

**БАКАЛАВРСЬКА РОБОТА**

**РБ. ІП-11.00.00.000 ІЗ**

**Група ІП-22-1**

**Котурлаш Світлана**

**2026**

**Івано-Франківський національний технічний університет нафти і газу**

**Факкультет інформаційних технологій**

**Кафедра інженерії програмного забезпечення**

**Котурлаш Світлана Василівна**

---

(прізвище, ім'я, по батькові)

УДК 004.942

(індекс)

## **БАКАЛАВРСЬКА РОБОТА**

**Розроблення веб-системи рекомендацій вибору фільмів засобами Ruby on**

**Rails/Postgres**

(назва роботи)

Інженерія програмного забезпечення

---

(назва освітньої програми)

121– Інженерія програмного забезпечення

---

(шифр і назва спеціальності)

**Робота містить результати власних досліджень, використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело:**

Здобувач освітнього ступеня Котурлаш С.В.  
(підпис, ініціали та прізвище здобувача)

Науковий керівник Процюк Василь Романович, к.т.н., доцент  
(підпис, прізвище, ім'я, по батькові, науковий ступінь, вчене звання керівника)

**Допущено до захисту**  
Завідувач кафедри

доц. Бандура В.В.  
(посада) (підпис) (дата) (ініціали та прізвище)

**Івано-Франківськ – 2026**

Івано-Франківський національний технічний університет нафти і газу  
Інститут, факультет інформаційних технологій  
Кафедра інженерії програмного забезпечення  
Ступінь вищої освіти бакалавр  
Спеціальність 121 – Інженерія програмного забезпечення

**ЗАТВЕРДЖУЮ:**

Зав. кафедрою ІІЗ  
доцент. В.В. Бандура

“      ”                      2026 р.

## **ЗАВДАННЯ**

### **НА ДИПЛОМНУ РОБОТУ БАКАЛАВРА СТУДЕНТЦІ**

***Котурлаш Світлані Василівні***

(прізвище, ім'я, по-батькові)

**1. Тема проекту (роботи)** **“Розроблення веб-системи рекомендацій вибору фільмів засобами Ruby on Rails/Postgres”**

керівник проекту (роботи) Процюк Василь Романович , к.т.н., доцент

затверджені наказом вищого навчального закладу від “ 21 ” квітня 2026 р. № 179/7

**2. Строк подання студентом проекту (роботи)** 10 червня 2026р.

**3. Вихідні дані до проекту (роботи)** Результати і матеріали отримані під час проходження переддипломної практики

**4. Зміст розрахунково - пояснювальної записки(перелік питань, які потрібно розробити)**

1 Вступ до проблеми

2 Огляд існуючих концепцій , рішень та сервісів в даній області

3 Побудова моделі або алгоритму власного рішення

4 Документування реалізації власного оригінального рішення вибраними засобами

**5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)**

1 Діаграма варіантів використання системи MovieMatch (рис. 2.1 ст. 23)

2 ER-модель бази даних MovieMatch (рис. 3.2, ст. 26)

3 Приклад заповнення таблиці «Основні компоненти програмної реалізації» (рис. 4.1, ст. 30)

4 Адміністративна панель MovieMatch (рис. 4.4 ст. 33)

5 Результат розгортання MovieMatch на сервері (рис. 5.1, ст. 37)

**1. Консультанти розділів проекту (роботи)**

| <i>Розділ</i> | <i>Консультант</i> | <i>Підпис, дата</i>   |                          |
|---------------|--------------------|-----------------------|--------------------------|
|               |                    | <i>Завдання видав</i> | <i>Завдання прийняла</i> |
|               |                    |                       |                          |
|               |                    |                       |                          |

**2. Дата видачі завдання** 2026 р.

Керівник

Процюк В.Р.

(підпис)

Завдання прийняла до виконання

Котурлаш С.В.

(підпис)

***КАЛЕНДАРНИЙ ПЛАН***

| <i>№ з/п</i> | <i>Назва етапів дипломного проекту (роботи)</i>                                                             | <i>Строк виконання етапів проекту</i> | <i>Примітка</i> |
|--------------|-------------------------------------------------------------------------------------------------------------|---------------------------------------|-----------------|
| 1            | Визначення та обґрунтування теми роботи                                                                     | 27.02.2026                            | виконано        |
| 2            | Огляд існуючих концепцій, рішень та сервісів в даній області та формування вимог до веб-системи             | 11.03.2026                            | виконано        |
| 3            | Проектування архітектури, бази даних і користувацьких сценаріїв                                             | 19.04.2026                            | виконано        |
| 4            | Реалізація функцій каталогу, профілю, рейтингу, коментарів і підбірок, розгортання застосунку та тестування | 02.05.2026                            | виконано        |
| 5            | Документування реалізації власного оригінального рішення вибраними засобами                                 | 25.05.2026                            | виконано        |
| 6            | Оформлення пояснювальної записки бакалаврської роботи                                                       | 09.06.2026                            | виконано        |

Студентка – дипломник

Котурлаш С.В.

(підпис)

(розшифровка підпису)

Керівник роботи

Процюк В.Р.

(підпис)

(розшифровка підпису)

## АНОТАЦІЯ

Бакалаврська робота містить 65 сторінок, 31 рисунок, 5 таблиць, список використаних джерел із 43 найменуваннями та додатки.

**Метою** бакалаврської роботи є розроблення веб-системи MovieMatch для рекомендаційного вибору фільмів із використанням Ruby on Rails та PostgreSQL

**Об'єктом дослідження** є процес вибору фільмів і серіалів користувачем у веб-середовищі.

**Предметом дослідження** є програмні засоби розроблення веб-системи рекомендаційного вибору фільмів із використанням Ruby on Rails та PostgreSQL.

**У першому розділі** розглянуто теоретичні основи рекомендаційних веб-систем та особливості їх застосування для вибору фільмів і серіалів.

**У другому розділі** визначено цільову аудиторію системи, основні користувацькі сценарії, функціональні та нефункціональні вимоги.

**У третьому розділі** спроектовано архітектуру веб-застосунку MovieMatch. Обґрунтовано використання Ruby on Rails і шаблону MVC.

**У четвертому розділі** описано реалізацію проєкту MovieMatch. Розглянуто структуру програмного коду, основні контролери, моделі, представлення та маршрути.

**У п'ятому розділі** протестовано та описано розгортання системи.

**Висновки** роботи містять основні результати розроблення, тестування та впровадження веб-системи MovieMatch. У роботі підтверджено, що Ruby on Rails і PostgreSQL є придатними засобами для створення веб-системи.

**Отримані результати** включають розроблений веб-застосунок MovieMatch, спроектовану базу даних PostgreSQL.

**КЛЮЧОВІ СЛОВА:** ВЕБ-СИСТЕМА, MOVIEMATCH, RUBY ON RAILS, POSTGRESQL, РЕКОМЕНДАЦІЇ ФІЛЬМІВ, КАТАЛОГ ФІЛЬМІВ, КОРИСТУВАЦЬКІ ОЦІНКИ, ПЕРСОНАЛЬНІ СПИСКИ, MVC.

## ANNOTATION

The bachelor's thesis contains 65 pages, 31 figures, 5 tables, a list of 43 references and appendices.

**The purpose** of the bachelor's thesis is to develop the MovieMatch web system for recommendation-oriented movie selection using Ruby on Rails and PostgreSQL.

**The object** of the research is the process of selecting movies and TV series by a user in a web environment.

**The subject** of the research is software tools for developing a web system for recommendation-oriented movie selection using Ruby on Rails and PostgreSQL.

**The first chapter** considers the theoretical foundations of recommendation web systems and the features of their application for selecting movies and TV series.

**The second chapter** defines the target audience of the system, the main user scenarios, functional and non-functional requirements.

**The third chapter** designs the architecture of the MovieMatch web application. The use of Ruby on Rails and the MVC pattern is substantiated.

**The fourth chapter** describes the implementation of the MovieMatch project. The structure of the program code, the main controllers, models, views and routes are considered.

**The fifth chapter** presents the testing of the system and describes its deployment.

**The conclusions** contain the main results of the development, testing and implementation of the MovieMatch web system. The thesis confirms that Ruby on Rails and PostgreSQL are suitable tools for creating this type of web system.

**The obtained results** include the developed MovieMatch web application and the designed PostgreSQL database.

**KEYWORDS:** WEB SYSTEM, MOVIEMATCH, RUBY ON RAILS, POSTGRESQL, MOVIE RECOMMENDATIONS, MOVIE CATALOG, USER RATINGS, PERSONAL LISTS, MVC.

## ЗМІСТ

|                                                                                              |           |
|----------------------------------------------------------------------------------------------|-----------|
| <b>ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ .....</b>                                                       | <b>9</b>  |
| <b>ВСТУП .....</b>                                                                           | <b>10</b> |
| <b>РОЗДІЛ 1. ОГЛЯД ІСНУЮЧИХ РІШЕНЬ ТА ТЕОРЕТИЧНИХ ОСНОВ РЕКОМЕНДАЦІЙНИХ ВЕБ-СИСТЕМ .....</b> | <b>12</b> |
| 1.1. Поняття веб-системи рекомендацій вибору фільмів .....                                   | 12        |
| 1.2. Аналіз аналогів і конкурентних підходів .....                                           | 15        |
| 1.3. Технологічні тенденції у розробленні каталогів медіаконтенту.....                       | 19        |
| 1.4. Висновки по розділу .....                                                               | 21        |
| <b>РОЗДІЛ 2. АНАЛІЗ ВИМОГ ТА ПОСТАНОВКА ЗАДАЧІ .....</b>                                     | <b>22</b> |
| 2.1. Цільова аудиторія та користувацькі сценарії .....                                       | 22        |
| 2.2. Функціональні та нефункціональні вимоги .....                                           | 24        |
| 2.3. Постановка задачі проєктування .....                                                    | 29        |
| 2.4. Висновки по розділу .....                                                               | 33        |
| <b>РОЗДІЛ 3. ПРОЄКТУВАННЯ СИСТЕМИ РЕКОМЕНДАЦІЙ ВИБОРУ ФІЛЬМІВ MOVIEMATCH.....</b>            | <b>34</b> |
| 3.1. Архітектура застосунку MovieMatch .....                                                 | 34        |
| 3.2. Модель даних PostgreSQL .....                                                           | 36        |
| 3.3. Проєктування інтерфейсу та користувацької взаємодії .....                               | 38        |
| 3.4. Висновки по розділу .....                                                               | 40        |
| <b>РОЗДІЛ 4. РЕАЛІЗАЦІЯ ПРОЄКТУ РЕКОМЕНДАЦІЙ ВИБОРУ ФІЛЬМІВ MOVIEMATCH.....</b>              | <b>41</b> |
| 4.1. Структура програмного коду .....                                                        | 41        |
| 4.2. Реалізація основних функцій .....                                                       | 43        |
| 4.3. Алгоритми пошуку, фільтрації та персональних списків .....                              | 45        |

|           |      |               |        |      |                                                                                     |                 |      |         |
|-----------|------|---------------|--------|------|-------------------------------------------------------------------------------------|-----------------|------|---------|
|           |      |               |        |      | РБ. ІП – 11.00.00.000 ІЗ                                                            |                 |      |         |
| Змн.      | Арк. | № докум.      | Підпис | Дата |                                                                                     |                 |      |         |
| Розроб.   |      | Котурлаш С.В. |        |      | Розроблення веб-системи рекомендацій вибору фільмів засобами Ruby on Rails/Postgres | Лім.            | Арк. | Акрушів |
| Перевір.  |      | Процюк В.Р.   |        |      |                                                                                     |                 |      | 71      |
| Реценз.   |      | Юрчишин В.М.  |        |      |                                                                                     | ІФНТУНГ ІП-22-1 |      |         |
| Н. Контр. |      | Піх М.М.      |        |      |                                                                                     |                 |      |         |
| Затверд.  |      | Бандура В.В.  |        |      |                                                                                     |                 |      |         |
|           |      |               |        |      | Пояснювальна записка                                                                |                 |      |         |

|                                |    |
|--------------------------------|----|
| 4.4. Висновки по розділу ..... | 47 |
|--------------------------------|----|

**РОЗДІЛ 5. ТЕСТУВАННЯ ТА РОЗГОРТАННЯ ПРОЄКТУ  
РЕКОМЕНДАЦІЙ ВИБОРУ ФІЛЬМІВ MOVIEWATCH..... 48**

|                                 |    |
|---------------------------------|----|
| 5.1. Стратегія тестування ..... | 48 |
|---------------------------------|----|

|                                         |    |
|-----------------------------------------|----|
| 5.2. Результати перевірки функцій ..... | 49 |
|-----------------------------------------|----|

|                                                            |    |
|------------------------------------------------------------|----|
| 5.3. Розгортання на сервері та оцінювання готовності ..... | 51 |
|------------------------------------------------------------|----|

|                                 |    |
|---------------------------------|----|
| 5.4. Висновки по розділу ... .. | 52 |
|---------------------------------|----|

|                       |           |
|-----------------------|-----------|
| <b>ВИСНОВКИ .....</b> | <b>53</b> |
|-----------------------|-----------|

|                                         |           |
|-----------------------------------------|-----------|
| <b>СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ .....</b> | <b>54</b> |
|-----------------------------------------|-----------|

**ДОДАТКИ**

**БІБЛІОГРАФІЧНА ДОВІДКА**

|      |      |          |        |      |                         |      |
|------|------|----------|--------|------|-------------------------|------|
|      |      |          |        |      | РБ.ІП - 11.00.00.000 ПЗ | Арк. |
|      |      |          |        |      |                         | 7    |
| Змн. | Арк. | № докум. | Підпис | Дата |                         |      |

## ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

**API-** Application Programming Interface, програмний інтерфейс взаємодії між системами.

**CRUD-** Create, Read, Update, Delete – базові операції над даними.

**MVC-** Model–View–Controller, архітектурний шаблон розділення відповідальностей.

**ORM-** Object-Relational Mapping, відображення об'єктів застосунку на таблиці БД.

**ПЗ-** програмне забезпечення.

**СКБД-** система керування базами даних.

**UI-** User Interface, користувацький інтерфейс.

**UX-** User Experience, користувацький досвід.

**HTTP-** Hypertext Transfer Protocol.

**ER-** Entity–Relationship, модель сутностей і зв'язків.

|      |      |          |        |      |                         |      |
|------|------|----------|--------|------|-------------------------|------|
|      |      |          |        |      | РБ.ІП - 11.00.00.000 ПЗ | Арк. |
|      |      |          |        |      |                         | 8    |
| Змн. | Арк. | № докум. | Підпис | Дата |                         |      |

## ВСТУП

У сучасному світі цифрові сервіси стали невід’ємною частиною повсякденного життя людини, зокрема у сфері пошуку та вибору медіаконтенту. Кількість фільмів, серіалів і платформ перегляду постійно зростає, що ускладнює процес самостійного вибору контенту для перегляду. Користувач часто стикається не з нестачею інформації, а з її надлишком, тому виникає потреба у зручних інструментах, які допомагають швидко знайти фільм, оцінити його, зберегти до особистого списку та повернутися до нього пізніше.

**Актуальність теми** зумовлена необхідністю створення веб-системи рекомендаційного вибору фільмів, яка забезпечить зручний, швидкий та інтуїтивно зрозумілий інтерфейс для користувача. Незважаючи на наявність існуючих сервісів, таких як The Movie Database, IMDb, JustWatch, Netflix та Megogo, значна частина з них орієнтована або на інформаційний каталог, або на комерційний перегляд контенту. У таких умовах розроблення веб-системи MovieMatch є актуальним, оскільки вона поєднує функції каталогу, персонального профілю, підбірки фільмів та адміністрування контенту.

**Метою роботи** є розробка веб-системи MovieMatch для рекомендаційного вибору фільмів і серіалів, яка забезпечує персоналізоване збереження медіаконтенту, оцінювання, коментування, пошук, фільтрацію, формування списків перегляду та адміністрування каталогу.

**Завданнями дослідження** є вивчення сучасних підходів до побудови рекомендаційних веб-систем, аналіз існуючих сервісів для пошуку та вибору фільмів, визначення ключових потреб користувачів, виявлення основних функціональних і нефункціональних вимог, проєктування архітектури веб-застосунку, розробка моделі бази даних PostgreSQL, реалізація веб-системи MovieMatch засобами Ruby on Rails, тестування основних сценаріїв роботи та розгортання системи на сервері.

|      |      |          |        |      |                         |      |
|------|------|----------|--------|------|-------------------------|------|
|      |      |          |        |      | РБ.ІП - 11.00.00.000 ПЗ | Арк. |
|      |      |          |        |      |                         | 9    |
| Змн. | Арк. | № докум. | Підпис | Дата |                         |      |

**Об'єктом дослідження** є процес вибору фільмів і серіалів користувачем у веб-середовищі.

**Предметом дослідження** є програмні засоби, технології та інструменти, що застосовуються на етапах життєвого циклу розробки веб-системи рекомендаційного вибору фільмів, зокрема аналіз і формалізація вимог користувачів, проектування архітектури застосунку, моделювання структури бази даних, реалізація функціональних можливостей, тестування та оцінка працездатності системи.

**Методи:** аналіз предметної області, порівняльний аналіз існуючих веб-сервісів вибору фільмів, моделювання вимог, об'єктно-орієнтоване проектування, реляційне моделювання бази даних, функціональне тестування, аналіз сценаріїв користувача та оцінювання зручності інтерфейсу. Використання цих методів дало змогу обґрунтувати структуру веб-системи MovieMatch, визначити її функціональні можливості та перевірити працездатність реалізованого застосунку.

**Наукова новизна роботи** полягає у розробці веб-системи рекомендаційного вибору фільмів та серіалів MovieMatch, яка поєднує функції інформаційного каталогу, персоналізованого формування списків перегляду, оцінювання та коментування контенту в межах єдиного програмного рішення адміністративну панель.

Розроблена веб-система передбачатиме можливість перегляду каталогу фільмів і серіалів, пошуку без залежності від реєстру, фільтрації за жанрами, відкриття сторінки фільму, оцінювання у зірках, додавання коментарів, формування персональних списків «хочу подивитись», «бачив/ла» та «не подобається», використання підбірки фільмів, випадкового вибору фільму, ручного додавання власного фільму та керування каталогом через адміністративну панель.

Бакалаврська робота містить 65 сторінок, 31 рисунок, 5 таблиць, список використаних джерел із 43 найменуваннями та додатки.

|      |      |          |        |      |                         |      |
|------|------|----------|--------|------|-------------------------|------|
|      |      |          |        |      | РБ.ІП - 11.00.00.000 ПЗ | Арк. |
|      |      |          |        |      |                         | 10   |
| Змн. | Арк. | № докум. | Підпис | Дата |                         |      |

# РОЗДІЛ 1. ОГЛЯД ІСНУЮЧИХ РІШЕНЬ ТА ТЕОРЕТИЧНИХ ОСНОВ РЕКОМЕНДАЦІЙНИХ ВЕБ-СИСТЕМ

## 1.1. Поняття веб-системи рекомендацій вибору фільмів

Рекомендаційна веб-система у межах цієї роботи розглядається як програмний продукт, що допомагає користувачу звужити простір вибору і зафіксувати персональне ставлення до контенту. На відміну від звичайного каталогу, така система зберігає історію взаємодії: оцінки, коментарі, список бажаного перегляду, позначки «бачив/ла» та негативні реакції. Саме ці дані формують інформаційну основу персоналізованої роботи системи [1; 2].

У сфері фільмів рекомендації можуть будуватися різними способами. Контентний підхід аналізує жанр, опис, рік, режисера, акторів і подібні метадані. Колаборативний підхід враховує подібність поведінки різних користувачів. Гібридні підходи поєднують обидва варіанти і застосовуються у великих комерційних системах [3; 4]. У бакалаврській роботі акцент зроблено не на складній математичній моделі, а на якісному зборі та структурованому збереженні користувацьких дій, оскільки без цього неможлива коректна персоналізація.

Особливістю задачі є те, що фільм як об'єкт має не лише технічні поля, а й культурний контекст: назву, оригінальну назву, жанри, опис, тривалість, рейтинг, трейлер, зовнішні посилання, зображення та статус перегляду. Тому модель даних повинна бути достатньо гнучкою, щоб підтримувати і фільми, і серіали, а інтерфейс – достатньо простим, щоб користувач не відчував перевантаження.

