інформації.

Модуль рейтингової системи забезпечує механізм виставлення оцінок користувачами та автоматичний перерахунок середнього рейтингу фільму. Під час проєктування модуля враховано необхідність уникнення дублювання оцінок одним користувачем, що дозволяє забезпечити достовірність рейтингових показників.

Адміністративний модуль призначений для управління контентом та контролю роботи системи. Основними функціями адміністратора є додавання нових фільмів, редагування інформації про контент, модерація користувацьких відгуків та управління користувачами системи.

Для відображення логіки роботи адміністративної частини системи доцільно використати UML-діаграму активності (рис. 2.6).

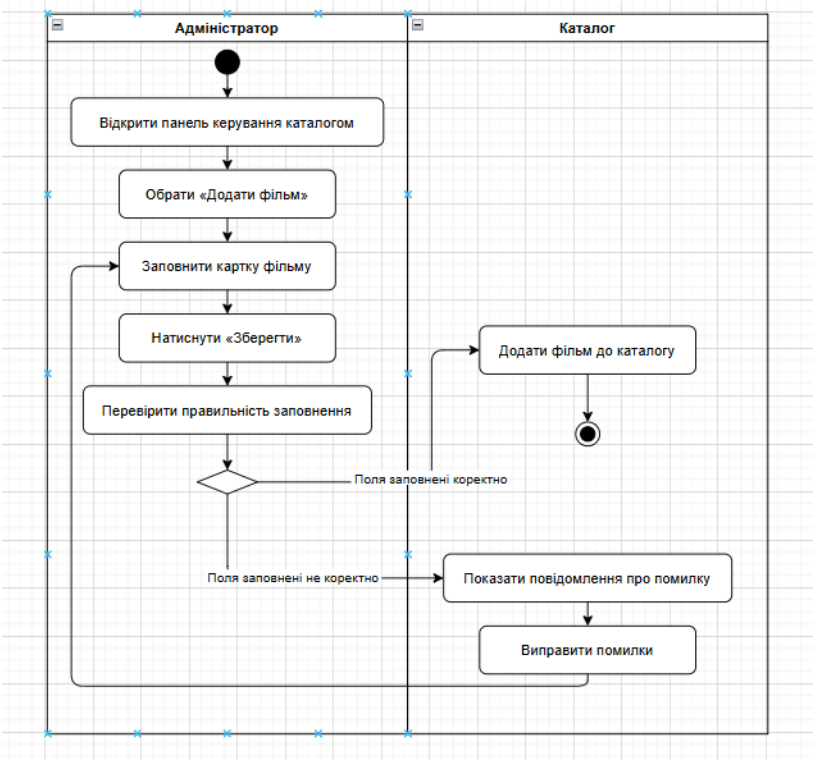

Рис. 2.6 Діаграма активності процесу додавання нового фільму адміністратором

Як показано на рис. 2.6, процес додавання нового фільму включає перевірку коректності введених даних, взаємодію із зовнішнім API та оновлення інформаційного наповнення системи. Використання діаграми активності дозволяє формалізувати послідовність виконання операцій адміністративного модуля.

Окремим функціональним компонентом є модуль взаємодії з базою даних, який забезпечує виконання CRUD-операцій та централізовану роботу із таблицями користувачів, фільмів, рейтингів і відгуків. Для взаємодії з базою даних використовується ORM SQLAlchemy, що дозволяє спростити роботу з

інформаційним середовищем системи та забезпечити незалежність програмного коду від конкретної СУБД [8].

Таким чином, проєктування функціональних модулів системи дозволило сформувати структуровану архітектуру програмного забезпечення, у якій кожен компонент виконує окремий набір функцій. Використання модульного підходу забезпечує масштабованість системи, спрощує супровід програмного забезпечення та створює можливість подальшого розвитку вебзастосунку.

2.5 Проєктування користувацького інтерфейсу

Проєктування користувацького інтерфейсу є одним із ключових етапів розробки веборієнтованого програмного забезпечення, оскільки саме інтерфейс забезпечує взаємодію користувача із системою та формує загальне враження від роботи застосунку. У межах розроблюваної системи аналізу рейтингів та відгуків до фільмів основна увага приділялася створенню інтуїтивно зрозумілого, адаптивного та функціонального інтерфейсу, який забезпечує швидкий доступ до основних можливостей системи та зручну навігацію між функціональними модулями.

Під час проєктування інтерфейсу враховувалися сучасні принципи UI/UX-дизайну, серед яких мінімізація кількості дій користувача для виконання основних операцій, логічне групування елементів, адаптивність до різних розмірів екранів та забезпечення швидкого зворотного зв’язку після виконання дій. Інтерфейс системи реалізовано у вигляді односторінкового вебзастосунку (Single Page Application), що дозволяє здійснювати динамічне оновлення контенту без повного перезавантаження сторінки.

Архітектура користувацького інтерфейсу побудована за модульним принципом, відповідно до якого кожна функціональна секція системи реалізує окремий набір задач. Перемикання між окремими сторінками реалізується за допомогою динамічного приховування та відображення HTML-секцій із

використанням JavaScript-функції showPage(), що забезпечує плавність навігації та зменшує навантаження на серверну частину системи.

Для забезпечення безпеки та розмежування функціональних можливостей у системі реалізовано механізм рольового доступу. Користувачі системи поділяються на три основні категорії: звичайний користувач, модератор та адміністратор. Для кожної ролі передбачено окремий набір інструментів та елементів інтерфейсу. Наприклад, адміністратор отримує доступ до модулів керування фільмами та інтеграції із зовнішнім API, тоді як модератор має доступ до інструментів перевірки та підтвердження відгуків.

З метою систематизації основних функціональних елементів інтерфейсу доцільно виділити ключові компоненти користувацької взаємодії, наведені у табл. 2.5.

Таблиця 2.5

Основні компоненти користувацького інтерфейсу системи

Компонент інтерфейсу	Призначення
Головна сторінка	Відображення каталогу фільмів, пошук, фільтрація, перегляд рейтингу
Сторінка фільму	Перегляд опису, рейтингу, постера та відгуків
Модуль авторизації	Реєстрація та автентифікація користувачів
Особистий профіль	Перегляд історії відгуків та персональних рекомендацій
Модуль аналітики	Відображення статистики та графіків тональності відгуків
Панель модерації	Перевірка та підтвердження користувацьких відгуків
Панель адміністратора	Додавання, редагування та видалення інформації про фільми

Центральним елементом інтерфейсу є головна сторінка системи, на якій реалізовано каталог фільмів із можливістю пошуку та фільтрації за жанрами. Для

кожного фільму формується окрема картка із постером, назвою, коротким описом та середнім рейтингом. При виборі конкретного фільму користувач переходить до детальної сторінки, де відображаються повний опис, список відгуків, статистичні показники та форма для додавання власного коментаря.

Окрему увагу приділено проєктуванню модуля аналітики, який забезпечує візуалізацію результатів аналізу текстових відгуків. Для побудови графіків використовується бібліотека Chart.js, що дозволяє формувати інтерактивні діаграми розподілу позитивних, нейтральних та негативних відгуків, а також відображати статистику модерації та рейтингових оцінок.

Після виконання користувачем будь-якої дії система автоматично оновлює відповідні елементи інтерфейсу без необхідності повторного завантаження сторінки. Такий підхід дозволяє забезпечити швидкий зворотний зв’язок і створює ефект інтерактивного «живого» застосунку. Наприклад, після додавання нового відгуку автоматично оновлюються списки коментарів, середній рейтинг фільму та графіки аналітики.

Важливою складовою інтерфейсу є відображення результатів автоматичного аналізу тексту. Після публікації відгуку система формує спеціальний інформаційний блок, у якому відображаються тональність коментаря, рівень позитивності, ключові слова та відповідні графічні індикатори. Це дозволяє користувачу одразу отримати результати аналізу власного повідомлення та підвищує інтерактивність системи.

Для організації взаємодії між клієнтською та серверною частинами застосунку використовується централізована функція apiFetch(), яка відповідає за формування HTTP-запитів, передачу токенів авторизації та обробку помилок мережевої взаємодії. Такий підхід забезпечує уніфікацію логіки взаємодії з сервером і спрощує супровід програмного коду.

У процесі проєктування користувацького інтерфейсу також враховувалися вимоги адаптивності. Інтерфейс коректно масштабується для різних типів пристроїв, зокрема персональних комп’ютерів, планшетів та мобільних телефонів. Адаптивна верстка реалізована із використанням CSS Flexbox та Grid

Layout, що дозволяє автоматично перебудовувати структуру сторінки залежно від розміру екрана користувача (Додаток А).

Для підвищення зручності використання у системі реалізовано механізми динамічного оновлення даних, візуального виділення активних елементів меню, повідомлень про успішне виконання операцій та обробки помилок введення даних. Це дозволяє підвищити загальну ергономічність програмного забезпечення та забезпечити комфортну взаємодію користувача із системою.

Таким чином, спроєктований користувацький інтерфейс забезпечує зручну навігацію, швидкий доступ до функціональних можливостей системи, інтерактивність взаємодії та ефективне відображення результатів аналізу рейтингів і текстових відгуків. Реалізований підхід дозволяє забезпечити адаптивність, масштабованість та можливість подальшого функціонального розширення вебзастосунку.

3. РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ СИСТЕМИ

3.1 Організація взаємодії компонентів програмної системи

Для забезпечення стабільної роботи програмної системи аналізу користувацьких відгуків була реалізована клієнт-серверна архітектура із розподілом функціональних компонентів між frontend-, backend- та інформаційним рівнями системи. Такий підхід дозволяє забезпечити незалежність компонентів, спростити супровід програмного забезпечення та підвищити масштабованість вебзастосунку.

На рис. 3.1 наведено схему взаємодії технологічних компонентів програмної системи.

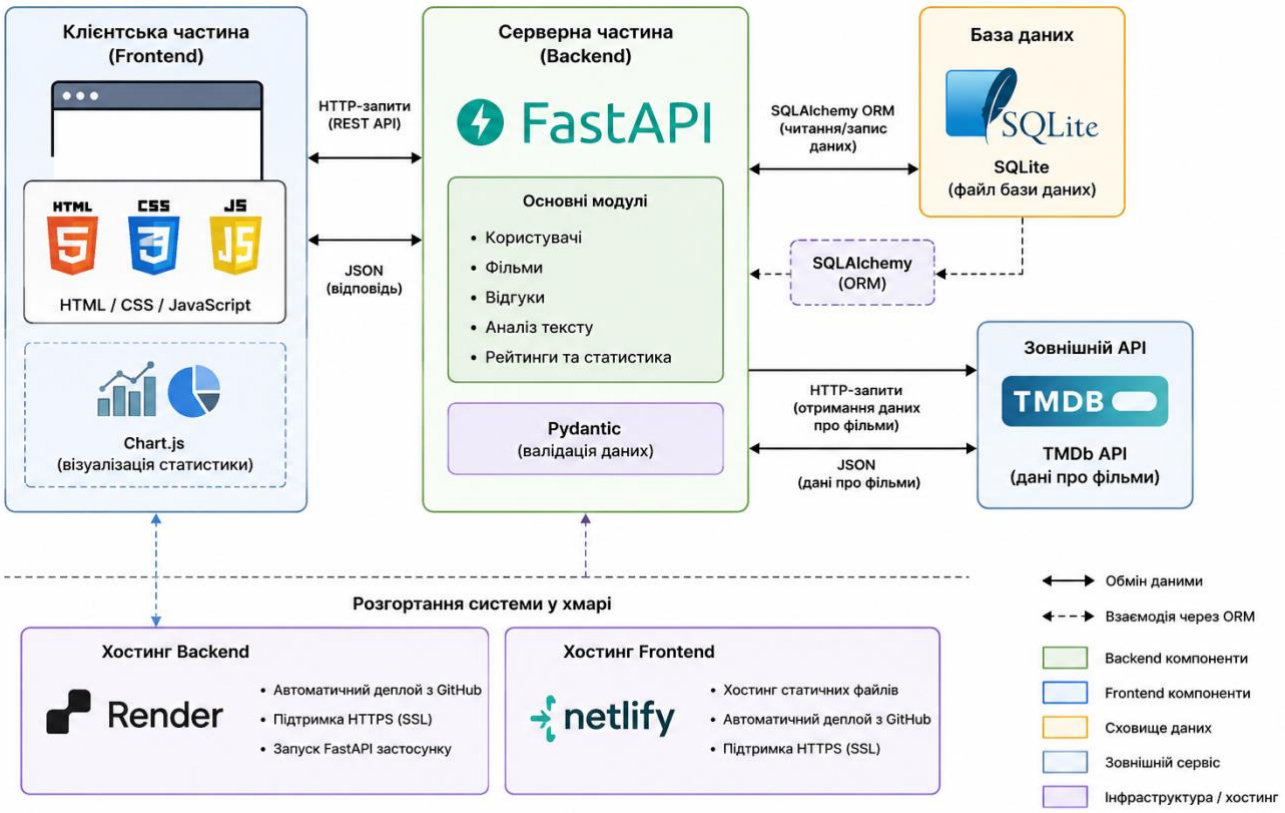

Рис. 3.1 Взаємодія технологічних компонентів програмної системи

Як показано на рис. 3.1, взаємодія технологічних компонентів системи реалізується за клієнт-серверним принципом із використанням REST API. Після

відкриття вебсторінки користувачем браузер завантажує frontend-компоненти системи, реалізовані засобами HTML, CSS та JavaScript. Клієнтська частина формує асинхронні HTTP-запити до серверного застосунку FastAPI для отримання інформації про фільми, рейтинги та користувацькі відгуки.

Frontend-компоненти забезпечують взаємодію користувача із системою, відображення списків фільмів, сторінок відгуків, статистичних даних та елементів аналітики. Для візуалізації статистичних показників використовується бібліотека Chart.js, яка дозволяє будувати інтерактивні графіки та діаграми без перезавантаження сторінки.

Після отримання запиту серверна частина виконує перевірку параметрів, обробку бізнес-логіки та взаємодію з базою даних SQLite через ORM SQLAlchemy. Для валідації вхідних даних та контролю структури JSON-запитів використовується бібліотека Pydantic, що дозволяє мінімізувати ризик передачі некоректних даних між компонентами системи.

Серверна частина містить набір функціональних модулів, серед яких модулі керування користувачами, обробки інформації про фільми, роботи з відгуками, аналізу тексту та формування статистичних показників. Такий підхід дозволяє реалізувати модульну структуру програмного забезпечення та забезпечити незалежність окремих функціональних компонентів.

Під час взаємодії із базою даних backend-компоненти здійснюють операції читання, додавання, оновлення та видалення інформації через ORM SQLAlchemy. Використання ORM-технології дозволяє спростити роботу з реляційною базою даних SQLite та забезпечує об’єктно-реляційне відображення структур даних програмної системи.

За необхідності backend також здійснює звернення до зовнішнього TMDb API для отримання актуальної інформації про кіноконтент. Зовнішній сервіс використовується для автоматичного отримання назв фільмів, описів, постерів, жанрів та рейтингових характеристик. Після завершення обробки сервер формує JSON-відповідь, яка передається клієнтській частині та використовується для оновлення інтерфейсу користувача.

У процесі створення користувацького відгуку додатково активується модуль аналізу тексту, який автоматично визначає емоційне забарвлення повідомлення, формує аналітичні показники та зберігає результати аналізу у базі даних. Отримані результати використовуються для оновлення рейтингу фільму та відображення статистичних даних у вебінтерфейсі системи.

