Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра програмного забезпечення

Бакалаврська кваліфікаційна робота

на тему: «Розробка методу та засобів реалізації вебсистеми кіноклубу для підбору й обговорення відеоконтенту»

Виконав: студент IV курсу
групи ЗПІ-21б
спеціальності
121 – Інженерія програмного забезпечення
(шифр і назва напряму підготовки, спеціальності)
Цимбал І. Ю.
(прізвище та ініціали)

Керівник: к.т.н., доц. каф. ПЗ Войтко В. В.
(прізвище та ініціали)

Рецензент: к.т.н., доц. каф. ОТ Городецька О. С.
(прізвище та ініціали)

Допущено до захисту

Зав. кафедри ПЗ Романюк О.Н.
«06» червня 2025 р.

ВНТУ – 2025

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення
Рівень вищої освіти перший (бакалаврський)
Галузь знань 12 «Інформаційні технології»
Спеціальність 121 «Інженерія програмного забезпечення»
Освітньо-професійна програма «Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
О. Н. Романюк
«21» березня 2025 р.

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

Цимбалу Івану Юрійовичу

1. Тема роботи – розробка методу та засобів реалізації вебсистеми кіноклубу для підбору й обговорення відеоконтенту

Керівник роботи: Войтко Вікторія Володимирівна, к.т.н., доцент кафедри ПЗ, затверджені наказом вищого навчального закладу від 20 березня 2025 р. №97.

2. Строк подання студентом роботи: 2 червня 2025 р.

3. Вихідні дані до роботи: середовище розробки – Visual Studio Code, мови розробки – JavaScript/TypeScript, HTML, SASS; фреймворки – NextJS, NodeJS/NestJS; операційна система – Windows 8/10/11.

4. Зміст текстової частини: вступ, аналіз стану розвитку систем кіноклубів для підбору та обговорення контенту та постановка завдань розробки, розробка методу, моделей та алгоритмів роботи вебсистеми, розробка програмних модулів, тестування вебсистеми та окремих модулів, висновки, список використаних джерел, додатки, ілюстративна частина.

5. Перелік ілюстративного матеріалу: титульний слайд – автор, науковий керівник бакалаврської кваліфікаційної роботи, тема; актуальність розробки; мета, предмет та об'єкт дослідження; постановка задачі; наукова новизна; практичне значення; аналіз аналогів; метод автоматизованого підбору; блок-схема алгоритму автоматизованого підбору; метод збору інформації про користувача та формування аналітичних графіків; блок-схема алгоритму збору інформації про користувача та формування аналітичних графіків; загальна блок-схема алгоритму роботи системи; схеми інтерфейсів; діаграма стану модуля обговорення контенту; використані

технології; апробація та публікація результатів дослідження; функціо можливості розробленої вебсистеми; висновки; кінцевий слайд.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завда прийн
1-4	Войтко В. В., к.т.н., доцент кафедри ПЗ	24.03.25	01.06.

7. Дата видачі завдання 24 березня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської кваліфікаційної роботи	Строк виконання етапів роботи	Приміт
1	Аналіз розвитку вебсистем для кіноклубів та постановка задач дослідження	25.03.25 – 29.03.25	Вик.
2	Розробка методів та алгоритмів роботи системи	30.03.25 – 10.04.25	Вик.
3	Розробка програмного забезпечення системи	11.04.25 – 10.05.25	Вик.
4	Тестування програми	11.05.25 – 14.05.25	Вик.
5	Оформлення матеріалів до захисту БКР	15.05.25 – 30.05.25	Вик.

Студент _____ **Цимба**
(підпис) (ініціали та пр

Керівник бакалаврської кваліфікаційної роботи _____ **Войтко В**
(підпис) (ініціали та пр

АНОТАЦІЯ

Бакалаврська кваліфікаційна робота складається з 68 сторінок формату А4, що містять 48 рисунків і 3 таблиці, список використаних джерел налічує 28 найменувань.

У бакалаврській кваліфікаційній роботі проаналізовано стан і рівень розвитку застосунків кіноклубів, що мають на меті полегшити комунікацію між постачальником контенту та кінцевим споживачем. Проведено аналіз аналогів розроблюваної вебсистеми, визначено їх основні переваги та недоліки та доведено доцільність розробки власного програмного продукту.

Розроблена вебсистема дозволяє покращити діяльність кіноклубу шляхом взаємодії із аудиторією: автоматизований підбір відеоконтенту на основі короткого опитування, обговорення улюблених жанрів у реальному часі та персоналізовані пропозиції, базуючись на вподобаннях користувача.

Вебсистему розроблено з використанням мов програмування «JavaScript» та «TypeScript» та фреймворків «NextJS» для клієнтської частини й «NestJS» для серверної. «Visual Studio Code» було обрано основним середовищем розробки. Для можливості взаємодії в реальному часі з іншими користувачами було використано протокол «Socket» й бібліотеку «Socket.io». Для відображення статистичних даних було використано бібліотеку «ChartsJS».

У результаті виконання бакалаврської кваліфікаційної роботи було розроблено вебсистему кіноклубу, що дасть можливість підвищити загальний рівень зацікавленості відвідувачів, заохотити нову аудиторію, таким чином збільшити прибутковість кіно бізнесу.

Ключові слова: вебсистема, кіноклуб, TypeScript, NextJS, NestJS, інтерфейс користувача.

ABSTRACT

The bachelor's qualification work consists of 68 pages of A4 format, containing 45 figures and 3 tables, the list of used sources includes 28 items.

The bachelor's qualification work analyzes the state and level of development of film club relations, which aim to facilitate communication between the content provider and the end consumer. An analysis of analogues of the developed web system was conducted, their main advantages and disadvantages were identified, and the feasibility of developing an own software product was proven.

The developed web system allows you to improve the activities of the film club through interaction with the audience: automated selection of video content based on a short survey, discussion of favorite genres in real time, and personalized offers based on user preferences.

The web system was developed using the «JavaScript» and «TypeScript» programming languages and the «NextJS» frameworks for the client part and «NestJS» for the server part. «Visual Studio Code» was chosen as the main development environment. To enable real-time interaction with other users, the «Socket» protocol and the «Socket.io» library were used. The «ChartsJS» library was used to display statistical data.

As a result of the bachelor's qualification work, a web system for a cinema club was developed, which will allow to increase the overall level of interest of visitors, to encourage a new audience, thus increasing the profitability of the cinema business.

Keywords: web system, cinema club, TypeScript, NextJS, NestJS, user interface.

ЗМІСТ

ВСТУП 4
1 АНАЛІЗ РОЗВИТКУ ВЕБСИСТЕМ ДЛЯ КІНОКЛУБІВ ТА ПОСТАНОВКА ЗАДАЧ ДОСЛІДЖЕННЯ 9
1.1 Аналіз предметної області 9
1.2 Порівняльний аналіз аналогів вебсистеми кіноклубу 10
1.3 Аналіз методів розв’язання задачі 14
1.4 Постановка задач дослідження 16
1.5 Висновки 16
2 РОЗРОБКА МЕТОДІВ, МОДЕЛЕЙ ТА АЛГОРИТМІВ РОБОТИ СИСТЕМИ 18
2.1 Аналіз даних 18
2.2 Розробка загальної моделі вебсистеми 20
2.3 Розробка загального алгоритму роботи програми і діаграм станів 22
2.4 Розробка методу автоматизованого підбору контенту 24
2.5 Розробка методу збору інформації про користувача та формування аналітичних графіків 28
2.6 Висновки 31
3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ СИСТЕМИ 32
3.1 Варіантний аналіз і обґрунтування вибору засобів для реалізації вебсистеми 32
3.2 Розробка модуля обговорення контенту в реальному часі 35
3.3 Розробка модуля улюбленого 40
3.4 Розробка модуля автоматизованого підбору контенту 42
3.6 Розробка модуля статистики 45
3.7 Розробка інтерфейсу користувача вебсистеми 48
3.8 Висновки 53
4 ТЕСТУВАННЯ ПРОГРАМИ 55
4.1 Тестування функціоналу вебсистеми 55
4.2 Розробка інструкції користувача 64
4.4 Висновки 66
ВИСНОВКИ 67

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 69
ДОДАТКИ 72
Додаток А. Технічне завдання 73
Додаток Б. Протокол перевірки бакалаврської кваліфікаційної роботи на наявність текстових запозичень 77
Додаток В. Лістинг програми 78
Додаток Г. Ілюстративна частина 100

ВСТУП

Обґрунтування вибору теми дослідження. Сьогодні кіноіндустрія – це не лише створення фільмів, але і їх прокат. Кожен приходить у кінобізнес з різними намірами, хтось для реалізації власних ідей, а хтось для розвитку власного бізнесу.

У зв'язку зі зростанням популярності фільмів і кінотеатрів серед широких верств населення зріс рівень конкуренції, що вимагає впровадження нових рішень, які могли б зацікавити споживачів. Згідно зі звітом «BBC», відео складає 58% глобального обсягу трафіку, за ним йде перегляд сторінок в інтернеті (17%), ігри (7,8%) та соціальні мережі (5,1%) [1].

Сучасний етап розвитку технологій і збільшення рівня очікування людей вимагає постійного вдосконалення методів аналізу даних користувача, особливо що стосується сектору економіки, що надає послуги. Однією з найбільших проблем власників бізнесів є залучення нової аудиторії та збільшення попиту на власну продукцію, те ж саме і стосується кінопрокату – не всі кінотеатри користуються прихильністю через брак потенційних відвідувачів. Соціальні мережі відкрили перед підприємцями безмежні можливості для спілкування зі своєю аудиторією. Вони дозволяють побудувати міцні відносини з клієнтами, показати унікальність свого бренду та ефективно конкурувати на ринку [2].

Традиційні методи ведення бізнесу в сфері кінопрокату давно не дають бажаних результатів, адже більшість людей перейшли у віртуальний світ і виграє той, хто вміє захопити увагу користувача, запропонувати йому альтернативу й тим самим змотивувати відвідати кінотеатр, а останні десятиліття підприємства рухаються в бік автоматизації та віртуалізації.

Розвиток технологій, таких як електронні обчислювальні машини, штучний інтелект, хмарні сховища та бази із даними роблять взаємодію із кінцевим споживачем набагато простішою: зникає потреба у паперових носіях, а всю

необхідну інформацію відвідувачі можуть отримати в електронному форматі, без потреби виходити з дому або звертатися до кінотеатрів безпосередньо. Завдяки базам даних та розробці вебзастосунків інформація про користувача може зберігатися безстроково з метою подальшої обробки, задля аналізу поведінки відвідувачів і впровадження потрібного функціоналу, базуючись на результатах обробки.

Наявні аналоги розроблюваної системи для кіноклубу з можливістю підбору й обговорення контенту не надають функціоналу, що здатен зацікавити корисувача. Більшість із готових рішень є доволі вузькоспеціалізованими й розширюють лише якусь певну частину. Технології змінюють життя людей і, таким чином, змінюють всі методи ведення бізнесу і операції. В результаті кожна галузь зараз зосереджується на впровадженні нових та інноваційних технологій в свої ділові підприємства [3]. Таким чином, актуальною є розробка власної вебсистеми кіноклубу, що покращить діяльність малого бізнесу й принесе дохід.

Зв’язок роботи з науковими програмами, планами, темами. Робота виконувалася згідно плану виконання наукових досліджень на кафедрі програмного забезпечення.

Мета та завдання дослідження. Метою роботи є підвищення відвідуваності у сфері кінопоказу за рахунок використання вебсистеми для обговорення та підбору контенту, що дозволить залучати нових відвідувачів та аудиторію в мережі Інтернет, відслідковувати і розуміти їх вподобання, тим самим підвищуючи рівень зацікавленості та прибуток.

Відповідно до поставленої мети потрібно вирішити такі завдання:

- провести аналіз стану розвитку застосунків кіноклубів;
- розробити загальну модель вебсистеми;
- розробити загальний алгоритм роботи платформи;
- розробити метод автоматизованого підбору контенту;

– розробити метод збору інформації про користувача та формування аналітичних графіків;

– розробити вебсистему кіноклубу, функціонал якої включає:

1) авторизацію;

2) формування власного списку вподобаних фільмів;

3) обговорення відеоконтенту за жанрами та вподобаннями, окремими групами, в режимі реального часу;

4) функціонал автоматизованого підбору контенту шляхом проведення короткого опитування користувача та аналізу його списку вподобаних фільмів;

5) каталог із усіма відомими фільмами та інформацією про них та їх акторів;

6) функціональність пошуку із можливістю застосування широкого спектру фільтрів;

7) можливість перегляду відеоконтенту;

8) статистичні дані про взаємодію користувача із системою.