Для користувача рекомендаційна веб-система виконує роль посередника між великою кількістю доступного контенту та конкретним рішенням про перегляд. Якщо звичайний каталог лише відображає перелік фільмів, то рекомендаційна система додатково враховує дії користувача, його оцінки, жанрові вподобання та попередні рішення. Завдяки цьому вибір стає більш

|      |      |          |        |      |                         |      |
|------|------|----------|--------|------|-------------------------|------|
|      |      |          |        |      | РБ.ІП - 11.00.00.000 ПЗ | Арк. |
| Змн. | Арк. | № докум. | Підпис | Дата |                         | 11   |

упорядкованим, а користувач отримує можливість швидше знайти фільм або серіал, який відповідає його поточному настрою чи інтересам.

У межах MovieMatch рекомендаційність реалізується не як складний математичний алгоритм, а як поєднання персональних дій користувача з інструментами пошуку, фільтрації та збереження контенту. Такий підхід є виправданим для бакалаврської роботи, оскільки він дозволяє показати повний цикл роботи веб-системи: від збереження даних у базі PostgreSQL до відображення результатів у користувацькому інтерфейсі [8]. Основою персоналізації є не зовнішній рейтинг фільму, а власна взаємодія користувача із системою.

Фільтрація за кількома жанрами, де активні кнопки показують вибрані жанри на рисунку 1.1.

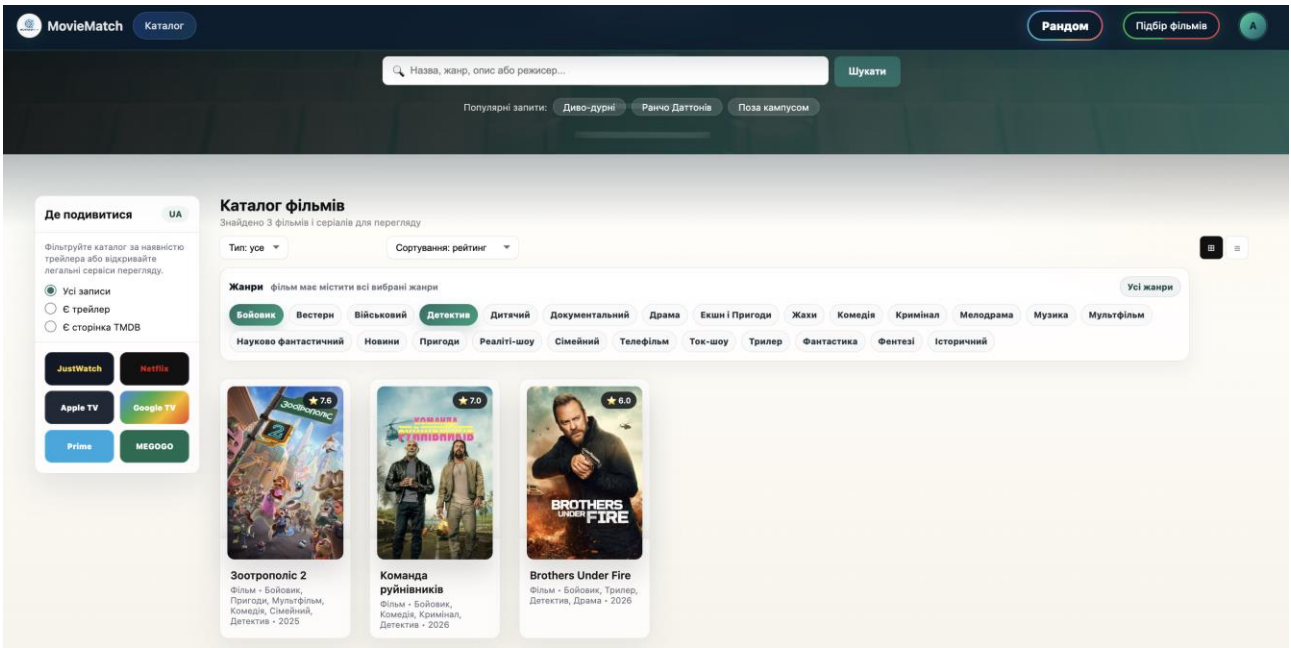


Рисунок 1.1 – Фільтрація фільмів

Сторінка фільму з трейлером, оцінкою у зірках, коментарями і посиланнями на сервіси перегляду на рисунку 1.2.

|      |      |          |        |      |                         |      |
|------|------|----------|--------|------|-------------------------|------|
|      |      |          |        |      | РБ.ІП - 11.00.00.000 ПЗ | Арк. |
|      |      |          |        |      |                         | 12   |
| Змн. | Арк. | № докум. | Підпис | Дата |                         |      |

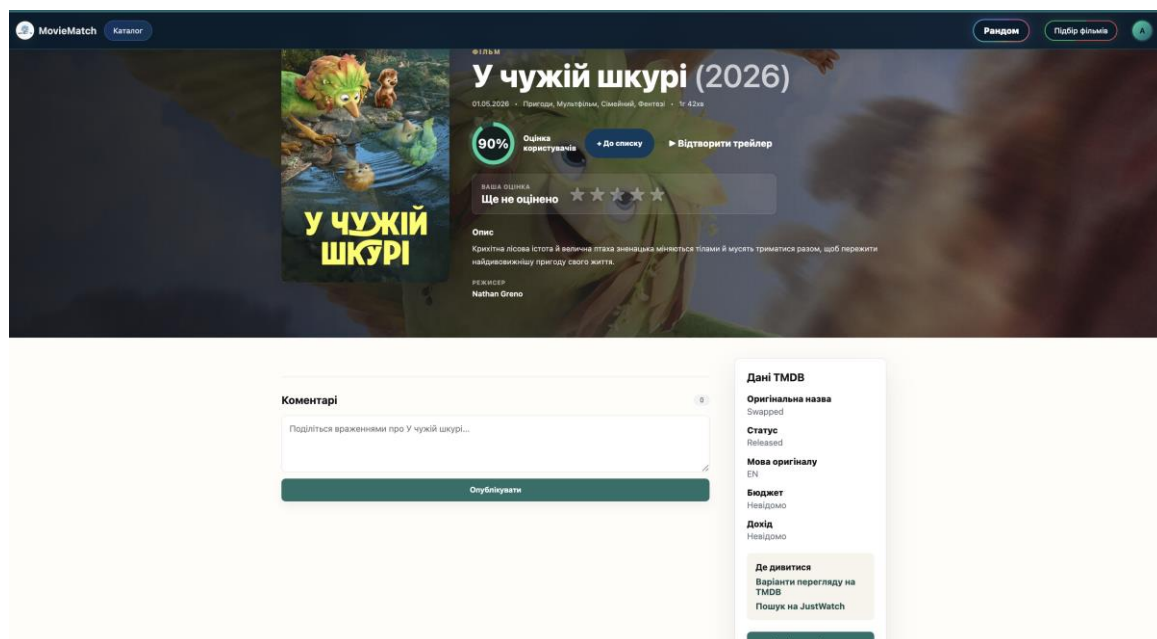


Рисунок 1.2 – Сторінка фільму

Важливою вимогою до такої системи є розмежування загальних і персональних даних. Загальна інформація про фільм, наприклад назва, опис, жанр, рік випуску чи постер, може бути спільною для всіх користувачів. Натомість оцінка у зірках, коментар, статус «хочу подивитись», «бачив/ла» або «не подобається» мають належати конкретному користувачу. Саме це забезпечує коректну роботу персонального профілю і не допускає ситуації, коли дії одного користувача впливають на особисту підбірку іншого.

Ще однією особливістю рекомендаційної веб-системи є необхідність підтримки різних сценаріїв вибору. Частина користувачів заходить на сайт із конкретним запитом і використовує пошук за назвою. Інші користувачі не мають визначеного фільму для перегляду, тому їм потрібні жанрові фільтри, випадковий вибір або підбірка у форматі свайпів [39]. Тому система MovieMatch поєднує кілька способів взаємодії: пошук, каталог, фільтрацію, персональні списки, рандомайзер і мобільний сценарій швидкого вибору.

У рекомендаційних веб-системах важливим є не лише факт наявності каталогу, а й спосіб взаємодії користувача з цим каталогом. Якщо користувач

бачить лише список фільмів без можливості зберегти власне ставлення до них, така система залишається звичайним довідником. Рекомендаційна система повинна накопичувати інформацію про інтереси користувача, його оцінки, переглянуті фільми, відхилені варіанти та бажаний контент. Саме ця інформація дозволяє зробити взаємодію персональною.

Для MovieMatch важливою є ідея поступового формування користувацького профілю. На початку користувач може не мати жодних збережених дій, тому система повинна бути корисною вже на рівні каталогу, пошуку та жанрових фільтрів. Надалі, коли користувач ставить оцінки, додає фільми до списків або позначає їх як нецікаві, система отримує більше інформації про його вподобання. Таким чином, персоналізація формується не одноразово, а поступово в процесі використання сайту [19].

Окрему роль відіграє сторінка фільму. Вона повинна поєднувати довідкову інформацію та персональні дії. До довідкової інформації належать назва, жанр, опис, постер, трейлер, дата виходу, тривалість та інші характеристики [26]. До персональних дій належать оцінювання, коментування, додавання до списку перегляду або позначення фільму як переглянутого. Такий поділ дозволяє користувачу не тільки отримати інформацію, а й одразу зафіксувати власне рішення.

Важливою складовою рекомендаційної системи є підтримка різних типів контенту. Користувачі часто обирають не лише повнометражні фільми, а й серіали, мультфільми, документальні роботи або інші медіаформати. Тому система MovieMatch передбачає поле типу медіаконтенту, що дозволяє відокремлювати фільми й серіали в каталозі. Це робить систему гнучкішою і ближчою до реального способу використання сучасних медіасервісів.

Також важливо, щоб рекомендаційна система не нав'язувала користувачу лише один спосіб вибору. Одні користувачі звикли шукати конкретну назву, інші переглядають жанри, а частина користувачів хоче отримати випадкову пропозицію. Саме тому MovieMatch поєднує пошук, фільтрацію, персональні

|      |      |          |        |      |                         |      |
|------|------|----------|--------|------|-------------------------|------|
|      |      |          |        |      | РБ.ІП - 11.00.00.000 ПЗ | Арк. |
| Змн. | Арк. | № докум. | Підпис | Дата |                         | 14   |

списки, рандомайзер і підбірку фільмів у форматі швидкого вибору [34]. Така багатоваріантність підвищує зручність системи для різних сценаріїв.

Отже, у межах цієї роботи веб-система рекомендацій вибору фільмів розглядається як програмний продукт, що поєднує каталог медіаконтенту, персональний профіль користувача та механізми збереження індивідуальних рішень. Її цінність полягає не тільки у відображенні фільмів, а й у можливості організувати власний досвід вибору та перегляду.

## 1.2. Аналіз аналогів і конкурентних підходів

Для обґрунтування функціональності MovieMatch проаналізовано кілька типів сервісів: інформаційні каталоги, платформи пошуку місць перегляду, стримінгові сервіси та соціальні сервіси для формування списків. Кожен тип має переваги, проте не завжди закриває потребу студента або звичайного користувача у простій персональній системі вибору відображено в таблиці 1.1.

Таблиця 1.1

Порівняння існуючих підходів до вибору фільмів

| Система            | Сильні сторони                                                                   | Обмеження                                                                              |
|--------------------|----------------------------------------------------------------------------------|----------------------------------------------------------------------------------------|
| The Movie Database | Детальні картки фільмів, трейлери, актори, API                                   | Не є персональною системою навчального рівня; власні списки не є центральною функцією  |
| JustWatch          | Показує, де легально переглянути контент                                         | Не замінює персональний профіль з оцінками і коментарями                               |
| IMDb               | Велика база фільмів, рейтинги, рецензії                                          | Орієнтація на глобальний рейтинг, а не на власну шкалу користувача                     |
| Netflix / Megogo   | Перегляд контенту і комерційні рекомендації                                      | Закрита екосистема, складна інтеграція для бакалаврського проєкту                      |
| MovieMatch         | Власний каталог, профіль, коментарі, рейтинги, свайпи, рандомайзер, адмін-панель | Навчальна реалізація зосереджена на персональному виборі та збереженні дій користувача |

Порівняння показує, що MovieMatch реалізовано не як копію комерційного стримінгу, а як рекомендаційно-каталогову систему. Такий підхід узгоджується з темою бакалаврської роботи і дозволяє реалістично виконати проєкт однією студенткою без команди, ліцензійних договорів і платіжної інфраструктури.

Важливою відмінністю є відмова від використання середніх оцінок інших людей як основного індикатора якості. У реалізованій системі користувач отримує можливість ставити власну оцінку у зірках, а профіль показує його персональну середню оцінку. Це робить систему ближчою до реальної поведінки людини, адже чужий рейтинг не завжди відповідає особистому смаку [39].

Аналіз аналогів показує, що більшість існуючих сервісів орієнтована або на надання інформації про фільм, або на комерційний перегляд контенту. Наприклад, The Movie Database зручно використовувати як джерело даних про фільми, акторів, трейлери, постери та рейтинги, однак цей сервіс не є персональним простором користувача для формування власної підбірки. JustWatch добре вирішує задачу пошуку місця перегляду, але його основна функція полягає саме у переході до легальних платформ, а не у збереженні індивідуальних оцінок, коментарів і реакцій користувача. IMDb містить велику базу фільмів і рейтингів, проте його рейтингова система переважно відображає загальну думку аудиторії.

Стримінгові платформи, такі як Netflix або Megogo, мають власні рекомендаційні механізми, однак вони працюють у межах закритої екосистеми. Користувач отримує рекомендації тільки щодо контенту, який доступний у конкретному сервісі, а сама логіка рекомендацій залишається прихованою. Крім того, такі платформи потребують ліцензійних договорів, платіжної інфраструктури, підтримки відеопотоку та значних ресурсів для розроблення. Для бакалаврського проєкту створення повного аналога стримінгової платформи є надмірним за складністю, тому функціональність MovieMatch зосереджено на каталогізації, персональному виборі та збереженні користувацьких дій [17].

|      |      |          |        |      |                         |      |
|------|------|----------|--------|------|-------------------------|------|
|      |      |          |        |      | РБ.ІП - 11.00.00.000 ПЗ | Арк. |
|      |      |          |        |      |                         | 16   |
| Змн. | Арк. | № докум. | Підпис | Дата |                         |      |

Під час проєктування MovieMatch було враховано переваги кожного з розглянутих підходів. Від інформаційних каталогів система переймає ідею структурованої картки фільму з назвою, жанром, описом, рейтингом, трейлером і постером. Від сервісів пошуку місця перегляду використано підхід із посиленнями на зовнішні платформи. Від соціальних і рекомендаційних сервісів взято ідею персональної взаємодії користувача з контентом через оцінки, коментарі, списки та реакції.

Основною відмінністю MovieMatch є поєднання каталогу фільмів із персональним профілем користувача. У системі користувач не лише переглядає інформацію про фільм, а й формує власну історію вибору. Для цього реалізовано списки «хочу подивитись», «бачив/ла» та «не подобається», а також можливість ставити власну оцінку у зірках. Така функціональність дозволяє зробити систему ближчою до реальної поведінки користувача, який часто обирає фільм не лише за загальним рейтингом, а й за власним настроєм, жанровими вподобаннями та попереднім досвідом.

Під час аналізу аналогів важливо враховувати не лише кількість функцій, а й те, яку саме задачу вирішує кожен сервіс [3]. Інформаційні каталоги зосереджуються на довідкових даних, стримінгові платформи — на перегляді контенту, а сервіси пошуку місця перегляду — на переході до легальних платформ. MovieMatch, у свою чергу, орієнтований на персональну організацію вибору, тому його функціональність не дублює повністю жодну з розглянутих систем.

The Movie Database є корисним джерелом структурованої інформації про фільми, акторів, постери й трейлери. Проте цей сервіс більше виконує роль відкритої бази даних і платформи для отримання метаданих. Для користувача, який хоче вести власний список перегляду або фіксувати особисті оцінки, TMDb не завжди є достатньо простим рішенням. Тому в MovieMatch частину підходів до картки фільму адаптовано, але основний акцент зроблено на персональній взаємодії.

|      |      |          |        |      |                         |      |
|------|------|----------|--------|------|-------------------------|------|
|      |      |          |        |      | РБ.ІП - 11.00.00.000 ПЗ | Арк. |
| Змн. | Арк. | № докум. | Підпис | Дата |                         | 17   |

JustWatch вирішує іншу задачу — показує, де саме можна легально переглянути фільм або серіал. Це корисно для користувача, який уже визначився з фільмом і шукає платформу перегляду. Однак JustWatch не є системою персонального оцінювання або збереження реакцій у тому вигляді, який потрібен для MovieMatch. Саме тому в розробленій системі ідея зовнішніх посилань використовується як допоміжна функція, а не як центральна [13].

Профіль користувача з вкладками «хочу подивитись», «бачив/ла», «не подобається» рисунок 1.3.

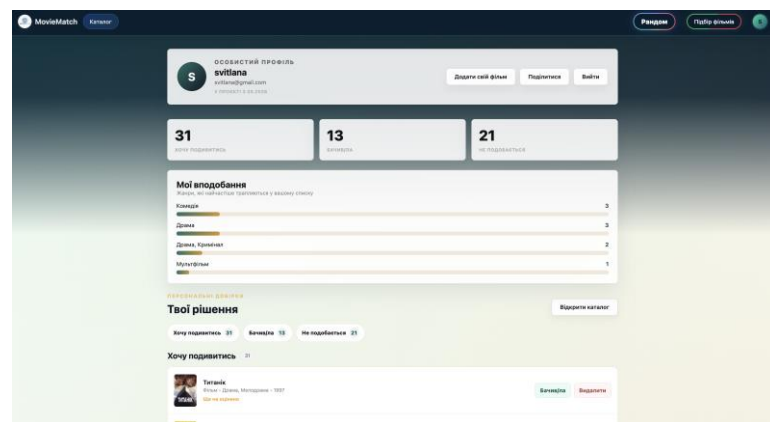


Рисунок 1.3 – Профіль користувача

– Підбірка фільмів зі свайпами у мобільному вигляді зображено на рисунку 1.4 та 1.5.

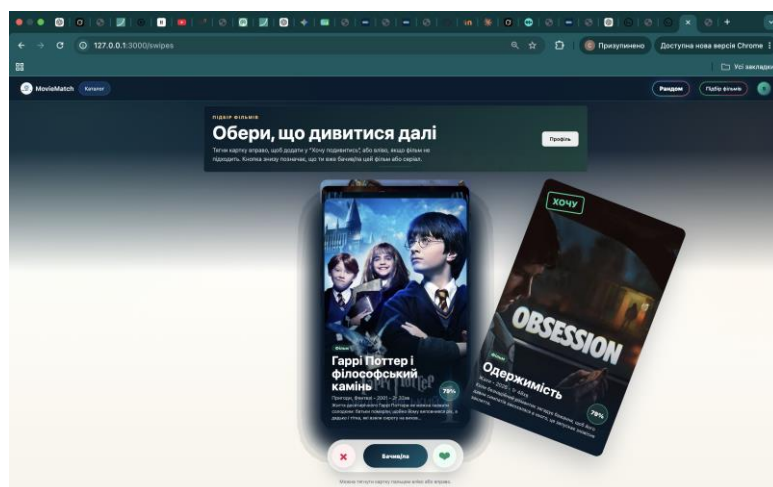


Рисунок 1.4 – Підбір фільмів, свайп вправо

|      |      |          |        |      |                         |      |
|------|------|----------|--------|------|-------------------------|------|
|      |      |          |        |      | РБ.ІП - 11.00.00.000 ПЗ | Арк. |
| Змн. | Арк. | № докум. | Підпис | Дата |                         | 18   |

IMDb і подібні рейтингові сервіси мають велику кількість даних та відгуків, однак вони часто акцентують увагу на загальному рейтингу. Такий рейтинг корисний для первинної оцінки якості фільму, але він не завжди відображає індивідуальні вподобання конкретного користувача. У MovieMatch персональна оцінка відокремлена від зовнішнього рейтингу, що дозволяє користувачу формувати власну шкалу вражень

Окремою перевагою MovieMatch є наявність адміністративної панелі, яка забезпечує контроль над каталогом і користувацьким контентом. Адміністратор може переглядати зареєстрованих користувачів, додані фільми, коментарі та дії в системі. Це відрізняє розроблений застосунок від простого навчального каталогу, оскільки в ньому передбачено не тільки користувацьку частину, а й механізм керування даними.