Розгортання програмної системи реалізовано із використанням хмарних платформ Render та Netlify. Backend-застосунок FastAPI розміщується на платформі Render, тоді як frontend-компоненти розгортаються через Netlify у вигляді статичних вебресурсів. Такий підхід дозволяє забезпечити автоматичне розгортання застосунку, підтримку HTTPS-з’єднання та спрощує процес супроводу програмної системи.

Таким чином, реалізована організація взаємодії компонентів забезпечує ефективний обмін даними між клієнтською та серверною частинами системи, підтримує модульність архітектури та створює основу для подальшого функціонального розширення програмного забезпечення.

3.2 Реалізація серверної частини програмної системи

Серверна частина програмної системи реалізує основну бізнес-логіку вебзастосунку та забезпечує обробку клієнтських запитів, роботу з базою даних, модерацію контенту, авторизацію користувачів і виконання аналізу текстових повідомлень. Програмна реалізація backend-компонентів побудована за модульним принципом, що дозволило розділити функціональні частини системи та спростити супровід програмного коду.

Під час реалізації серверної частини особлива увага приділялася забезпеченню розширюваності програмного забезпечення, повторному використанню функціональних компонентів та розмежуванню рівнів доступу до даних, бізнес-логіки й API-маршрутів.

Структура серверної частини організована у вигляді окремих Python-модулів, кожен із яких виконує визначене функціональне призначення. Основні модулі backend-компонентів наведено у табл. 3.1.

Таблиця 3.1

Основні модулі серверної частини програмної системи

Модуль	Призначення
models.py	Опис моделей даних та зв’язків між таблицями
crud.py	Реалізація операцій роботи з даними
routers/	Реалізація REST API-маршрутів
schemas.py	Валідація вхідних і вихідних даних
utils.py	Допоміжні функції та модулі аналізу тексту
auth.py	Реалізація авторизації та JWT-перевірок

Для роботи із даними використовується ORM-підхід, у межах якого таблиці бази даних представлені у вигляді Python-класів. Це дозволяє реалізовувати операції створення, редагування та отримання інформації без використання великої кількості SQL-запитів. Основними моделями системи є User, Movie, Review, Rating та Analysis. Модель User використовується для зберігання інформації про користувачів та їх ролі доступу. Модель Movie містить дані про фільми та рейтингові показники. Модель Review забезпечує зберігання текстових відгуків користувачів, а модель Rating використовується для числових оцінок фільмів. Результати автоматичного аналізу текстових повідомлень зберігаються у моделі Analysis.

Між ORM-моделями реалізовано зв’язки типу «один-до-багатьох» та «один-до-одного». Один користувач може створювати декілька відгуків та оцінок, тоді як кожен відгук пов’язаний лише з одним результатом аналізу тексту. Така структура дозволяє забезпечити цілісність даних та автоматизувати навігацію між взаємопов’язаними сутностями.

Для організації серверної логіки використовується система REST API-маршрутів, згрупованих за функціональними категоріями. Частина маршрутів відповідає за роботу із фільмами, частина — за користувачів, модерацію відгуків,

статистику та аналітичні функції. Основні маршрути серверної частини наведено у табл. 3.2.

Таблиця 3.2

Основні API-маршрути серверної частини

Маршрут	Функціональне призначення
/movies	Робота з інформацією про фільми
/reviews	Створення та отримання відгуків
/reviews/{id}/moderate	Модерація повідомлень
/auth/login	Авторизація користувачів
/auth/register	Реєстрація користувачів
/users/me	Отримання профілю користувача
/stats/reviews	Формування статистики
/tmdb/search	Пошук фільмів через зовнішній сервіс

Під час створення нового відгуку серверна частина автоматично виконує запуск модуля аналізу тексту. Після обробки текстового повідомлення система визначає емоційне забарвлення відгуку, обчислює рівень позитивності та формує перелік ключових слів. Отримані результати автоматично зберігаються у базі даних та використовуються для оновлення рейтингових показників фільму.

Для реалізації механізмів безпеки використовується JWT-автентифікація. Після успішної авторизації сервер формує токен доступу, який використовується під час подальших звернень до захищених API-маршрутів. Додатково реалізовано перевірку ролей користувачів, що дозволяє обмежити доступ до адміністративних функцій та модерації відгуків.

Окрему роль у серверній частині відіграє модуль валідації даних, який забезпечує перевірку структури HTTP-запитів та коректності параметрів. Це дозволяє мінімізувати ризик обробки некоректних або неповних даних та підвищує стабільність роботи програмної системи.

Таким чином, серверна частина програмної системи забезпечує централізовану реалізацію бізнес-логіки, підтримку механізмів авторизації,

модерації та аналізу текстових повідомлень, а також ефективну взаємодію між клієнтськими компонентами, базою даних та зовнішніми сервісами.

3.3 Реалізація інформаційного забезпечення та бази даних

Інформаційне забезпечення програмної системи реалізоване на основі реляційної бази даних SQLite, яка використовується для зберігання користувацьких даних, інформації про фільми, текстових відгуків, рейтингових оцінок та результатів аналізу текстових повідомлень. Використання реляційного підходу дозволило забезпечити структуроване зберігання інформації, підтримку зв’язків між сутностями та цілісність даних у межах програмної системи [8, 17].

Фізичне зберігання даних реалізується у вигляді локального файлу database.db, який автоматично створюється під час першого запуску серверної частини застосунку. Формування структури таблиць виконується засобами ORM SQLAlchemy шляхом автоматичної генерації схем бази даних на основі описаних моделей. Для створення таблиць, зовнішніх ключів та службових зв’язків використовується механізм Base.metadata.create_all(), що дозволяє автоматизувати початкове розгортання інформаційної бази та спростити супровід програмної системи.

Структура інформаційного забезпечення побудована відповідно до функціональних потреб системи та включає таблиці користувачів, фільмів, відгуків, рейтингових оцінок і результатів аналізу текстових повідомлень. Основні таблиці бази даних наведено у табл. 3.3.

Таблиця users містить інформацію про користувачів програмної системи, зокрема ім’я користувача, адресу електронної пошти, хешований пароль, роль доступу та дату реєстрації. Для забезпечення унікальності облікових записів реалізовано обмеження унікальності для поля username.

Таблиця movies використовується для зберігання інформації про кіноконтент, отриманий із зовнішнього сервісу TMDb API або доданий адміністратором вручну. Структура таблиці включає ідентифікатор фільму у

TMDb, назву, опис, рік випуску, адресу постера, жанри та середній рейтинговий показник.

Таблиця 3.3

Основні таблиці інформаційної бази програмної системи

Таблиця	Призначення
users	Зберігання облікових записів користувачів
movies	Зберігання інформації про фільми
reviews	Зберігання текстових відгуків
ratings	Зберігання рейтингових оцінок
analysis	Зберігання результатів аналізу тексту

У таблиці reviews зберігаються текстові повідомлення користувачів, статуси модерації, дата створення запису та зовнішні ключі для зв’язку із таблицями users і movies. Для контролю процесу модерації використовується перелік статусів pending, approved та rejected.

Таблиця ratings призначена для зберігання числових оцінок користувачів. Виділення рейтингових показників в окрему таблицю дозволяє реалізувати незалежне оновлення оцінок та спрощує формування статистичних показників системи.

Для зберігання результатів автоматичного аналізу текстових повідомлень використовується таблиця analysis. У ній зберігаються тональний бал повідомлення, категорія емоційного забарвлення, рівень позитивності та перелік ключових слів. Дані таблиці використовуються для формування аналітичних показників і візуалізації результатів аналізу користувацьких відгуків.

Зв’язки між таблицями реалізовані засобами зовнішніх ключів Foreign Key, що дозволяє забезпечити цілісність інформаційної бази та підтримку узгодженості даних [18]. Один користувач може створювати декілька відгуків і рейтингових оцінок, тоді як один фільм може містити множину пов’язаних відгуків та оцінок. Для таблиці analysis реалізовано зв’язок типу «один-до-

одного» із таблицею reviews, оскільки кожному текстовому повідомленню відповідає лише один результат автоматичного аналізу.

Для підвищення продуктивності роботи системи реалізовано індексацію полів, які найчастіше використовуються під час пошуку та фільтрації інформації. Індекси створені для зовнішніх ключів, а також для полів username та tmdb_id, що дозволяє прискорити виконання операцій авторизації користувачів, пошуку фільмів та отримання пов’язаних записів.

Підтримка цілісності даних додатково забезпечується на рівні ORM SQLAlchemy шляхом використання каскадного видалення залежних записів. Наприклад, у разі видалення користувача автоматично видаляються його відгуки та рейтингові оцінки, що дозволяє уникнути появи неузгоджених записів у базі даних.

Використання SQLite у поєднанні із SQLAlchemy забезпечує достатню продуктивність роботи програмної системи, спрощує розгортання програмного забезпечення та дозволяє реалізувати ефективну взаємодію між серверною частиною застосунку та інформаційною базою [7, 8]. Реалізована структура інформаційного забезпечення підтримує цілісність даних, забезпечує ефективне зберігання інформації та створює основу для подальшого функціонального розширення програмної системи.

3.4 Реалізація клієнтської частини програмної системи

Клієнтська частина програмної системи реалізована у вигляді односторінкового вебзастосунку (Single Page Application, SPA), який забезпечує взаємодію користувача із серверною частиною системи, відображення інформації про фільми, роботу з користувацькими відгуками та візуалізацію статистичних показників. Реалізація frontend-компонентів виконана засобами HTML, CSS та JavaScript без використання додаткових frontend-фреймворків або систем автоматичного збирання проєкту. Такий підхід дозволив зменшити

складність програмної реалізації та забезпечити швидке завантаження вебзастосунку у браузері [12, 13].

Архітектура клієнтської частини побудована за принципом динамічного оновлення вмісту сторінки без її повного перезавантаження. Для реалізації механізму навігації використовується функція showPage(page), яка забезпечує перемикання між функціональними секціями застосунку шляхом приховування та відображення відповідних HTML-компонентів. Усі сторінки системи реалізовані у вигляді окремих блоків , що дозволяє організувати SPA-архітектуру та забезпечити швидке оновлення інтерфейсу користувача.

Основні функціональні сторінки клієнтської частини програмної системи наведено у табл. 3.4.

Таблиця 3.4

Основні сторінки клієнтської частини системи

Сторінка	Призначення
Головна сторінка	Відображення популярних фільмів
Каталог фільмів	Перегляд інформації про фільми
Сторінка відгуків	Робота з користувацькими повідомленнями
Авторизація	Вхід до системи
Реєстрація	Створення облікового запису
Панель модерації	Керування статусами відгуків
Сторінка статистики	Візуалізація аналітичних показників

Приклади реалізованого інтерфейсу користувача наведені у Додатку А.

Для організації взаємодії із серверною частиною використовується набір асинхронних HTTP-запитів, реалізованих засобами JavaScript. Усі звернення до REST API виконуються через централізовану функцію apiFetch(), яка забезпечує формування HTTP-запитів, передачу JWT-токенів авторизації та обробку помилок мережевої взаємодії. Використання єдиного механізму взаємодії із сервером дозволяє спростити програмний код та уніфікувати обробку API-відповідей.

Після успішної авторизації JWT-токен зберігається у localStorage браузера та використовується для доступу до захищених функціональних можливостей системи. Клієнтська частина також реалізує механізм розмежування ролей користувачів. Під час формування навігаційного меню перевіряється роль поточного користувача, у результаті чого окремі елементи інтерфейсу, пов’язані з адміністративними функціями або модерацією відгуків, відображаються лише для користувачів із відповідними правами доступу.

Формування динамічного контенту виконується засобами JavaScript шляхом генерації HTML-компонентів на основі отриманих JSON-відповідей сервера. Для створення елементів інтерфейсу використовуються шаблонні рядки та методи document.createElement(). Після отримання інформації про фільми клієнтська частина автоматично формує картки фільмів із використанням назв, постерів, року випуску та рейтингових показників. Аналогічним чином реалізовано механізм відображення користувацьких відгуків та результатів аналізу тексту.

Для кожного відгуку система додатково відображає результати автоматичного аналізу текстових повідомлень, зокрема категорію емоційного забарвлення, рівень позитивності та ключові слова. Візуалізація результатів аналізу реалізована у вигляді окремих інформаційних блоків із використанням прогрес-барів та стилізованих HTML-компонентів. Такий підхід дозволяє підвищити інформативність вебінтерфейсу та спрощує сприйняття результатів аналітичної обробки даних користувачами.

Обробка користувацьких форм реалізована в асинхронному режимі без перезавантаження сторінки. Після успішного виконання операції клієнтська частина автоматично оновлює відповідні елементи інтерфейсу: списки відгуків, рейтингові показники, статистичні блоки та результати аналізу тексту. Такий механізм дозволяє забезпечити інтерактивність вебзастосунку та покращує зручність взаємодії користувача із системою [14].

Для візуалізації статистичних показників використовується бібліотека Chart.js, яка забезпечує побудову інтерактивних графіків і діаграм у браузері [13].

Дані для візуалізації отримуються із серверного маршруту /stats/reviews та відображаються у вигляді кругових і стовпчикових діаграм. Зокрема, система дозволяє відображати співвідношення позитивних, нейтральних і негативних відгуків, а також статистику модерації повідомлень. Приклади реалізації графічної візуалізації статистичних показників наведені у Додатку А.

Для стилізації інтерфейсу використовується CSS із адаптивним розміщенням елементів сторінки. Це дозволяє забезпечити коректне відображення вебзастосунку на різних типах пристроїв та підтримати базові принципи адаптивного вебдизайну [12].

Таким чином, реалізована клієнтська частина програмної системи забезпечує інтерактивну взаємодію користувача із серверними компонентами, підтримує асинхронне оновлення контенту, реалізує механізми авторизації та розмежування прав доступу, а також забезпечує візуалізацію результатів аналітичної обробки користувацьких відгуків.

3.5 Реалізація модуля аналізу тексту

Однією з ключових функціональних особливостей розробленої програмної системи є модуль автоматичного аналізу текстових відгуків користувачів. Його основним призначенням є визначення емоційного забарвлення повідомлень, формування показників позитивності тексту та виділення ключових слів, що характеризують загальне ставлення користувача до фільму. Реалізація такого модуля дозволяє автоматизувати обробку текстової інформації та забезпечити формування додаткових аналітичних характеристик відгуків [14, 18].

Загальна схема роботи модуля аналізу тексту наведена на рис. 3.2.

Для реалізації аналізу текстових повідомлень було обрано rule-based підхід, заснований на використанні словників позитивних і негативних лексичних конструкцій. На відміну від складних моделей машинного навчання, такий підхід характеризується простотою реалізації, прозорістю алгоритму та низькими вимогами до обчислювальних ресурсів.