– розробити інтуїтивний інтерфейс користувача;

– провести тестування роботи вебсистеми.

Об'єктом дослідження є процес розробки вебсистеми кіноклубу для підбору й обговорення відеоконтенту.

Предметом дослідження є методи і засоби реалізації вебсистеми кіноклубу для обговорення і підбору контенту.

Методи дослідження. У процесі дослідження використовувалися такі методи:

– методи створення UI/UX для розробки інтерфейсу користувача;

– методи програмування з використанням фреймворку «NestJS» для розробки серверної частини вебсистеми;

– методи аналізу даних для підбору і формування списку персоналізованого контенту;

– методи розробки сховища даних для збереження інформаційних рескрсів;

– методи тестування для перевірки роботи вебсистеми.

Новизна отриманих результатів.

1. Подальшого розвитку отримав метод збору інформації про користувача та формування аналітичних графіків, який, на відміну від існуючих, відслідковує активність користувача щоденно, будуючи на основі даних стовпчасту діаграму із найпопулярнішими категоріями, з якими взаємодіяв відвідувач вебсистеми за обраний проміжок часу, що дозволяє користувачу відслідковувати динаміку зміни своїх інтересів у процесі підбору персоналізованого контенту.

2. Подальшого розвитку отримав метод автоматизованого підбору контенту, який, на відміну від існуючих, враховує не лише результати опитування і відслідковує вподобання користувача, а й визначає пріоритетність обраного наповнення у процесі аналізу й підбору рекомендованих відео, що дозволяє сформувати персоналізований список запропонованих для перегляду фільмів з урахуванням вподобань та пріоритетності побажань користувача.

Практичне значення роботи полягає у можливості використання створеної вебсистеми у сфері кінопоказу, яка дозволить кіноклубу взаємодіяти з аудиторію й знаходити цільового споживача для певного виду контенту шляхом його підбору й можливості обговорення в реальному часі з іншими учасниками. Розроблені програмні модулі можуть бути використані не лише для кіноклубу, а у будь-якому напрямку сфери послуг, де важливе значення відіграє кінцевий споживач і бізнес залежить від аудиторії. Також розроблені програмні модулі можуть інтегруватися розробниками у будь-які інші застосунки, що потребують аналізу дій користувача.

Особистий внесок здобувача. Усі результати, подані в бакалаврській кваліфікаційній роботі, були отримані автором особисто. У науковій праці [4], опублікованій у співавторстві, автору належить розробка діаграми діяльності. У науковій праці [5], опублікованій у співавторстві, автору належать такі результати:

таблиця із порівнянням функціональних можливостей власного продукту та наявних аналогів, блок-схема загального алгоритму роботи вебсистеми, діаграма станів модулів підбору контенту шляхом опитування та обговорення фільмів у реальному часі. У науковій праці [6], яку опубліковано співавторстві, автору належать такі результати: метод автоматизованого пріоритетного підбору персоналізованого контенту та блок-схема алгоритму його роботи.

Апробація результатів роботи. Результати бакалаврської кваліфікаційної роботи доповідалися на Всеукраїнській науково-технічній конференції підрозділів Вінницького національного технічного університету (НТКП ВНТУ–2025) та на Міжнародній науково-практичній конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2025)».

Публікації. Результати дослідження були опубліковані у матеріалах Всеукраїнської науково-технічної конференції підрозділів Вінницького національного технічного університету (НТКП ВНТУ–2025) [4, 5] та у матеріалах Міжнародної науково-практичної інтернет-конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2025)» [6].

Аналіз. У пояснювальній записці до бакалаврської кваліфікаційної роботи було розглянуто чотири розділи та було використано 27 літературних джерел.

1 АНАЛІЗ РОЗВИТКУ ВЕБСИСТЕМ ДЛЯ КІНОКЛУБІВ ТА ПОСТАНОВКА ЗАДАЧ ДОСЛІДЖЕННЯ

1.1 Аналіз предметної області

Сучасний етап розвитку людства й технологій вимагає постійного вдосконалення методів взаємодії з користувачами в мережі Інтернет, особливо, що стосується бізнесу, який безпосередньо залежить від попиту серед людей, тому так важливо покращувати функціонал наявного програмного забезпечення, додаючи різноманітні алгоритми, що здатні захопити увагу й залишити користувача на платформі якомога довше, що в свою чергу покращує метрики вебсистеми й просуває її у рейтингу в пошукових системах. Тому у вище описаній ситуації надзвичайно важливо пропонувати користувачеві те, що актуально для нього й надавати можливість ділитися думкою [7].

Одним із ключових аспектів розвитку вище згаданих технологій є розквіт у сфері маркетингу й винаходження різноманітних стратегій, що націлені на заохочування пересічних громадян для взаємодії з контентом. Тому так важливо брати до уваги вподобання користувача, аби пропонувати те, що дійсно принесе користь, як йому так і бізнесу.

Впровадження алгоритму для підбору фільмів має значну перевагу над традиційним показом контенту.

Головною перевагою використання алгоритму для підбору контенту й імплементації можливості взаємодії у реальному часі є те, що це дозволяє більш точно знайти та привернути увагу цільової аудиторію для того чи іншого виду наповнення й надати можливість обговорити його, не виходячи з дому й знайти однодумців. Алгоритм підбору контенту відіграє ключову роль у формуванні інформаційної екосистеми, адже він вирішує чи побачить користувач певний контент [8].

Ще однією перевагою використання технологій для взаємодії у реальному часі для популяризації кіноконтенту є можливість розвитку за кордоном й привертання уваги закордонної аудиторії та заохочування інвестицій і фінансових надходжень у даний сектор бізнесу, а це може відіграти неабияку роль у формуванні економіки країни, що займається розвитком й впровадженням даної вебсистеми.

Отже, аналіз стану наявних технологій для підбору контенту й обговорень в реальному часі підтверджує важливість переходу до більш радикальних, надійних та цільових методів у сфері надання послуг кінопрокату й кіноклубу, що точно принесуть результат. Розробка такого функціоналу стає необхідністю в реаліях сьогодення і розвитку сфери маркетингу й обслуговування. Саме розроблюваний функціонал надасть перевагу бізнесу, що впровадить даний функціонал, серед конкурентів й допоможе користуватися прихильністю серед користувачів у мережі Інтернет. Таким чином, можна підсумувати й сказати, що надзвичайно важливо впроваджувати останні наявні технології у вебсистему кіноклубу, аби не залишатися осторонь й завжди мати попит серед відвідувачів.

1.2 Порівняльний аналіз аналогів вебсистеми кіноклубу

Завдяки розвитку технологій та їх використання у кіноіндустрії значно підвищує ефективність та рентабельність цієї галузі, залучаючи нових працівників та відвідувачів. На просторах мережі «Інтернет» існує багато інтерактивних застосунків та систем, що містять безліч відеоконтенту на різну тематику, тим самим приваблюючи та об’єднуючи користувачів з різними поглядами та інтересами.

До найбільш відомих застосунків, що використовуються кіноклубами відносять:

- FandonGo;
- MyVue;

– IMDb.

Одним із прикладів використання алгоритму підбору персоналізованого контенту є вебзастосунок «IMDb» (див. рисунок 1.1), що розроблено всесвітньо відомою компанією «Amazon» – майданчик, що містить інформацію про кіновсесвіт й пропонує контент для будь-кого й на будь-який смак, аналізуючи вподобання користувача [9]. Система надає можливість ділитися думками, щодо контенту, але не в реальному часі – це і є недоліком.

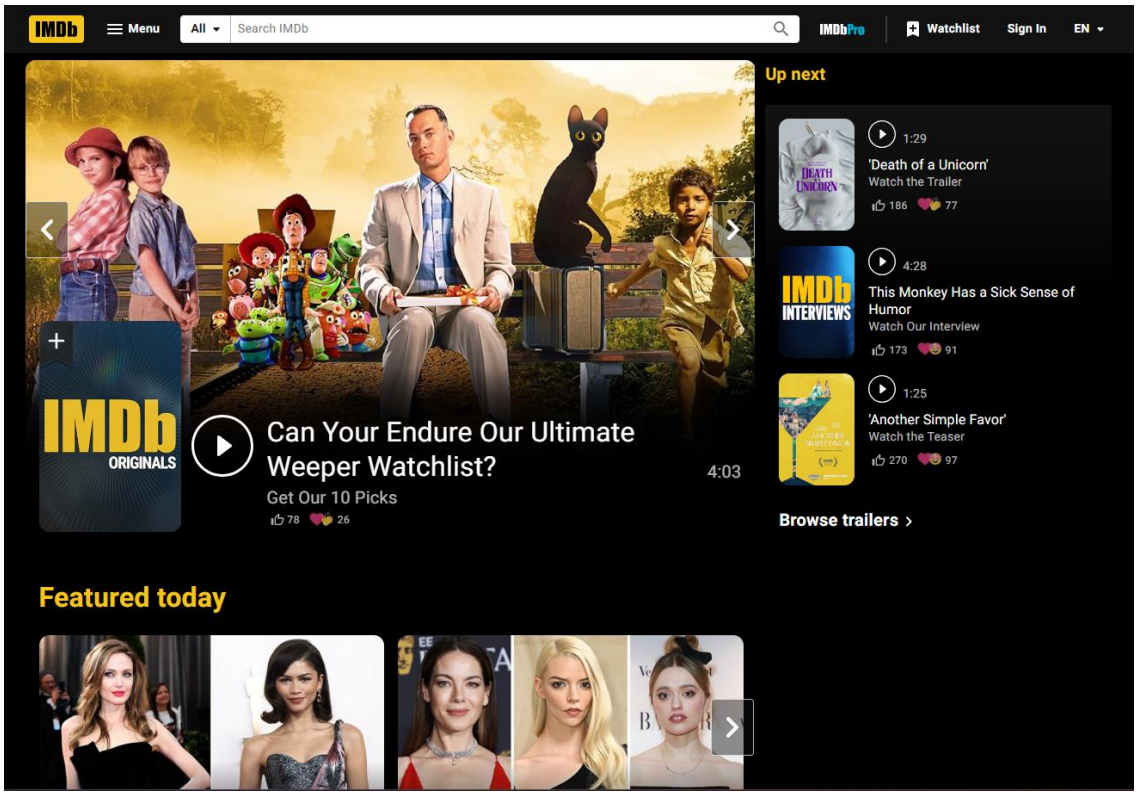

Рисунок 1.1 – Приклад роботи застосунку «IMDb»

Наступний приклад – вебсистема під назвою «MyVue» [10]. Це популярна мережа кінотеатрів у Великій Британії, яка також має зручний онлайн-сервіс для перегляду афіші та вибору сеансів, також наявна система лояльності, подарункові сертифікати та інформація про відеоконтент, що допоможе заохотити нову аудиторію. Але на превеликий жаль система має доволі обмежений функціонал й

не дозволяє додавати контент у список «Улюблене» або ж обговорювати його. Таким чином, можна зробити висновок, що даний сервіс має лише демонстративний характер, не наділений інтерактивністю «MyVue» продемонстровано на рис. 1.2.