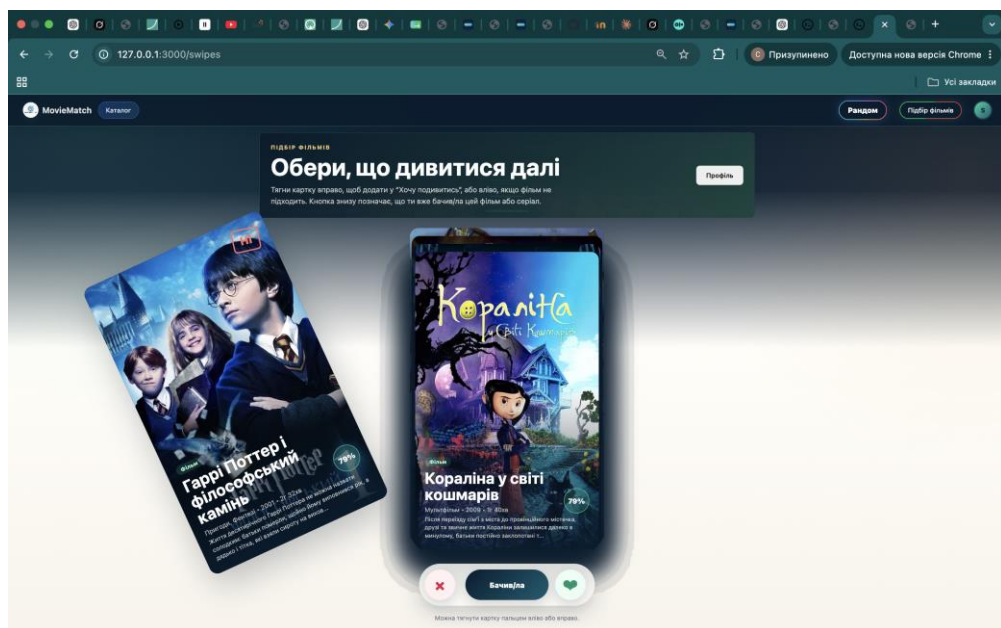


Рисунок 1.5 – Підбір фільмів, свайп вліво

Таким чином, MovieMatch займає проміжну позицію між інформаційним каталогом фільмів і повноцінною стримінговою платформою. Система не реалізує перегляд відео всередині сайту та не потребує комерційної інфраструктури, але забезпечує основні сценарії вибору фільмів: пошук,

|      |      |          |        |      |                         |      |
|------|------|----------|--------|------|-------------------------|------|
|      |      |          |        |      | РБ.ІП - 11.00.00.000 ПЗ | Арк. |
| Змн. | Арк. | № докум. | Підпис | Дата |                         | 19   |

фільтрацію, перегляд картки, оцінювання, коментування, збереження у списки, підбірку фільмів і перехід до зовнішніх сервісів перегляду. Саме така функціональна межа відповідає темі бакалаврської роботи та дозволяє практично реалізувати веб-систему рекомендаційного вибору фільмів засобами Ruby on Rails і PostgreSQL [27].

### 1.3. Технологічні тенденції у розробленні каталогів медіаконтенту

Сучасні веб-застосунки для роботи з каталогами зазвичай поєднують серверну логіку, реляційну або документну базу даних, адаптивний інтерфейс, пошукові механізми та інтеграцію із зовнішніми API [6]. Для MovieMatch обрано Ruby on Rails, оскільки фреймворк має зрілу MVC-структуру, Active Record для роботи з базою даних, маршрутизацію, шаблони представлення і зручну підтримку CRUD-операцій [7].

PostgreSQL використано як основну СКБД через підтримку транзакцій, індексів, зв'язків, обмежень цілісності та широкі можливості для подальшого пошуку й аналітики [10]. Для рекомендаційної системи це особливо важливо, тому що дані користувачів, оцінок, коментарів і списків мають бути узгодженими. Наприклад, оцінка не може існувати без користувача і фільму, а реакція користувача на один фільм повинна бути унікальною.

Окремим технологічним напрямом є адаптивний інтерфейс. Користувачі часто відкривають подібні сервіси з телефона, тому функції каталогу, профілю та свайпів мають бути зручними на вузькому екрані. У MovieMatch це враховано через мобільно орієнтовану сторінку підбору, кнопки дій під постером і зрозумілу навігацію.

Важливою тенденцією у розробленні сучасних каталогів медіаконтенту є відокремлення загальних даних від персональних дій користувача. Загальні дані описують сам фільм або серіал: назву, жанр, рік випуску, опис, постер, трейлер, тривалість та інші характеристики. Персональні дані відображають взаємодію конкретного користувача з цим контентом: оцінку, коментар, додавання до

|      |      |          |        |      |                         |      |
|------|------|----------|--------|------|-------------------------|------|
|      |      |          |        |      | РБ.ІП - 11.00.00.000 ПЗ | Арк. |
| Змн. | Арк. | № докум. | Підпис | Дата |                         | 20   |

списку, позначку «бачив/ла» або негативну реакцію. Такий поділ дозволяє будувати систему так, щоб один і той самий фільм міг мати різне значення для різних користувачів.

Адмін-панель з користувачами, діями, коментарями та фільмами каталогу зображена на рисунку 1.6, 1.7.

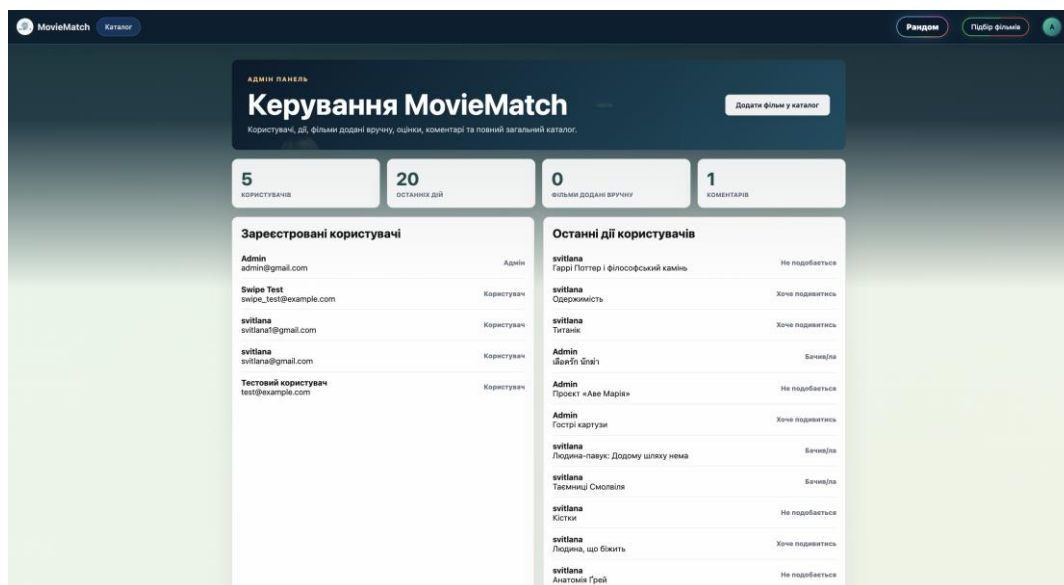


Рисунок 1.6 – Адмін-панель

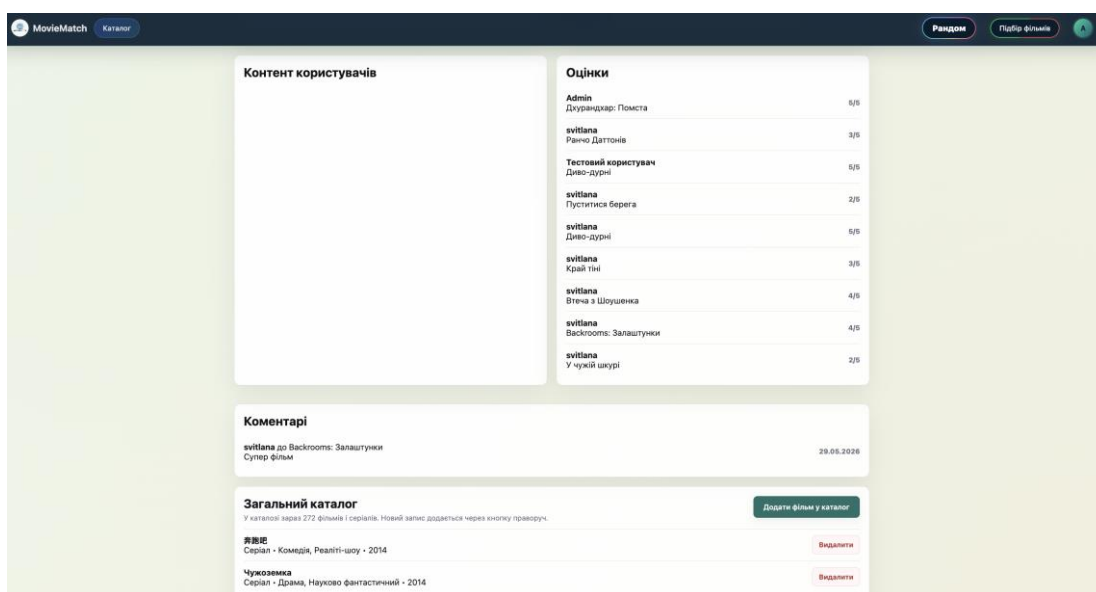


Рисунок 1.7 – Продовження адмін-панель

Ще одним важливим напрямом є використання серверних фреймворків, які підтримують швидке створення CRUD-функцій і роботу з базою даних. Ruby on Rails добре підходить для такого типу веб-застосунків, оскільки має вбудовані засоби маршрутизації, шаблонізації, роботи з моделями, валідації даних і організації контролерів [21]. Для MovieMatch це дозволило реалізувати каталог фільмів, профіль користувача, коментарі, оцінки, персональні списки та адміністративну панель у єдиній структурі MVC.

Для каталогів медіаконтенту також важливою є швидкість пошуку та зручність фільтрації. Користувач часто не знає точної назви фільму або вводить її з маленької літери, тому пошук має бути нечутливим до регістру та підтримувати часткові збіги. Крім того, фільтрація за жанрами повинна працювати зрозуміло: якщо користувач обирає кілька жанрів одночасно, система має показувати фільми, які відповідають усім вибраним умовам. У MovieMatch ця логіка використовується для підвищення точності результатів і зменшення кількості нерелевантних записів.

Значна частина користувачів відкриває подібні сервіси зі смартфона, тому інтерфейс має бути зручним на малому екрані. У MovieMatch це враховано через мобільний сценарій підбірки фільмів, розміщення кнопок під постером, зрозумілу навігацію та карткове представлення контенту. Такий підхід відповідає сучасній практиці створення веб-застосунків.

Актуальною тенденцією є також використання простих інтерактивних механізмів для зменшення складності вибору. Великий каталог може перевантажувати користувача, тому функції швидкого вибору стають важливою частиною інтерфейсу [25]. У MovieMatch для цього використано рандомайзер і підбірку фільмів у форматі свайпів. Вони не замінюють каталог, але доповнюють його і дають користувачу альтернативний спосіб взаємодії.

З технічної точки зору Ruby on Rails дозволяє швидко створювати такі системи завдяки готовим механізмам маршрутизації, роботі з базою через Active Record, шаблонам представлення та підтримці MVC-структури. PostgreSQL, у

|      |      |          |        |      |                         |      |
|------|------|----------|--------|------|-------------------------|------|
|      |      |          |        |      | РБ.ІП - 11.00.00.000 ПЗ | Арк. |
|      |      |          |        |      |                         | 22   |
| Змн. | Арк. | № докум. | Підпис | Дата |                         |      |

свою чергу, забезпечує надійне збереження пов'язаних даних, індекси, обмеження цілісності та можливість подальшого розширення пошуку. Поєднання цих технологій є придатним для реалізації рекомендаційно-каталогового веб-застосунку [19].

#### 1.4. Висновки по розділу

У першому розділі розглянуто поняття рекомендаційної веб-системи, проаналізовано аналоги та визначено технологічні тенденції, релевантні для MovieMatch. Обґрунтовано формат системи персонального вибору фільмів і серіалів з каталогом, профілем, оцінками, коментарями, підбірками та базовими рекомендаційними сценаріями [32].

Порівняння існуючих сервісів показало, що інформаційні каталоги, платформи пошуку місця перегляду та стримінгові сервіси вирішують різні частини задачі вибору фільму. Водночас вони не завжди забезпечують простий персональний простір, у якому користувач може зберігати власні оцінки, списки та реакції незалежно від конкретної платформи перегляду. Саме ця потреба стала основою для розроблення MovieMatch.

У розділі визначено технологічну основу майбутньої системи. Ruby on Rails обрано через підтримку MVC-архітектури, маршрутизації, шаблонів представлення та зручної роботи з моделями. PostgreSQL використовується як реляційна база даних, що забезпечує збереження користувачів, фільмів, оцінок, коментарів і персональних списків [17]. Подальші розділи роботи зосереджені на формуванні вимог, проєктуванні архітектури, моделі PostgreSQL, реалізації та перевірці працездатності такого рішення.

У розділі також обґрунтовано технологічний вибір. Ruby on Rails забезпечує швидку розробку веб-застосунку, зрозумілу MVC-структуру, маршрутизацію та роботу з моделями. PostgreSQL використовується як надійна реляційна база даних для збереження фільмів, користувачів, оцінок, коментарів.

|      |      |          |        |      |                         |      |
|------|------|----------|--------|------|-------------------------|------|
|      |      |          |        |      | РБ.ІП - 11.00.00.000 ПЗ | Арк. |
|      |      |          |        |      |                         | 23   |
| Змн. | Арк. | № докум. | Підпис | Дата |                         |      |

## РОЗДІЛ 2. АНАЛІЗ ВИМОГ ТА ПОСТАНОВКА ЗАДАЧІ

### 2.1. Цільова аудиторія та користувацькі сценарії

Цільовою аудиторією MovieMatch є користувачі, які хочуть швидко обрати фільм або серіал для перегляду, зберегти власні рішення і повернутися до них пізніше. Типовими користувачами можуть бути студенти, молоді спеціалісти, сім'ї або групи друзів, які обирають контент для вечірнього перегляду [42]. Для таких користувачів важлива не складність алгоритму, а прозорість дії: знайти, переглянути картку, оцінити, додати у список, поділитися.

Під час формування вимог враховано кілька сценаріїв. Перший сценарій – швидкий пошук: користувач вводить назву фільму незалежно від регістру і очікує побачити релевантний результат. Другий сценарій – каталоговий перегляд: користувач застосовує фільтри за типом, жанрами і сортуванням. Третій сценарій – персональне збереження: користувач додає фільм до списку «хочу подивитись», переносить його до «бачив/ла» або вилучає.

Четвертий сценарій – ігровий вибір через підбірку фільмів. Користувач бачить картку, свайпає вправо для додавання, вліво для відхилення або натискає кнопку «бачив/ла». Такий формат знижує когнітивне навантаження і робить вибір більш природним на мобільному пристрої. П'ятий сценарій – адміністрування: адміністратор переглядає користувачів, їх дії, доданий контент, коментарі й може керувати загальним каталогом [9].

Оскільки система MovieMatch орієнтована на повсякденне використання, важливо врахувати різні рівні підготовки користувачів. Частина аудиторії може добре орієнтуватися у фільмах, жанрах, рейтингах і стримінгових сервісах, а інша частина користувачів може просто шукати швидку відповідь на питання, що подивитися ввечері. Тому інтерфейс системи має бути зрозумілим без додаткових інструкцій, а основні дії повинні виконуватися за мінімальну кількість кроків.

Для незареєстрованого користувача основним сценарієм є ознайомлення з каталогом. Такий користувач може переглядати фільми й серіали, відкривати

|      |      |          |        |      |                         |      |
|------|------|----------|--------|------|-------------------------|------|
|      |      |          |        |      | РБ.ІП - 11.00.00.000 ПЗ | Арк. |
| Змн. | Арк. | № докум. | Підпис | Дата |                         | 24   |

сторінку окремого фільму, бачити опис, жанр, рік випуску, постер, трейлер і посилання на зовнішні сервіси перегляду [26]. Цей сценарій важливий, оскільки він дозволяє користувачу оцінити можливості системи до створення облікового запису. Якщо каталог є зручним і зрозумілим, користувач має більшу мотивацію зареєструватися для збереження власних дій.

Зареєстрований користувач має ширший набір сценаріїв. Після входу в систему він може ставити фільмам персональні оцінки у зірках, залишати коментарі, додавати фільми до списку «хочу подивитись», переносити їх до списку «бачив/ла» або позначати як такі, що не зацікавили. Усі ці дії зберігаються у базі даних і прив'язуються до конкретного облікового запису. Завдяки цьому профіль користувача стає персональним простором для організації власного досвіду перегляду [18].

Сторінка ручного додавання фільму з URL або завантаженням зображення на рисунку 2.1.

The screenshot shows a web interface for 'MovieMatch'. At the top, there's a navigation bar with 'Каталог', 'Рандом', and 'Підбір фільмів'. The main content area is titled 'Додати фільм у загальний каталог' (Add movie to the general catalog) with a subtitle 'Цей фільм бачитимуть усі користувачі.' (This movie will be seen by all users). The form contains the following fields: 'Назва' (Name) with a placeholder 'Назва фільму'; 'Тип' (Type) with a dropdown menu showing 'Фільм'; 'Жанр' (Genre) with a dropdown menu showing 'Бойовик, Драма'; 'Рік' (Year) and 'Тривалість, хв' (Duration, min) as separate input fields; 'Рейтинг 0-10' (Rating 0-10) with a numeric input field showing '0'; 'URL постера' (Poster URL) with a placeholder 'https://...'; a note 'Або завантажте постер з комп'ютера нижче.' (Or upload the poster from your computer below.); 'Файл постера' (Poster file) with a 'Вибрати файл' (Select file) button and a 'Файл не вибрано' (File not selected) status; and a large 'Опис' (Description) text area. At the bottom of the form is a green button labeled 'Додати в каталог' (Add to catalog).

Рисунок 2.1 – Додавання фільму вручну

Результат відкриття сайту з телефона за IP-адресою сервера на рисунку 2.2.

|      |      |          |        |      |                         |      |
|------|------|----------|--------|------|-------------------------|------|
|      |      |          |        |      | РБ.ІП - 11.00.00.000 ПЗ | Арк. |
| Змн. | Арк. | № докум. | Підпис | Дата |                         | 25   |

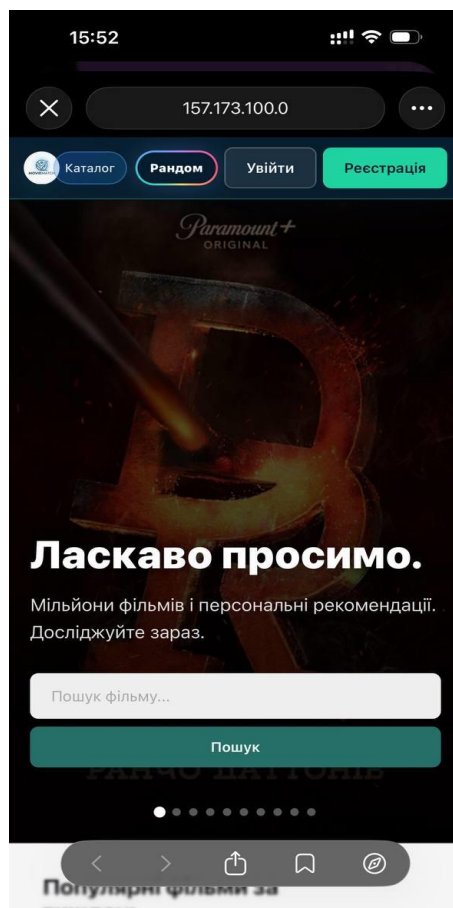


Рисунок 2.2 – Додавання фільму вручну

Окремим сценарієм є використання підбірки фільмів у форматі свайпів. Такий підхід розрахований насамперед на мобільних користувачів, які не хочуть довго переглядати великий каталог. Система послідовно показує картки фільмів, а користувач швидко приймає рішення: додати фільм до бажаних, відхилити його або позначити як переглянутий. Це спрощує процес вибору і робить взаємодію з сайтом більш динамічною.