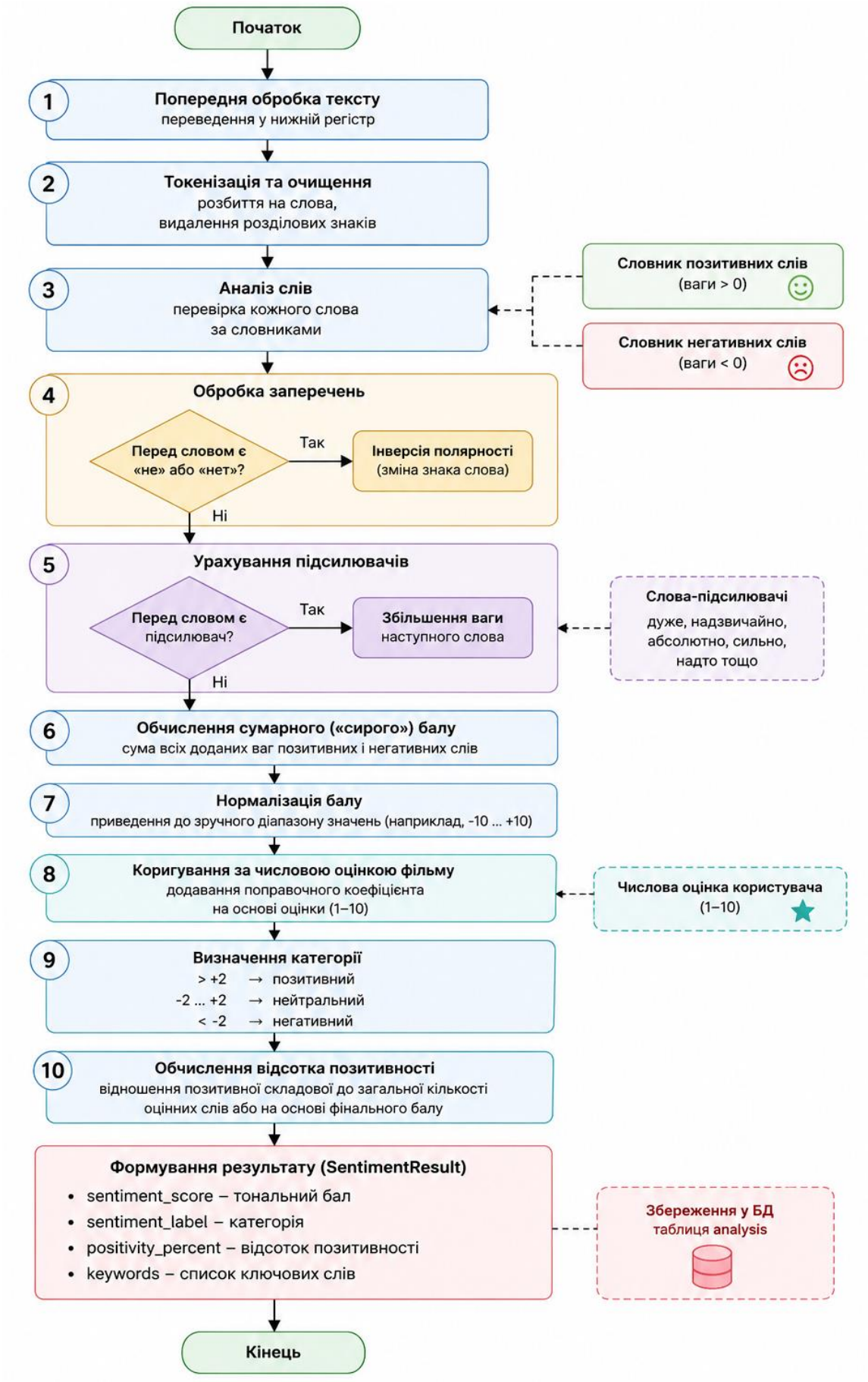

Рис. 3.2 Алгоритм функціонування модуля аналізу текстових повідомлень

Крім того, rule-based аналіз дозволяє контролювати логіку формування результатів та легко модифікувати набір словникових конструкцій відповідно до потреб системи [18].

Модуль аналізу тексту реалізований у серверному модулі utils.py та використовує набір словників позитивних і негативних слів. До позитивних конструкцій належать слова, які характеризують позитивне ставлення користувача до фільму, наприклад «чудовий», «захопливий», «талановитий» або «сподобався». Негативний словник містить слова на кшталт «нудний», «жахливий», «розчарував» або «слабкий». Кожному слову призначається відповідний ваговий коефіцієнт, який використовується під час обчислення загального тонального показника.

На початковому етапі аналізу текст повідомлення переводиться у нижній регістр та проходить процедуру попередньої обробки. Після цього виконується токенізація тексту шляхом розбиття повідомлення на окремі слова та очищення від розділових знаків. Для кожного слова виконується перевірка його належності до позитивного або негативного словника.

Під час аналізу формується сумарний тональний бал повідомлення. У разі виявлення позитивного слова до загального результату додається позитивний коефіцієнт, тоді як негативні слова формують від’ємний внесок у підсумковий показник. Додатково модуль підтримує обробку заперечних конструкцій. Якщо перед оціночним словом знаходиться заперечення типу «не» або «нет», емоційне значення слова інвертується. Такий підхід дозволяє підвищити точність інтерпретації текстових повідомлень та зменшити кількість помилкових результатів аналізу.

Окремо реалізовано механізм урахування слів-підсилювачів, зокрема «дуже», «надзвичайно», «абсолютно» та інших подібних конструкцій. У разі виявлення таких слів ваговий коефіцієнт наступного оціночного слова збільшується, що дозволяє точніше враховувати інтенсивність емоційного забарвлення повідомлення.

Після завершення аналізу виконується нормалізація сумарного тонального показника, що дозволяє привести результати до уніфікованого діапазону значень. На основі отриманого результату система визначає категорію повідомлення: позитивне, нейтральне або негативне. Основні категорії результатів аналізу наведено у табл. 3.5.

Таблиця 3.5

Категорії результатів аналізу тексту

Тональний бал	Категорія повідомлення
> +2	Позитивне
від -2 до +2	Нейтральне
< -2	Негативне

Додатково модуль аналізу враховує числову оцінку, яку користувач встановлює фільму під час створення відгуку. Рейтингова оцінка використовується як поправочний коефіцієнт для коригування фінального тонального показника. Такий підхід дозволяє підвищити коректність аналізу коротких текстових повідомлень, у яких може бути недостатньо оціночної лексики для формування об’єктивного результату.

Результатом роботи модуля є об’єкт SentimentResult, який містить сумарний тональний бал, категорію повідомлення, рівень позитивності та перелік ключових слів. Отримані результати автоматично зберігаються у таблиці analysis та використовуються для формування статистичних показників і візуалізації результатів аналізу у клієнтській частині системи.

Модуль аналізу тексту виконується безпосередньо на серверній стороні під час створення нового відгуку. Це дозволяє забезпечити миттєве отримання результатів аналізу без необхідності використання зовнішніх сервісів або тривалих обчислень. Після завершення аналізу користувач одразу отримує оновлену інформацію про тональність повідомлення та відповідні аналітичні показники.

Приклади результатів аналізу текстових повідомлень та їх відображення у вебінтерфейсі наведені у Додатку А.

Таким чином, реалізований модуль аналізу тексту забезпечує автоматизовану обробку користувацьких повідомлень, визначення емоційного забарвлення відгуків та формування аналітичних характеристик текстових даних, що дозволяє підвищити інформативність програмної системи та розширити її функціональні можливості.

3.6 Алгоритмізація роботи програмної системи

Алгоритмізація програмної системи спрямована на забезпечення послідовної взаємодії між клієнтською та серверною частинами застосунку, автоматизацію обробки користувацьких запитів, реалізацію механізмів авторизації та підтримку аналітичної обробки текстових повідомлень. Реалізовані алгоритми забезпечують узгоджену роботу всіх функціональних компонентів системи та підтримують обробку даних у режимі реального часу.

Загальна схема функціонування програмної системи базується на послідовності взаємодії користувача із frontend-компонентами, передачі HTTP-запитів до серверної частини та подальшої обробки даних на рівні backend-модулів і бази даних. Загальний алгоритм роботи системи наведено на рис. 3.3.

На початковому етапі користувач відкриває вебзастосунок через браузер, після чого виконується завантаження статичних frontend-компонентів із хмарного середовища Netlify. Після ініціалізації клієнтської частини система перевіряє наявність JWT-токена у локальному сховищі браузера localStorage. У разі наявності токена frontend-компоненти надсилають запит до маршруту /users/me для отримання інформації про поточного користувача та визначення його ролі доступу.

На основі отриманої ролі автоматично формується структура навігаційного меню та перелік доступних функціональних можливостей системи.

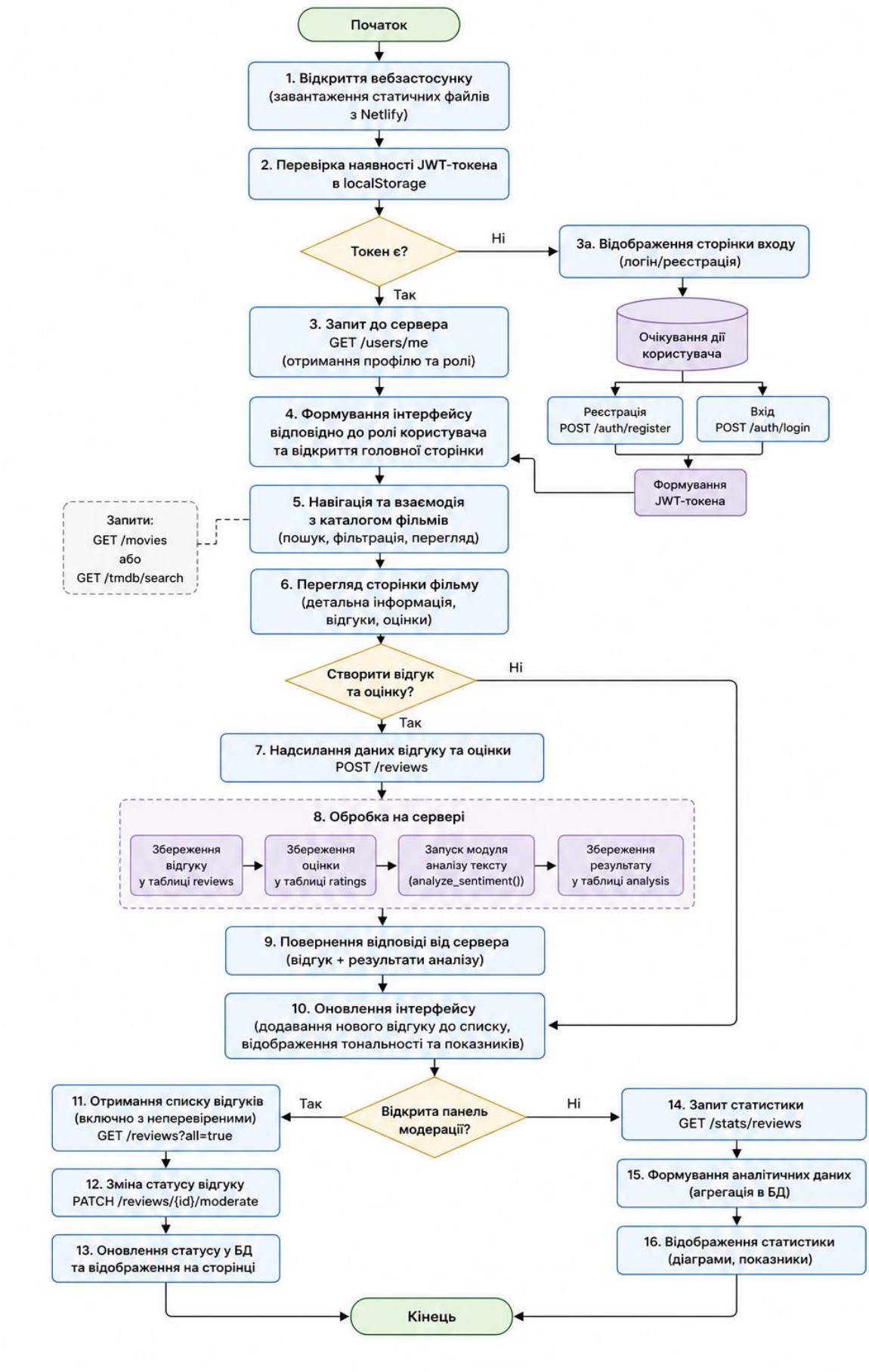

Рис. 3.3 Загальний алгоритм роботи програмної системи

Для користувачів із роллю admin або moderator додатково активуються сторінки модерації контенту та керування інформацією про фільми. У разі відсутності авторизації користувачу доступний лише базовий функціонал перегляду інформації про фільми та відгуки.

Після переходу до каталогу фільмів користувач може виконувати пошук інформації через відповідні форми інтерфейсу. У процесі пошуку frontend-компоненти надсилають HTTP-запити до серверних маршрутів /movies або /tmdb/search. Серверна частина виконує обробку запиту, звертається до локальної бази даних або зовнішнього сервісу TMDb API та повертає структуровану JSON-відповідь із результатами пошуку.

Під час відкриття сторінки конкретного фільму система додатково виконує отримання пов’язаних відгуків, рейтингових оцінок та результатів аналізу текстових повідомлень. Отримані дані використовуються для динамічного формування інформаційних блоків інтерфейсу користувача.

Алгоритм створення нового відгуку включає декілька взаємопов’язаних етапів. Після введення текстового повідомлення та рейтингової оцінки frontend-компоненти формують POST-запит до маршруту /reviews. Серверна частина виконує перевірку коректності отриманих даних, після чого інформація про відгук та оцінку зберігається у відповідних таблицях бази даних.

Після успішного збереження текстового повідомлення автоматично запускається модуль аналізу тексту. У процесі аналізу система визначає тональний бал повідомлення, категорію емоційного забарвлення, рівень позитивності та формує перелік ключових слів. Результати аналізу автоматично записуються до таблиці analysis та зв’язуються із відповідним відгуком через зовнішній ключ review_id.

Після завершення серверної обробки frontend-компоненти отримують JSON-відповідь, яка містить інформацію про новий відгук та результати аналізу тексту. Клієнтська частина автоматично оновлює вміст сторінки без її перезавантаження, додаючи новий відгук до загального списку повідомлень та відображаючи результати аналізу у вигляді окремого аналітичного блоку.

Для реалізації модерації використовується алгоритм перевірки статусів повідомлень, у межах якого адміністратор або модератор може змінювати статус відгуків через маршрут /reviews/{id}/moderate з подальшим оновленням записів у базі даних. Формування статистичних показників базується на агрегованих SQL-запитах до таблиць reviews, ratings та analysis. Після відкриття сторінки аналітики frontend-компоненти надсилають запит до маршруту /stats/reviews, а серверна частина формує статистику тональності та модерації повідомлень для побудови графічних діаграм. У поточній версії системи оновлення статистики виконується шляхом повторного отримання даних під час відкриття сторінки без використання WebSocket-технологій, що дозволяє забезпечити актуальність інформації без ускладнення архітектури застосунку [12, 13].Для узагальнення основних алгоритмічних процесів системи доцільно виділити основні етапи обробки даних, наведені у табл. 3.6.

Таблиця 3.6

Основні алгоритмічні процеси програмної системи

Процес	Результат виконання
Авторизація користувача	Формування JWT-токена
Пошук фільмів	Отримання JSON-відповіді
Створення відгуку	Збереження тексту та рейтингу
Аналіз тексту	Формування тонального результату
Модерація повідомлень	Оновлення статусу відгуку
Формування статистики	Побудова аналітичних показників

Таким чином, реалізована алгоритмізація програмної системи забезпечує узгоджену взаємодію між функціональними компонентами, автоматизує процеси обробки користувацьких запитів, підтримує аналіз текстових повідомлень та забезпечує формування статистичних і аналітичних показників у межах вебзастосунку.

4. ТЕСТУВАННЯ ТА ВПРОВАДЖЕННЯ ПРОГРАМНОЇ СИСТЕМИ

4.1 Тестування та перевірка працездатності системи

Одним із завершальних етапів розробки програмної системи є проведення тестування, метою якого є перевірка коректності функціонування програмного забезпечення, виявлення можливих помилок та оцінювання стабільності роботи системи в умовах, наближених до реальної експлуатації. Проведення тестування дозволяє підтвердити відповідність реалізованого функціоналу поставленим вимогам, а також оцінити зручність використання вебзастосунку для різних категорій користувачів.