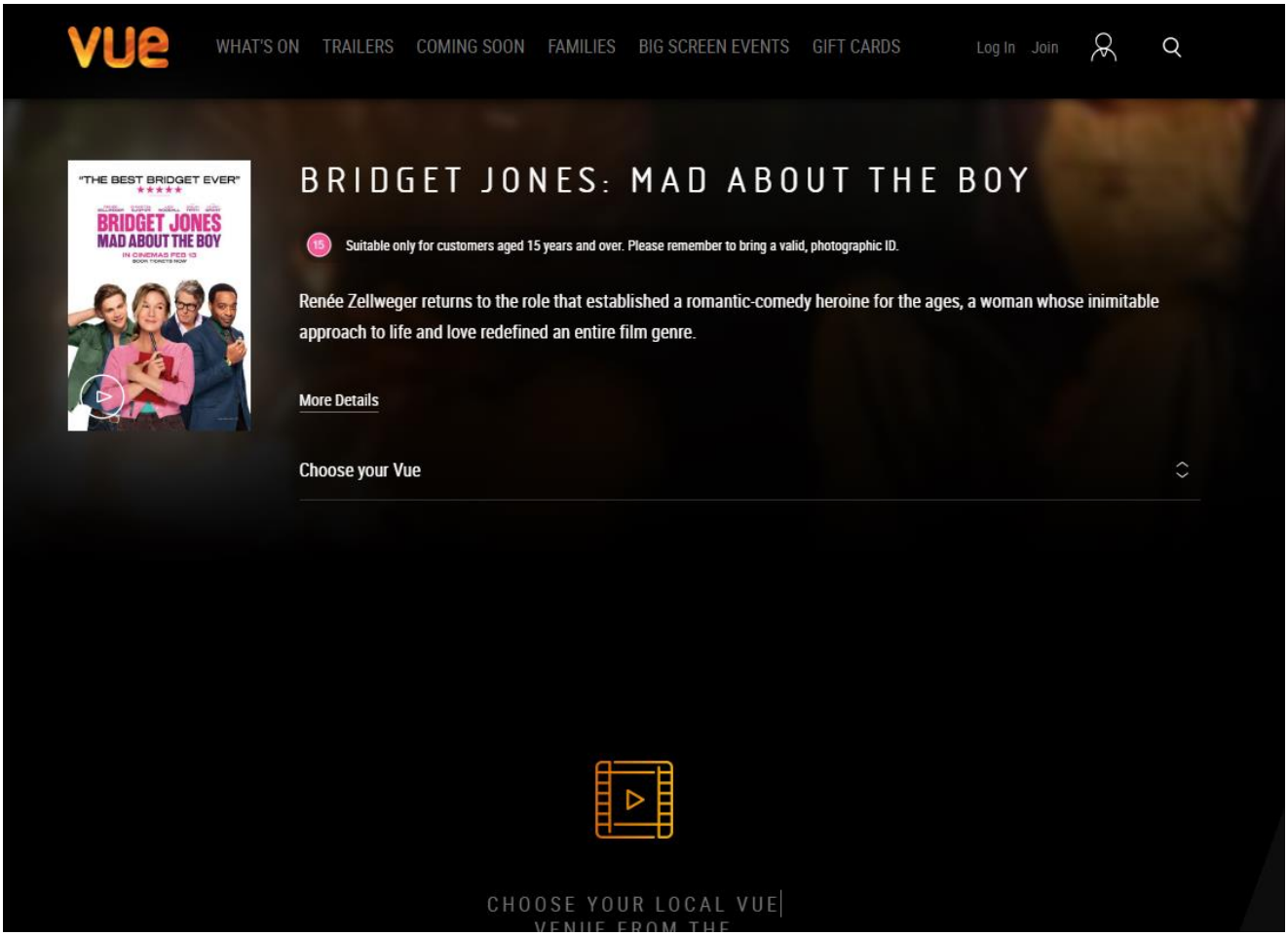

Рисунок 1.2 – Приклад роботи застосунку «MyVue»

«FanDanGo» (див. рисунок 1.3) є одним із найкращих запропонованих аналогів [11], адже цей сервіс є інтерактивним, на відміну від попередніх, і надає можливість взаємодії з контентом, обговорюючи та коментуючи його, але також не в режимі реального часу.

Також тут наявний функціонал рекомендації контенту, що відрізняє «FandonGo» від «MyVue» та «IMDb».

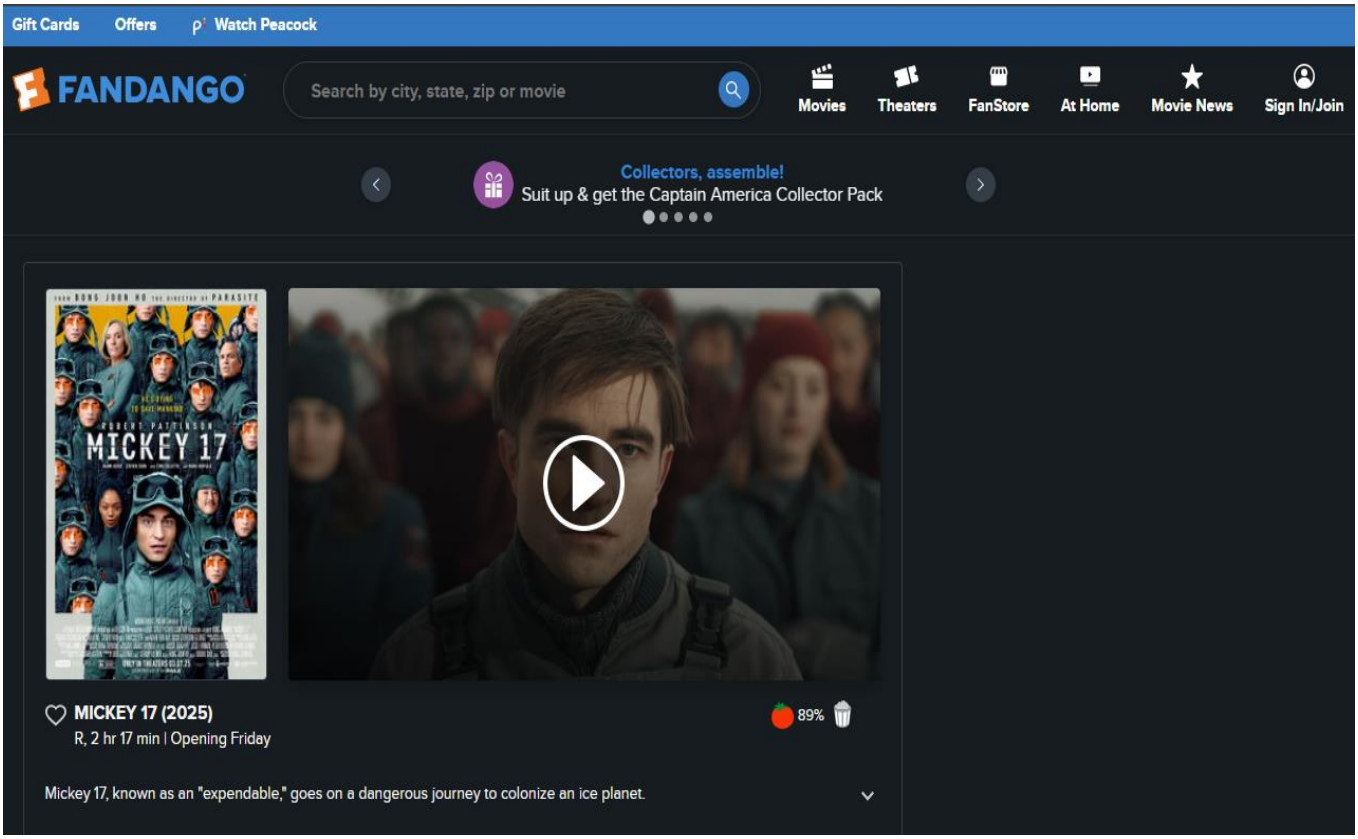

Рисунок 1.3 – Приклад роботи застосунку «FanDanGo»

Отже, після знайомства з усіма аналогами, було детально проаналізовано та вирішено, яким саме функціоналом варто наділити створювану вебсистему для кіноклубу.

Результати порівняння аналогів зведено в таблицю 1.1.

Таблиця 1.1 – Порівняльна характеристика аналогів

Критерій оцінювання	FandonGo	MyVue	IMDb	Власна розробка
Функціонал для комунікації в реальному часі із учасниками системи	0	0	0	1
Функціонал автопідбору контенту	1	1	1	1
Точність підбору контенту	1	1	1	1

Продовження таблиці 1.1

Критерій оцінювання	FandonGo	MyVue	IMDb	Власна розробка
Можливість додавати контент до списку «Улюблене»	1	0	1	1
Статистика користувача	0	0	0	1
Сумарний коефіцієнт	3	2	3	5

Відповідно до таблиці з порівнянням функціональних можливостей аналогів із власною розробкою для кіноклубу із можливістю підбору й обговорення контенту можна підсумувати, що власна розробка є доречною і доцільною для реалізації й впровадження. Розроблювана система має потенціал покрити недоліки усіх наявних систем й розшири їх можливості, ставши найкращою вебсистемою, де кожен зможе підібрати бажаний відеоконтент й обговорити його.

Отже, розробка програмних засобів для кіноклубу з можливістю підбору й обговорення контенту відкриє перед бізнесом кінопрокату широкі двері у новий етап розвитку й надасть більше можливостей для взаємодії з аудиторією, а це є одним із важливих критеріїв успішного розвитку власної справи.

1.3 Аналіз методів розв’язання задачі

Розробка вебсистеми кіноклубу вимагає обдумування кожного аспекту майбутньої програми. На просторах мережі Інтернет існує безліч готових рішень, але кожне із них має власні переваги та недоліки, що впливає на загальне враження користувачів.

Один із варіантів створення вебсистеми є використання відомої стратегії «SPA», що розшифровується як «Single Page Application», і має на меті полегшити розробку вебзастосунків [12].

Основним принципом підходу «SPA» є те, що браузер завантажує одну сторінку, кешує її, й динамічно змінює її вміст, без потреби повного

перезавантажувати застосунку. Також всі запити та додаткова логіка компілюється на клієнтській стороні. Такий підхід має перевагу в тому, що після першого завантаження сторінки, навігуючись по інших розділах, користувач буде вимушений чекати меншу кількість часу, адже сторінку вже було завантажено перед цим і контент змінюється динамічно, без повного перезавантаження. Але є декілька надзвичайно вагомих недоліків даної стратегії, що стають на заваді її використанню [13]:

1. SEO (search engine optimization) є надзвичайно низькою, адже весь вміст сторінки завантажується лише на клієнті, а пошукові роботи сканують лише серверний вміст сторінки, без запуску додаткового JavaScript коду.

2. Перше завантаження сторінки може займати до 5 секунд, адже браузер користувача повинен завантажити усі файли і виконати логіку задля відображення контенту. Таким чином, протягом великої кількості часу користувач бачитиме пусту сторінку.

3. Швидкість відображення контенту напряму залежить від якості з’єднання користувача із мережею Інтернет.

Інший, і набагато кращий підхід, полягає у використанні серверного підходу до рендерингу контенту вебсистеми. Такий підхід гарантує, що [14]:

1. Вебсистема матиме високий рейтинг у пошукових списках, за рахунок контенту на серверній частині.

2. Користувач бачитиме контент вже через 1-2 секунди, із усіма даними.

3. Швидкість відображення контенту менше залежатиме від швидкості Інтернету користувача, адже вся логіка та запити будуть виконані на сервері.

Таким чином, проаналізувавши переваги та недоліки кожного із підходів до розробки вебсервісів було вирішено використовувати стратегію серверного рендерингу контенту та виконання запитів. Використання серверного підходу є правильним рішенням, адже забезпечить швидке завантаження контенту та високі

рейтинги у пошукових списках, а своєю чергою допоможе заощадити кошти на просуванні та рекламі вебсистеми.

1.4 Постановка задач дослідження

Проаналізувавши переваги і недоліки наявних програмних рішень для кіноклубів і кінотеатрів з використання вебтехнологій було поставлено наступні цілі, що повинні бути виконані для успішної реалізації задуманого:

– реалізувати алгоритм автоматизованого підбору контенту на основі короткого опитування користувача і аналізі його взаємодії з системою;

– розробити алгоритм збору інформації про користувача з метою побудови аналітичних графіків;

– розробити функціонал візуалізації статистичних даних, шляхом щоденного відслідковування активності користувача;

– розробити систему із використанням фреймворку «NextJS», що потенційно пришвидшить рендеринг сторінок;

– розробити графічний інтерфейс користувача вебсистеми, що, першочергово, буде інтуїтивно зрозумілим для відвідувачів із будь-яким рівнем підготовки;

– розробити функціонал, що надасть можливість взаємодії користувачів у реальному часі задля обговорення контенту;

– протестувати коректність роботи вебсистеми.

Технічне завдання на розробку наведено в додатку А.

1.5 Висновки

У першому розділі було розглянуто наявні програмні засоби і рішення для кінотеатрів з використанням вебсистем. Проаналізовано функціонал наявних рішень. Було розглянуто наступні платформи: «MyVue», «FanDanGo» та «IMDb».

Проаналізувавши переваги і недоліки існуючих систем, було висунуто вимоги до власної розробки, аби скомбінувати в ній найкращий функціонал, що допоможе заохотити користувачів до відвідування кінотеатру. Поставлені цілі включають: розробку алгоритму й архітектури для взаємодії користувачів у реальному часі для проведення обговорень, автоматизовано підбору контенту на основі короткого опитування та аналізі вподобань і дій користувача, розробку інтуїтивного та простого інтерфейсу, імплементація візуалізації статистичних даних користувача.