Ще одним важливим сценарієм є ручне додавання фільму користувачем. Така можливість потрібна тоді, коли певного фільму немає в загальному каталозі або користувач хоче зберегти власний запис для себе [38]. У MovieMatch вручну доданий фільм не потрапляє автоматично до загального каталогу, а залишається видимим для користувача, який його створив. Це дозволяє не змішувати приватний контент із публічним каталогом і зберігати впорядкованість загальних даних.

|      |      |          |        |      |                         |      |
|------|------|----------|--------|------|-------------------------|------|
|      |      |          |        |      | РБ.ІП - 11.00.00.000 ПЗ | Арк. |
|      |      |          |        |      |                         | 26   |
| Змн. | Арк. | № докум. | Підпис | Дата |                         |      |

Фільтрація за жанрами повинна працювати зрозуміло та передбачувано. Якщо користувач обирає один жанр, система показує фільми, які відповідають цьому жанру. Якщо вибрано кілька жанрів, результат має містити фільми, що відповідають усім вибраним умовам, а не кожному жанру окремо. Такий підхід дозволяє точніше звужити вибір і знайти контент, який відповідає конкретному запиту користувача.

Оцінювання у зірках реалізує персональне ставлення користувача до фільму. На відміну від зовнішнього рейтингу, який може формуватися великою кількістю інших людей, власна оцінка відображає індивідуальну думку конкретного користувача. Тому вона повинна зберігатися окремо і бути прив'язаною до пари користувач-фільм. Це дозволяє в профілі показувати саме персональну середню оцінку, а не загальний рейтинг із зовнішніх джерел.

Коментарі дають користувачу можливість залишити текстову реакцію на фільм. Вони доповнюють оцінювання, оскільки числова оцінка не завжди пояснює причину ставлення до контенту. Коментар може містити коротке враження, зауваження або пояснення, чому фільм сподобався чи не сподобався. Для адміністратора коментарі також є частиною користувацької активності, яку можна контролювати через панель керування.

Персональний профіль повинен об'єднувати основні дії користувача в одному місці. У профілі відображаються фільми, які користувач хоче подивитися, уже переглянув або позначив як нецікаві. Такий поділ допомагає організувати власний список перегляду та не втрачати фільми, які були обрані раніше. Профіль також демонструє практичну цінність системи, оскільки саме в ньому видно результат персональної взаємодії користувача з каталогом.

Підбірка фільмів у форматі свайпів є додатковим сценарієм вибору. Вона потрібна для випадків, коли користувач не має конкретного запиту і хоче швидко переглянути кілька варіантів. Свайп вправо додає фільм до списку бажаного перегляду, свайп вліво позначає його як нецікавий, а кнопка «бачив/ла» переносить фільм до списку переглянутих. Такий сценарій особливо зручний для

|      |      |          |        |      |                         |      |
|------|------|----------|--------|------|-------------------------|------|
|      |      |          |        |      | РБ.ІП - 11.00.00.000 ПЗ | Арк. |
|      |      |          |        |      |                         | 27   |
| Змн. | Арк. | № докум. | Підпис | Дата |                         |      |

мобільного використання.

Рандомайзер виконує функцію швидкого випадкового вибору фільму. Він корисний тоді, коли користувач не хоче самостійно переглядати каталог або застосовувати фільтри. Після натискання система обирає випадковий публічний фільм і відкриває його сторінку. Така функція робить взаємодію з сайтом більш динамічною і доповнює класичний каталог.

Адміністративна панель необхідна для керування загальним станом системи. Через неї адміністратор може переглядати користувачів, додані фільми, коментарі та дії в системі. Також адміністратор має можливість додавати або видаляти фільми із загального каталогу. Наявність адміністративної частини важлива, оскільки без неї система не мала б механізму контролю якості даних.

Нефункціональні вимоги визначають якість реалізації зазначених функцій. До них належать продуктивність, безпека, зручність використання, підтримуваність, переносимість і стабільність роботи. Для рекомендаційно-каталогової системи ці вимоги є важливими, оскільки користувач очікує не лише наявності функцій, а й швидкої, зрозумілої та надійної взаємодії з сайтом.

Вимога продуктивності полягає в тому, що система повинна відкривати каталог фільмів і серіалів без значних затримок. MovieMatch не має звертатися до зовнішнього API під час кожного завантаження сторінки каталогу, оскільки нестабільність або недоступність стороннього сервісу може негативно вплинути на роботу застосунку. Основні дані про фільми повинні зберігатися у власній базі PostgreSQL, щоб каталог залишався доступним незалежно від стану зовнішніх джерел.

Безпека системи пов'язана насамперед із захистом персональних дій користувача. Авторизація повинна гарантувати, що оцінки, коментарі, персональні списки та вручну додані фільми належать конкретному обліковому запису. Користувач не повинен мати доступу до редагування або видалення чужих даних. Особливо важливо, щоб вручну доданий фільм був видимим тільки його власнику, якщо адміністратор не вніс цей запис до загального каталогу.

|      |      |          |        |      |                         |      |
|------|------|----------|--------|------|-------------------------|------|
|      |      |          |        |      | РБ.ІП - 11.00.00.000 ПЗ | Арк. |
| Змн. | Арк. | № докум. | Підпис | Дата |                         | 28   |

## 2.2. Функціональні та нефункціональні вимоги

Нефункціональні вимоги визначають не окремі дії користувача, а загальні характеристики якості веб-системи MovieMatch зображено в таблиці 2.1

Таблиця 2.1

Функціональні вимоги до MovieMatch

| Код | Вимога                            | Пояснення                                                                |
|-----|-----------------------------------|--------------------------------------------------------------------------|
| F1  | Реєстрація та вхід                | Користувач повинен мати персональний обліковий запис                     |
| F2  | Каталог фільмів і серіалів        | Система відображає загальний публічний каталог                           |
| F3  | Пошук без залежності від регістру | Запит «inception» має знаходити «Inception»                              |
| F4  | Фільтрація за жанрами             | Користувач може вибрати кілька жанрів, які мають одночасно бути у фільмі |
| F5  | Оцінювання у зірках               | Оцінка зберігається як персональна оцінка користувача                    |
| F6  | Коментарі                         | Користувач може залишати коментар до фільму                              |
| F7  | Персональний профіль              | У профілі зберігаються списки: хочу подивитись, бачив/ла, не подобається |
| F8  | Підбірка фільмів                  | Свайп-механіка додає фільми у відповідні списки                          |
| F9  | Рандомайзер                       | Система випадково обирає фільм і перекидає на його сторінку              |
| F10 | Адмін-панель                      | Адміністратор керує загальним каталогом і бачить активність користувачів |

До них належать продуктивність, безпека, зручність використання, підтримуваність, переносимість і стабільність роботи. Для рекомендаційно-каталогової системи ці вимоги є важливими, оскільки користувач очікує не лише наявності функцій, а й швидкої, зрозумілої та надійної взаємодії з сайтом [28].

Вимога продуктивності полягає в тому, що система повинна відкривати каталог фільмів і серіалів без значних затримок. MovieMatch не має звертатися до зовнішнього API під час кожного завантаження сторінки каталогу, оскільки

нестабільність або недоступність стороннього сервісу може негативно впливати на роботу застосунку. Основні дані про фільми повинні зберігатися у власній базі даних PostgreSQL [30], щоб каталог залишався доступним незалежно від стану зовнішніх джерел. Звернення до зовнішніх сервісів доцільне на етапі наповнення або оновлення даних, але не як обов'язкова умова кожного відкриття сторінки.

Безпека системи пов'язана насамперед із захистом персональних дій користувача. Авторизація повинна гарантувати, що оцінки, коментарі, персональні списки та вручну додані фільми належать конкретному обліковому запису. Користувач не повинен мати доступу до редагування або видалення чужих даних. Особливо важливо, щоб вручну доданий фільм був видимим тільки його власнику, якщо адміністратор не вніс цей запис до загального каталогу [34]. Такий підхід дозволяє розмежувати приватний користувацький контент і публічну частину системи.

Функціональні вимоги визначають конкретні можливості, які повинна надавати система користувачам і адміністратору. Для MovieMatch такі вимоги сформовано на основі сценаріїв, описаних у попередньому підрозділі. Основна увага приділена тим діям, які безпосередньо пов'язані з вибором фільмів, збереженням персональних рішень та керуванням каталогом. Саме тому до функціональних вимог включено не лише перегляд каталогу, а й пошук, фільтрацію, оцінювання, коментування, роботу з персональними списками, підбірку фільмів і адміністративний контроль.

Реєстрація та вхід необхідні для того, щоб система могла відокремлювати персональні дані різних користувачів. Без авторизації неможливо коректно зберігати оцінки, коментарі, списки перегляду або вручну додані фільми. Тому обліковий запис є основою персоналізації. Після входу користувач отримує доступ до власного профілю, де зберігаються його дії та сформовані списки.

Каталог фільмів і серіалів є центральною функцією MovieMatch. Він повинен відображати публічні записи, доступні всім користувачам. Кожна картка фільму має містити достатню кількість інформації для первинного вибору: назву,

|      |      |          |        |      |                         |      |
|------|------|----------|--------|------|-------------------------|------|
|      |      |          |        |      | РБ.ІП - 11.00.00.000 ПЗ | Арк. |
|      |      |          |        |      |                         | 30   |
| Змн. | Арк. | № докум. | Підпис | Дата |                         |      |

жанр, тип контенту, рік випуску, постер або інше зображення, а також короткі відомості про фільм. Каталог повинен бути зручним для перегляду як на комп'ютері, так і на мобільному пристрої.

Пошук без залежності від реєстру є важливою вимогою, оскільки користувачі не завжди вводять назву фільму точно так, як вона збережена в базі даних. Наприклад, запит з маленької літери має знаходити фільм, навіть якщо в каталозі його назва починається з великої. Така поведінка робить систему більш дружньою до користувача і зменшує кількість ситуацій, коли потрібний фільм є в базі, але не знаходиться через формат введення [10].

Фільтрація за жанрами повинна працювати зрозуміло та передбачувано. Якщо користувач обирає один жанр, система показує фільми, які відповідають цьому жанру. Якщо вибрано кілька жанрів, результат має містити фільми, що відповідають усім вибраним умовам, а не кожному жанру окремо. Такий підхід дозволяє точніше звужити вибір і знайти контент, який відповідає конкретному запиту користувача.

Оцінювання у зірках реалізує персональне ставлення користувача до фільму. На відміну від зовнішнього рейтингу, який може формуватися великою кількістю інших людей, власна оцінка відображає індивідуальну думку конкретного користувача. Тому вона повинна зберігатися окремо і бути прив'язаною до пари користувач-фільм. Це дозволяє в профілі показувати саме персональну середню оцінку, а не загальний рейтинг із зовнішніх джерел [35].

Коментарі дають користувачу можливість залишити текстову реакцію на фільм. Вони доповнюють оцінювання, оскільки числова оцінка не завжди пояснює причину ставлення до контенту. Коментар може містити коротке враження, зауваження або пояснення, чому фільм сподобався чи не сподобався. Для адміністратора коментарі також є частиною користувацької активності, яку можна контролювати через панель керування.

Персональний профіль повинен об'єднувати основні дії користувача в одному місці. У профілі відображаються фільми, які користувач хоче

|      |      |          |        |      |                         |      |
|------|------|----------|--------|------|-------------------------|------|
|      |      |          |        |      | РБ.ІП - 11.00.00.000 ПЗ | Арк. |
|      |      |          |        |      |                         | 31   |
| Змн. | Арк. | № докум. | Підпис | Дата |                         |      |

подивитися, уже переглянув або позначив як нецікаві [26]. Такий поділ допомагає організувати власний список перегляду та не втрачати фільми, які були обрані раніше. Профіль також демонструє практичну цінність системи, оскільки саме в ньому видно результат персональної взаємодії користувача з каталогом.

Підбірка фільмів у форматі свайпів є додатковим сценарієм вибору. Вона потрібна для випадків, коли користувач не має конкретного запиту і хоче швидко переглянути кілька варіантів. Свайп вправо додає фільм до списку бажаного перегляду, свайп вліво позначає його як нецікавий, а кнопка «бачив/ла» переносить фільм до списку переглянутих. Такий сценарій особливо зручний для мобільного використання [31].

Рандомайзер виконує функцію швидкого випадкового вибору фільму. Він корисний тоді, коли користувач не хоче самотійно переглядати каталог або застосовувати фільтри. Після натискання система обирає випадковий публічний фільм і відкриває його сторінку. Така функція робить взаємодію з сайтом більш динамічною і доповнює класичний каталог.

Адміністративна панель необхідна для керування загальним станом системи. Через неї адміністратор може переглядати користувачів, додані фільми, коментарі та дії в системі. Також адміністратор має можливість додавати або видаляти фільми із загального каталогу [19]. Наявність адміністративної частини важлива, оскільки без неї система не мала б механізму контролю якості даних.

Нефункціональні вимоги визначають якість реалізації зазначених функцій. До них належать продуктивність, безпека, зручність використання, підтримуваність, переносимість і стабільність роботи [35]. Для рекомендаційно-каталогової системи ці вимоги є важливими, оскільки користувач очікує не лише наявності функцій, а й швидкої, зрозумілої та надійної взаємодії з сайтом.

Вимога продуктивності полягає в тому, що система повинна відкривати каталог фільмів і серіалів без значних затримок. MovieMatch не має звертатися до зовнішнього API під час кожного завантаження сторінки каталогу, оскільки нестабільність або недоступність стороннього сервісу може негативно вплинути

|      |      |          |        |      |                         |      |
|------|------|----------|--------|------|-------------------------|------|
|      |      |          |        |      | РБ.ІП - 11.00.00.000 ПЗ | Арк. |
| Змн. | Арк. | № докум. | Підпис | Дата |                         | 32   |

на роботу застосунку. Основні дані про фільми повинні зберігатися у власній базі PostgreSQL, щоб каталог залишався доступним незалежно від стану зовнішніх джерел.

Безпека системи пов'язана насамперед із захистом персональних дій користувача. Авторизація повинна гарантувати, що оцінки, коментарі, персональні списки та вручну додані фільми належать конкретному обліковому запису. Користувач не повинен мати доступу до редагування або видалення чужих даних. Особливо важливо, щоб вручну доданий фільм був видимим тільки його власнику, якщо адміністратор не вніс цей запис до загального каталогу [24].

Зручність використання є однією з ключових нефункціональних вимог. Інтерфейс повинен мати зрозумілі назви кнопок, логічне розміщення навігації, читабельні картки фільмів і передбачувану поведінку після натискання на елементи керування. Наприклад, кнопки додавання до списку, оцінювання, коментування або переходу до профілю мають бути помітними й не повинні перекриватися зображеннями постерів.

Окрему увагу необхідно приділити адаптивності інтерфейсу. Користувачі можуть відкривати сайт не лише з комп'ютера, а й з мобільного телефона, тому сторінки каталогу, профілю, фільму та підбірки повинні коректно відображатися на екранах різного розміру. Для мобільного сценарію важливо, щоб кнопки дій були достатньо великими, розміщувалися нижче постера та не заважали перегляду основної інформації [20].

Підтримуваність системи забезпечується дотриманням MVC-структури Ruby on Rails. Розділення коду на моделі, контролери та представлення дозволяє легше змінювати окремі частини застосунку без порушення всієї системи. Моделі відповідають за роботу з даними та зв'язки між сутностями, контролери обробляють запити користувача, а представлення формують сторінки інтерфейсу.

Переносимість означає можливість запуску MovieMatch як у локальному середовищі розробки, так і на віддаленому сервері. Для цього застосунок має

|      |      |          |        |      |                         |      |
|------|------|----------|--------|------|-------------------------|------|
|      |      |          |        |      | РБ.ІП - 11.00.00.000 ПЗ | Арк. |
| Змн. | Арк. | № докум. | Підпис | Дата |                         | 33   |

використовувати стандартні засоби Ruby on Rails, PostgreSQL, Puma та Nginx. У локальному середовищі система працює з базою розробки, а на сервері — з production-базою. Такий поділ дозволяє безпечно розробляти і тестувати зміни локально, не впливаючи на серверну версію сайту.

Стабільність роботи системи також є важливою вимогою. Застосунок повинен коректно обробляти типові помилки користувача: порожній пошуковий запит, відсутність результатів фільтрації, спробу відкрити неіснуючий фільм або дію без авторизації. У таких випадках система має не завершувати роботу з помилкою, а показувати зрозуміле повідомлення або перенаправляти користувача на відповідну сторінку [29].

Таким чином, функціональні вимоги визначають, які саме можливості повинна мати MovieMatch, а нефункціональні вимоги визначають якість реалізації цих можливостей. Разом вони формують основу для проектування архітектури, бази даних і користувацького інтерфейсу системи.

### 2.3. Постановка задачі проєктування

Необхідно розробити веб-систему MovieMatch, яка працює як каталог і персональний простір користувача для вибору фільмів та серіалів. Система повинна підтримувати базові ролі: гість, зареєстрований користувач та адміністратор. Гість може переглядати каталог і відкривати сторінки фільмів. Зареєстрований користувач додатково може оцінювати, коментувати, додавати фільми до персональних списків, створювати власний фільм і ділитися підбіркою. Адміністратор керує каталогом і переглядає активність [45].

Межі системи визначено так, щоб проєкт був реалістичним для бакалаврської роботи. MovieMatch не реалізує оплату, ліцензійний перегляд відео всередині сайту та повну інтеграцію зі стрімінгами. Натомість система може надавати посилання на партнерські або інформаційні сервіси перегляду, зокрема JustWatch, Netflix, Apple TV, Google TV, Prime Video, MEGOGO та TMDb. Такий підхід відповідає рівню навчального проєкту і не потребує

|      |      |          |        |      |                         |      |
|------|------|----------|--------|------|-------------------------|------|
|      |      |          |        |      | РБ.ІП - 11.00.00.000 ПЗ | Арк. |
| Змн. | Арк. | № докум. | Підпис | Дата |                         | 34   |

комерційних договорів.

Таким чином, цільова аудиторія MovieMatch охоплює кілька груп користувачів: гостей, зареєстрованих користувачів і адміністратора. Для кожної групи визначено власні сценарії взаємодії із системою. Такий поділ ролей дозволяє чітко визначити функціональні вимоги до системи та перейти до формування її архітектури й моделі бази даних, зображено на рисунку 2.1.

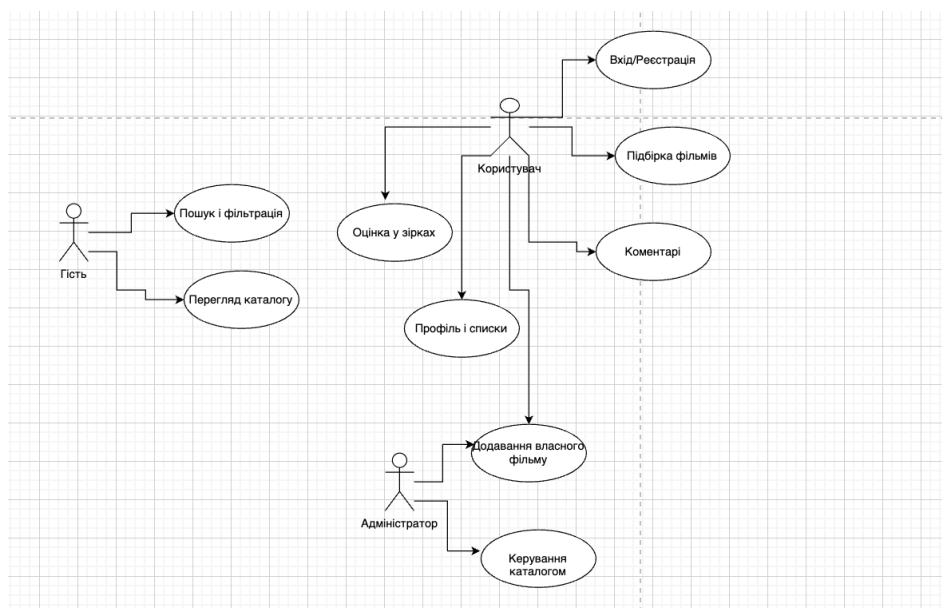


Рисунок 2.1 – Діаграма варіантів використання системи MovieMatch

Очікуваним результатом є працездатний веб-застосунок із PostgreSQL-базою, який можна запускати локально та на сервері. Результат має демонструвати повний цикл інженерії програмного забезпечення: аналіз вимог, проектування архітектури, створення моделі даних, реалізацію основних функцій, тестування та розгортання [19].