У межах даної роботи основну увагу було приділено функціональному тестуванню програмної системи. Враховуючи навчальний характер проєкту та обмежений обсяг реалізації, автоматизоване модульне тестування не застосовувалося. Перевірка працездатності виконувалася шляхом послідовного проходження основних сценаріїв взаємодії із системою від імені звичайного користувача, модератора та адміністратора. Такий підхід дозволив оцінити коректність роботи як клієнтської частини застосунку, так і серверної логіки, механізмів авторизації, взаємодії з базою даних та зовнішніми API.

Під час тестування перевірялося завантаження вебінтерфейсу, коректність відображення сторінок у сучасних браузерах, стабільність отримання даних із сервера, а також правильність роботи функцій пошуку та фільтрації інформації. Окремо здійснювалася перевірка взаємодії із зовнішнім API сервісу TMDb, що використовується для отримання інформації про фільми. У процесі тестування було підтверджено, що результати пошуку коректно завантажуються та відображаються у користувацькому інтерфейсі без критичних помилок навіть при використанні різних пошукових запитів.

Важливим етапом перевірки стала оцінка роботи механізму додавання нових фільмів до локальної бази даних системи. Було протестовано як

автоматичне наповнення каталогу популярними фільмами, так і ручне створення записів адміністратором. У результаті перевірки встановлено, що після виконання відповідних операцій нові записи коректно зберігаються у базі даних та відображаються у каталозі вебзастосунку.

Окрему увагу приділено тестуванню підсистеми роботи з відгуками користувачів. Було перевірено можливість створення текстових відгуків різного емоційного забарвлення, виставлення оцінок фільмам, а також подальшого редагування або повторного надсилання інформації. Після публікації відгуків перевірялося автоматичне формування результатів аналізу тональності тексту, зокрема визначення категорії емоційного забарвлення, рівня позитивності та набору ключових слів.

Результати тестування показали, що модуль аналізу тональності коректно визначає позитивні та негативні відгуки для більшості типових текстів користувачів. Водночас складні конструкції з іронією або змішаними емоціями можуть інтерпретуватися менш точно, що пояснюється використанням rule-based підходу на основі словникового аналізу.

Під час тестування також було перевірено механізми модерації відгуків. Підтверджено, що модератор може переглядати відгуки, змінювати їх статус та керувати публікацією контенту. Відгуки, які не пройшли модерацію, не відображалися на сторінках фільмів.

Окремо здійснювалася перевірка системи розмежування прав доступу. Спроби виконання адміністративних функцій без відповідних прав блокувалися системою, що підтвердило коректність роботи механізмів авторизації.

Для оцінювання стійкості системи перевірялася обробка некоректних даних, зокрема порожніх відгуків, надмірно довгих текстів та неіснуючих ідентифікаторів фільмів. У всіх випадках система коректно обробляла помилки та відображала зрозумілі повідомлення користувачу.

У ході тестування було виявлено окремі незначні недоліки роботи системи. Зокрема, при багаторазовому швидкому натисканні кнопки надсилання відгуку існувала можливість дублювання записів. Для усунення даної проблеми було

реалізовано тимчасове блокування кнопки на період виконання HTTP-запиту до сервера, що дозволило усунути повторне створення однакових записів.

Для узагальнення результатів тестування доцільно представити основні перевірені функції системи у вигляді табл. 4.1.

Таблиця 4.1

Результати функціонального тестування системи

Функція системи	Сценарій перевірки	Результат
Завантаження вебінтерфейсу	Відкриття системи у браузерах Chrome, Firefox, Edge	Працює коректно
Пошук фільмів	Виконання пошуку через TMDb API	Результати відображаються коректно
Додавання фільмів	Створення записів адміністратором	Дані успішно зберігаються
Робота з відгуками	Створення та редагування відгуків	Працює стабільно
Аналіз тональності	Перевірка позитивних та негативних текстів	Результати відповідають очікуванням
Модерація	Зміна статусу відгуків	Непідтверджені записи приховуються
Розмежування доступу	Спроба доступу без відповідних прав	Доступ блокується
Обробка помилок	Надсилання некоректних даних	Помилки обробляються коректно

Приклади користувацького інтерфейсу системи наведено у Додатку А, а результати функціонального тестування та перевірки працездатності системи — у Додатку Б.

Таким чином, проведене тестування підтвердило працездатність програмної системи та коректність реалізації основного функціоналу.

Вебзастосунок забезпечує стабільну роботу механізмів пошуку, управління фільмами, аналізу користувацьких відгуків, модерації контенту та формування статистичних даних, що свідчить про досягнення поставленої мети розробки.

4.2 Розгортання системи та вимоги до середовища

Розгортання програмної системи є важливим етапом впровадження вебзастосунку, оскільки саме на цьому етапі забезпечується доступність сервісу для кінцевих користувачів у глобальній мережі Інтернет. Оскільки розроблений застосунок має веборієнтовану архітектуру, для його використання не потрібне встановлення додаткового спеціалізованого програмного забезпечення на стороні користувача. Доступ до функціональних можливостей системи здійснюється через стандартний веббраузер.

Мінімальні вимоги до апаратного та програмного забезпечення користувацького середовища наведено у табл. 4.2.

Таблиця 4.2

Вимоги до користувацького середовища

Категорія	Мінімальні вимоги
Апаратне забезпечення	Комп’ютер, ноутбук або мобільний пристрій
Оперативна пам’ять	Від 2 ГБ
Мережеве підключення	Доступ до мережі Інтернет
Операційна система	Windows, Linux, macOS, Android, iOS
Веббраузер	Google Chrome, Mozilla Firefox, Microsoft Edge, Safari

Проведене тестування підтвердило коректну роботу системи у браузерах Google Chrome та Mozilla Firefox. Завдяки використанню адаптивного вебінтерфейсу застосунок може використовуватися як на персональних комп’ютерах, так і на мобільних пристроях.

Для локального запуску серверної частини застосунку необхідне встановлення середовища Python версії 3.10 або новішої, а також програмних залежностей, перелік яких визначений у файлі requirements.txt. Запуск серверної частини здійснюється за допомогою ASGI-сервера Uvicorn. Клієнтська частина системи не потребує спеціального серверного програмного забезпечення та може бути розгорнута як набір статичних HTML-, CSS- та JavaScript-файлів.

З метою забезпечення віддаленого доступу до системи було реалізовано хмарне розгортання вебзастосунку. Архітектура системи побудована за дворівневим принципом та передбачає окреме розміщення клієнтської і серверної частин. Такий підхід дозволяє підвищити масштабованість, спростити супровід системи та забезпечити незалежне оновлення окремих компонентів.

Серверна частина застосунку, реалізована із використанням FastAPI, була розгорнута на хмарній платформі Render. Для автоматизації процесу оновлення застосовано інтеграцію з GitHub-репозиторієм, що дозволяє автоматично виконувати повторне розгортання системи після внесення змін до програмного коду. Запуск застосунку виконується за допомогою команди:

uvicorn main:app --host 0.0.0.0 --port $PORT

У середовищі Render також було налаштовано змінні оточення для зберігання конфіденційних параметрів системи, зокрема ключів доступу до зовнішніх сервісів та параметрів конфігурації. Додатковою перевагою використання Render є автоматичне налаштування SSL-сертифіката, що забезпечує передачу даних за захищеним протоколом HTTPS.

Клієнтська частина системи була розміщена на платформі Netlify, яка оптимізована для доставки статичного вебконтенту. Під час розгортання було налаштовано механізм перенаправлення маршрутів для підтримки SPA-архітектури застосунку та забезпечення коректної роботи навігації після оновлення сторінок браузера. У конфігурації фронтенду також було визначено базову адресу серверної частини для надсилання HTTP-запитів до API.

База даних SQLite розташована у файловій системі серверного середовища платформи Render та використовується для зберігання інформації про

користувачів, фільми, відгуки та результати аналізу тональності. Для поточної реалізації цього рішення є достатньо, однак у перспективі доцільним є перехід до використання окремої серверної СУБД, наприклад PostgreSQL, що дозволить підвищити надійність та масштабованість системи.

Архітектуру розгортання програмної системи наведено на рис. 4.1.

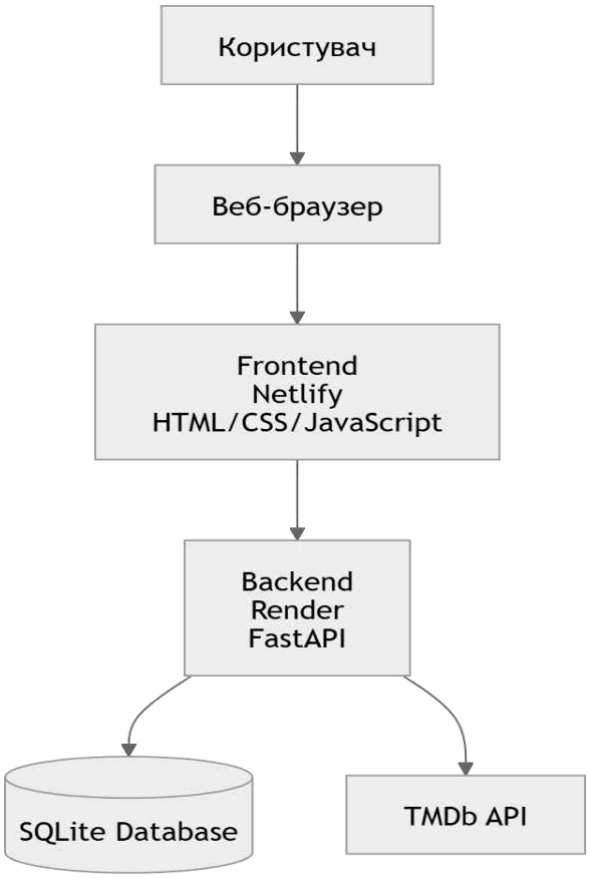

Рис. 4.1 Діаграма розгортання системи

Діаграма демонструє взаємодію між клієнтською частиною, сервером застосунку, базою даних та зовнішнім сервісом TMDb API. Обмін даними між фронтендом та бекендом реалізовано за допомогою REST API, що забезпечує стандартизовану взаємодію між компонентами системи.

У результаті проведеного розгортання система стала доступною через глобальну мережу Інтернет та може використовуватися без встановлення додаткового програмного забезпечення. Користувачу достатньо мати веббраузер та доступ до мережі Інтернет. Додаткові приклади розгортання системи та фрагменти конфігурації наведено у Додатках А та Б.

4.3 Інсталяція, експлуатація та супровід системи

Розроблена програмна система орієнтована на використання у вебсередовищі та не потребує встановлення спеціалізованого програмного забезпечення на стороні кінцевого користувача. Для початку роботи достатньо перейти за вебадресою сервісу через браузер та пройти процедуру авторизації або реєстрації нового облікового запису. Після входу користувач отримує доступ до каталогу фільмів, функцій пошуку, перегляду інформації про фільми та створення власних відгуків.

Основні функціональні можливості системи реалізовані у вигляді інтуїтивного вебінтерфейсу, що забезпечує просту взаємодію користувача із сервісом без необхідності додаткового навчання або спеціальної підготовки. Для створення відгуку користувачу достатньо перейти на сторінку вибраного фільму, вказати оцінку та текст коментаря, після чого система автоматично виконує аналіз тональності повідомлення та формує відповідні аналітичні результати.

Для користувачів із ролями адміністратора або модератора передбачено додаткові функції управління контентом системи. Зокрема, реалізовано можливість додавання нових фільмів, перегляду відгуків користувачів, а також модерації публікацій шляхом зміни їх статусу. Такий підхід забезпечує контроль якості контенту та підтримання коректної роботи інформаційної системи.

Склад основних компонентів програмної системи наведено у табл. 4.3.

Таблиця 4.3

Основні компоненти системи

Компонент	Призначення
Frontend	Користувацький вебінтерфейс
Backend API	Обробка запитів та бізнес-логіки
SQLite Database	Зберігання даних системи
TMDb API	Отримання інформації про фільми
Модуль аналізу тональності	Аналіз текстових відгуків

Під час експлуатації система продемонструвала стабільну роботу та коректне виконання основних функцій. Завдяки використанню хмарного розгортання сервіс доступний через мережу Інтернет цілодобово та не залежить від конкретного користувацького пристрою. Додатковою перевагою є мінімальні вимоги до клієнтської частини, оскільки основна обробка даних виконується на сервері.

Порівняно з існуючими аналогами система характеризується спрощеним інтерфейсом, відсутністю надлишкових елементів та інтеграцією автоматизованого аналізу користувацьких відгуків. Використання зовнішнього сервісу TMDb дозволяє підтримувати актуальність каталогу фільмів без необхідності ручного наповнення бази даних. Крім цього, реалізована архітектура із розділенням клієнтської та серверної частин створює можливість подальшого розширення функціональних можливостей системи без суттєвої зміни загальної структури застосунку.

У системі також реалізовано базові механізми захисту даних, зокрема використання JWT-авторизації, хешування паролів та обмеження доступу до адміністративних функцій. Це дозволяє забезпечити базовий рівень інформаційної безпеки та захисту користувацьких даних.

Отже, розроблена система є придатною до практичного використання, забезпечує стабільну роботу основного функціоналу та може слугувати основою для подальшого розвитку і вдосконалення програмного продукту. Додаткові приклади роботи системи та результати її експлуатації наведено у Додатках А та Б.

ВИСНОВКИ

У кваліфікаційній роботі вирішено актуальне завдання розробки веборієнтованої інформаційної системи аналізу користувацьких відгуків із реалізацією інтерактивного механізму формування рейтингів для кіноглядачів. Проведене дослідження підтвердило доцільність використання сучасних засобів програмної інженерії, технологій клієнт-серверної взаємодії та методів аналізу текстової інформації для автоматизації процесів обробки користувацького контенту у сфері мультимедійних сервісів.

У процесі виконання роботи було проведено аналіз предметної області та сучасних аналогів інформаційних систем, призначених для оцінювання та аналізу кіноконтенту. Виконане дослідження дозволило визначити основні переваги та недоліки існуючих платформ, сформувати функціональні та нефункціональні вимоги до програмного забезпечення, а також обґрунтувати необхідність створення системи, орієнтованої на автоматизований аналіз текстових відгуків користувачів.

На етапі проєктування було розроблено архітектуру програмної системи, що базується на клієнт-серверному підході та використанні REST API. Спроєктовано структуру бази даних, визначено взаємозв’язки між основними сутностями системи та реалізовано механізми збереження і обробки інформації про користувачів, фільми, рейтинги та текстові рецензії. Використання UML-діаграм, ER-моделі та компонентного підходу дозволило забезпечити структурованість, масштабованість і логічну узгодженість програмного рішення.

У межах програмної реалізації розроблено серверну частину застосунку із використанням мови програмування Python та фреймворку FastAPI. Реалізовано механізми обробки HTTP-запитів, авторизації користувачів, роботи з базою даних та інтеграції із зовнішнім сервісом TMDb для отримання актуальної інформації про фільми. Клієнтська частина системи забезпечує зручну взаємодію користувачів із програмним забезпеченням, підтримує перегляд інформації про

фільми, формування рейтингів, створення відгуків та візуалізацію результатів аналізу.