Було доведено доцільність власної розробки шляхом аналізу існуючих імплементацій схожих вебсистем і сформовано список завдань, що необхідно виконати.

2 РОЗРОБКА МЕТОДІВ, МОДЕЛЕЙ ТА АЛГОРИТМІВ РОБОТИ СИСТЕМИ

2.1 Аналіз даних

Сьогодні всі додатки, не залежно від призначення та сфери використання, обов’язково включають функціонал для збору даних користувача з різною метою, як от наприклад для подальшого аналізу активності та прийняття правильних рішень задля покращення системи.

Процес передачі особистої інформації є неминучим: щонайменше усі сучасні вебсистеми зберігають власні файли кукі, аби розуміти вподобання користувача [15]. Користувач повинен пройти процес авторизації, щоб мати можливість взаємодіяти з таким функціоналом, як: створення власного списку улюбленого, прийняття участі в обговореннях різних тем в реальному часі, та отримати більш об’єктивні результати в процесі автоматизованого підбору відео контенту. Усі дані: об’єкти з персональною інформацією, аналітичні дані та фото зберігаються у хмарному сховищі від компанії «Google» [16].

Модуль авторизації приймає такі вхідні дані від користувача: ім’я, прізвище, нікнейм, електронна пошта, пароль та зображення профілю. Перед початком процесу збереженням інформації у сховищі, пароль буде закодовано із використанням 10-ти разового хешування та бібліотеки «Argon2» [17], що надає ці інструменти.

Після завершення реєстрації, користувач має персональний акаунт, в який може здійснювати вхід з використанням пошти та паролю.

Модуль улюбленого приймає на вхід нікнейм користувача та, власне, унікальний ідентифікатор фільму, завдяки якому буде знайдено інформацію у базі, де зберігається увесь контент.

Наявний функціонал також дозволяє видаляти фільми, таким чином редагуючи список, вхідні дані такі ж самі, так як потрібно розуміти, зі списку якого саме користувача потрібно видалити елемент. Кожен елемент списку включає: назву, рік випуску та рейтинг серед глядачів. Також користувач має можливість перейти на обраний фільм. Вихідними даними є графічний елемент з оновлений списком, що містить доданий елемент, або ж без видаленого.

Модуль із обговореннями в реальному часі користувачі мають змогу безперерервно обмінюватися думками у вигляді коротких текстових повідомлень, без потреби перезавантажувати діалог. Модуль містить статичний список кімнат для обговорення, що може бути змінено лише командою підтримки вебсистеми. Вхідними даними є нікнейм користувача та ідентифікатор кімнати, до якої користувач здійснює процес під’єднання. Вихідними даними є графічний інтерфейс, що демонструє розмову та усі повідомлення від учасників кімнати, зберігаючи історію та початкову послідовність діалогу.

Модуль із автоматизованим підбором відео контенту дозволяє користувачеві отримати список із фільмами, що повинні йому сподобатися. На вхід приймається інформація про нещодавню активність користувача, його список улюблених фільмів та, власне, відповіді на питання із анкети.

Найвищий пріоритет мають жанри, з якими користувач найбільше взаємодіяв останнім часом, далі відповіді на питання та список улюбленого. Система аналізує ці дані, визначаючи список жанрів, використовуючи відповідний метод. Вихідними даними є інтерфейс користувача, що відображає список підібраного контенту на основі проаналізованої інформації.

Модуль із статистикою дозволяє побачити преференції користувача за обраний період часу. Вхідними даними для збору статистики є кожен фільм з яким взаємодіє користувач та його список улюбленого. Вихідними даними є інтерфейс користувача із графіком, що демонструє статистичну інформацію.

2.2 Розробка загальної моделі вебсистеми

Для глибшого розуміння роботи системи було створено її модель у вигляді діаграми класів з усіма основними сутностями, що використовуються (рис. 2.1).

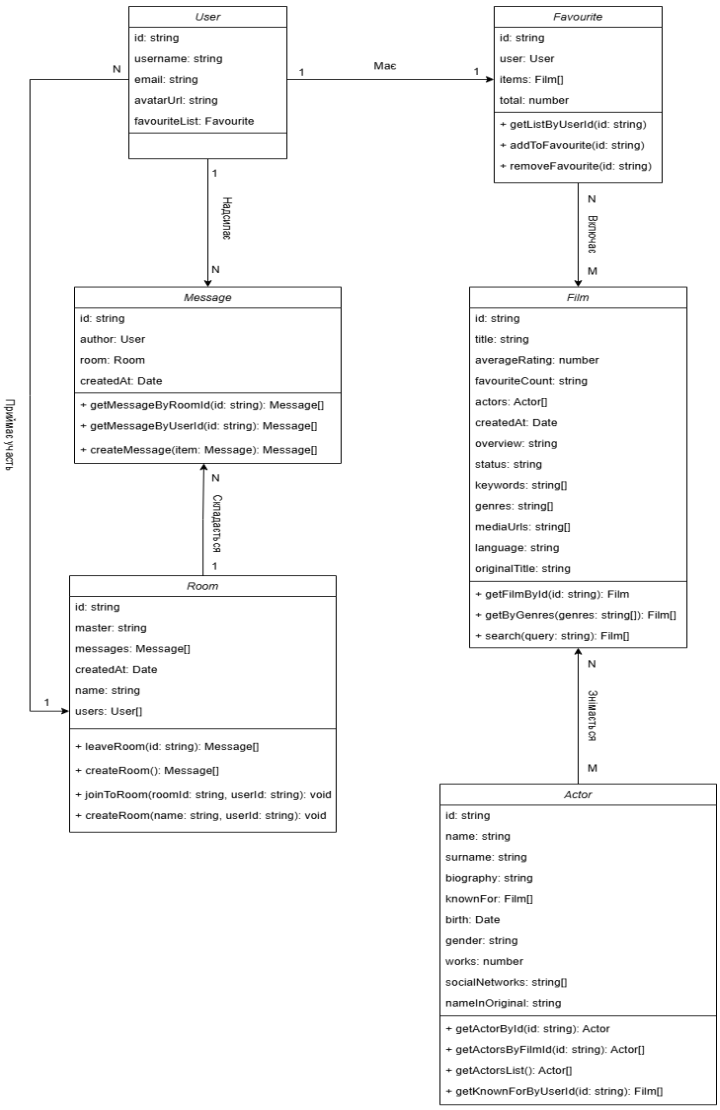

Рисунок 2.1 – Модель вебсистеми

Однією із основних сутностей системи є «User», що містить інформацію про користувача. Дана сутність містить «id», що є унікальним ідентифікатором користувача, «username» і «email», що є іменем і поштою, «avatarUrl» – картинка аккаунту і «favouriteList» це об’єкт типу класу «Favourite». Так як кожен користувач

обов’язково має список «Улюблене», навіть якщо він пустий, то екземпляр класу «User» має зв’язок із класом «Favourite» із типом зв’язку «один-до-одного», бо для кожного користувача створюється власний список з унікальним ідентифікатором.

Наступним по важливості класів є «Film», адже головна ідея вебсистеми – це кіноклуб для обговорення і підбору контенту, таким чином більшість із наповнення системи - фільми. Клас складається із великої кількості службових полів, що можна побачити на рисунку 2.1, але основні наступні: «title» і «originalTitle» – назва фільму в перекладі й оригіналі, «actors» – список акторів, що приймали участь у зйомках, це поле є масивом об’єктів типу «Actor», поле «genre» є масивом зі списком жанрів. Також клас «Film» містить методи, що спеціалізуються лише на роботі із основними завданнями класу і відповідає механізму об’єктно-орієнтованого програмування «інкапсуляція» [18]. Клас має зв’язок типу «багато-до-багатьох» із класом «FavouriteList», отже багато фільмів можуть відноситися до багатьох списків із улюбленим і навпаки.

Клас «Actor» є репрезентацією актора, що безпосередньо приймає участь у фільмах. Даний клас містить наступні поля: «name», «surname» що є представленням імені і прізвища, «biography», «knownFor» масив фільмів, де актор приймав участь, і безліч інших службових полів, що потрібні для відображення повної інформації про актора. Також даний клас містить методи для роботи з власними полями і відповідає лише за власний функціонал, тобто відповідає одному із принципів «SOLID», а саме «Single Responsibility». Має зв’язок типу «багато-до-багатьох» із класом «Film», так як один актор може бути учасником багатьох фільмів, а один фільм містить багато акторів.

Клас «Room» є кімнатою-чатом, що використовуватиметься у модулі із обговореннями контенту у реальному часі. Даний клас містить такі поля: «users» список учасників обговорення, «messages» – список повідомлень в обговорені, «name» що є іменем кімнати, і додаткові поля для інформації, що потрібна під час

розробки. Клас має зв’язок з типом «один-до-багатьох» із класом «Message». Саме цей тип зв’язку має підставу, бо один чат має відношення до багатьох повідомлень. Також клас має зв’язок «один-до-багатьох» з «User», адже багато користувачів можуть відноситися до одного обговорення.

Клас «Message», що відповідає за повідомлення у чатах з обговореннями контенту, містить такі поля: «text» - власне текст, «author» – автор, «room», аби розуміти до якої кімнати дане повідомлення відноситься і відмальовувати потрібну інформацію у правильному чаті. Має зв’язок типу «багато-до-одного», бо багато повідомлень можуть відноситися лише до одного чату, а не до багатьох одночасно.

Отже, було розроблено діаграму класів, де відображено основні сутності вебсистеми кіноклубу й зв’язки між ними. Додатково було описано тип зв’язків між сутностями вебсистеми. Усі діаграми та блок-схеми було розроблено у середовищі «Visio» [19].

2.3 Розробка загального алгоритму роботи програми і діаграм станів

Загальний алгоритм роботи програми у вигляді діаграми є основою для процесу успішної розробки будь-якого застосунку, а особливо, що стосується вебсистеми для кіноклубу, де наявні безліч сторінок і різноманітний функціонал.

Що стосується алгоритму, то перш за все система перевіряє чи користувач був раніше авторизований, шляхом зчитування токена доступу, що має термін придатності, близько 7-ми днів, тобто через тиждень користувач буде вимушений ввести дані знову. Вебсистема пропонує користувачеві створити акаунт, і якщо процес реєстрації пройдено успішно, то користувач буде перенаправлений на головну сторінку, де зможе взаємодіяти із контентом та головним меню. Якщо користувач зареєстрований, то йому відкриються додаткові функції, а саме доступ до модуля із обговореннями контенту та формування власного списку улюблених фільмів.

Загальний алгоритм роботи програми можна побачити на рисунку 2.2.