Необхідно розробити веб-систему MovieMatch, яка працює як каталог і персональний простір користувача для вибору фільмів та серіалів. Система повинна поєднувати загальнодоступну частину, доступну гостям, персональну частину для зареєстрованих користувачів і адміністративну частину для керування даними. Такий поділ дозволяє чітко визначити ролі користувачів і межі їхніх можливостей у системі.

Гість повинен мати можливість переглядати каталог фільмів і серіалів, відкривати сторінки окремих фільмів, ознайомлюватися з описом, жанрами, рейтингом, постером, трейлером і посиланнями на зовнішні сервіси перегляду. Для гостя система виконує роль інформаційного каталогу, який дозволяє швидко знайти фільм і зрозуміти, чи варто переходити до реєстрації.

Зареєстрований користувач повинен отримати доступ до персональних функцій. До них належать оцінювання фільмів у зірках, додавання коментарів, створення списків перегляду, позначення фільмів як переглянутих або нецікавих, використання підбірки фільмів і випадкового вибору. Усі ці дії мають зберігатися у базі даних і бути пов'язаними з конкретним користувачем.

Окремою задачею є реалізація ручного додавання фільмів користувачем. Така функція потрібна для випадків, коли фільм відсутній у загальному каталозі, але користувач хоче зберегти його у власному списку. При цьому вручну доданий запис не повинен автоматично ставати публічним для всіх користувачів. Він має залишатися приватним, доки адміністратор не вирішить додати його до загального каталогу.

Адміністратор повинен мати доступ до панелі керування, де відображаються користувачі, їхні дії, коментарі, доданий контент і фільми загального каталогу. Через адміністративну панель має здійснюватися контроль якості даних, додавання нових фільмів до публічного каталогу та видалення некоректних записів. Наявність адміністративної ролі дозволяє підтримувати систему у впорядкованому стані.

Межі системи визначено з урахуванням реалістичності бакалаврського проєкту. MovieMatch не реалізує оплату, ліцензійний перегляд відео всередині сайту та повну інтеграцію зі стримінговими платформами. Натомість система надає посилання на зовнішні сервіси перегляду або інформаційні сторінки фільмів. Такий підхід дозволяє зосередитися на основній темі роботи — рекомендаційному виборі фільмів, персональних списках, оцінках і збереженні даних у PostgreSQL.

|      |      |          |        |      |                         |      |
|------|------|----------|--------|------|-------------------------|------|
|      |      |          |        |      | РБ.ІП - 11.00.00.000 ПЗ | Арк. |
| Змн. | Арк. | № докум. | Підпис | Дата |                         | 36   |

Технічною задачею є створення Rails-застосунку, який використовує архітектурний шаблон MVC. Моделі повинні описувати основні сутності системи: користувачів, фільми, оцінки, коментарі, списки та реакції. Контролери мають обробляти запити, пов'язані з каталогом, пошуком, фільтрацією, профілем, авторизацією, коментарями, оцінками, підбіркою фільмів і адміністративною панеллю. Представлення повинні формувати зрозумілий і адаптивний інтерфейс. Головна сторінка MovieMatch з темним меню і кнопками «Рандом», «Каталог», «Підбірка фільмів» на рисунку 2.2 та 2.3.

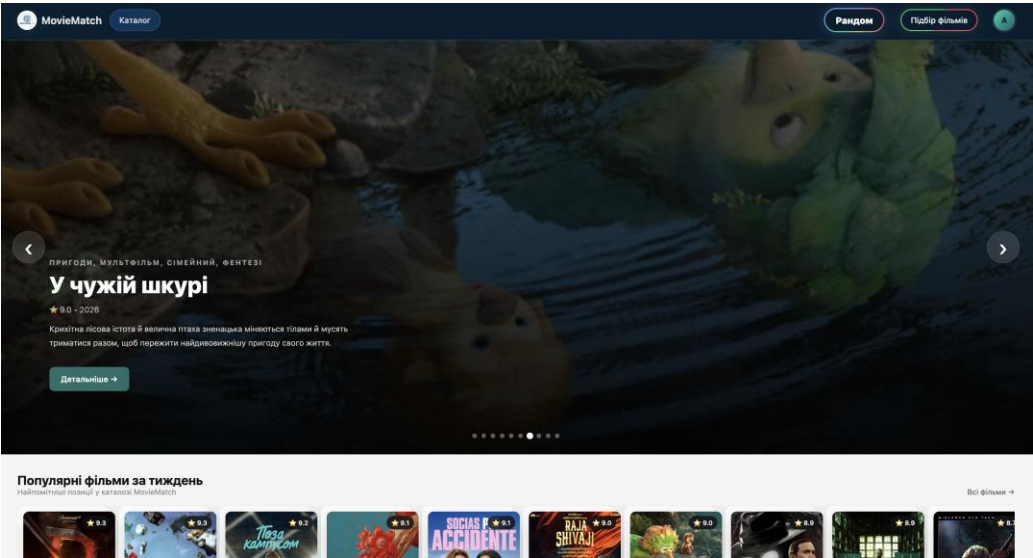


Рисунок 2.2 – Головна сторінка MovieMatch

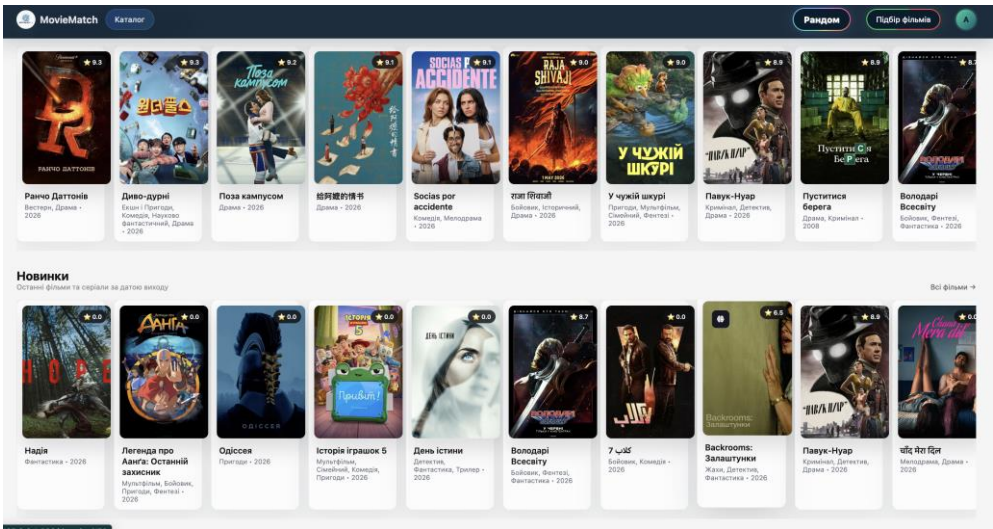


Рисунок 2.3 – Продовження Головної сторінки

База даних PostgreSQL повинна забезпечувати збереження як загальних даних каталогу, так і персональних дій користувача. Для цього необхідно передбачити зв'язки між таблицями, які дозволяють коректно прив'язувати оцінку, коментар або реакцію до конкретного користувача і конкретного фільму. Також необхідно врахувати унікальність певних дій, наприклад, один користувач не повинен мати кілька оцінок для одного й того самого фільму.

Під час проєктування також необхідно врахувати вимогу до доступності системи з різних пристроїв. Користувач може відкривати сайт з ноутбука, стаціонарного комп'ютера або мобільного телефона, тому інтерфейс повинен бути адаптивним. Особливо це стосується сторінки підбору фільмів, де користувач взаємодіє з карткою через свайпи або кнопки дій. Неправильне розміщення елементів може ускладнити використання сайту на малому екрані.

Постановка задачі також передбачає розгортання системи на сервері. Це необхідно для перевірки працездатності не лише в локальному середовищі, а й у production-умовах. На сервері застосунок повинен працювати разом із PostgreSQL-базою, приймати HTTP-запити та бути доступним за публічною IP-адресою. Така перевірка підтверджує, що розроблений застосунок можна використовувати з інших пристроїв, а не тільки на комп'ютері розробника.

Для підтвердження виконання задачі необхідно провести тестування основних сценаріїв. Перевірці підлягають авторизація, відкриття каталогу, пошук, фільтрація, оцінювання, коментування, додавання фільмів до списків, свайп-підбірка, рандомайзер, ручне додавання фільму та робота адміністративної панелі. Результати тестування повинні показати, що система відповідає поставленим вимогам і коректно працює з базою даних.

Очікуваним результатом є працездатний веб-застосунок із PostgreSQL-базою, який можна запускати локально та на сервері. Результат має демонструвати повний цикл інженерії програмного забезпечення: аналіз вимог, проєктування архітектури, створення моделі даних, реалізацію основних функцій, тестування та розгортання. У підсумку MovieMatch повинен працювати

|      |      |          |        |      |                         |      |
|------|------|----------|--------|------|-------------------------|------|
|      |      |          |        |      | РБ.ІП - 11.00.00.000 ПЗ | Арк. |
| Змн. | Арк. | № докум. | Підпис | Дата |                         | 38   |

як рекомендаційно-каталогова система, що допомагає користувачу обирати фільми та зберігати власні рішення. Каталог фільмів із лівою панеллю «Де подивитися» та сіткою фільмів справа на рисунку 2.4.

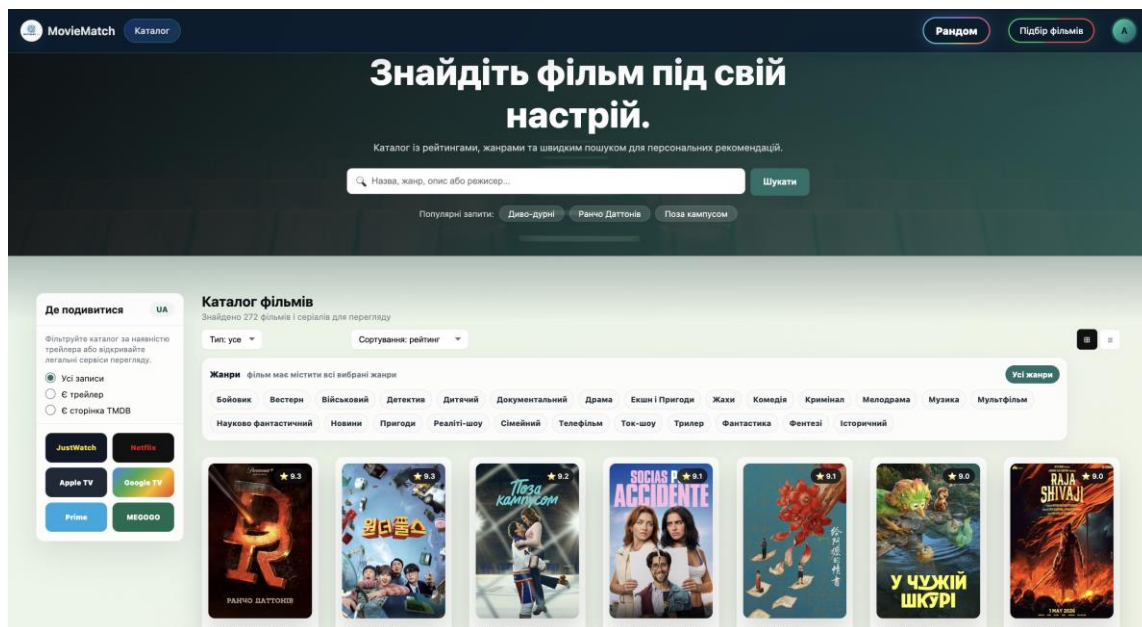


Рисунок 2.4 – Каталог фільмів

## 2.4. Висновки по розділу

У другому розділі визначено цільову аудиторію MovieMatch, основні сценарії взаємодії із системою, функціональні та нефункціональні вимоги, а також постановку задачі проєктування. Розгляд цільової аудиторії показав, що система повинна бути зручною для гостей, зареєстрованих користувачів і адміністратора. Кожна з цих ролей має власний набір дій, тому поділ користувачів є важливим для подальшого проєктування.

Функціональні вимоги охоплюють основні можливості MovieMatch: реєстрацію, вхід, перегляд каталогу, пошук, фільтрацію, оцінювання, коментарі, персональні списки, підбірку фільмів, рандомайзер і адміністративну панель.

|      |      |          |        |      |                         |      |
|------|------|----------|--------|------|-------------------------|------|
|      |      |          |        |      | РБ.ІП - 11.00.00.000 ПЗ | Арк. |
| Змн. | Арк. | № докум. | Підпис | Дата |                         | 39   |

## РОЗДІЛ 3. ПРОЄКТУВАННЯ СИСТЕМИ РЕКОМЕНДАЦІЙ ВИБОРУ ФІЛЬМІВ MOVIEMATCH

### 3.1. Архітектура застосунку MovieMatch

MovieMatch реалізовано як монолітний веб-застосунок Ruby on Rails. Для бакалаврського проєкту монолітна архітектура є обґрунтованою, оскільки вона дозволяє зосередити бізнес-логіку, представлення, маршрутизацію і роботу з базою даних в одному репозиторії. На відміну від мікросервісів, такий підхід не створює зайвої складності розгортання, мережевої взаємодії та синхронізації між сервісами [11].

Архітектурний шаблон MVC розділяє відповідальності між моделями, контролерами та представленнями, зображено на рисунку 3.1. Моделі описують сутності домену і правила зв'язків, контролери приймають HTTP-запити та готують дані для відображення, а представлення формують HTML-інтерфейс [7]. У MovieMatch така структура використовується для фільмів, користувачів, оцінок, коментарів, реакцій, списків перегляду, колекцій і адміністративної частини.

У розгортанні система працює на Linux-сервері. Запити приймає Nginx, який проксує їх до Rails-сервера Puma. Rails-застосунок зберігає дані в PostgreSQL. Така схема є типовою для production-запуску Rails і дозволяє відокремити публічний вебсервер від застосункового процесу.

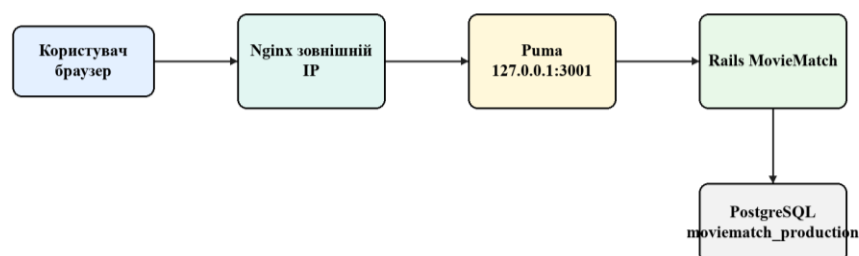


Рисунок 3.1 – Узагальнена архітектура розгортання MovieMatch

Під час розгортання MovieMatch було враховано, що на одному сервері може працювати кілька сервісів. Тому застосунок винесено на окремий порт, що дозволяє запускати його паралельно з іншими проєктами та не порушувати їхню роботу. Така організація є важливою для практичного використання, оскільки сервер може одночасно обслуговувати кілька веб-застосунків або допоміжних сервісів [16].

Архітектура MovieMatch побудована як цілісний монолітний Rails-застосунок із PostgreSQL-базою даних, MVC-структурою, окремими контролерами для основних функцій і серверною схемою Nginx — Puma — Rails — PostgreSQL. Такий підхід відповідає масштабу бакалаврської роботи, забезпечує зрозумілу організацію коду, підтримує основні користувацькі сценарії та дозволяє розгорнути систему на сервері для доступу з інших пристроїв.

У межах проєктування архітектури важливо було визначити, як саме окремі частини системи взаємодіють між собою. MovieMatch містить кілька основних функціональних блоків: каталог фільмів і серіалів, модуль авторизації, профіль користувача, систему оцінювання, коментарі, персональні списки, підбірку фільмів, рандомайзер та адміністративну панель. Усі ці блоки працюють із єдиною базою даних, тому монолітна структура є зручною для реалізації та підтримки такого застосунку.

Вибір Ruby on Rails обґрунтований тим, що цей фреймворк уже містить багато вбудованих механізмів, потрібних для створення веб-системи. Rails забезпечує маршрутизацію запитів, роботу з контролерами, шаблони представлення, взаємодію з базою даних через Active Record, підтримку міграцій і зручну організацію структури проєкту. Завдяки цьому значна частина типових технічних задач не реалізується вручну, а виконується засобами фреймворку.

Архітектура MVC дозволяє розділити відповідальність між основними частинами програми. Модель відповідає за дані та правила роботи з ними, контролер обробляє запити користувача, а представлення формує сторінку, яку бачить користувач у браузері. Для MovieMatch це особливо важливо, оскільки

одна і та сама сутність, наприклад фільм, використовується в різних сценаріях: у каталозі, на сторінці фільму, у профілі користувача, у підбірці та в адміністративній панелі.

На рівні контролерів система розподілена за основними сценаріями. Контролер фільмів відповідає за каталог, пошук, фільтрацію, сторінки окремих фільмів і ручне додавання власного фільму. Контролер користувачів формує профіль і персональні списки. Контролери оцінок, коментарів і реакцій обробляють дії користувача щодо конкретного фільму. Окремо реалізовано контролери для рандомайзера, підбірки фільмів і адміністративної частини.

Важливою частиною архітектури є маршрутизація. Саме маршрути визначають, яка URL-адреса відповідає за конкретну дію в системі. Наприклад, окремі маршрути використовуються для відкриття каталогу, сторінки фільму, профілю користувача, публічної підбірки, сторінки свайпів і адміністративної панелі. Така структура робить застосунок зрозумілим як для користувача, так і для подальшої підтримки коду.

У production-середовищі MovieMatch працює за схемою Nginx — Puma — Rails — PostgreSQL. Nginx приймає зовнішні HTTP-запити та передає їх до Puma. Puma запускає Rails-застосунок і обробляє запити на рівні серверної логіки. Rails отримує або змінює дані через PostgreSQL і повертає сформовану HTML-сторінку користувачу. Така схема є типовою для Rails-застосунків і дозволяє відокремити вебсервер від процесу застосунку.

Окремий порт для MovieMatch було використано для того, щоб застосунок міг працювати на одному сервері паралельно з іншими сервісами. Це важливо для практичного розгортання, оскільки сервер може одночасно обслуговувати кілька веб-застосунків або допоміжних процесів. Такий підхід дозволяє не порушувати роботу інших проєктів і водночас забезпечити доступ до MovieMatch з інших пристроїв.

Таким чином, архітектура MovieMatch побудована як цілісний Rails-застосунок, у якому всі основні функції пов'язані через єдину базу даних і

|      |      |          |        |      |                         |      |
|------|------|----------|--------|------|-------------------------|------|
|      |      |          |        |      | РБ.ІП - 11.00.00.000 ПЗ | Арк. |
| Змн. | Арк. | № докум. | Підпис | Дата |                         | 42   |

спільну MVC-структуру. Це забезпечує зрозумілу організацію коду, спрощує реалізацію користувацьких сценаріїв і дозволяє розгорнути систему на сервері для практичного використання.