Важливою складовою роботи стала реалізація модуля аналізу текстових відгуків користувачів. Використання методів аналізу текстової інформації дозволило автоматизувати визначення емоційного забарвлення рецензій та забезпечити формування узагальнених рейтингових показників на основі користувацького контенту. Реалізований підхід підвищує інформативність системи та створює передумови для подальшого розвитку рекомендаційних механізмів.

У процесі тестування підтверджено працездатність програмного забезпечення, коректність реалізації основних функціональних модулів та стабільність взаємодії між компонентами системи. Проведене розгортання вебзастосунку у хмарному середовищі забезпечило можливість віддаленого доступу до програмного продукту та підтвердило готовність системи до практичного використання.

Практичне значення одержаних результатів полягає у створенні працездатної інформаційної системи, яка може бути використана для автоматизації аналізу користувацьких відгуків та формування інтерактивних рейтингів у сфері кіноконтенту. Розроблене програмне забезпечення може слугувати основою для подальшого розширення функціональних можливостей, інтеграції інтелектуальних методів аналізу даних та створення повноцінних рекомендаційних сервісів.

Отже, поставлену мету кваліфікаційної роботи досягнуто, а всі визначені завдання виконано у повному обсязі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Sommerville I. Software Engineering. 10th ed. Boston : Pearson, 2021. 811 p.
2. Pressman R. S., Maxim B. R. Software Engineering: A Practitioner’s Approach. 9th ed. New York : McGraw-Hill Education, 2022. 880 p.
3. Fowler M. Patterns of Enterprise Application Architecture. 2nd ed. Boston : Addison-Wesley Professional, 2021. 560 p.
4. Richards M., Ford N. Fundamentals of Software Architecture: An Engineering Approach. Sebastopol : O’Reilly Media, 2020. 432 p.
5. Newman S. Building Microservices: Designing Fine-Grained Systems. 2nd ed. Sebastopol : O’Reilly Media, 2021. 620 p.
6. Ramalho L. Fluent Python. 2nd ed. Sebastopol : O’Reilly Media, 2022. 1014 p.
7. Lubanovic B. Introducing Python: Modern Computing in Simple Packages. 2nd ed. Sebastopol : O’Reilly Media, 2023. 624 p.
8. FastAPI Documentation. URL: https://fastapi.tiangolo.com/.
9. SQLAlchemy Documentation. URL: https://docs.sqlalchemy.org/.
10. SQLite Documentation. URL: https://www.sqlite.org/docs.html.
11. Mozilla Developer Network. REST APIs. URL: https://developer.mozilla.org/en-US/docs/Glossary/REST.
12. Fielding R. T. Architectural Styles and the Design of Network-based Software Architectures : doctoral dissertation. University of California, Irvine, 2021. 162 p.
13. Richardson L., Amundsen M., Ruby S. RESTful Web APIs. Sebastopol : O’Reilly Media, 2021. 408 p.
14. Grinberg M. Flask Web Development: Developing Web Applications with Python. 2nd ed. Sebastopol : O’Reilly Media, 2022. 398 p.
15. TMDb API Documentation. URL: https://developer.themoviedb.org/docs.
16. Jurafsky D., Martin J. H. Speech and Language Processing. 3rd ed. Draft. Stanford University, 2023. URL: https://web.stanford.edu/~jurafsky/slp3/.
17. Bird S., Klein E., Loper E. Natural Language Processing with Python. Sebastopol : O’Reilly Media, 2021. 504 p.

18. Zhang A., Lipton Z. C., Li M., Smola A. J. Dive into Deep Learning. Cambridge : Cambridge University Press, 2023. 842 p.
19. Cambria E., White B. Jumping NLP Curves: A Review of Natural Language Processing Research. IEEE Computational Intelligence Magazine. 2021. Vol. 16, No. 2. P. 48–57.
20. Medhat W., Hassan A., Korashy H. Sentiment Analysis Algorithms and Applications: A Survey. Ain Shams Engineering Journal. 2021. Vol. 12, No. 2. P. 1093–1113.
21. Devlin J., Chang M. W., Lee K., Toutanova K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. Proceedings of NAACL-HLT. 2021. P. 4171–4186.
22. Chollet F. Deep Learning with Python. 2nd ed. Shelter Island : Manning Publications, 2022. 504 p.
23. Freeman E., Robson E. Head First Design Patterns. 2nd ed. Sebastopol : O'Reilly Media, 2021. 694 p.
24. Gamma E., Helm R., Johnson R., Vlissides J. Design Patterns: Elements of Reusable Object-Oriented Software. Updated edition. Boston : Addison-Wesley Professional, 2021. 416 p.
25. Mozilla Developer Network. JavaScript Guide. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide.
26. Chart.js Documentation. URL: https://www.chartjs.org/docs/latest/.
27. PostgreSQL Documentation. URL: https://www.postgresql.org/docs/.
28. OWASP Foundation. OWASP Top 10: Web Application Security Risks. URL: https://owasp.org/www-project-top-ten/.
29. Stallings W. Effective Cybersecurity: A Guide to Using Best Practices and Standards. Boston : Addison-Wesley Professional, 2021. 576 p.
30. Docker Documentation. URL: https://docs.docker.com/.
31. GitHub Docs. URL: https://docs.github.com/.
32. IEEE Software Engineering Standards Committee. Guide to the Software Engineering Body of Knowledge SWEBOK. Version 4.0. IEEE, 2024. 390 p.

33. Биков В. Ю., Спірін О. М., Пінчук О. П. Цифрова трансформація освіти і сучасні інформаційні технології. Київ : Компринт, 2022. 284 с.
34. Грибюк О. О. Аналіз та обробка даних у сучасних інформаційних системах. Київ : Ліра-К, 2021. 312 с.
35. Бондаренко М. Ф., Шабельник Т. В. Проєктування інформаційних систем і баз даних. Харків : ХНУРЕ, 2023. 356 с.
36. Коваленко І. І. Вебтехнології та розробка сучасних вебзастосунків. Львів : Новий Світ-2000, 2022. 298 с.

ДОДАТОК А

Інтерфейс користувача системи

У додатку наведено приклади інтерфейсу користувача розробленої системи аналізу користувацьких відгуків та рейтингів для кіноглядачів. Представлені скріншоти демонструють основні функціональні можливості веб-застосунку, включаючи перегляд фільмів, систему оцінювання, модуль рекомендацій, а також адміністративні функції системи.