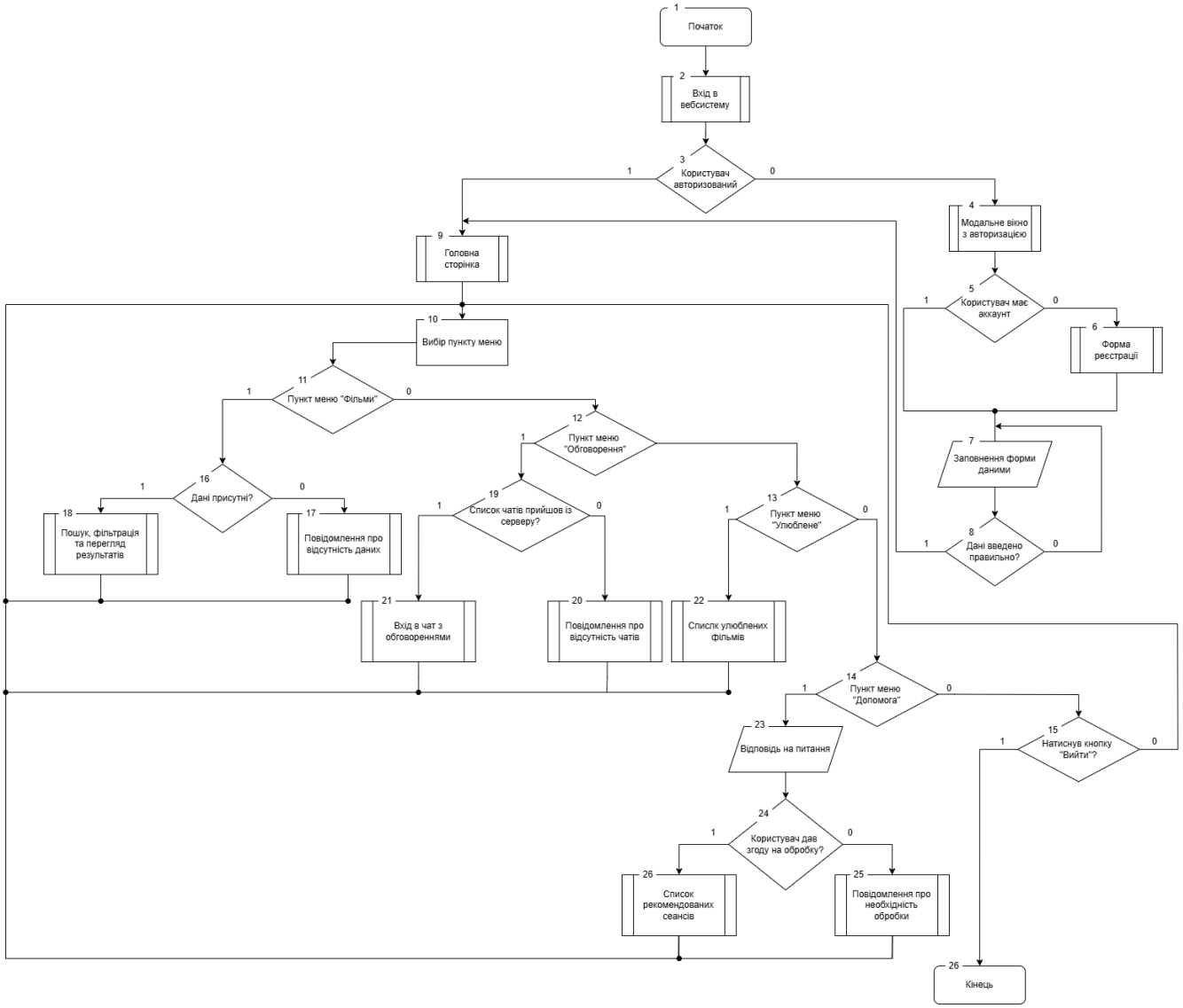

Рисунок 2.2 – Загальний алгоритм роботи вебсистеми кіноклубу

На рисунку 2.3 продемонстровано діаграму станів модуля обговорень контенту. Таким чином, користувач постійно отримує вхідні повідомлення від серверної частини з текстом і відгуком, що надсилають інші користувачі в реальному часі. Коли особа покидає чат, то інші користувачі отримують повідомлення із серверу з певним типом, що дозволяє реагувати на них потрібним чином і фільтрувати список тих, хто приймає участь в обговоренні, змінюючи кількість юзерів.

Рисунок 2.3 – Діаграма станів модуля обговорення контенту

Завдяки мережевому протоколу «WebSockets» є можливість повідомляти учасників обговорення, коли поточний користувач надсилає повідомлення, і так само, коли співбесідники діляться думкою, то клієнту прийде сигнал із сервера і список із повідомленнями оновиться автоматично.

Було продемонстровано загальний алгоритм роботи програми, де можна побачити увесь функціонал, що містить вебсистема. Також продемонстровано діаграми станів модулів обговорення.

2.4 Розробка методу автоматизованого підбору контенту

Вебсистема надає можливість взаємодіяти із функціоналом автоматизованого підбору контенту кожному відвідувачу, але для неавторизованих користувачів результати підбору базуватимуться лише, безпосередньо, на відповідях.

Метод автоматизованого підбору контенту надасть змогу користувачам почуватися почутими, бо система охоплюватиме усі аспекти їх активності:

1. Найвищий пріоритет під час підбору контенту має, безпосередньо, активність користувача протягом усього часу. Система відслідковуватиме контент, з яким взаємодіє користувач, і записуватиме у сховище категорії, що використовуватимуться в майбутньому.

2. Наступний за пріоритетністю є список улюбленого. В більшості випадків цей список є відображенням довгострокових вподобань користувача, на відміну від активності, що може змінюватися протягом певного часу.

3. Відповіді користувача під час опитування також дають змогу зрозуміти його вподобання в конкретний момент часу.

Таким чином, беручи до уваги усі 3 складові, вебсистема може запропонувати контент, що здатний зацікавити користувача.

Для авторизованих користувачів метод працює таким чином:

1. Вебсистема приймає відповіді користувача на запитання, що передаються через форму, що створено з використанням бібліотеки «Formik».

2. Після підтвердження форми з опитуванням викликається метод «onSubmit()», що приймає відповіді в ролі вхідного аргументу функції.

3. Так як відповіді на питання є об’єктом, де кожен ключ – назва поля у формі, то для обробки використовується метод «Object.keys()» та «Object.values()», що дають можливість отримати ключі та їх значення. Після отримання масиву з назвами ключів форми використовується «forEach()» метод, що на вхід приймає ключ форми і витягує назву жанру зі статичного списку, залежно від відповіді користувача. Завдяки використанню засобів мови «JavaScript» усі операції та цикли виконуються надзвичайно швидко.

4. Наступним етапом перевіряється наявність поля у «LocalStorage», яке має назву «recent» і містить в собі 5 жанрів з якими нещодавно взаємодіяв поточний користувач.

5. Далі отримується інформація про список улюблених фільмів і з використанням циклів обробляється кожен елемент і витягується список жанрів кожного фільму.

6. Два списки жанрів: отримані з форми та списку улюбленого зливаються в один масив, з використанням методу «Array.flat()» і настає час підрахунку частоти кожного виду.

7. Для підрахунку частоти кожного жанру використовується метод «Array.reduce()», що формує об’єкт із полями, де ключ – це назва жанру, а значення – його частота.

8. Під час фільтрації об’єкта беруться лише жанри, що зустрічаються більше 3х разів у двох списках: результати опитування та улюблене.

9. Активність користувача має найвищий пріоритет, отже 5 жанрів із сховища додаються безумовно.

10. З використанням мережевого протоколу «HTTP» та бібліотеки «Axios» здійснюється запит на сервер, результати якого проходять через парсинг, а саме через функцію, що має назву «parseMovies()». Результат роботи функції «parseMovies()» передається у відповідний компонент, що відповідає за відображення результатів.

Вище описаний метод вимагає від користувача проходження процедури авторизації для більш точного підбору, адже потрібно отримати доступ до особистої інформації: список вподобаного та нещодавня активність.

В основі методу лежить алгоритм, що реалізовано з використанням засобів та нативних методів, що надаються мовою програмування «JavaScript». Блок-схему алгоритму підбору контенту продемонстровано на рисунку 2.4.

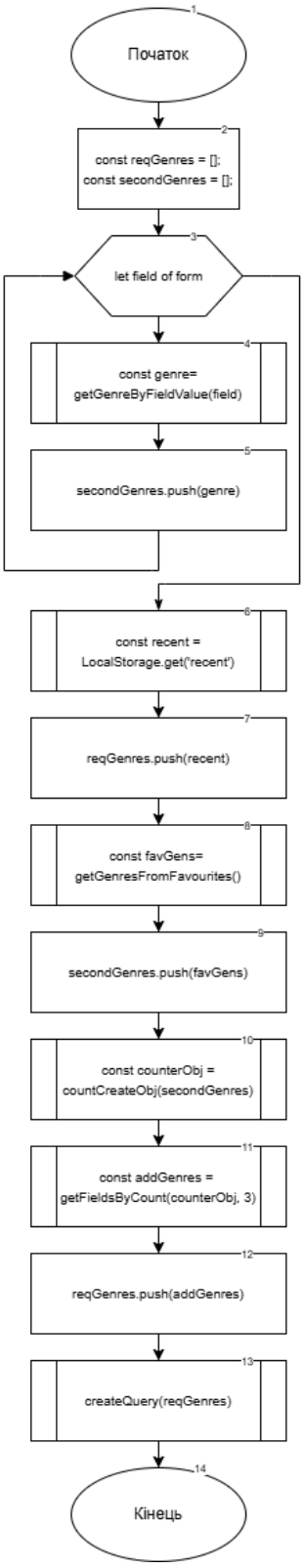

Рисунок 2.4 – Блок-схема алгоритму роботи методу автоматизованого підбору контенту

2.5 Розробка методу збору інформації про користувача та формування аналітичних графіків

Користувачі мають змогу переглядати власну статистику щодо вподобань жанрів. Таким чином кожен має змогу відслідковувати зміну своїх смаків та поглядів і аналізувати їх.

Статистика доступна не лише для користувачів, але і для власників кіноклубу, таким чином вони можуть аналізувати інформацію, щоб розуміти активність користувачів. Саме звітність про найбільш популярні жанри кіно у вигляді графіків за певний період дають змогу якнайкраще “відчути” аудиторію: пропонувати рекламу з відповідною тематикою, наповнювати головні сторінки контентом із популярними жанрами, розуміти власні помилки.

Метод не лише відслідковує той контент, що додається до списку «Улюблене», але й всю активність користувача, що допомагає формувати аналітичні дані якнайточніше.

Аналітичні дані зберігаються у сховищі, що надається компанією «Google».

Метод складається із таких кроків:

1. Використовуючи «FireBase» «Software Development Kit» (SDK), і метод «getIsAuthPassed()» система перевіряє, що користувач авторизований.

2. Якщо користувач авторизований, то розпочинається динамічне завантаження модуля із логікою звітності.

3. Здійснюється запит до бази даних із аналітичною інформацією для поточного користувача, і береться поле «general», в якому зберігається статистика по жанрах, що базується саме на списку «Улюблене». Якщо такої інформації немає, а у користувача є список улюбленого, то формується об’єкт та відправляється до бази даних, використовуючи метод «storage.set()», куди передається ідентифікатор користувача та інформація. Кожен раз, коли користувач додає або видаляє елемент зі списку із улюбленим – виконується дана логіка.

4. При кожному заході на сторінку фільму виконується функція, що отримує список жанрів поточного фільму.

5. З використанням «FireBase SDK» виконується запит до бази даних, де зберігається інформація по кожному користувачу, включно із аналітикою. В тіло запиту передається ідентифікатор користувача, щоб база розуміла, яку інформацію потрібно надати.

6. Після отримання інформації, використовуючи засоби мови JavaScript виконуються маніпуляції над отриманими даними у методі «getAnalyticsParsed()»: береться поле, що відповідає сьогоднішній даті, якщо його немає, то воно створюється. Далі використовуючи «for of» цикл і в об’єкт записується ключ, що є назвою жанру, а значення – кількість взаємодій із ним.

7. Для оновлення інформації у базі даних викликається метод «storage.set()», куди передається ідентифікатор користувача та оброблена інформація. Таким чином дані завжди будуть синхронізовані.

Завдяки тому, що інформація корелюється із датою, отримання аналітики за певний період стає ще легшою і менш ресурсозатратною зі сторони клієнта, адже усі дані посортовані, безпосередньо, у сховищі «FireBase».

Зовнішнє сховище надає змогу зберігання без загрози видалення. Таким чином, дані стають незалежними від функціонування вебсистеми і доступні в будь-який час.

Описаний метод містить в собі кроки взаємодії із зовнішнім сховищем з даними, їх отримання та обробки потрібним чином. Усі дані зберігаються у сховищі в форматі «JavaScript Object Notation» (JSON), що дозволяє передавати інформацію у запитах, наприклад, використовуючи мережевий протокол «HTTP».

На рисунку 2.5 продемонстровано блок-схему алгоритму формування звітності імплементованого засобами мови програмування «JavaScript» та «SDK FireBase».

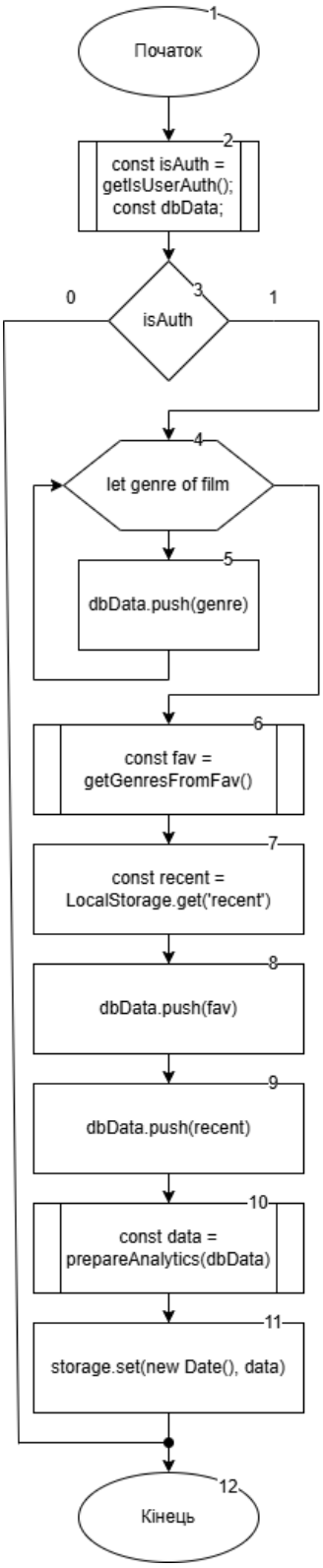

Рисунок 2.5 – Блок-схема алгоритму роботи методу збору інформації про користувача та формування аналітичних графіків

Завдяки інструментам, що надаються «FireBase» взаємодія із сховищем стає набагато легшою. Цей інструмент надає вже імплементовані методи, що містять необхідну логіку, необхідно лише здійснити інсталяцію пакету. Отже, розроблений метод лежить в основі модуля із аналітичною інформацією.