### 3.2. Модель даних PostgreSQL

Модель даних побудовано навколо сутностей User, Movie, Watchlist, Rating, Comment і MovieReaction. Сутність User зберігає облікові дані, ім'я, токен для поширення підбірки та ознаку адміністратора [28]. Сутність Movie містить метадані фільму або серіалу: назву, рейтинг, жанри, рік, постер, опис, тривалість, дату релізу, TMDb-ідентифікатор, режисера, тип медіа, трейлер, обмеження за віком та ознаку ручного додавання.

Зв'язки реалізовано через зовнішні ключі. Коментар належить користувачу і фільму. Оцінка належить користувачу і фільму та має унікальний індекс за парою user\_id/movie\_id, щоб один користувач не створював кілька оцінок для одного фільму. MovieReaction також має унікальний індекс за парою user\_id/movie\_id і статус, що дозволяє переносити фільм між списками «хочу подивитись», «бачив/ла» і «не подобається» [15].

Таблиця 3.1

Основні таблиці бази даних MovieMatch

| Таблиця         | Призначення                            | Ключові поля                                                   |
|-----------------|----------------------------------------|----------------------------------------------------------------|
| Users           | Облікові записи користувачів           | email, password_digest, name, share_token, admin               |
| Movies          | Фільми та серіали каталогу             | title, genre, media_type, trailer_key, tmdb_id, manually_added |
| Ratings         | Персональні оцінки                     | user_id, movie_id, score                                       |
| comments        | Коментарі до фільмів                   | user_id, movie_id, body                                        |
| watchlists      | Зв'язок користувача з фільмом у списку | user_id, movie_id                                              |
| movie_reactions | Статус взаємодії користувача з фільмом | user_id, movie_id, status                                      |

Використання PostgreSQL є обґрунтованим через потребу у зв'язках, транзакційності, індексах і стабільності. Структура таблиць залишає можливість розширювати пошук засобами PostgreSQL, формувати популярні підбірки та накопичувати історію взаємодій без зміни базової архітектури [9; 10].

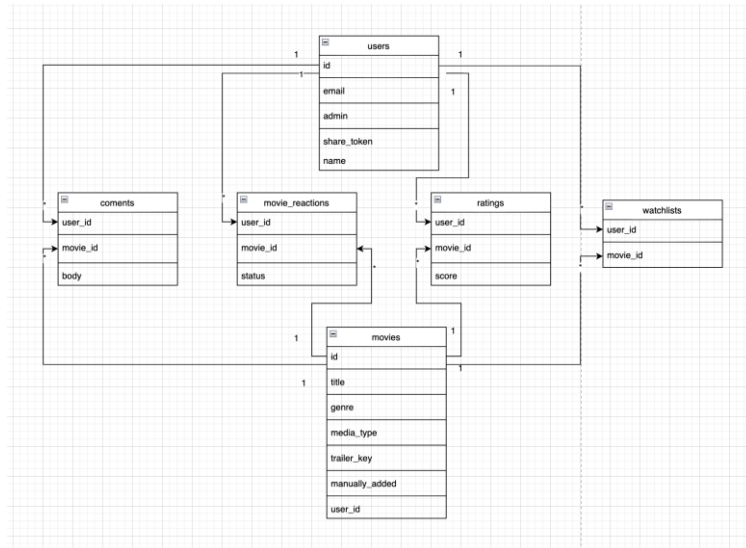


Рисунок 3.2 – ER-модель бази даних MovieMatch

Окремою особливістю є ручне додавання фільмів. Якщо користувач додає власний фільм, такий запис має містити зв'язок із користувачем через `user_id` і ознаку `manually_added`. Це дозволяє відокремити приватні записи користувача від загального каталогу. Такий підхід важливий для збереження впорядкованості публічної частини системи.

Використання PostgreSQL у цьому проєкті є доцільним через потребу в надійних зв'язках між таблицями, підтримці індексів, обмежень цілісності та стабільному збереженні даних. Реляційна модель добре підходить для MovieMatch, оскільки всі основні сутності мають чіткі зв'язки: користувачі створюють оцінки, коментарі й реакції, а фільми є об'єктами цих дій.

Таким чином, модель даних MovieMatch забезпечує збереження загальної інформації про медіаконтент і персональної активності користувачів. Вона дозволяє реалізувати каталог, профіль, оцінки, коментарі, списки, підбірку

фільмів і адміністративний контроль. Саме ця модель є основою для подальшої реалізації функцій системи.

### 3.3. Проєктування інтерфейсу та користувацької взаємодії

Інтерфейс MovieMatch спроектовано так, щоб перший екран одразу вів користувача до дії: переглянути каталог, скористатися рандомайзером або перейти до підбірки фільмів [24]. Верхнє меню виконано у темній візуальній стилістиці, оскільки тема кіно природно асоціюється з затемненим переглядом, постерами та контрастними акцентами.

Каталог містить компактну панель «Де подивитися» зліва і основну область карток фільмів справа. Така композиція відповідає очікуванням користувача: фільтри не займають увесь екран, а основний простір залишається для контенту. Фільтри за жанрами виконано у вигляді активних кнопок, що дозволяє вибирати кілька жанрів одночасно. Логіка фільтрації працює за принципом перетину: якщо вибрано «драма» і «комедія», система шукає фільми, які мають обидва жанри.

Для мобільного сценарію важливою є підбірка фільмів. Кнопки дій розміщено нижче постера, щоб вони не перекривали зображення. Анімація свайпу має бути плавною, оскільки різке переміщення картки створює відчуття незавершеності. Рандомайзер реалізовано як окрему виразну кнопку: після натискання користувач бачить коротке завантажувальне вікно, після чого система відкриває випадковий фільм [37].

Проєктування інтерфейсу виконано з урахуванням основних сценаріїв користувача: першого переходу на сайт, перегляду каталогу, відкриття сторінки фільму, роботи з профілем і мобільної підбірки. Узагальнені схеми екранів наведено на рисунках 3.3 і 3.4.

|      |      |          |        |      |                         |      |
|------|------|----------|--------|------|-------------------------|------|
|      |      |          |        |      | РБ.ІП - 11.00.00.000 ПЗ | Арк. |
|      |      |          |        |      |                         | 45   |
| Змн. | Арк. | № докум. | Підпис | Дата |                         |      |

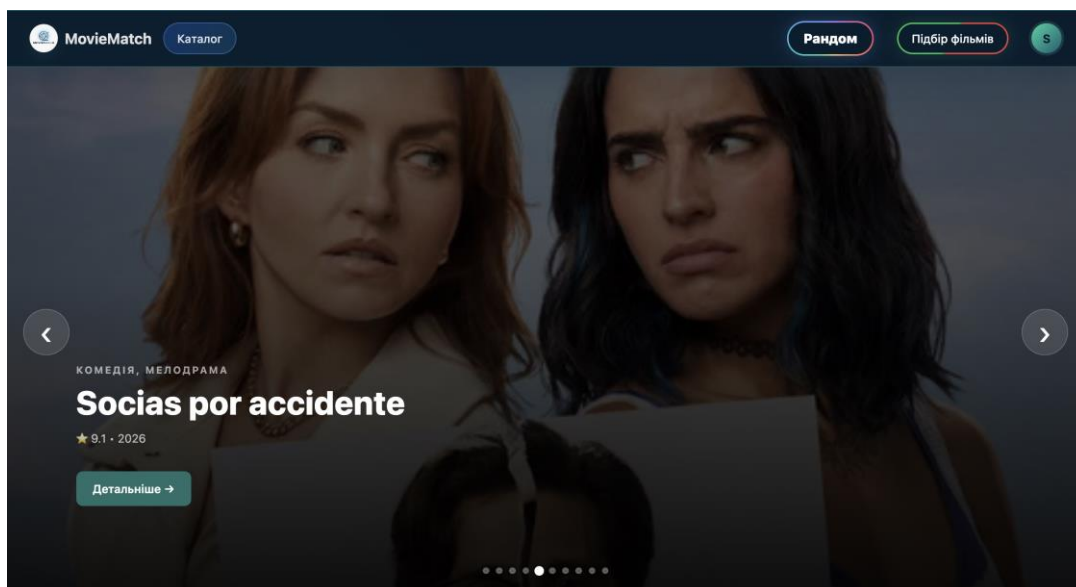


Рисунок 3.3 – Головна сторінка MovieMatch

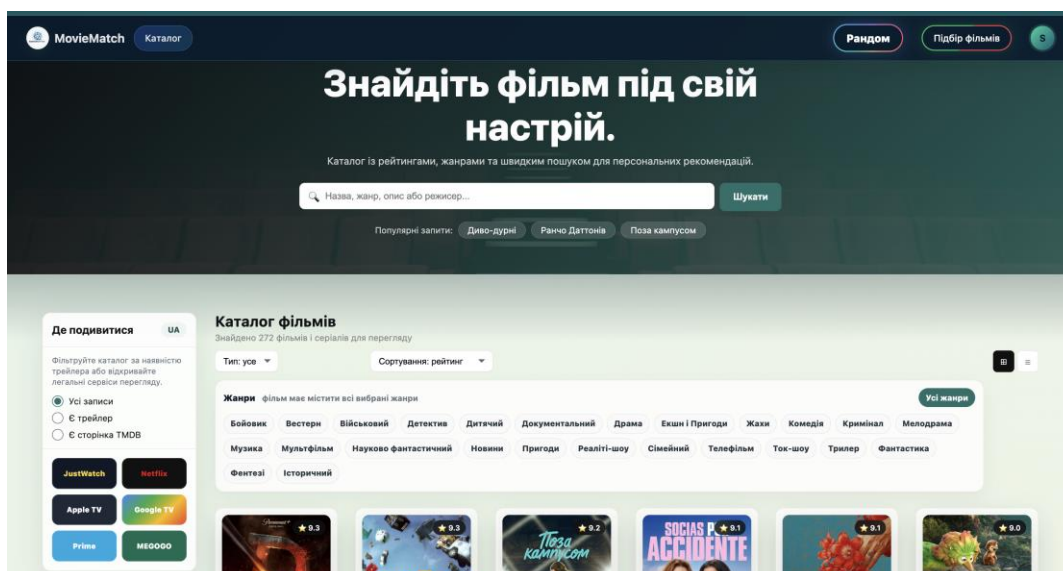


Рисунок 3.4 – Каталог фільмів із панеллю перегляду та жанровими фільтрами

Профіль користувача проєктується як персональний простір. У ньому повинні відображатися списки фільмів за статусами: «хочу подивитись», «бачив/ла» та «не подобається». Така структура є зрозумілою для користувача, оскільки вона відповідає реальному процесу вибору контенту. Користувач може спочатку зберегти фільм, після перегляду перенести його до переглянутих або видалити зі списку.

|      |      |          |        |      |                         |      |
|------|------|----------|--------|------|-------------------------|------|
|      |      |          |        |      | РБ.ІП - 11.00.00.000 ПЗ | Арк. |
|      |      |          |        |      |                         | 46   |
| Змн. | Арк. | № докум. | Підпис | Дата |                         |      |

Особливе значення має мобільний сценарій. Підбірка фільмів у форматі свайпів повинна бути зручною саме на телефоні, тому кнопки дій розміщуються нижче постера і не перекривають зображення. Користувач має швидко прийняти рішення: додати фільм до бажаних, відхилити або позначити як переглянутий. Такий формат зменшує кількість дій і робить вибір більш природним для мобільного пристрою.

Рандомайзер у системі виконує роль швидкого способу вибору фільму. Його кнопка має бути помітною, оскільки це один із додаткових сценаріїв взаємодії. Після натискання користувач бачить коротке повідомлення про підбір фільму, а потім система відкриває випадкову сторінку. Такий сценарій додає елемент несподіваності й допомагає користувачу тоді, коли він не хоче довго переглядати каталог.

Таким чином, інтерфейс MovieMatch спроектовано навколо ключових сценаріїв: перегляд каталогу, пошук, фільтрація, відкриття сторінки фільму, оцінювання, робота з профілем, мобільна підбірка та адміністрування.

Структура бази даних MovieMatch охоплює таблиці users, movies, ratings, comments, watchlists і movie\_reactions. ER-модель наведено в основній частині роботи на рисунку 3.5

moviematch\_development=# \d movies

| Column             | Type                           | Collation | Nullable | Default           |
|--------------------|--------------------------------|-----------|----------|-------------------|
| id                 | bigint                         |           | not null | nextval('movies_i |
| d_seq'::regclass)  |                                |           |          |                   |
| title              | character varying              |           |          |                   |
| rating             | double precision               |           |          |                   |
| created_at         | timestamp(6) without time zone |           | not null |                   |
| updated_at         | timestamp(6) without time zone |           | not null |                   |
| genre              | character varying              |           |          |                   |
| release_year       | integer                        |           |          |                   |
| poster_url         | character varying              |           |          |                   |
| description        | text                           |           |          |                   |
| runtime            | integer                        |           |          |                   |
| release_date       | date                           |           |          |                   |
| tmdb_poster_path   | character varying              |           |          |                   |
| tmdb_id            | integer                        |           |          |                   |
| director           | character varying              |           |          |                   |
| media_type         | character varying              |           | not null | 'movie'::characte |
| r_varying          |                                |           |          |                   |
| original_title     | character varying              |           |          |                   |
| original_language  | character varying              |           |          |                   |
| tagline            | character varying              |           |          |                   |
| status             | character varying              |           |          |                   |
| budget             | bigint                         |           |          |                   |
| revenue            | bigint                         |           |          |                   |
| tmdb_backdrop_path | character varying              |           |          |                   |
| trailer_key        | character varying              |           |          |                   |
| content_rating     | character varying              |           |          |                   |
| user_id            | bigint                         |           | not null | false             |
| manually_added     | boolean                        |           |          |                   |

Indexes:

- "movies\_pkey" PRIMARY KEY, btree (id)
- "index\_movies\_on\_manually\_added" btree (manually\_added)
- "index\_movies\_on\_media\_type" btree (media\_type)
- "index\_movies\_on\_media\_type\_and\_tmdb\_id" btree (media\_type, tmdb\_id)
- "index\_movies\_on\_user\_id" btree (user\_id)

Foreign-key constraints:

- "fk\_rails\_2c7f519372" FOREIGN KEY (user\_id) REFERENCES users(id)

Referenced by:

- "fk\_rails\_1del911646" FOREIGN KEY (movie\_id) REFERENCES movie

TABLE "watchlists" CONSTRAINT "fk\_rails\_1del911646" FOREIGN KEY (movie\_id) REFERENCES movie

Рисунок 3.5 – Структура таблиці movies

| Column                                                                                              | Type                           | Collation | Nullable | Default              |
|-----------------------------------------------------------------------------------------------------|--------------------------------|-----------|----------|----------------------|
| id                                                                                                  | bigint                         |           | not null | nextval('users_id_se |
| email                                                                                               | character varying              |           |          |                      |
| password_digest                                                                                     | character varying              |           |          |                      |
| created_at                                                                                          | timestamp(6) without time zone |           | not null |                      |
| updated_at                                                                                          | timestamp(6) without time zone |           | not null |                      |
| name                                                                                                | character varying              |           |          |                      |
| share_token                                                                                         | character varying              |           | not null |                      |
| admin                                                                                               | boolean                        |           | not null | false                |
| Indexes:                                                                                            |                                |           |          |                      |
| "users_pkey" PRIMARY KEY, btree (id)                                                                |                                |           |          |                      |
| "index_users_on_admin" btree (admin)                                                                |                                |           |          |                      |
| "index_users_on_share_token" UNIQUE, btree (share_token)                                            |                                |           |          |                      |
| Referenced by:                                                                                      |                                |           |          |                      |
| TABLE "comments" CONSTRAINT "fk_rails_03de2dc08c" FOREIGN KEY (user_id) REFERENCES users(id)        |                                |           |          |                      |
| TABLE "watchlists" CONSTRAINT "fk_rails_0dcla4cbcb" FOREIGN KEY (user_id) REFERENCES users(id)      |                                |           |          |                      |
| TABLE "movies" CONSTRAINT "fk_rails_2c7f519372" FOREIGN KEY (user_id) REFERENCES users(id)          |                                |           |          |                      |
| TABLE "movie_reactions" CONSTRAINT "fk_rails_943c14353d" FOREIGN KEY (user_id) REFERENCES users(id) |                                |           |          |                      |
| TABLE "ratings" CONSTRAINT "fk_rails_a7dfeb9f5f" FOREIGN KEY (user_id) REFERENCES users(id)         |                                |           |          |                      |

Рисунок 3.6 – Структура таблиці users

| Column                                                                    | Type                           | Collation | Nullable | Default                   |
|---------------------------------------------------------------------------|--------------------------------|-----------|----------|---------------------------|
| id                                                                        | bigint                         |           | not null | nextval('ratings_id_seq': |
| user_id                                                                   | bigint                         |           | not null |                           |
| movie_id                                                                  | bigint                         |           | not null |                           |
| score                                                                     | integer                        |           | not null |                           |
| created_at                                                                | timestamp(6) without time zone |           | not null |                           |
| updated_at                                                                | timestamp(6) without time zone |           | not null |                           |
| Indexes:                                                                  |                                |           |          |                           |
| "ratings_pkey" PRIMARY KEY, btree (id)                                    |                                |           |          |                           |
| "index_ratings_on_movie_id" btree (movie_id)                              |                                |           |          |                           |
| "index_ratings_on_user_id" btree (user_id)                                |                                |           |          |                           |
| "index_ratings_on_user_id_and_movie_id" UNIQUE, btree (user_id, movie_id) |                                |           |          |                           |
| Foreign-key constraints:                                                  |                                |           |          |                           |
| "fk_rails_62d9686c7c" FOREIGN KEY (movie_id) REFERENCES movies(id)        |                                |           |          |                           |
| "fk_rails_a7dfeb9f5f" FOREIGN KEY (user_id) REFERENCES users(id)          |                                |           |          |                           |

Рисунок 3.7 – Структура таблиці ratings

### 3.4. Висновки по розділу

У третьому розділі спроектовано архітектуру MovieMatch, модель даних PostgreSQL та логіку користувацької взаємодії. Обрана монолітна Rails-архітектура відповідає масштабу бакалаврського проєкту, а реляційна модель забезпечує надійне збереження персональних дій користувача. Отримані результати проєктування створюють основу для реалізації системи. На основі визначеної архітектури, моделі даних і користувацького інтерфейсу у наступному розділі описується програмна реалізація MovieMatch, структура коду, основні контролери, моделі, представлення та механізми роботи з даними.

## РОЗДІЛ 4. РЕАЛІЗАЦІЯ ПРОЄКТУ РЕКОМЕНДАЦІЙ ВИБОРУ ФІЛЬМІВ MOVIEMATCH

### 4.1. Структура програмного коду

Код MovieMatch організовано за типовою структурою Ruby on Rails. Контролери розміщено у каталозі `app/controllers`, моделі – в `app/models`, представлення – в `app/views`, стилі – в `app/assets/stylesheets`. Такий поділ спрощує підтримку, оскільки кожна частина системи має зрозумілу відповідальність [23].

Основними контролерами є `MoviesController`, `UsersController`, `SessionsController`, `WatchlistsController`, `RatingsController`, `CommentsController`, `SwipesController`, `RandomizerController`, `CollectionsController` та контролери простору `admin` в таблиці 4.1. `MoviesController` відповідає за каталог, пошук, фільтрацію, сторінку фільму та створення власного фільму користувача. `UsersController` формує профіль користувача і персональні списки. `Admin::DashboardController` дає адміністратору огляд користувачів, контенту та дій.

Окрему роль у структурі коду відіграють маршрути, які описані у файлі `config/routes.rb`. Вони визначають, які URL-адреси відповідають за відкриття каталогу, сторінки фільму, профілю користувача, підбірки фільмів, публічної колекції та адміністративної панелі [31]. Завдяки маршрутизації Rails кожна дія користувача спрямовується до відповідного контролера, а структура сайту залишається зрозумілою і впорядкованою. Це спрощує як навігацію для користувача, так і подальшу підтримку програмного коду.

У моделях визначено зв'язки між сутностями. Наприклад, `Movie` має багато коментарів, оцінок, реакцій і записів `watchlist`, а `User` має пов'язані оцінки, коментарі та реакції. Такі зв'язки дозволяють писати код на рівні доменних об'єктів, не формуючи вручну кожен SQL-запит.