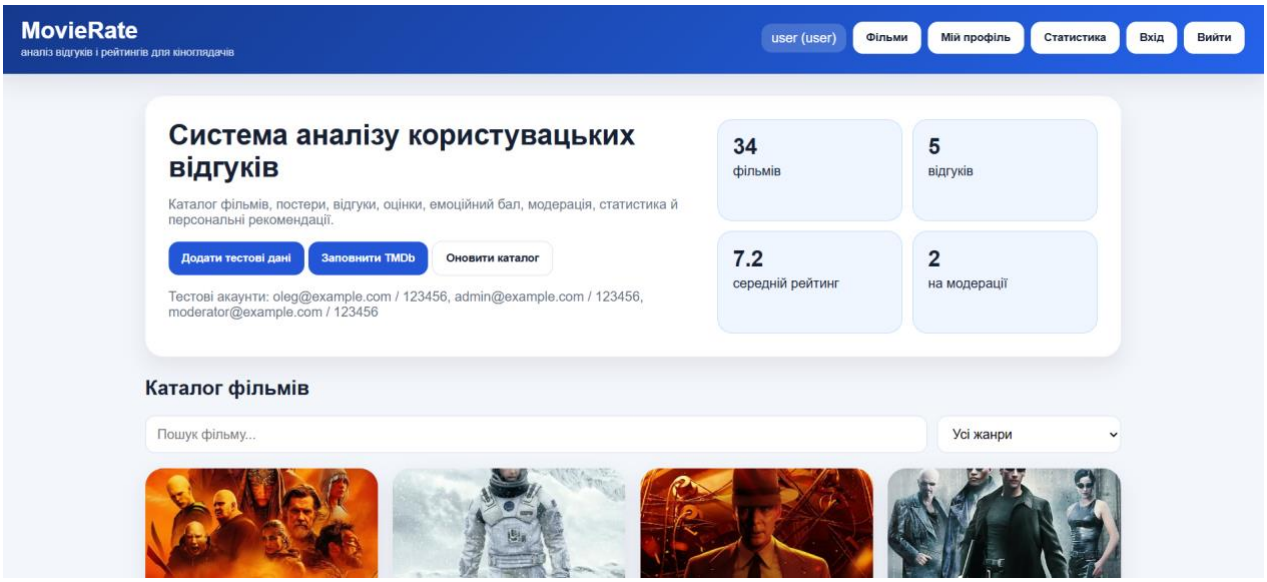

Рис. А.1 – Головна сторінка системи

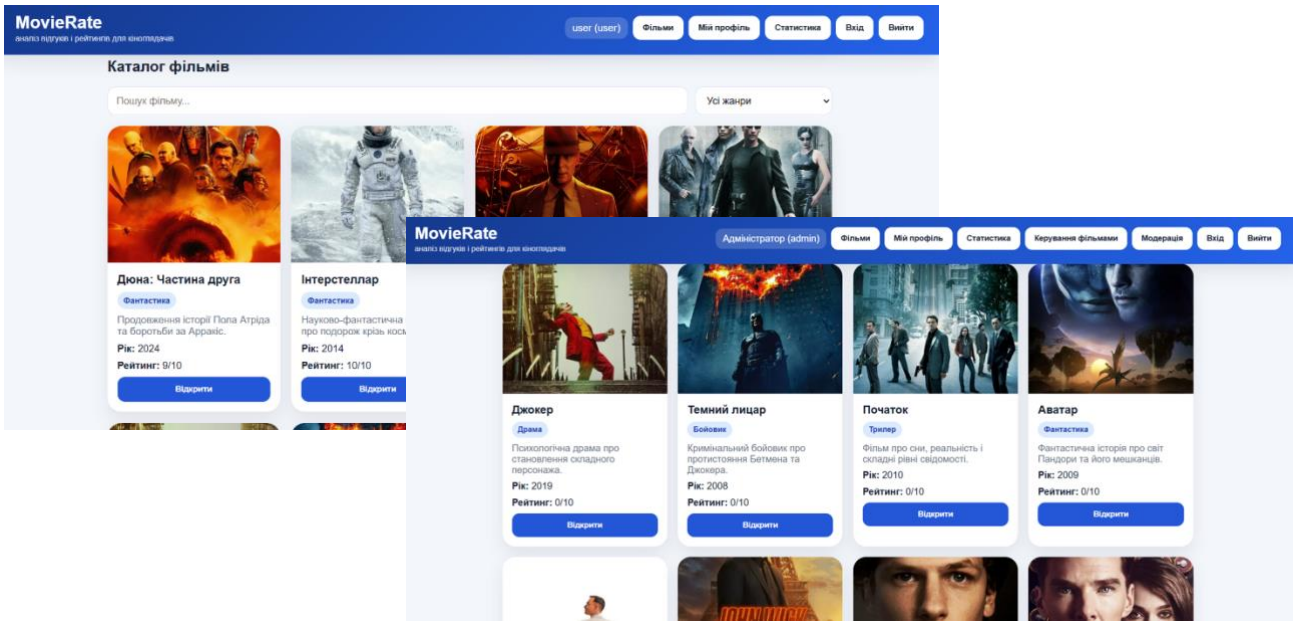

Рис. А.2 – Сторінка перегляду каталогу фільмів

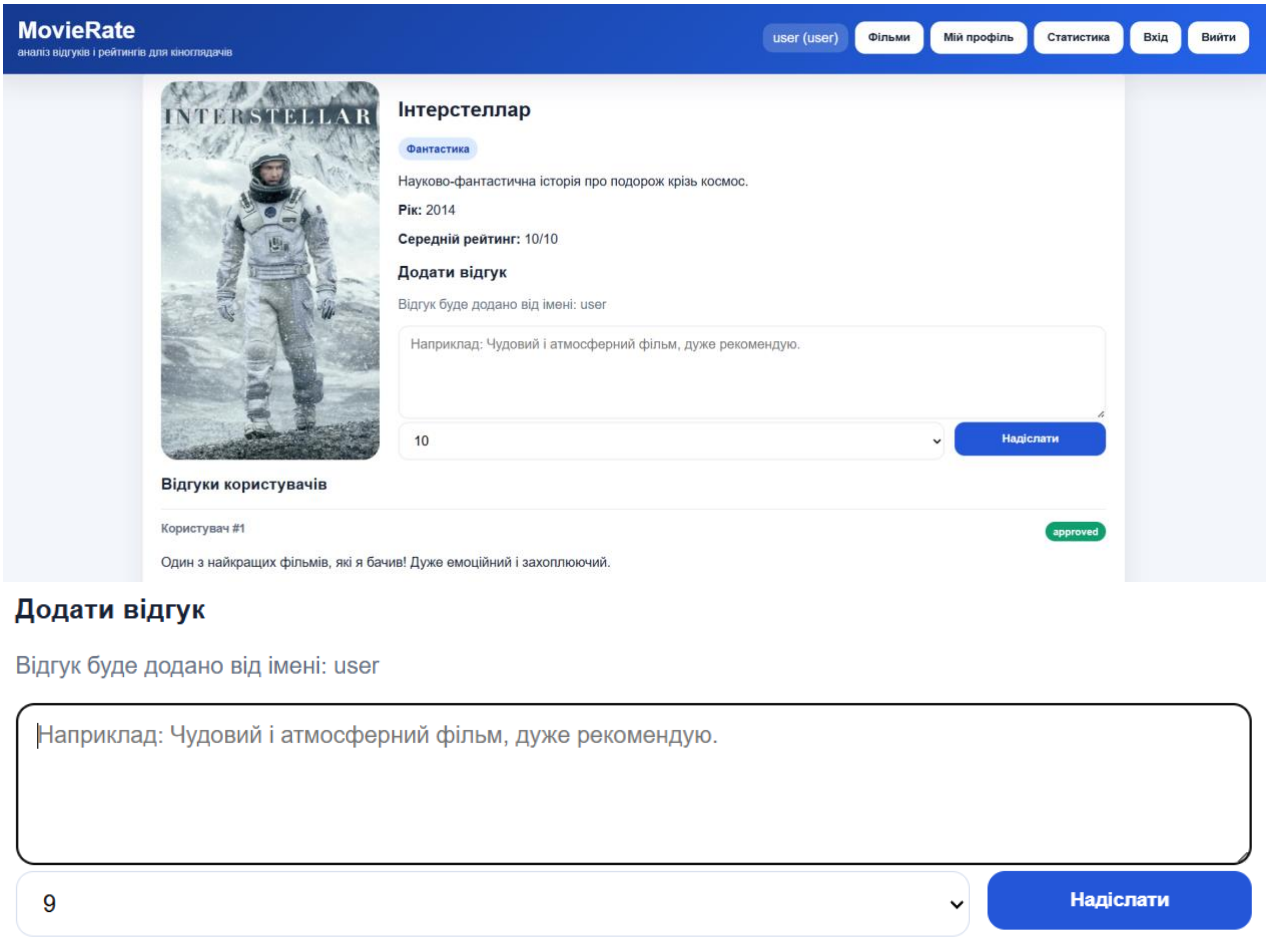

Рис. А.3 – Інтерфейс додавання відгуку та оцінки

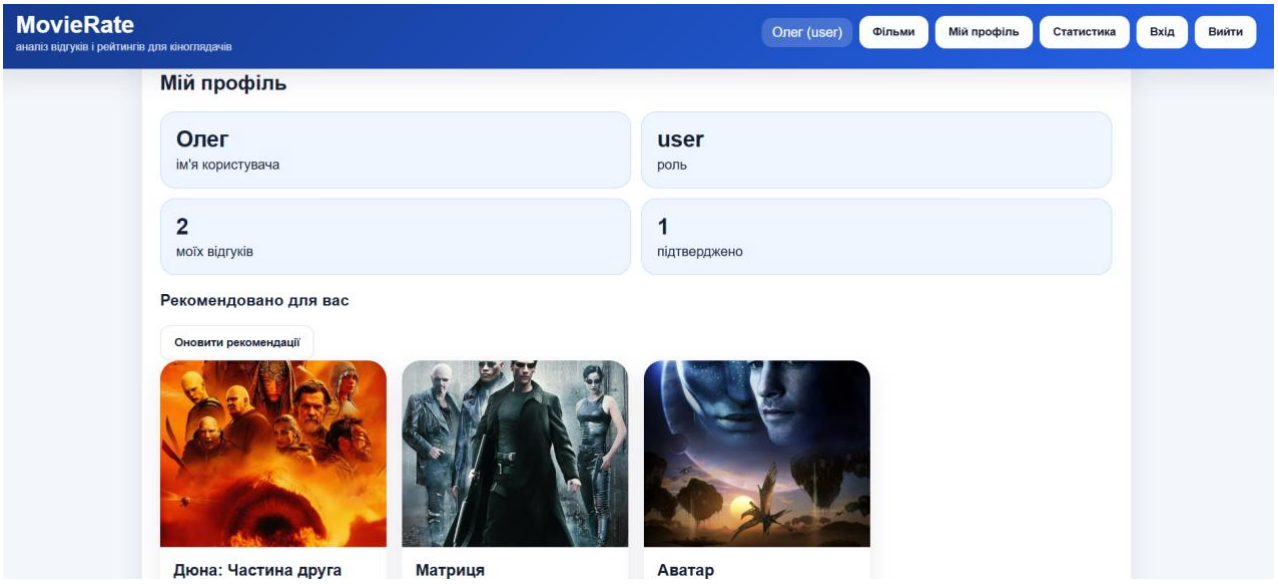

Рис. А.4 – Профіль користувача

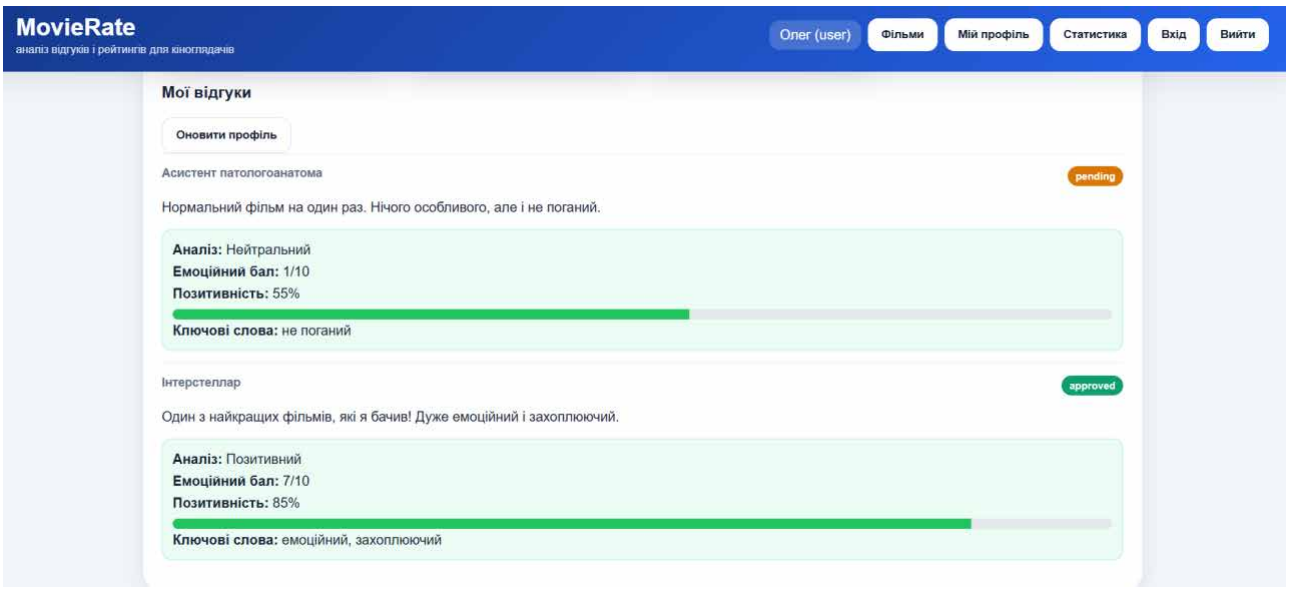

Рис. А.5 – Панель відгуків користувача

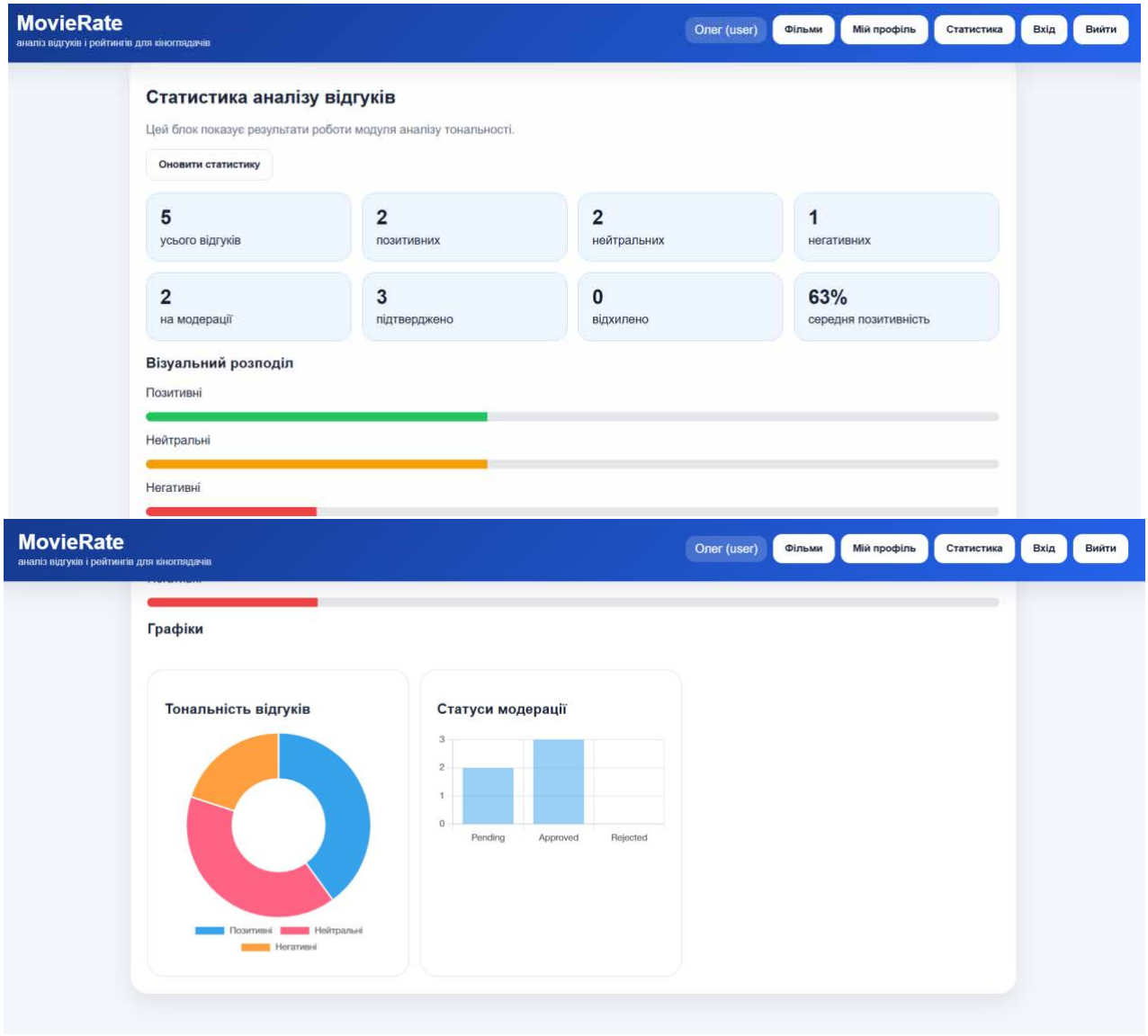

Рис. А.6 – Статистика аналізу відгуків

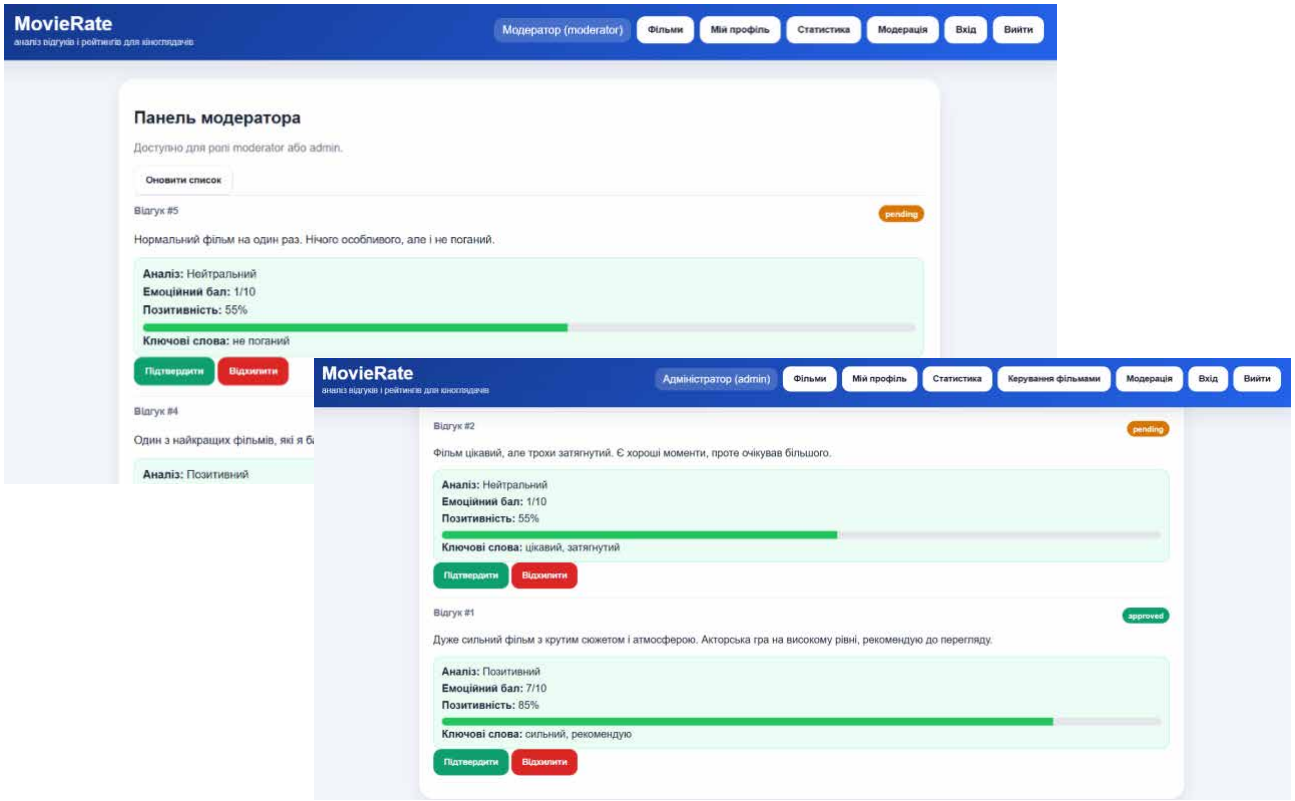

Рис. А.7 – Інтерфейс модерації відгуків

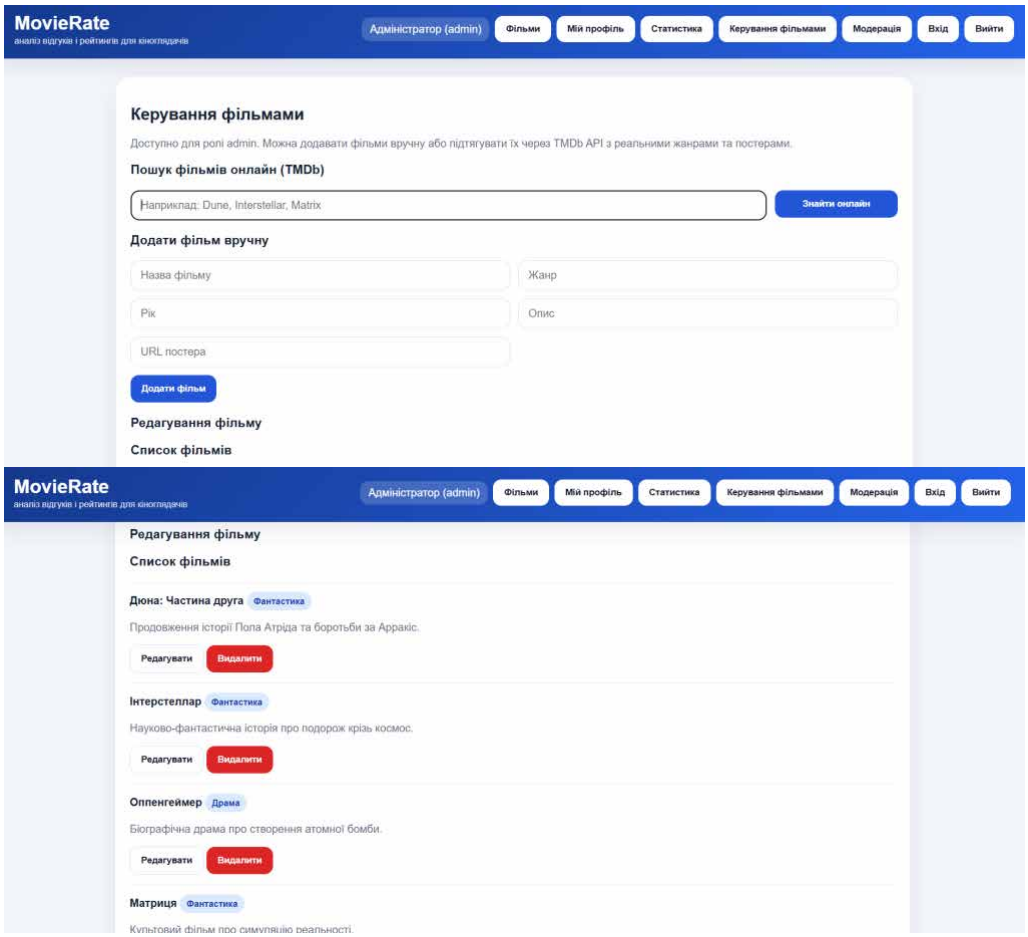

Рис. А.8 – Панель керування фільмами

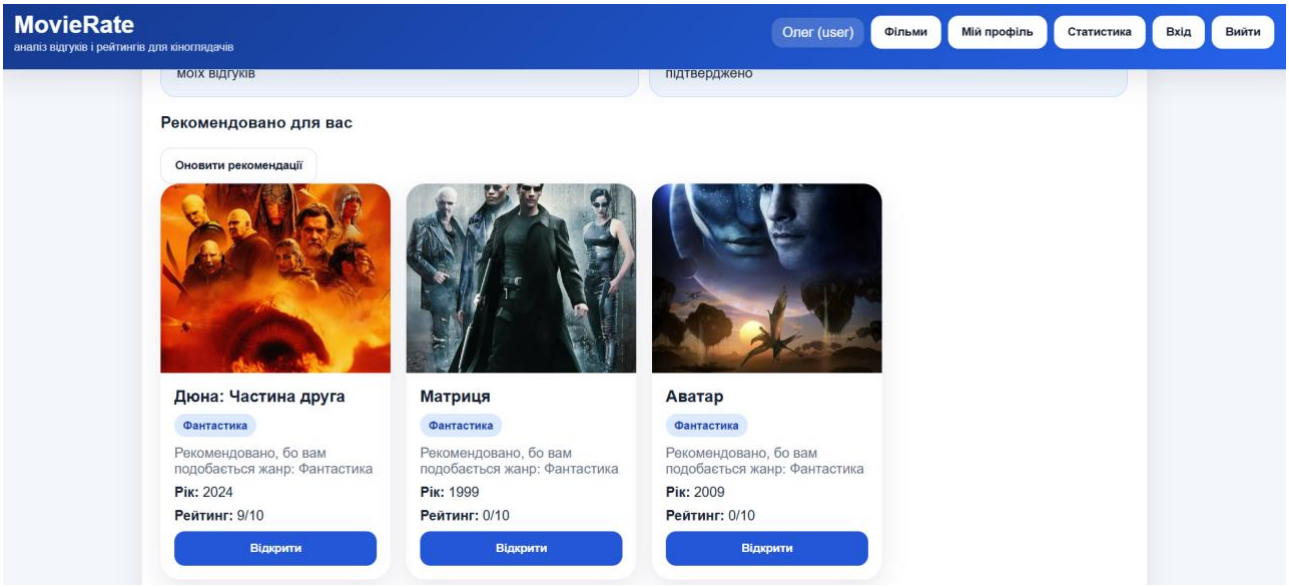

Рис. А.9 – Блок рекомендацій для користувача

ДОДАТОК Б

REST API програмної системи аналізу користувацьких відгуків для кіноглядачів

У даному додатку наведено приклади реалізованого REST API програмної системи, побудованого із використанням фреймворку FastAPI [6].