2.6 Висновки

У другому розділі було проведено аналіз вхідних та вихідних даних вебсистеми кіноклубу для обговорення і підбору контенту. Створено та пояснено загальний алгоритм роботи вебсистеми. Розроблено такі алгоритми:

1. Алгорим підбору персоналізованого контенту, на основі короткого опитування користувача та аналізі його дій.

2. Алгоритм обміну повідомленнями у реальному часі під час обговорення фільмів.

Розроблено методи: автоматизованого підбору контенту, збору інформації про користувача та формування аналітичних графіків. Логіку обох методів візуалізовано у вигляді блок-схеми.

Додатково продемонстровано діаграму станів модуля обговорення. Описано розроблені методи автоматизованого підбору контенту та збору інформації про користувача з метою формування аналітичних графіків.

Усі діаграми, включаючи модель роботи вебсистеми, було розроблено за допомогою «UML».

3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ СИСТЕМИ

3.1 Варіантний аналіз і обґрунтування вибору засобів для реалізації вебсистеми

Починаючи або плануючи розробку вебсистеми кіноклубу для обговорення і підбору контенту надзвичайно важливо обрати найнеобхідніші технології та бібліотеки, аби не збільшувати розмір кінцевого проєкту, адже це безпосередньо впливає на швидкість завантаження сторінок.

«NextJs» є популярним і потужним фреймворком для розробки вебсистем і був створений компанією «Vercel», що націлені не лише на зовнішній вигляд інтерфейсу, а також на швидкість роботи системи та її показників «SEO», що допоможе підвищити її позиції у пошукових списках. Даний фреймворк є найкращим рішенням, так як він базується на основі бібліотеки для створення користувацьких інтерфейсів, що має назву «React». Таким чином, в одному фреймворку присутні інструменти для оптимізації вебсистеми та створення привабливого користувацького інтерфейсу.

Основною мовою програмування було обрано «TypeScript», що було створено на основі «JavaScript». «TypeScript» є статично типізованою мовою програмування і дозволяє легко дотримуватися раніше описаних інтерфейсів, аби у місця, де потрібні цифри, не потрапили рядки, або там, де потрібно передати масив – не потрапив об’єкт. Типізація дозволяє пришвидшити розробку та зменшити кількість помилок під час експлуатації. У таблиці 3.1 наведено порівняння «JavaScript» з іншими мовами.

Таблиця 3.1 – Порівняльна мов програмування

Критерії	Java	C++	Python	JavaScript
Економія пам’яті	0	0	0	1
Легкість засвоєння мови	0	0	0	1

Продовження таблиці 3.1

Критерії	Java	C++	Python	JavaScript
Методи для взаємодії із браузерним середовищем	0	0	0	1
Нативна підтримка асинхронних операцій	1	1	1	1
Кроссплатформенність	1	0	1	1
Сумарний коефіцієнт	2	1	2	5

Відповідно до порівняння у таблиці 3.1 - «JavaScript» разом із типізацією є найкращим рішенням для розробки вебсистеми, так як містить наттивні методи для взаємодії із браузером.

Для імплементації логіки серверної частини буде використано фреймворк «NestJs», що полегшує створення необхідного функціоналу, шляхом готових функцій, що надаються ним.

MaterialUI [20] – це бібліотека компонентів «React» з відкритим кодом, яка реалізує «Material Design» від «Google». Для побудови привабливого інтерфейсу користувача буде використано бібліотеку «MaterialUI», адже вона містить готові елементи: кнопки, поля для вводу, таблиці, випадаючі списки, контейнери та багато інших речей, що стануть в нагоді під час розробки.

Здійснювати запити на сервер, використовуючи протокол «HTTP» і основні методи: «GET», «POST», «PUT», «DELETE», для отримання контенту допоможе «Axios».

ChartsJs [21] – це популярна бібліотека «JavaScript» для створення інтерактивних та адаптивних графіків на вебсторінках. Вона використовує «HTML5» і дозволяє легко візуалізувати дані у вигляді різних типів діаграм. Тому дана бібліотека використовуватиметься для візуалізації аналітичних даних вебсистеми: рейтинг акторів, відповідно до країни.

Socket.IO [22] – «JavaScript» бібліотека для вебзастосунків і обміну даними в реальному часі. Складається з двох частин: клієнтської, яка запускається в браузері

і серверної для «Node.js». Обидва компоненти мають схожий прикладний програмний інтерфейс. Функціонал, що надається бібліотекою «Socket.IO» дозволить імплементувати необхідну основу для встановлення безперервного з’єднання, як на клієнті, так і на серверній частині.

SWR [23] – це стратегія, яка спочатку повертає дані з кешу (застарілі), потім надсилає запит (ревалідація), і в результаті повертає актуальні дані. З «SWR», компоненти отримають потік даних, що постійно і автоматично оновлюються. І інтерфейс користувача завжди буде швидким і реактивним. Дана стратегія надає широкий спектр налаштувань, починаючи від часу, через який буде проведено ревалідацію, закінчуючи створенням нескінченного списку з динамічним підвантаженням інформації. Власне, саме дана стратегія використовуватиметься для створення каталогів з фільмами та акторами, адже у базі може бути присутня інформація про десятки тисяч осіб та кінокартин, тому важливо відвантажувати їх поступово, на вимогу користувача.

«Formik» і «Yup» це бібліотеки, що зазвичай працюють разом, надаючи засоби для створення реактивних форм з валідацією. «Formik» і «Yup» використовуватимуться для створення форми реєстрації та авторизації.

Лише поєднання вище описаних бібліотек та стратегій дозволить розробити дійсно реактивну та оптимізовану вебсистему кіноклубу для обговорення і підбору контенту. «Socket.IO» надасть інструменти для імплементації можливості взаємодії у реальному часі між великою кількістю користувачів, а «Axios» та «SWR» для отримання контенту з серверу, під час формування списку з контентом для користувача, на основі короткого опитування.

Для зберігання інформації про користувачів: персональні дані, список вподобаних фільмів й інша технічна інформація, було використано сховище від компанії «Google», що має назву «Firebase» і є безкоштовним при лімітованому використанні і запитах.

Firebase [24] – це платформи розробки мобільних та вебзастосунків. Він містить у собі все необхідне для створення успішного додатку. «Cloud», «Hosting», «TestLab», «Performance Monitoring», «CrashReporting», «A/B Testing» та багато іншого всередині. Дана платформа також надає можливість автентифікації через сторонні ресурси: «Facebook», «Google» або «Instagram», що дає можливість створити платформу, що працюватиме без обмежень.

Visual Studio Code [25] – це безкоштовне інтегроване середовище розробки (IDE) від «Microsoft». Розробка сайтів і додатків включає «Visual Studio Code». «VS Code» має розширення для роботи з різними мовами програмування, включаючи «PHP», і надає інструменти для редагування коду та роботи з системами контролю версій.

Отже, вибір технологій для розробки програмних засобів вебсистеми кіноклубу, що вміщає в собі спілкування у реальному часі та підбір контенту, включатиме такі фреймворки: «NextJs» та «NestJs», для здійснення запитів – стратегія «Stale While Revalidate». Для відображення аналітичних даних використовуватиметься «ChartJs» і декілька інших бібліотек, але саме вище описані мають вирішальну роль під час розробки, створюючи підґрунтя для масштабованої і реактивної вебсистеми. Поєднання цих інструментів надає комплексний підхід до розробки, що дозволяє створювати продукт з високим рівнем функціональності та надійності.

3.2 Розробка модуля обговорення контенту в реальному часі

Модуль для обговорення контенту в реальному часі містить функціонал для дискусій вподобань та фільмів у створених «кімнатах». Кожен користувач може знайти «кімнату» для обговорення, відповідно до своїх вподобань та улюблених жанрів. Кожен авторизований користувач може з легкістю під’єднатися до будь-якої розмови.

Під час реалізації модуля для обговорення контенту в реальному часі варто враховувати ряд аспектів. Перш за все, варто правильно обрати мережевий протокол, щоб була можливість обміну повідомленнями в реальному часі, без потреби перезавантажувати сторінку. Найкращим варіантом для імплементації потрібної логіки є протокол «WebSockets» і бібліотека «Socket.IO», що містить необхідні інструменти для створення потрібних функцій, як на серверній частині, так і на клієнтській.

Для підтримки протоколу «WebSockets» необхідно встановити бібліотеку «Socket.IO» і створити сервер, використовуючи клас, що імпортується з бібліотеки, і має назву «Server».

На рисунку 3.1 продемонстровано процес впровадження підтримки «WebSockets» протоколу. Спершу створюється, власне, сервер на основі змінної «app». Потім, змінна «server«, що є репрезентацією звичайного сервера, передається параметром у конструктор класу «Server», що імпортується із бібліотеки «Socket.IO», і додає підтримку потрібного протоколу на бек-енд.

```
const { Server } = require('socket.io');

app.use(cors());
app.use(express.json());
app.use(express.urlencoded({ extended: true }));

const server = http.createServer(app);
const io = new Server(server, {
    cors: {
        origin: 'http://localhost:3000',
        methods: ['GET', 'POST'],
    },
});
```

Рисунок 3.1 – Створення сервера з підтримкою «WebSockets»

Далі, необхідно створити назви подій, що відсилатимуться клієнтською і серверною частинами для індикації певних процесів. Таким чином, основні події і їх визначення:

1. «ROOM:CONNECT» – це подія, що повинна відправити клієнтська частина, коли користувач намагається ввійти до кімнати із обговореннями.

2. «ROOM:JOINED» – тип події, що надсилається серверною частиною для усіх учасників кімнати, задля індикації того, що з’явився новий співрозмовник. «ROOM:JOINED» відсилається одразу після того, як клієнтська частина надіслала «ROOM:CONNECT». Той хто приєднався отримає актуальні дані, і учасники також будуть проінформовані, шляхом відправки «ROOM:JOINED».

3. «CHAT:SEND_MESSAGE» – подія, що надсилається клієнтом, коли користувач відправляє повідомлення. Дані, що передаються з клієнта обов’язково повинні містити ідентифікатор кімнати і саме повідомлення, аби серверна частина розуміла, до якої кімнати відноситься надісланий текст.

4. «CHAT:MESSAGE:RECEIVE» – подія для всіх співрозмовників, що виступає індикатором того, що необхідно оновити інформацію, окрім відправника, із його текстом. Відправляється серверною частиною одразу після отримання сигналу із клієнта з повідомленням «CHAT:SEND_MESSAGE».

5. «CHAT:EXIT» – індикатор того, що користувач має намір покинути розмову. Містить ідентифікатор користувача і кімнату, аби сервер очистив потрібний список, відповідно до переданих даних.

6. «CHAT:LEAVE» – надсилається серверною частиною для того, аби всі учасники дискусії оновили список тих, хто все ще присутній. На клієнт передається ідентифікатор користувача, що покинув розмову, після чого вебсистема профільтрує список учасників і прибере відповідного.

На клієнтській частині необхідно впровадити підтримку таких повідомлень із серверної частини: «ROOM:JOINED», «CHAT:MESSAGE:RECEIVE»,

«CHAT:LEAVE». Як тільки користувач приєднується до обговорення, він автоматично починає прослуховувати всі події, для того, аби одразу реагувати на них, якщо вони надходять із сервера.

Схему інтерфейсу користувача модуля для обговорень контенту, а саме початкового екрану наведено на рисунку 3.2.