Фрагменти структури бази даних і приклади програмного коду винесено до додатків Б та В. Це дає змогу не перевантажувати основний текст

|      |      |          |        |      |                         |      |
|------|------|----------|--------|------|-------------------------|------|
|      |      |          |        |      | РБ.ІП - 11.00.00.000 ПЗ | Арк. |
| Змн. | Арк. | № докум. | Підпис | Дата |                         | 49   |

великими технічними вставками, але зберегти доказову базу реалізації.

Таблиця 4.1

Основні компоненти програмної реалізації

| Компонент               | Функція                                                              |
|-------------------------|----------------------------------------------------------------------|
| MoviesController        | Каталог, пошук, фільтри, сторінки фільмів, створення власного фільму |
| RatingsController       | Збереження персональної оцінки користувача                           |
| CommentsController      | Створення і видалення коментарів                                     |
| SwipesController        | Сценарій підбору фільмів за свайпами                                 |
| RandomizerController    | Вибір випадкового фільму                                             |
| Admin::MoviesController | Додавання і видалення фільмів у загальному каталозі                  |

Під час організації структури коду важливо було зберегти логіку, характерну для Ruby on Rails, щоб кожна частина застосунку мала чітке місце у проєкті. Контролери відповідають за приймання запитів і підготовку даних, моделі описують сутності та зв'язки між ними, а представлення формують HTML-сторінки для користувача. Такий поділ дозволяє не змішувати бізнес-логіку, роботу з базою даних і візуальне відображення.

У проєкті MovieMatch окремо виділено користувацьку частину та адміністративний простір. Користувацькі контролери забезпечують роботу з каталогом, профілем, оцінками, коментарями, підбіркою фільмів і персональними списками. Адміністративні контролери винесено в окремий простір admin, що дозволяє відокремити звичайні сценарії користувача від сценаріїв керування системою. Це робить структуру проєкту зрозумілішою та безпечнішою.

Важливим елементом структури є також наявність окремих представлень для різних сценаріїв. Наприклад, сторінка каталогу, сторінка фільму, профіль користувача, сторінка входу, підбірка фільмів і адміністративна

панель мають власні шаблони. Це дозволяє змінювати вигляд окремої сторінки без порушення роботи інших частин застосунку. Такий підхід спрощує подальше редагування інтерфейсу і підтримку коду.

## 4.2. Реалізація основних функцій

Авторизація реалізована через сесію користувача і `password_digest`. Після входу користувач отримує доступ до профілю, оцінювання, коментування, ручного додавання фільмів та персональних списків. Адміністратор визначається булевим полем `admin` у таблиці `users`. Для демонстрації створено адміністративний обліковий запис `admin@gmail.com`.

Каталог показує тільки публічні фільми, тобто записи, де `manually_added` дорівнює `false`. Це принципово важливо, оскільки вручну додані користувачем фільми не повинні потрапляти до загального каталогу і ставати видимими для інших користувачів. Такий поділ підтримує приватність персонального контенту.

Оцінювання реалізовано через таблицю `ratings`, де `score` зберігається як число від 1 до 5. На відміну від глобального рейтингу TMDb, ця оцінка належить конкретному користувачу. У профілі розраховується персональна середня оцінка тільки за власними оцінками користувача [39]. Таким чином виправлено проблему, коли користувач бачив чужий середній рейтинг замість свого ставлення до фільму.

Коментарі зберігаються у таблиці `comments` і прив'язані до користувача та фільму. На сторінці фільму користувач залишає власну думку, а адміністратор у панелі бачить активність і може контролювати створений користувачами контент.

Публічне поширення підбірки реалізовано через `share_token` користувача. Замість відкриття всього профілю система формує сторінку колекції за токеном, де інший користувач може побачити добірку без входу в акаунт. Для окремого фільму використовується звичайне посилання на сторінку фільму, зображено на рис 4.1, 4.2.

|      |      |          |        |      |                         |      |
|------|------|----------|--------|------|-------------------------|------|
|      |      |          |        |      | РБ.ІП - 11.00.00.000 ПЗ | Арк. |
| Змн. | Арк. | № докум. | Підпис | Дата |                         | 51   |



Рисунок 4.1 – Сторінка фільму MovieMatch

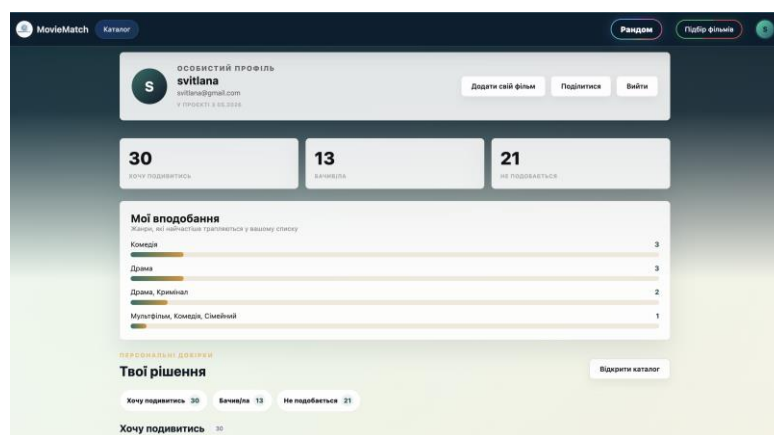


Рисунок 4.2 – Профіль користувача та об'єднаний список

Однією з ключових функцій системи є робота з каталогом фільмів і серіалів. Каталог відображає лише публічні записи, доступні всім користувачам, тоді як вручну додані фільми залишаються приватними для власника. Такий поділ реалізовано для того, щоб користувацький контент не змішувався із загальним каталогом і не впливав на досвід інших користувачів без адміністративного підтвердження.

Реалізація персональних списків дозволяє користувачу впорядковувати власний досвід перегляду. Фільм може бути доданий до списку «хочу подивитись», перенесений до «бачив/ла» або позначений як «не подобається». Для цього використовується окрема логіка реакцій, яка зберігає статус взаємодії

|      |      |          |        |      |                         |      |
|------|------|----------|--------|------|-------------------------|------|
|      |      |          |        |      | РБ.ІП - 11.00.00.000 ПЗ | Арк. |
|      |      |          |        |      |                         | 52   |
| Змн. | Арк. | № докум. | Підпис | Дата |                         |      |

користувача з конкретним фільмом. Завдяки цьому профіль користувача відображає не загальні дані системи, а саме його персональні рішення.

Функція поширення підбірки реалізована через спеціальний токен користувача. Це дозволяє створити публічне посилання на добірку без відкриття всього профілю і без потреби входу в акаунт для перегляду. Такий підхід є зручним для демонстрації списку фільмів іншим людям, але водночас не розкриває зайві персональні дані користувача.

#### 4.3. Алгоритми пошуку, фільтрації та персональних списків

Пошук у MovieMatch нормалізує запит користувача до нижнього регістру і порівнює його з кількома полями фільму: title, original\_title, description, genre та director. Такий підхід дозволяє знаходити фільм навіть тоді, коли користувач вводить назву з маленької літери або шукає не за повним збігом [35].

Алгоритм пошуку побудовано з урахуванням типових помилок користувача під час введення запиту. Користувач може вводити назву з маленької літери, не повністю або з іншим регістром, тому система нормалізує пошуковий рядок і порівнює його з кількома полями фільму. Це підвищує ймовірність знаходження потрібного запису навіть тоді, коли запит не повністю збігається з назвою в базі даних.

Фільтрація за жанрами реалізована як перевірка наявності всіх вибраних жанрів у полі genre. Якщо вибрано один жанр, система працює як звичайний фільтр. Якщо вибрано кілька жанрів, результатом є перетин, а не об'єднання [21]. Це відповідає очікуванню користувача: вибір «драма» і «комедія» означає пошук драматичних комедій, а не всіх драм і всіх комедій окремо.

Свайп-механіка використовує статуси реакції. Якщо користувач свайпає вправо, створюється або оновлюється реакція want\_to\_watch. Якщо вліво – not\_interested. Якщо натиснуто «бачив/ла» – watched рисунок 4.3. У профілі ці статуси розкладаються в окремі вкладки, тому користувач одразу бачить, які фільми хоче переглянути, які вже переглянув і які не зацікавили.

|      |      |          |        |      |                         |      |
|------|------|----------|--------|------|-------------------------|------|
|      |      |          |        |      | РБ.ІП - 11.00.00.000 ПЗ | Арк. |
| Змн. | Арк. | № докум. | Підпис | Дата |                         | 53   |

[illegible][illegible]

|      |      |          |        |      |                         |      |
|------|------|----------|--------|------|-------------------------|------|
|      |      |          |        |      | РБ.ІП - 11.00.00.000 ПЗ | Арк. |
|      |      |          |        |      |                         | 54   |
| Змн. | Арк. | № докум. | Підпис | Дата |                         |      |

#### 4.4. Висновки по розділу

У четвертому розділі описано структуру коду та реалізацію ключових функцій MovieMatch. Розроблений застосунок підтримує основні сценарії роботи користувача й адміністратора рисунок 4.4, 4.5 зберігає персональні дії у PostgreSQL та забезпечує практичну реалізацію рекомендаційно-каталогової логіки.

У процесі реалізації особливу увагу приділено персональним діям користувача. Оцінки, коментарі, списки перегляду, реакції та вручну додані фільми зберігаються з прив'язкою до конкретного облікового запису. Це дозволяє системі працювати не просто як загальний каталог, а як персональний інструмент для вибору і впорядкування фільмів. Проект розміщено на GitHub зображення на рисунку 4.6

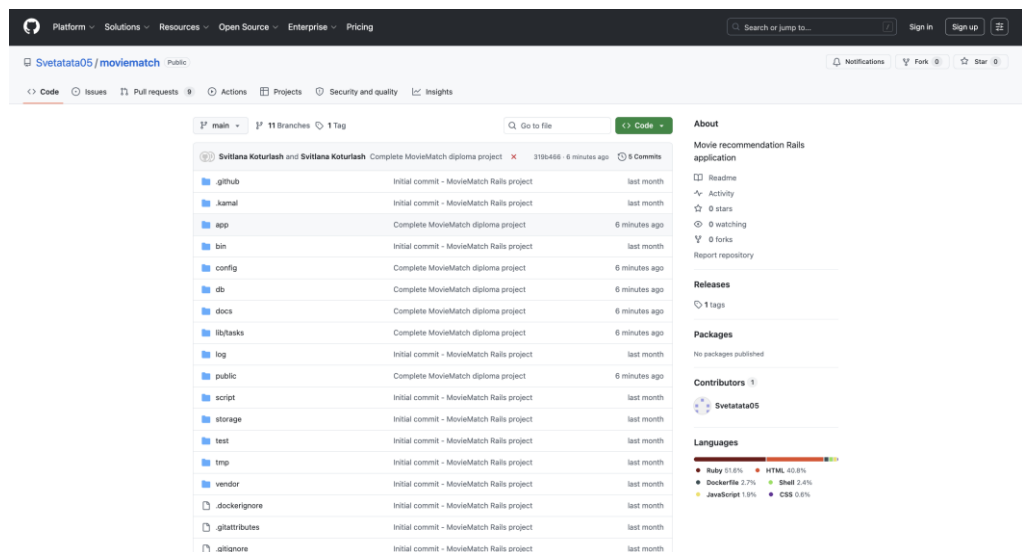


Рисунок 4.6 – Проект на GitHub

Реалізовані функції підтверджують, що MovieMatch відповідає поставленим вимогам. Система підтримує каталог, пошук, фільтрацію, оцінювання, коментарі, підбірку фільмів, рандомайзер, публічне поширення добірки та адміністративне керування. Це створює основу для подальшого тестування працездатності застосунку та перевірки його роботи на сервері.

|      |      |          |        |      |                         |      |
|------|------|----------|--------|------|-------------------------|------|
|      |      |          |        |      | РБ.ІП - 11.00.00.000 ПЗ | Арк. |
| Змн. | Арк. | № докум. | Підпис | Дата |                         | 55   |

## РОЗДІЛ 5. ТЕСТУВАННЯ ТА РОЗГОРТАННЯ ПРОЄКТУ РЕКОМЕНДАЦІЙ ВИБОРУ ФІЛЬМІВ MOVIEMATCH

### 5.1. Стратегія тестування

Тестування MovieMatch проведено за функціональними сценаріями, які відповідають вимогам розділу 2. Основна увага приділялася входу користувача, перегляду каталогу, пошуку, фільтрації, оцінюванню, коментарям, роботі профілю, ручному додаванню фільмів, свайпам, рандомайзеру, адмін-панелі та розгортанню на сервері [14].

Окремо перевірялася стабільність після перенесення бази даних на production-сервер. Під час цього було виявлено дві важливі проблеми: конфлікт CSRF через невідповідність HTTP/HTTPS і дублювання класу MoviesController у service-файлі. Обидві проблеми усунено: для запуску через IP без HTTPS вимкнено `force_ssl/assume_ssl`, а зайвий контролер вилучено з `tmdb_client.rb`.

Перевірка каталогу виконувалася не тільки відкриттям сторінки, а й аналізом HTTP-статусів. Після виправлення сервер повертає 200 OK для `/movies`, а HTML сторінки містить повідомлення «Знайдено 272». Це підтверджує, що каталог завантажується з PostgreSQL і не залежить від поточного доступу до зовнішнього API [37].

Під час інтерпретації результатів тестування використовувалися технічні підходи з офіційної документації Rails, PostgreSQL, Nginx і OWASP; відповідні джерела наведено у списку використаних джерел.

Стратегія тестування була побудована так, щоб перевірити не окремі сторінки ізолювано, а повні користувацькі сценарії. Наприклад, для оцінювання фільму перевірявся не лише факт натискання на зірку, а й збереження оцінки у базі даних, її відображення на сторінці фільму та врахування у профілі користувача. Аналогічно для коментарів перевірялося створення запису, прив'язка до конкретного користувача і фільму, а також відображення коментаря після оновлення сторінки.

|      |      |          |        |      |                         |      |
|------|------|----------|--------|------|-------------------------|------|
|      |      |          |        |      | РБ.ІП - 11.00.00.000 ПЗ | Арк. |
| Змн. | Арк. | № докум. | Підпис | Дата |                         | 56   |

Окремо перевірялися сценарії для різних ролей користувачів. Для гостя важливим було відкриття каталогу та перегляд сторінок фільмів без доступу до персональних дій. Для зареєстрованого користувача перевірялися оцінки, коментарі, списки перегляду, підбірка фільмів, рандомайзер і ручне додавання власного фільму. Для адміністратора перевірявся доступ до панелі керування, перегляд користувачів, коментарів, доданого контенту та керування каталогом.

Також у процесі тестування враховувалися різні ролі користувачів. Для гостя перевірявся доступ до каталогу та сторінок фільмів без можливості виконувати персональні дії. Для зареєстрованого користувача перевірялися оцінки, коментарі, списки перегляду, підбірка фільмів і ручне додавання власного фільму. Для адміністратора окремо перевірявся доступ до панелі керування, перегляд користувачів, коментарів, доданого контенту та можливість керування загальним каталогом.

## 5.2. Результати перевірки функцій

Результати функціонального тестування показали, що основні сценарії користувача виконуються відповідно до поставлених вимог. Каталог відкривається, пошук знаходить фільми незалежно від реєстру, фільтрація працює за вибраними жанрами, а персональні дії користувача зберігаються у відповідних таблицях бази даних. Це підтверджує, що реалізована логіка працює не тільки на рівні інтерфейсу, а й на рівні даних.

Особливу увагу під час перевірки приділено сценаріям, пов'язаним із персональністю інформації. Власна оцінка користувача не змішується із зовнішнім рейтингом, вручну доданий фільм не потрапляє автоматично до загального каталогу, а статуси реакцій правильно розподіляють фільми між вкладками профілю. Це важливо для MovieMatch, оскільки персоналізація є однією з основних особливостей системи.

Адміністративні функції також були перевірені окремо. Адміністратор має доступ до панелі керування, бачить користувачів, коментарі, доданий

|      |      |          |        |      |                         |      |
|------|------|----------|--------|------|-------------------------|------|
|      |      |          |        |      | РБ.ІП - 11.00.00.000 ПЗ | Арк. |
| Змн. | Арк. | № докум. | Підпис | Дата |                         | 57   |

контент і може керувати загальним каталогом. Успішна перевірка цих функцій підтверджує, що система має не лише користувацьку частину, а й механізм супроводу та контролю даних.

Таблиця 5.1

Результати функціонального тестування

| Сценарій                 | Очікуваний результат                    | Фактичний результат                        | Статус  |
|--------------------------|-----------------------------------------|--------------------------------------------|---------|
| Вхід адміністратора      | Перехід на /movies після авторизації    | HTTP 302 на /movies                        | успішно |
| Відкриття каталогу       | Сторінка повертає 200 OK                | Каталог відкривається, знайдено 272 записи | успішно |
| Пошук з маленької літери | Фільм знаходиться незалежно від реєстру | Пошук нормалізує введення                  | успішно |
| Оцінювання у зірках      | Зберігається персональна оцінка         | Оцінка прив'язана до user_id/movie_id      | успішно |
| Ручне додавання фільму   | Фільм видно тільки власнику             | manually_added=true і user_id власника     | успішно |
| Свайп вправо             | Фільм потрапляє у «хочу подивитись»     | Створюється реакція want_to_watch          | успішно |
| Кнопка «бачив/ла»        | Фільм переноситься до переглянутих      | Статус змінюється на watched               | успішно |
| Адмін-панель             | Адмін бачить користувачів і контент     | Доступ за admin=true                       | успішно |

Під час перевірки функцій особливу увагу приділено збереженню даних після виконання користувацьких дій. Для кожного сценарію важливо було підтвердити не лише зміну інтерфейсу, а й коректне створення або оновлення записів у PostgreSQL, таблиця 5.1. Наприклад, після оцінювання фільму перевірялося, що оцінка прив'язана до відповідного user\_id і movie\_id, після свайпу створюється реакція з потрібним статусом, а вручну доданий фільм має ознаку приватного запису та не потрапляє до загального каталогу. Це дозволило переконатися, що система працює не тільки візуально, а й коректно зберігає дані на рівні бази [35].

Результати тестування показали, що основні функції відповідають поставленим вимогам. Виявлені під час розгортання помилки не були пов'язані з логікою користувацьких сценаріїв, а стосувалися production-конфігурації та випадкового дублювання класу. Їх усунення підвищило стабільність системи.

Описані сценарії тестування сформували контрольний набір перевірок для демонстрації працездатності системи.

### 5.3. Розгортання на сервері та оцінювання готовності

MovieMatch розгорнуто на Linux-сервері поряд з іншим веб-сервісом і Telegram-ботом. Щоб не порушити роботу наявного сервісу, MovieMatch винесено на окремий порт 3001. Nginx приймає зовнішні запити на IP-адресі сервера і проксуює їх до Puma, який слухає 127.0.0.1:3001.

База даних production створена в PostgreSQL як окрема база moviematch\_production з окремим користувачем. Локальні дані з 272 фільмами перенесено через dump/restore. Після відновлення додано службові таблиці Rails для cache, queue і cable, оскільки production-режим Rails 8 використовує відповідні компоненти.

Готовність системи оцінено за трьома критеріями: доступність сторінок, коректність ключових сценаріїв і цілісність даних. Доступність підтверджено HTTP-відповідями 200 OK для каталогу. Коректність логіну підтверджено відповіддю 302 на /movies. Цілісність даних підтверджено кількістю 272 публічних фільмів у production-базі. Публічне посилання <http://157.173.100.0:3001>.

Для збереження вихідного коду та демонстрації завершеності проєкту застосунок MovieMatch також розміщено у віддаленому репозиторії GitHub. Це дозволяє зберігати історію змін, контролювати версії програмного коду та за потреби швидко відновити або перенести проєкт в інше середовище [15]. Наявність репозиторію підтверджує, що результат роботи оформлено не лише як розгорнутий веб-застосунок, а й як повноцінний програмний проєкт із

|      |      |          |        |      |                         |      |
|------|------|----------|--------|------|-------------------------|------|
|      |      |          |        |      | РБ.ІП - 11.00.00.000 ПЗ | Арк. |
| Змн. | Арк. | № докум. | Підпис | Дата |                         | 59   |

можливістю подальшого супроводу, перевірки структури коду та повторного розгортання

Розгортання на сервері дозволило перевірити роботу MovieMatch у середовищі, наближеному до реального використання. На відміну від локального запуску, серверна версія повинна коректно працювати з production-базою даних, зовнішніми HTTP-запитами, налаштуваннями Rails і проксуванням через Nginx. Саме тому цей етап є важливою частиною оцінювання готовності системи.