API забезпечує взаємодію між клієнтською та серверною частинами застосунку, а також дозволяє виконувати операції отримання, створення, редагування та аналізу даних.

Для тестування та документування API використовується автоматично згенерований Swagger-інтерфейс.

Загальний Swagger

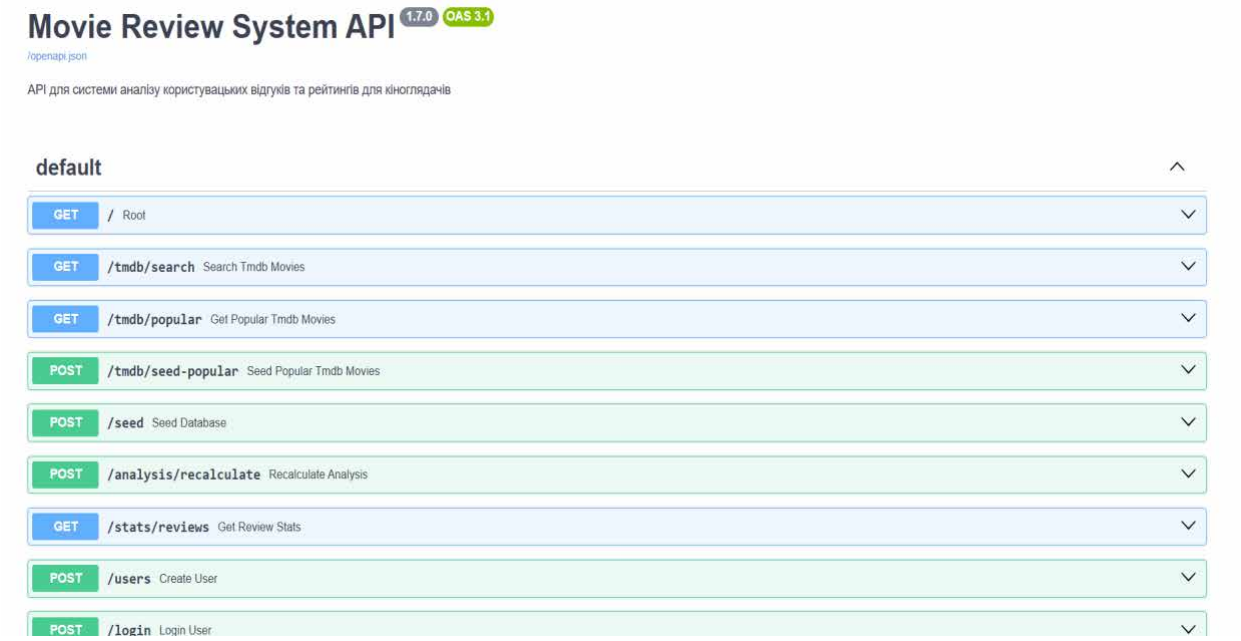

Рис. Б.1 – Swagger-інтерфейс REST API системи

GET /tmdb/popular

Endpoint використовується для отримання популярних фільмів із зовнішнього сервісу TMDb.

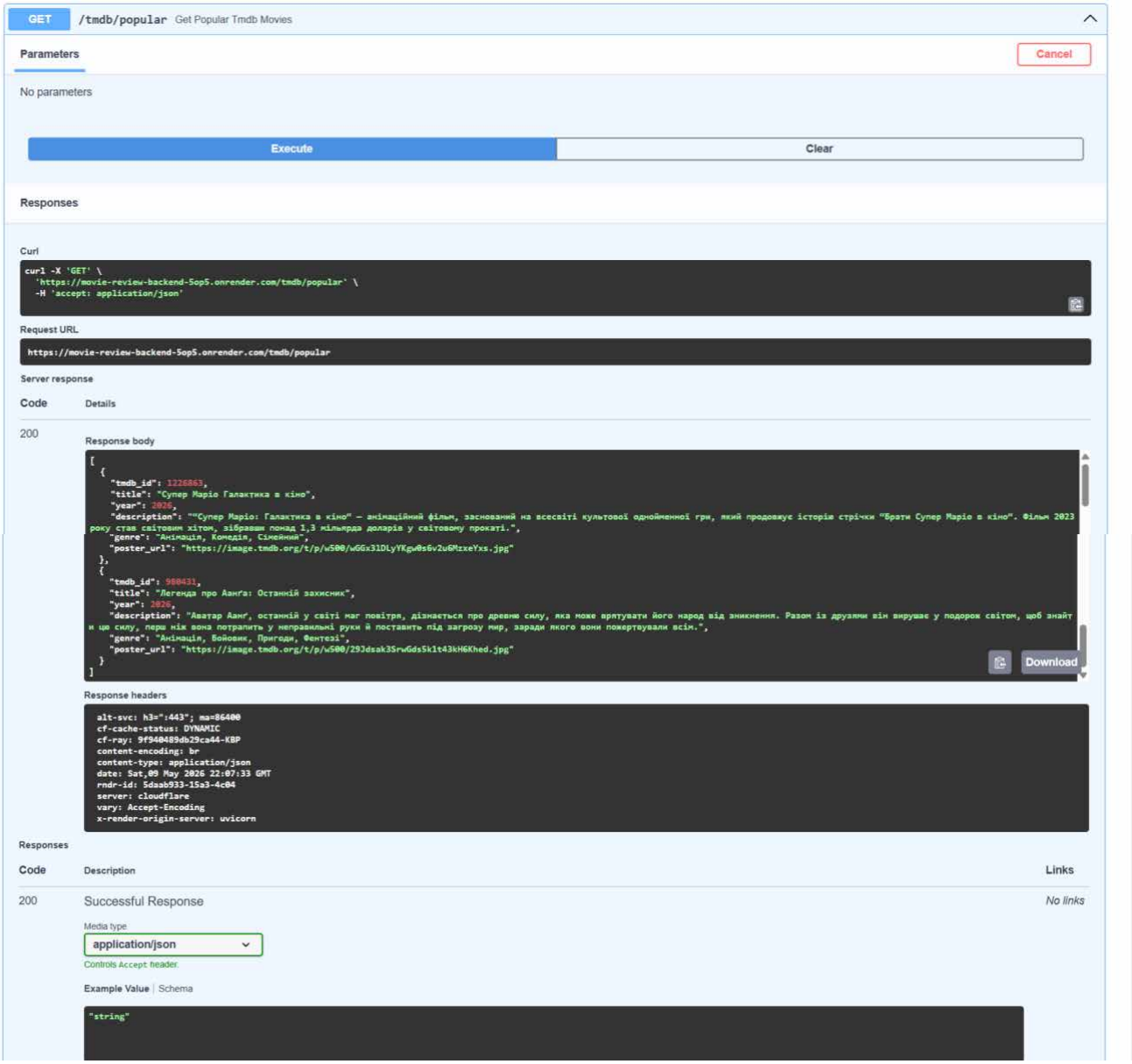

Рис. Б.2 – Отримання списку популярних фільмів

POST /reviews

Даний endpoint забезпечує створення нового користувацького відгуку та передачу даних на серверну частину системи.

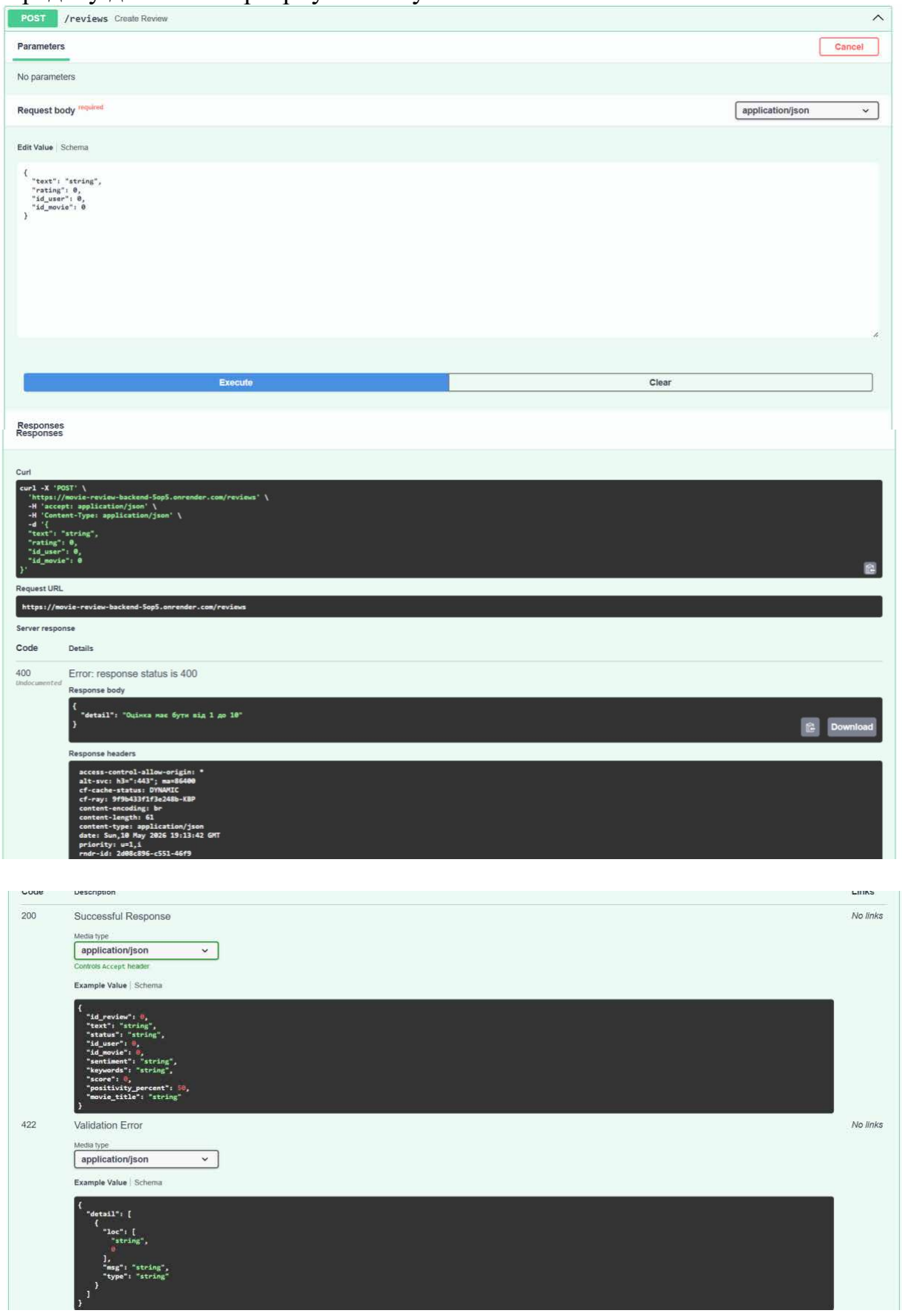

Рис. Б.3 – Створення нового відгуку

GET /stats/reviews

Endpoint використовується для формування статистики відгуків та отримання аналітичної інформації про активність користувачів.

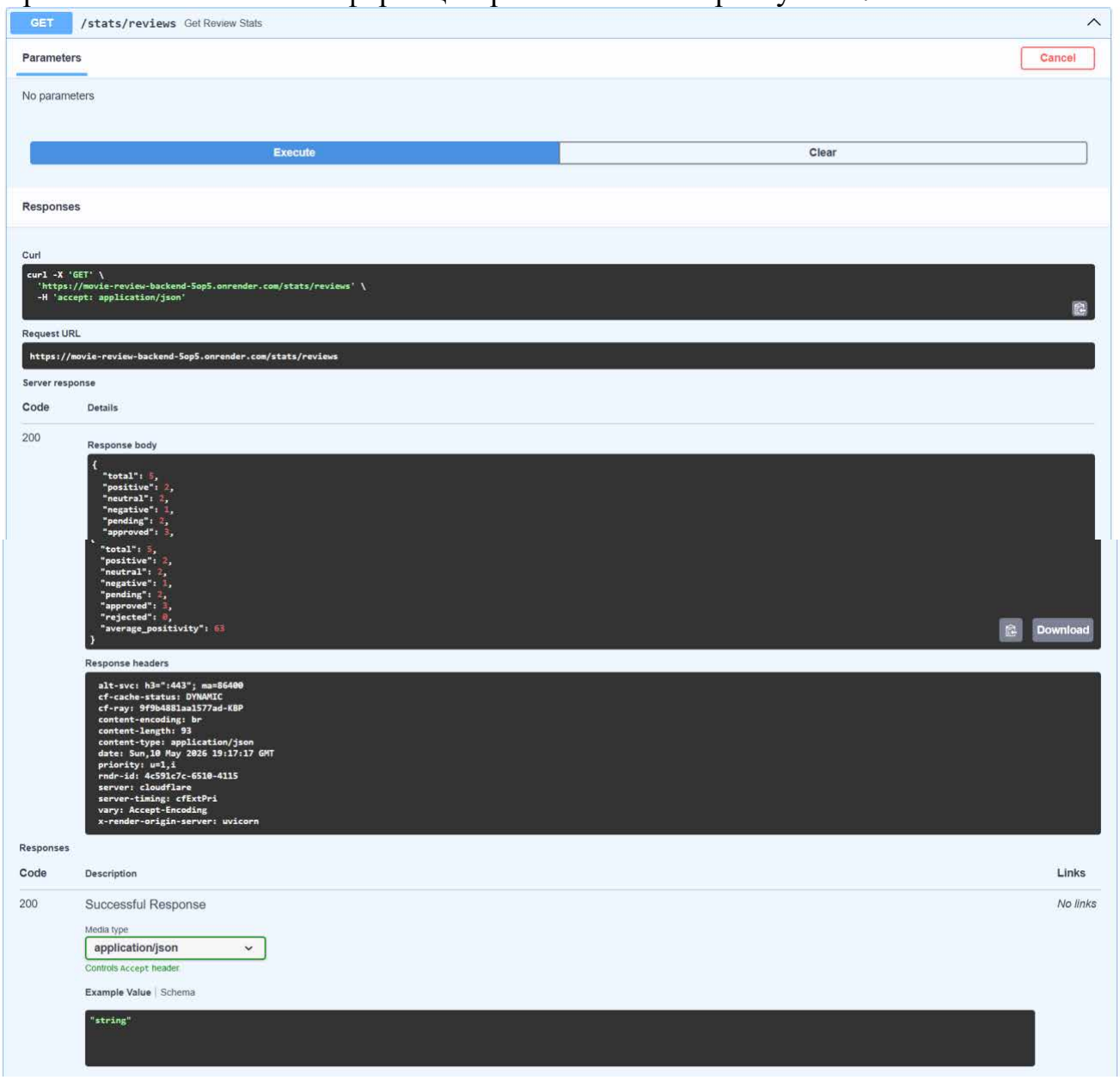

Рис. Б.4 – Отримання статистики відгуків

POST /analysis/recalculate

Endpoint призначений для повторного запуску аналізу текстових відгуків та оновлення результатів аналізу у базі даних.