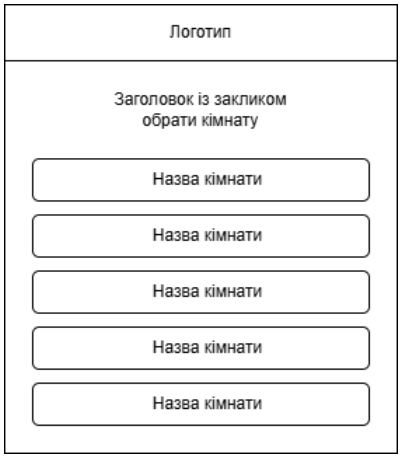

Рисунок 3.2 – Схема інтерфейсу користувача початкового екрану вікна з обговореннями

На рисунку 3.3 наведено готову реалізацію інтерфейсу користувача початкового екрану вікна з обговореннями.

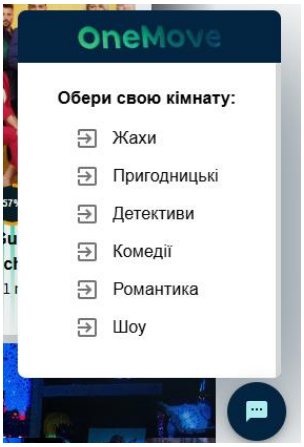

Рисунок 3.3 – Інтерфейс користувача початкового екрану вікна з обговореннями

На рисунку 3.4 продемонстровано схему інтерфейсу кімнати з обговореннями певного жанру.

Рисунок 3.4 – Схема інтерфейсу кімнати з обговореннями

На рисунку 3.5 продемонстровано реалізацію інтерфейсу користувача кімнати з обговореннями.

Рисунок 3.5 – Інтерфейс кімнати з обговореннями

Інтерфейс модуля обговорень контенту в реальному часі було виконано у стриманих кольорах, інтуїтивно зрозумілим, дотримано загальноприйнятих вимог щодо структури «HTML» розмітки та використано лише семантичні теги, що надасть змогу людям із обмеженими можливостями використовувати вебсистему.

3.3 Розробка модуля улюбленого

Надзвичайно важливо надавати змогу користувачеві зберігати контент, що йому сподобався.

Не менш важливо зберігати таку інформацію на незалежному носії, щоб дані були незалежними від працездатності вебсистеми. На рисунку 3.6 продемонстровано алгоритм роботи модуля «Улюблене».

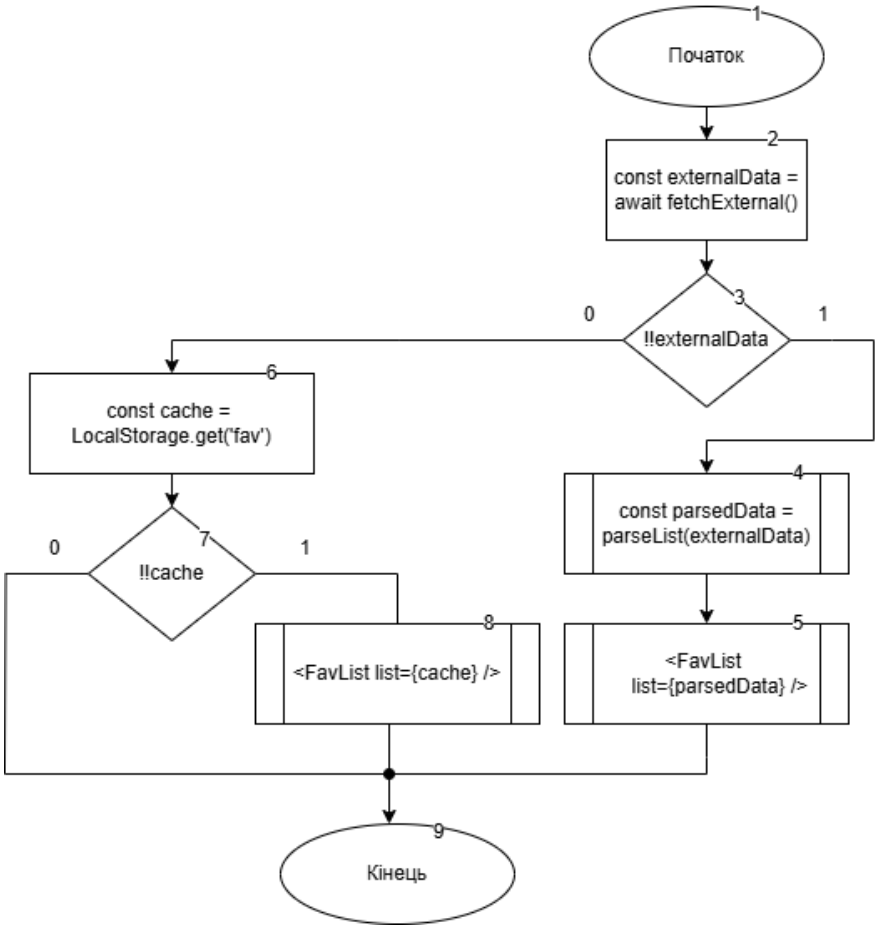

Рисунок 3.6 – Алгоритм роботи модуля «Улюблене»

У ролі зовнішнього сховища було використано «FireBase Storage», що надається компанією «Google» і постійно знаходиться в працюючому стані. Важливо, щоб зовнішній носій завжди функціонував справно, адже модуль, що відповідає за список «Улюблене» постійно взаємодіє із зовнішнім сервісом.

Задля підвищення рівня незалежності вебсистеми, інформація про улюблений контент додатково зберігається у браузерному сховищі «LocalStorage», і буде відображена у разі несправного функціонування зовнішніх сервісів.

На рисунку 3.7 продемонстровано схему інтерфейсу списку «Улюблене».

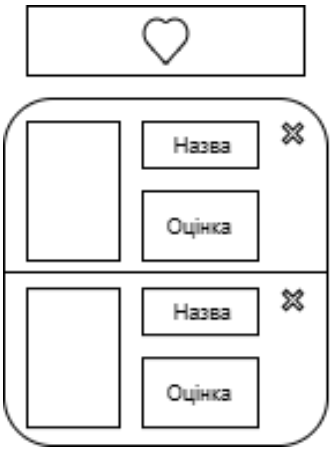

Рисунок 3.7 – Схема інтерфейсу списку улюбленого

На рисунку 3.8 готова імплементація списку улюбленого контенту.

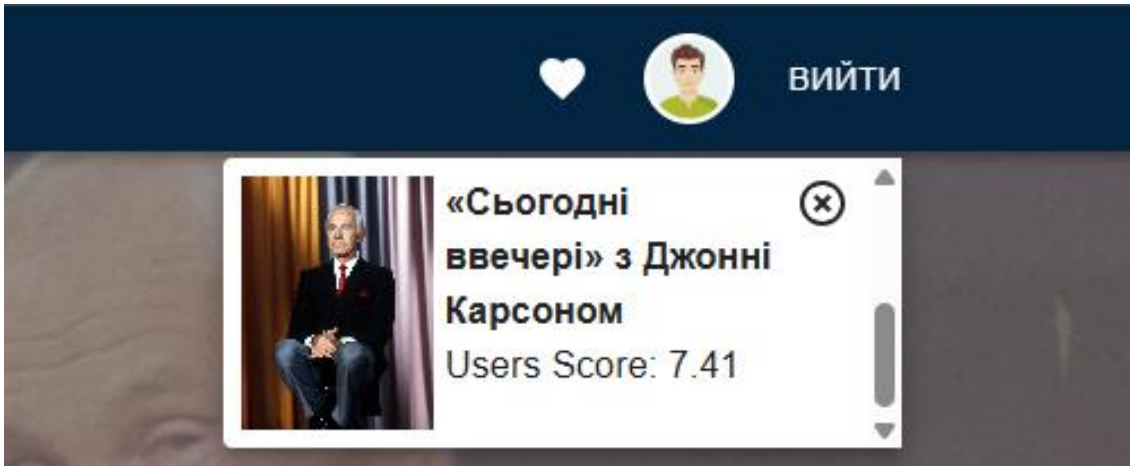

Рисунок 3.8 – Інтерфейс списку улюбленого

Інтерфейс користувача надає змогу не лише додавати елементи до списку, в односторонньому порядку, але й видаляти, редагуючи його.

Під час видалення елементу здійснюється запит до сховища із фільтрованим списком, де відсутній видалений елемент.

Якщо у користувача відсутні вподобані фільми, то в такому випадку список не буде продемонстровано.

3.4 Розробка модуля автоматизованого підбору контенту

Основне призначення модуля підбору контенту полягає в допомозі користувачеві у знаходженні фільмів або серіалів, що відповідатимуть його вподобанням. Такий підхід до взаємодії з користувачами допомагає почуватися їм краще, адже вони відчувають зацікавленість кіноклубу у залученні нової аудиторії і добросовісне ставлення до власного продукту.

Таким чином, форма із питаннями для користувача повинна виглядати лаконічно, бути інтуїтивно зрозумілою та добре продуманою. На рисунку 3.6 продемонстровано схему інтерфейсу форми з питаннями для користувача.

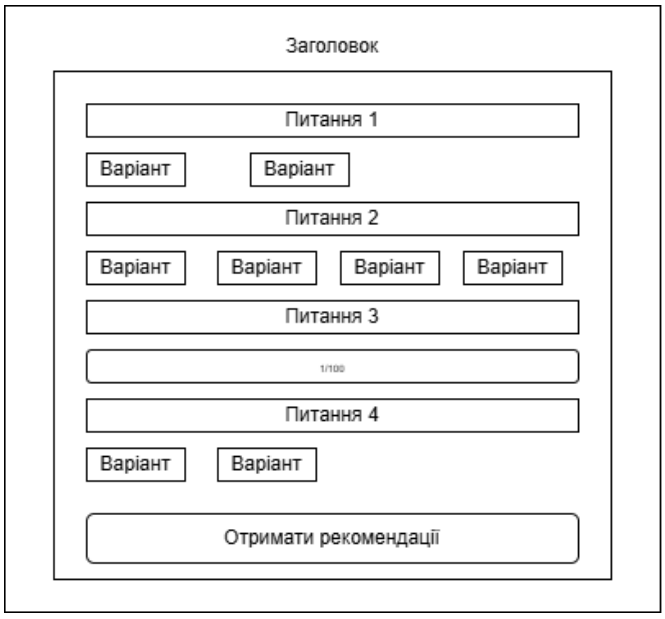

Рисунок 3.6 – Схема інтерфейсу форми з опитуванням

Анкета є інтуїтивною і легкою у використанні. Вона складається із заголовка, щонайменше 10-ти питань різного формату і кнопки для підтвердження відповідей і отримання результатів.

Під час створення форми була використана бібліотека «MaterialUI», що містить готові компоненти зі стилями. Для створення, власне, форми був використаний «Formik», а для валідації введеної інформації «Yup». Таким чином, у співпраці «Formik» та «Yup» є надзвичайно сильним засобом для контролю форм, їх станів та допомагає налаштовувати спосіб реагування на різноманітні події.

Готовий інтерфейс форми продемонстровано на рисунку 3.7.

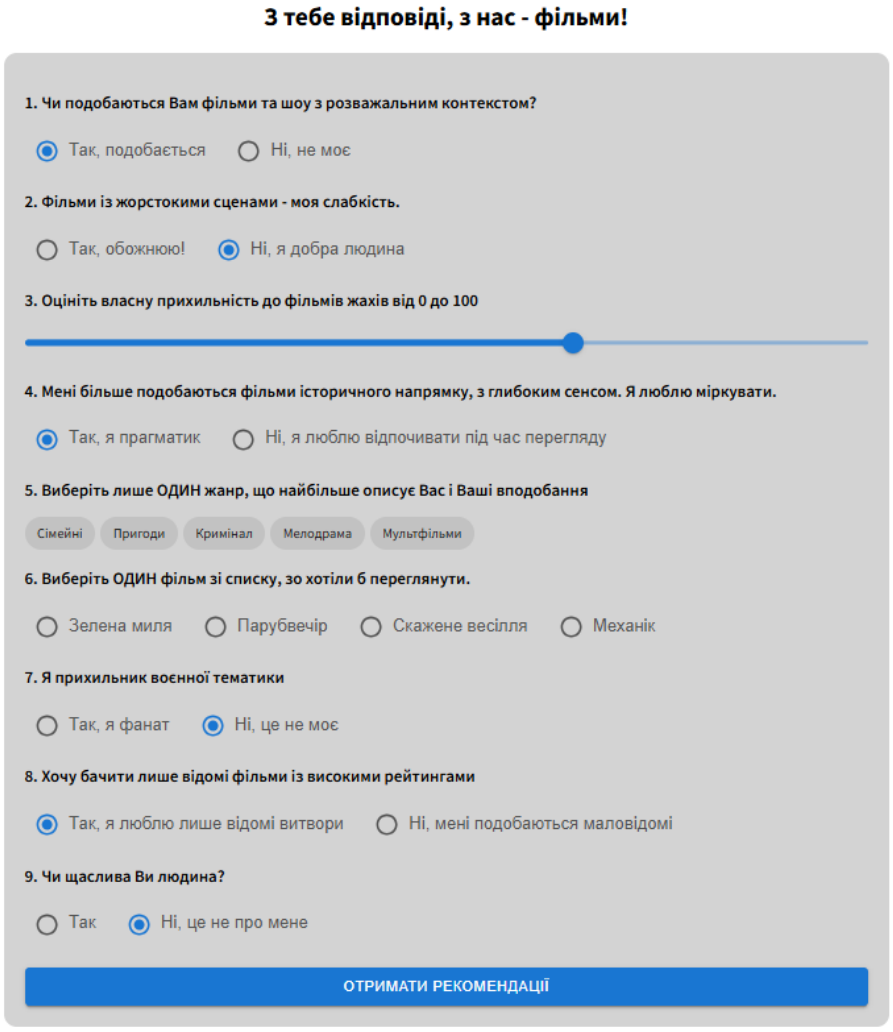

Рисунок 3.7 – Форма з опитуванням для користувача

Після підтвердження обраних відповідей, запускається механізм, в основі якого лежить алгоритм підбору контенту, що бере до уваги (порядок списку відповідає пріоритетності):

- нещодавня активність користувача;
- жанри фільмів, що додані до списку вподобаного;
- відповіді на питання анкети.