Стратегія тестування MovieMatch була побудована так, щоб перевірити не лише окремі сторінки, а повні користувацькі сценарії. Наприклад, для оцінювання фільму перевірявся не тільки факт натискання на зірку, а й збереження оцінки в базі даних, її відображення на сторінці фільму та врахування у профілі користувача. Такий підхід дозволяє переконатися, що функція працює повністю, а не лише в інтерфейсі.

Окремо перевірялися сценарії для різних ролей користувачів. Для гостя важливим було відкриття каталогу та перегляд сторінок фільмів без доступу до персональних дій. Для зареєстрованого користувача перевірялися оцінки, коментарі, списки перегляду, підбірка фільмів, рандомайзер і ручне додавання власного фільму. Для адміністратора перевірявся доступ до панелі керування, перегляд користувачів, коментарів, доданого контенту та керування каталогом.

Також у процесі тестування враховувалася стабільність після розгортання. Локально система може працювати коректно, але після перенесення на сервер можуть з'явитися проблеми з конфігурацією, авторизацією, доступом до бази даних або маршрутизацією. Тому перевірка виконувалася не лише в середовищі розробки, а й після запуску production-версії.

Окремим результатом є перенесення даних у production-базу PostgreSQL. Наявність 272 публічних фільмів у базі підтверджує, що система працює з реальним набором даних, а не з порожнім тестовим каталогом. Це дозволяє перевірити продуктивність, коректність фільтрації, пошуку та відкриття сторінок фільмів у практичних умовах.

|      |      |          |        |      |                         |      |
|------|------|----------|--------|------|-------------------------|------|
|      |      |          |        |      | РБ.ІП - 11.00.00.000 ПЗ | Арк. |
|      |      |          |        |      |                         | 60   |
| Змн. | Арк. | № докум. | Підпис | Дата |                         |      |

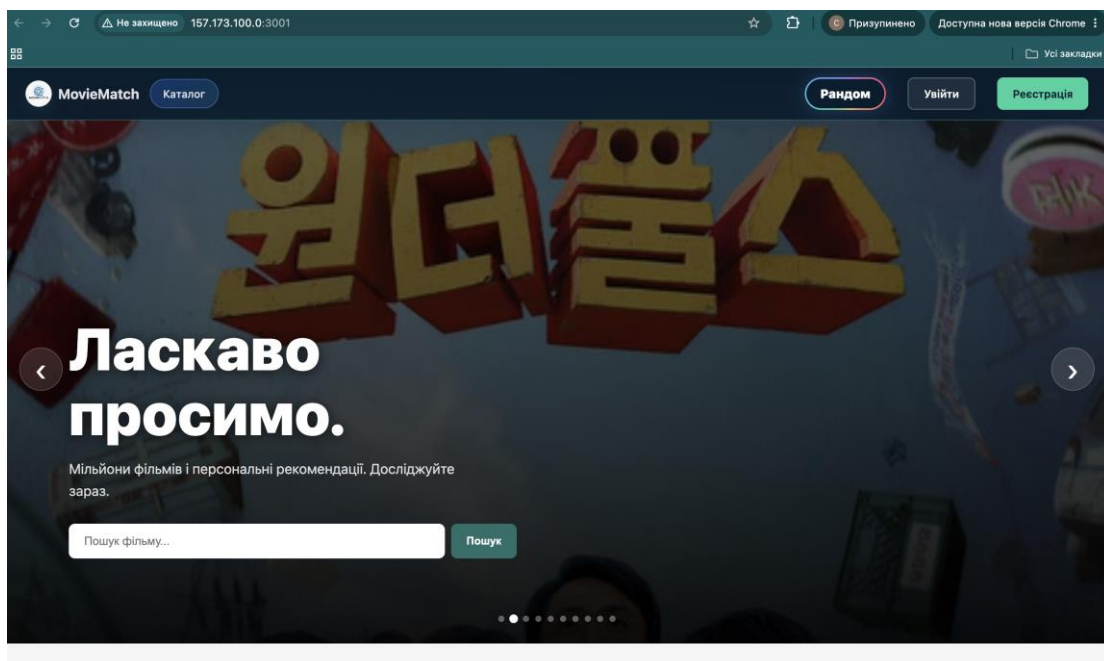


Рисунок 5.1 – Результат розгортання MovieMatch на сервері

#### 5.4. Висновки по розділу

У п'ятому розділі описано тестування та розгортання MovieMatch. Перевірка показала, що основні функції системи працюють, каталог завантажує 272 записи з PostgreSQL, авторизація відновлена, а production-конфігурація дозволяє використовувати сайт із зовнішніх пристроїв за IP-адресою сервера.

Результати тестування підтвердили, що основні функції відповідають поставленим вимогам. Виявлені під час розгортання проблеми стосувалися конфігурації production-середовища, а не логіки користувацьких сценаріїв. Після їх усунення система стабільно відкриває каталог, працює з PostgreSQL-базою та доступна через публічну IP-адресу.

Таким чином, тестування і розгортання підтвердили готовність MovieMatch як навчального веб-застосунку. Система реалізує повний цикл роботи: збереження даних, відображення каталогу, персональні дії користувача, адміністративне керування і доступ із зовнішніх пристроїв. Це підтверджує практичну придатність розробленого рішення в межах бакалаврської роботи.

|      |      |          |        |      |                         |      |
|------|------|----------|--------|------|-------------------------|------|
|      |      |          |        |      | РБ.ІП - 11.00.00.000 ПЗ | Арк. |
|      |      |          |        |      |                         | 61   |
| Змн. | Арк. | № докум. | Підпис | Дата |                         |      |

## ВИСНОВКИ

У бакалаврській роботі розроблено веб-систему MovieMatch для рекомендаційного вибору фільмів і серіалів. Робота охопила повний цикл створення програмного продукту: аналіз предметної області, порівняння аналогів, формування вимог, проєктування архітектури і бази даних, реалізацію функцій, тестування та розгортання на сервері.

У процесі виконання роботи встановлено, що для бакалаврського проєкту оптимальною є не спроба створити аналог Netflix, а розроблення персональної рекомендаційно-каталогової системи. Такий підхід дозволив реалізувати практично значущі функції: каталог фільмів і серіалів, пошук, жанрову фільтрацію, власні оцінки, коментарі, персональні списки, підбірку фільмів, рандомайзер, публічне поширення підбірки та адміністративну панель.

Розроблена модель даних PostgreSQL забезпечує збереження основних сутностей і зв'язків: користувачів, фільмів, оцінок, коментарів, списків перегляду та реакцій. Особливу увагу приділено персональності даних: власна оцінка користувача не зміщується із зовнішнім рейтингом, а вручну доданий фільм не потрапляє до загального каталогу без адміністративної дії.

Практична реалізація на Ruby on Rails підтвердила доцільність MVC-підходу для навчального веб-проєкту. Active Record спростив роботу з базою даних, маршрутизація Rails забезпечила зрозумілу структуру URL, а шаблони представлення дали можливість швидко формувати сторінки каталогу, профілю, фільму, підбірки і адмін-панелі.

Під час розгортання було вирішено реальні production-проблеми: конфігурацію авторизації при доступі через IP без HTTPS, перенесення повної локальної бази даних, створення службових таблиць Rails і усунення дубльованого контролера. Це показує, що результат роботи не обмежився локальним прототипом, а був доведений до стану, у якому сайт можна відкривати з інших пристроїв.

|      |      |          |        |      |                         |      |
|------|------|----------|--------|------|-------------------------|------|
|      |      |          |        |      | РБ.ІП - 11.00.00.000 ПЗ | Арк. |
| Змн. | Арк. | № докум. | Підпис | Дата |                         | 62   |

## СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ruby Programming Language Documentation. URL: <https://www.ruby-lang.org/en/documentation/> (дата звернення: 18.01.2026).
2. Active Record Basics. Ruby on Rails Guides. URL: [https://guides.rubyonrails.org/active\\_record\\_basics.html](https://guides.rubyonrails.org/active_record_basics.html) (дата звернення: 28.01.2026).
3. Action Controller Overview. Ruby on Rails Guides. URL: [https://guides.rubyonrails.org/action\\_controller\\_overview.html](https://guides.rubyonrails.org/action_controller_overview.html) (дата звернення: 19.01.2026).
4. Методичні рекомендації та вимоги щодо виконання та оформлення бакалаврських робіт для студентів спеціальності 121 «Інженерія програмного забезпечення». ІФНТУНГ, 2024.
5. Ruby on Rails Guides. URL: <https://guides.rubyonrails.org/> (дата звернення: 20.01.2026).
6. Ruby on Rails API Documentation. URL: <https://api.rubyonrails.org/> (дата звернення: 01.03.2026).
7. Ruby Programming Language Documentation. URL: <https://www.ruby-lang.org/en/documentation/> (дата звернення: 03.03.2026).
8. PostgreSQL Documentation. URL: <https://www.postgresql.org/docs/> (дата звернення: 04.02.2026).
9. PostgreSQL Indexes Documentation. URL: <https://www.postgresql.org/docs/current/indexes.html> (дата звернення: 06.03.2026).
10. PostgreSQL Full Text Search Documentation. URL: <https://www.postgresql.org/docs/current/textsearch.html> (дата звернення: 08.03.2026).
11. Puma Web Server Documentation. URL: <https://puma.io/> (дата звернення: 09.02.2026).

|      |      |          |        |      |                         |      |
|------|------|----------|--------|------|-------------------------|------|
|      |      |          |        |      | РБ.ІП - 11.00.00.000 ПЗ | Арк. |
| Змн. | Арк. | № докум. | Підпис | Дата |                         | 63   |

12. Nginx Documentation. URL: <https://nginx.org/en/docs/> (дата звернення: 09.03.2026).

13. Ubuntu Server Documentation. URL: <https://documentation.ubuntu.com/server/> (дата звернення: 10.03.2026).

14. OWASP Application Security Verification Standard. URL: <https://owasp.org/www-project-application-security-verification-standard/> (дата звернення: 12.02.2026).

15. OWASP Top Ten. URL: <https://owasp.org/www-project-top-ten/> (дата звернення: 14.03.2026).

16. OWASP Cheat Sheet Series: Authentication. URL: [https://cheatsheetseries.owasp.org/cheatsheets/Authentication\\_Cheat\\_Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/Authentication_Cheat_Sheet.html) (дата звернення: 16.02.2026).

17. OWASP Cheat Sheet Series: Session Management. URL: [https://cheatsheetseries.owasp.org/cheatsheets/Session\\_Management\\_Cheat\\_Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/Session_Management_Cheat_Sheet.html) (дата звернення: 18.03.2026).

18. OWASP Cheat Sheet Series: Cross-Site Request Forgery Prevention. URL: [https://cheatsheetseries.owasp.org/cheatsheets/CrossSite\\_Request\\_Forgery\\_Prevention\\_Cheat\\_Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/CrossSite_Request_Forgery_Prevention_Cheat_Sheet.html) (дата звернення: 20.03.2026).

19. The Movie Database API Documentation. URL: <https://developer.themoviedb.org/docs> (дата звернення: 21.03.2026).

20. The Movie Database Website. URL: <https://www.themoviedb.org/> (дата звернення: 19.03.2026).

21. JustWatch. URL: <https://www.justwatch.com/> (дата звернення: 22.03.2026).

22. IMDb. URL: <https://www.imdb.com/> (дата звернення: 23.02.2026).

23. Netflix. URL: <https://www.netflix.com/> (дата звернення: 25.02.2026).

24. MEGOGO. URL: <https://megogo.net/> (дата звернення: 08.03.2026).

25. Apple TV. URL: <https://tv.apple.com/> (дата звернення: 33.03.2026).

26. Google TV. URL: <https://tv.google/> (дата звернення: 09.04.2026).

|      |      |          |        |      |                         |      |
|------|------|----------|--------|------|-------------------------|------|
|      |      |          |        |      | РБ.ІІ - 11.00.00.000 ІЗ | Арк. |
| Змн. | Арк. | № докум. | Підпис | Дата |                         | 64   |

27. Amazon Prime Video. URL: <https://www.primevideo.com/> (дата звернення: 10.04.2026).
28. Fowler M. Patterns of Enterprise Application Architecture. Addison-Wesley, 2002.
29. Gamma E., Helm R., Johnson R., Vlissides J. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, 1994.
30. Pressman R. S., Maxim B. R. Software Engineering: A Practitioner's Approach. McGraw-Hill Education, 2020.
31. Sommerville I. Software Engineering. Pearson, 2016.
32. Beck K. Test Driven Development: By Example. Addison-Wesley, 2002.
33. Freeman S., Pryce N. Growing Object-Oriented Software, Guided by Tests. Addison-Wesley, 2009.
34. Nielsen J. Usability Engineering. Morgan Kaufmann, 1993.
35. Krug S. Don't Make Me Think, Revisited. New Riders, 2014.
36. Norman D. The Design of Everyday Things. Basic Books, 2013.
37. Resnick P., Varian H. R. Recommender Systems. Communications of the ACM. 1997. Vol. 40, No. 3. P. 56–58.
38. Ricci F., Rokach L., Shapira B. Recommender Systems Handbook. Springer, 2022.
39. Aggarwal C. C. Recommender Systems: The Textbook. Springer, 2016.
40. Jannach D., Zanker M., Felfernig A., Friedrich G. Recommender Systems: An Introduction. Cambridge University Press, 2010.
41. ISO/IEC 25010:2011. Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE).
42. MDN Web Docs: HTTP. URL: <https://developer.mozilla.org/en-US/docs/Web/HTTP> (дата звернення: 08.05.2026).
43. Git Репозиторій. URL: <https://github.com/Svetatata05/moviematch.git> (дата звернення: 08.06.2026).

|      |      |          |        |      |                         |      |
|------|------|----------|--------|------|-------------------------|------|
|      |      |          |        |      | РБ.ІІ - 11.00.00.000 ПЗ | Арк. |
|      |      |          |        |      |                         | 65   |
| Змн. | Арк. | № докум. | Підпис | Дата |                         |      |

# ДОДАТКИ

## Додаток А. Фрагменти структури бази даних

Структура бази даних MovieMatch охоплює таблиці users, movies, ratings, comments, watchlists і movie\_reactions. ER-модель наведено в основній частині роботи на рисунку 3.2. У додатку фіксується перелік ключових сутностей, які забезпечують зв'язок між користувачем, фільмом і персональними діями.

```
moviematch_development=# \d comments
      Column |          Type          | Collation | Nullable | Default
-----+-----+-----+-----+-----
id           | bigint                |           | not null | nextval('comments_id_seq'
::regclass)
user_id      | bigint                |           | not null |
movie_id     | bigint                |           | not null |
body         | text                  |           | not null |
created_at   | timestamp(6) without time zone |           | not null |
updated_at   | timestamp(6) without time zone |           | not null |
Indexes:
    "comments_pkey" PRIMARY KEY, btree (id)
    "index_comments_on_movie_id" btree (movie_id)
    "index_comments_on_movie_id_and_created_at" btree (movie_id, created_at)
    "index_comments_on_user_id" btree (user_id)
Foreign-key constraints:
    "fk_rails_03de2dc08c" FOREIGN KEY (user_id) REFERENCES users(id)
    "fk_rails_56963e5c80" FOREIGN KEY (movie_id) REFERENCES movies(id)
```

Рисунок А.1 Структура таблиці comments

```
moviematch_development=# \d watchlists
      Column |          Type          | Collation | Nullable | Default
-----+-----+-----+-----+-----
id           | bigint                |           | not null | nextval('watchlists_id_se
q'::regclass)
user_id      | integer               |           | not null |
movie_id     | integer               |           | not null |
created_at   | timestamp(6) without time zone |           | not null |
updated_at   | timestamp(6) without time zone |           | not null |
Indexes:
    "watchlists_pkey" PRIMARY KEY, btree (id)
    "index_watchlists_on_movie_id" btree (movie_id)
    "index_watchlists_on_user_id" btree (user_id)
Foreign-key constraints:
    "fk_rails_0dc1a4cbcb" FOREIGN KEY (user_id) REFERENCES users(id)
    "fk_rails_1de1911646" FOREIGN KEY (movie_id) REFERENCES movies(id)
```

Рисунок А.2 Структура таблиці watchlists

## Додаток Б. Фрагменти коду

Програмна реалізація включає контролер пошуку, модель Movie, контролер оцінювання та логіку підбірки фільмів. Повні файли коду

зберігаються у репозиторії проєкту, а в пояснювальній записці описано їх призначення і взаємозв'язок.

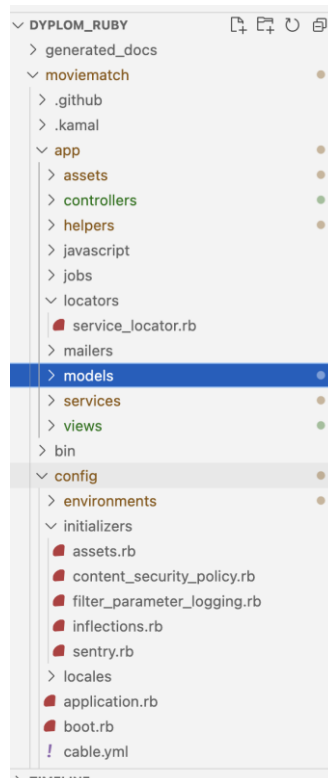


Рисунок Б.1 Структура проєкту

## Код файлу `moviematch/app/views/users/_profile_movie_list.html.erb`

```
<% if movies.any? %>
<div class="wl-list profile-movie-list">
  <% movies.each do |movie| %>
    <div class="wl-item profile-movie-item">
      <div class="wl-poster">
        <% if movie.poster_url.present? %>
          ">
        <% else %>
          <div class="wl-ph">POSTER</div>
        <% end %>
      </div>
      <div class="wl-info">
        <h3><%= link_to movie.title, movie_path(movie) %></h3>
        <p><%= [movie.media_label, movie.genre, movie.year].compact.join(" • ") %></p>
        <% if movie.manual? %>
          <span class="manual-badge">Додано вручну</span>
        <% end %>
      </div>
    </div>
  </div>
</div>
```

```

    <% if (own_rating = current_user.ratings.find_by(movie: movie)) %>
      <span class="wl-rating">Моя оцінка: <%= own_rating.score %>/5</span>
    <% else %>
      <span class="wl-rating">Ще не оцінено</span>
    <% end %>
  </div>
  <div class="profile-movie-actions">
    <% if local_assigns[:list_status] == "want" %>
      <%= button_to "Бачив/ла", swipes_path, method: :post,
        params: { movie_id: movie.id, movie_reaction: { status: MovieReaction::STATUSES[:seen] } },
        class: "btn-seen" %>
    <% end %>
    <% if (reaction_item = current_user.movie_reactions.find_by(movie: movie)) %>
      <%= button_to "Видалити", movie_reaction_path(reaction_item), method: :delete, class: "btn-remove" %>
    <% elsif (watchlist_item = current_user.watchlists.find_by(movie: movie)) %>
      <%= button_to "Видалити", watchlist_path(watchlist_item), method: :delete, class: "btn-remove" %>
    <% end %>
  </div>
</div>
<% end %>
</div>
<% else %>
  <p class="empty-msg"><%= empty_text %> <%= link_to "Почати підбір", swipes_path %></p>
<% end %>

```

## **БІБЛІОГРАФІЧНА ДОВІДКА**

Тема бакалаврської роботи: "Розроблення веб-системи рекомендацій вибору фільмів засобами Ruby on Rails/Postgres "

Обсяг пояснювальної записки: 65 аркушів

Дата закінчення бакалаврської роботи 10 червня 2026р.

Підпис студента \_\_\_\_\_ Котурлаш С.В.