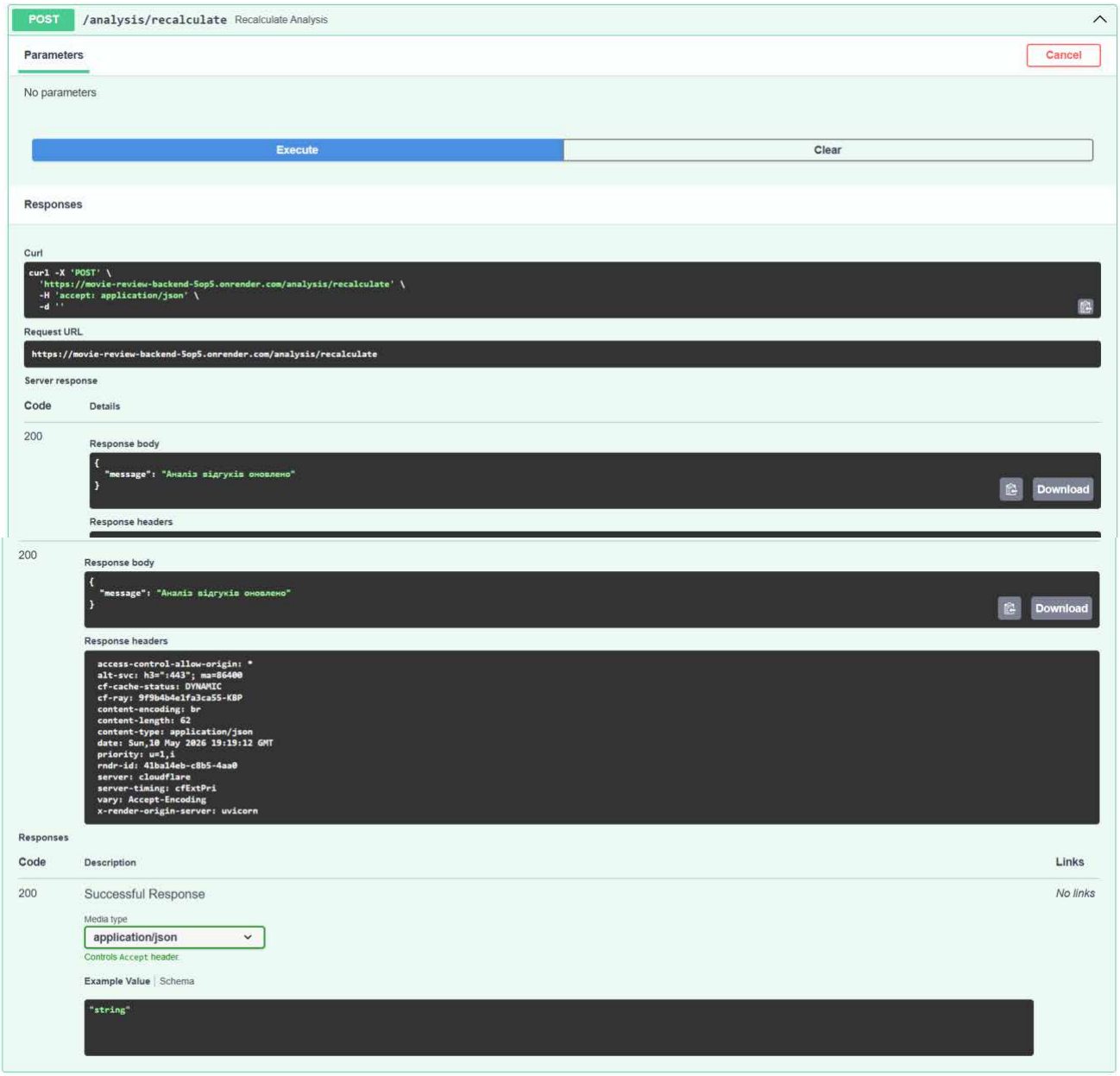

Рис. Б.5 – Повторний запуск аналізу відгуків

ДОДАТОК В

Фрагменти програмного коду системи

У додатку наведено фрагменти програмного коду, які реалізують основні функції системи аналізу користувацьких відгуків та рейтингів для кіноглядачів. Представлені лістинги демонструють структуру серверної частини застосунку, обробку відгуків, інтеграцію із зовнішнім API TMDb, модуль аналізу тексту та систему рекомендацій.

Лістинг В.1 – Ініціалізація серверної частини системи та отримання інформації про фільми через TMDb API

```
from fastapi import FastAPI, Depends, HTTPException, Query
from fastapi.middleware.cors import CORSMiddleware
from sqlalchemy.orm import Session
from sqlalchemy import text
import requests
import os

import models
import schemas
import crud
from database import engine, get_db

models.Base.metadata.create_all(bind=engine)

with engine.begin() as connection:
    connection.execute(text("ALTER TABLE analysis ADD COLUMN IF NOT
EXISTS score INTEGER DEFAULT 0 NOT NULL"))
    connection.execute(text("ALTER TABLE analysis ADD COLUMN IF NOT
EXISTS positivity_percent INTEGER DEFAULT 50 NOT NULL"))
    connection.execute(text("ALTER TABLE movies ADD COLUMN IF NOT
EXISTS poster_url TEXT"))

TMDB_API_KEY = "**************************"

TMDB_GENRES_UK = {
    28: "Бойовик",
    12: "Пригоди",
    16: "Анімація",
    35: "Комедія",
    80: "Кримінал",
    99: "Документальний",
    18: "Драма",
```

```
    10751: "Сімейний",
    14: "Фентезі",
    36: "Історичний",
    27: "Жахи",
    10402: "Музика",
    9648: "Детектив",
    10749: "Романтика",
    878: "Фантастика",
    10770: "Телефільм",
    53: "Трилер",
    10752: "Військовий",
    37: "Вестерн"
}

def genre_names_from_ids(genre_ids):
    names = [TMDB_GENRES_UK.get(gid) for gid in genre_ids or [] if
TMDB_GENRES_UK.get(gid)]
    return ", ".join(names) if names else "Фільм"

app = FastAPI(
    title="Movie Review System API",
    description="API для системи аналізу користувацьких відгуків та рейтингів
для кіноглядачів",
    version="1.7.0"
)

app.add_middleware(
    CORSMiddleware,
    allow_origins=["*"],
    allow_credentials=False,
    allow_methods=["*"],
    allow_headers=["*"],
)

@app.get("/")
def root():
    return {"message": "Movie Review System API працює"}

@app.get("/tmdb/search")
def search_tmdb_movies(query: str = Query(..., min_length=1)):
    url = "https://api.themoviedb.org/3/search/movie"
    params = {
        "api_key": TMDB_API_KEY,
        "query": query,
        "language": "uk-UA",
```

```
        "include_adult": "false"
    }

    response = requests.get(url, params=params, timeout=10)
    data = response.json()

    results = []
    for movie in data.get("results", [])[:10]:
        release_date = movie.get("release_date") or ""
        poster_path = movie.get("poster_path")

        results.append({
            "tmdb_id": movie.get("id"),
            "title": movie.get("title") or movie.get("original_title") or "Без назви",
            "year": int(release_date[:4]) if release_date[:4].isdigit() else None,
            "description": movie.get("overview") or "Опис відсутній.",
            "genre": genre_names_from_ids(movie.get("genre_ids")),
            "poster_url": f"https://image.tmdb.org/t/p/w500{poster_path}" if poster_path
else None
        })

    return results
```

Лістинг В.2 – Функція аналізу емоційного забарвлення відгуку

```
def analyze_sentiment(text: str, rating: int | None = None) -> tuple[str, str, int, int]:
    lower_text = text.lower()

    score = 0
    found_keywords = []

    # Обробка фраз із запереченням
    neutral_positive_phrases = {
        "не поганий": 1,
        "непоганий": 1,
        "не нудний": 1,
        "не слабкий": 1,
        "не жахливий": 1,
        "не розчарував": 1
    }

    for phrase, value in neutral_positive_phrases.items():
        if phrase in lower_text:
            score += value
            found_keywords.append(phrase)
            lower_text = lower_text.replace(phrase, "")
```

```
    for word, value in POSITIVE_WORDS.items():
        if word in lower_text:
            score += value
            found_keywords.append(word)

    for word, value in NEGATIVE_WORDS.items():
        if word in lower_text:
            score += value
            found_keywords.append(word)

    # Додатково враховуємо числову оцінку користувача
    if rating is not None:
        if rating >= 9:
            score += 2
        elif rating >= 7:
            score += 1
        elif rating <= 3:
            score -= 2
        elif rating <= 5:
            score -= 1

    words = re.findall(r"[а-яА-ЯіІїЇєЄґҐa-zA-Z]+", lower_text)
    normalized_score = max(-10, min(10, score))
    positivity_percent = int(round(((normalized_score + 10) / 20) * 100))

    if normalized_score >= 3:
        sentiment = "Позитивний"
    elif normalized_score <= -3:
        sentiment = "Негативний"
    else:
        sentiment = "Нейтральний"

    if not found_keywords:
        candidate_words = sorted(set(words), key=len, reverse=True)
        found_keywords = candidate_words[:3]

    return sentiment, ", ".join(found_keywords), normalized_score, positivity_percent
```

Лістинг В.3 – Створення користувацького відгуку

```
def create_review_with_rating(db: Session, review: schemas.ReviewCreate):
    db_review = models.Review(text=review.text, id_user=review.id_user,
id_movie=review.id_movie, status="pending")
    db.add(db_review)
    db.commit()
```

```
    db.refresh(db_review)

    db_rating = models.Rating(rating=review.rating, id_user=review.id_user,
id_movie=review.id_movie)
    db.add(db_rating)

    sentiment, keywords, score, positivity_percent = analyze_sentiment(review.text,
review.rating)
    db_analysis = models.Analysis(sentiment=sentiment, keywords=keywords,
score=score, positivity_percent=positivity_percent, id_review=db_review.id_review)
    db.add(db_analysis)

    db.commit()
    db.refresh(db_review)

    db_review.sentiment = sentiment
    db_review.keywords = keywords
    db_review.score = score
    db_review.positivity_percent = positivity_percent
    return db_review

def get_reviews_by_movie(db: Session, movie_id: int):
    reviews = db.query(models.Review).filter(models.Review.id_movie ==
movie_id).order_by(models.Review.id_review.desc()).all()
    for review in reviews:
        fill_review_analysis(review)
    return reviews

def get_reviews_by_user(db: Session, user_id: int):
    reviews = db.query(models.Review).filter(models.Review.id_user ==
user_id).order_by(models.Review.id_review.desc()).all()
    result = []
    for review in reviews:
        fill_review_analysis(review)
        review.movie_title = review.movie.title if review.movie else "Невідомий
фільм"
        result.append(review)
    return result
```

Лістинг В.4 – Формування рекомендацій для користувача

```
def get_recommendations_for_user(db: Session, user_id: int):
    user_ratings = db.query(models.Rating).filter(models.Rating.id_user ==
user_id).all()
    watched_movie_ids = {rating.id_movie for rating in user_ratings}
    liked_genres = {}
```

```
    for rating in user_ratings:
        if rating.rating >= 7 and rating.movie:
            liked_genres[rating.movie.genre] = liked_genres.get(rating.movie.genre, 0) +
rating.rating

    user_reviews = db.query(models.Review).filter(models.Review.id_user ==
user_id).all()
    for review in user_reviews:
        if review.analysis and review.analysis.sentiment == "Позитивний" and
review.movie:
            liked_genres[review.movie.genre] = liked_genres.get(review.movie.genre, 0)
+ 3

    if liked_genres:
        sorted_genres = sorted(liked_genres.items(), key=lambda item: item[1],
reverse=True)
        preferred_genres = [genre for genre, _ in sorted_genres]
        movies =
db.query(models.Movie).filter(~models.Movie.id_movie.in_(watched_movie_ids),
models.Movie.genre.in_(preferred_genres)).all()
        recommendations = []
        for movie in movies:
            fill_movie_rating(db, movie)
            movie.recommendation_reason = f"Рекомендовано, бо вам подобається
жанр: {movie.genre}"
            recommendations.append(movie)
        recommendations.sort(key=lambda movie:
(preferred_genres.index(movie.genre), -movie.average_rating))
        return recommendations[:8]

    movies = db.query(models.Movie).all()
    recommendations = []
    for movie in movies:
        if movie.id_movie not in watched_movie_ids:
            fill_movie_rating(db, movie)
            movie.recommendation_reason = "Популярний фільм з каталогу"
            recommendations.append(movie)
    recommendations.sort(key=lambda movie: movie.average_rating, reverse=True)
    return recommendations[:8]
```

ДОДАТОК Д

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ
Факультет інформаційних технологій

Декларація про академічну доброчесність та використання ШІ

Я, Савош Олег Ігорович, здобувач НУБіП України, усвідомлюючи відповідальність за порушення академічної доброчесності, цією заявою підтверджую, що:

1. Моя бакалаврська кваліфікаційна робота на тему «Програмне забезпечення системи аналізу користувацьких відгуків на основі рейтингу кіноглядачів» є результатом моєї особистої праці.
2. Усі запозичення з текстів інших авторів мають відповідні посилання згідно з чинними стандартами.
3. Щодо використання інструментів ШІ зазначаю, що (обрати необхідне):

- [ ] інструменти генеративного ШІ не використовувалися.
- [ ] інструменти ШІ (вказати назву: **ChatGPT**, **Gemini**, Claude, DeepL, **DeepSeek** інші (вказати назву)) були використані для:
  - [ ] перекладу іншомовних джерел;
  - [x] стилістичного редагування та перевірки граматики;
  - [x] допомоги у написанні/відлагодженні програмного коду;
  - [x] інше: отримання консультацій щодо використання окремих технологій та бібліотек; генерації прикладів UML-діаграм; пошуку можливих варіантів реалізації окремих функцій системи.

Я розумію, що надання недостовірної інформації може призвести до скасування результатів захисту та відрахування з числа здобувачів НУБіП України.

« » 2026 р. _____ **Олег САВОШ**
(підпис)