Вказані складові обробляються відповідним чином, після чого використовується цикл, щоб сформувати квері параметр. Якщо рядок із параметрами сформовано успішно, то здійснюється запит на сервер для отримання відповідного контенту, а оброблені результати опитування – в тіло запиту.

Після отримання результатів із сервера, на клієнті зникає форма із питаннями і демонструється список із контентом, що прийшов.

На рисунку 3.8 наведено схему інтерфейсу з отриманими пропозиціями контенту.

Рисунок 3.8 – Схема інтерфейсу списку із пропозиціями контенту відповідно до результатів опитування

На рисунку 3.9 продемонстровано готову реалізацію інтерфейсу, відповідно до розробленої схеми на рисунку 3.8, із пропозиціями контенту на основі опитування. Кожен елемент списку містить лише найнеобхіднішу інформацію:

заголовок, рік випуску, індикатор рейтингу й, найголовніше, обкладинка фільму. При бажанні, користувач має змогу перегенерувати підібраний список контенту, натиснувши на відповідну кнопку. Під час регенерації попередні відповіді на питання будуть взяті до уваги і запит на сервер буде здійснено з інформацією, що базується на них. Список контенту виконано у вигляді слайдера із можливістю гортання вправо-вліво, таким чином дана секція не буде займати багато місця, що є комфортним для користувача.

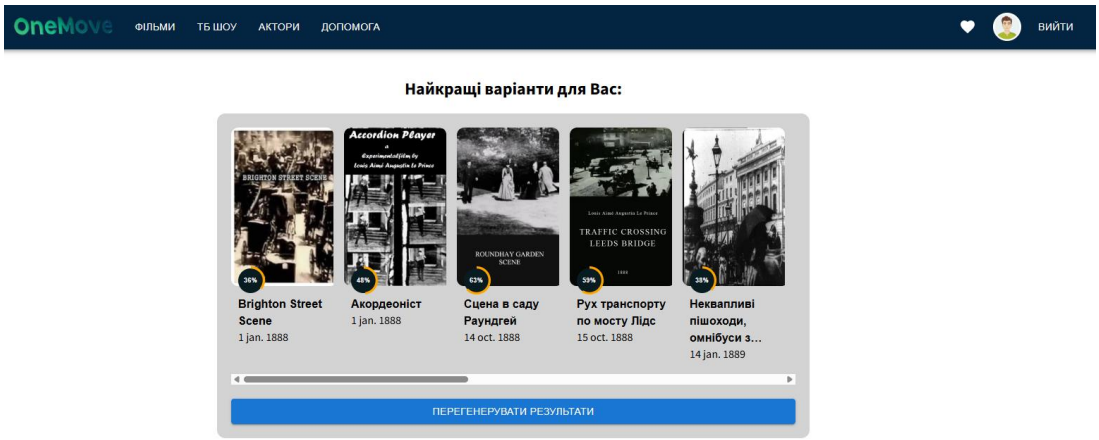

Рисунок 3.9 – Списку із пропозиціями контенту відповідно до результатів опитування

3.6 Розробка модуля статистики

Призначення статистичного модуля є різним для обох сторін:

1. Для користувача – це перегляд зміни власних вподобань, щоб краще розуміти себе.

2. Для власників – це незамінний інструмент, що допомагає розуміти загальні тенденції вподобань користувачів усієї вебсистеми. Саме розуміння аудиторії допомагає краще пропонувати персоналізовану рекламу.

Для кращого розуміння роботи модуля, на рисунку 3.10 продемонстрованого загальний алгоритм роботи аналітики.

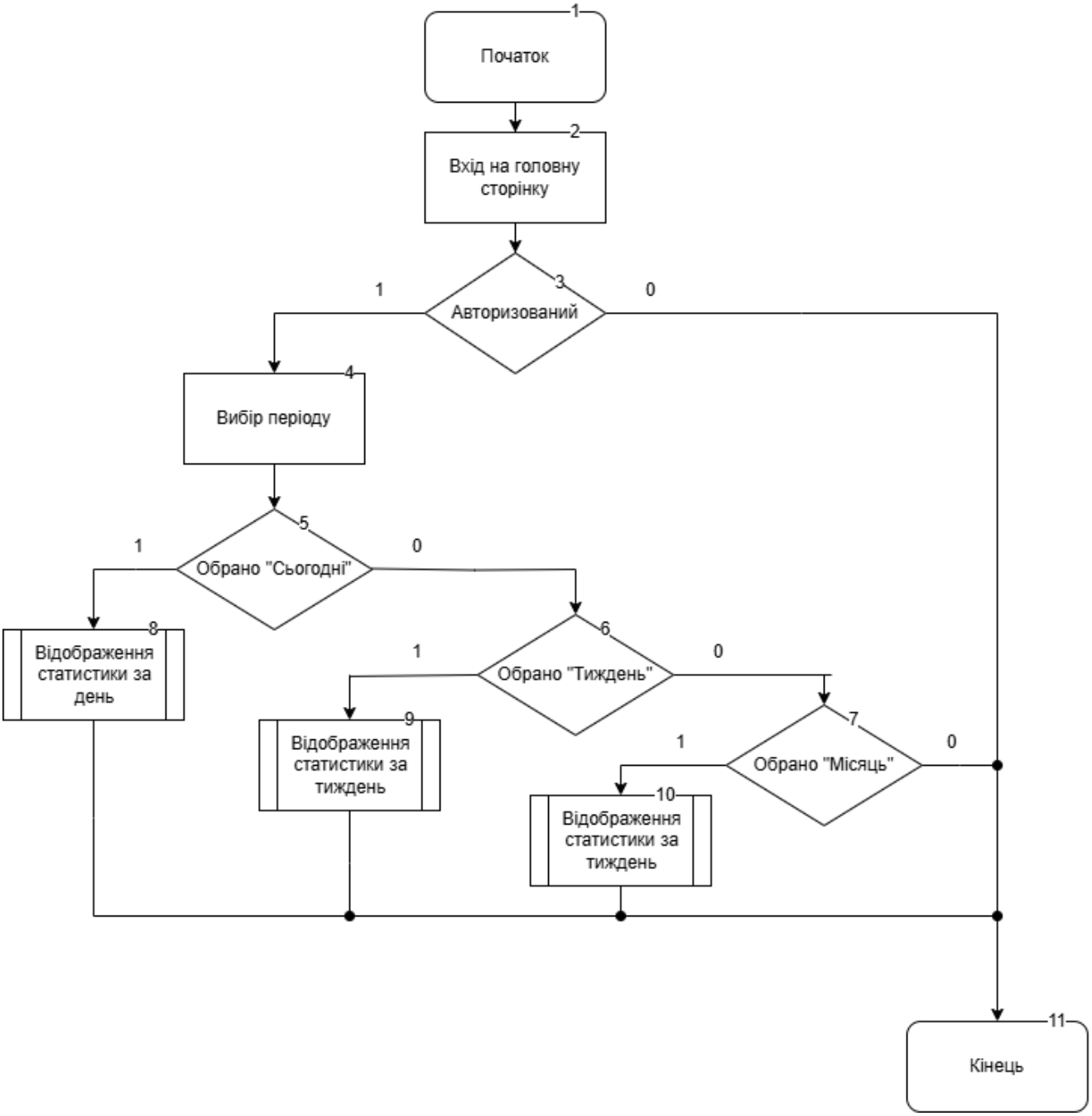

Рисунок 3.10 – Загальний алгоритм роботи статистичного модуля

Користувач має змогу обирати період за який бажає переглянути інформацію: день, місяць або тиждень. В залежності від обраного проміжку часу графік із категоріями будуватиметься по-різному, так як оброблятиметься інша кількість даних.

На рисунку 3.11 продемонстровано схему інтерфейсу користувача.

Рисунок 3.11 – Схема інтерфейсу користувача модуля аналітики

На рисунку 3.12 продемонстровано готову імплементацію інтерфейсу.

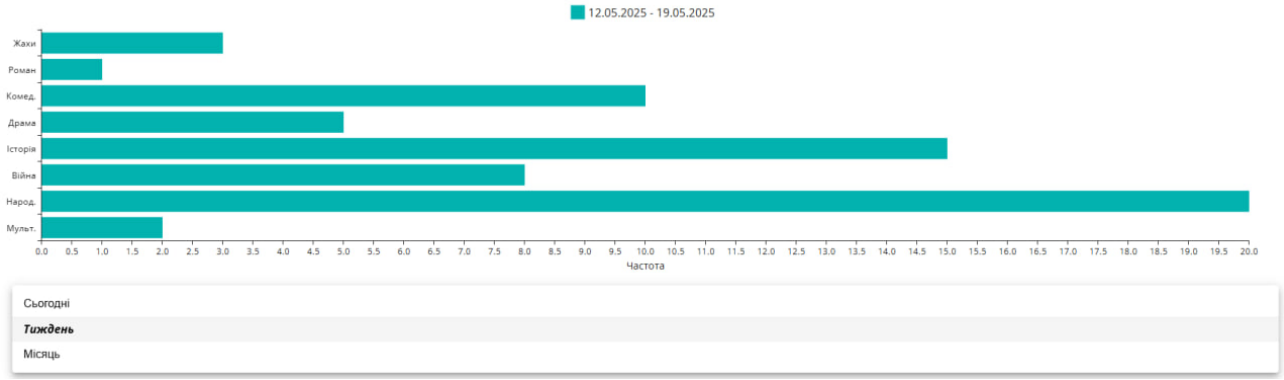

Рисунок 3.12 – Інтерфейс користувача модуля аналітики

Інтерфейс є повністю інтуїтивно зрозумілим. Користувач із будь-яким рівнем підготовки зможе взаємодіяти з ним, що робить його універсальним.

Зверху продемонстровано дату, яку охоплює обраний період. Нижня шкала є відображенням частоти, а права – назви категорій.

3.7 Розробка інтерфейсу користувача вебсистеми

Під час розробки вебсистеми кіноклубу було важливо створити інтерфейс, що не лише зручний у використанні, а й передає атмосферу кіно. Основним кольором обрано темний відтінок синього, що близький до бірюзового, який асоціюється з кінозалом, вечірнім переглядом фільмів та спокійною атмосферою. Такий колір допомагає зосередитися на контенті та додає інтерфейсу відчуття глибини, не відволікаючи увагу користувача від важливої інформації.

Контрастні допоміжні кольори – білий і чорний — забезпечують зручність читання, виділення ключових елементів і створюють візуальний баланс. Разом ця палітра формує сучасний та привабливий дизайн, який підкреслює унікальність платформи для поціновувачів кіно.

Після відкриття вебсистеми, користувач потрапляє на головну сторінку, де продемонстровано актуальні фільми та шоу у кожній із сфер: кінотеатри, телебачення, телешоу. Також на головній сторінці користувач має можливість відкрити список із кімнатами для обговорення контенту. На рисунку 3.13 продемонстровано головну сторінку вебсистеми кіноклубу.

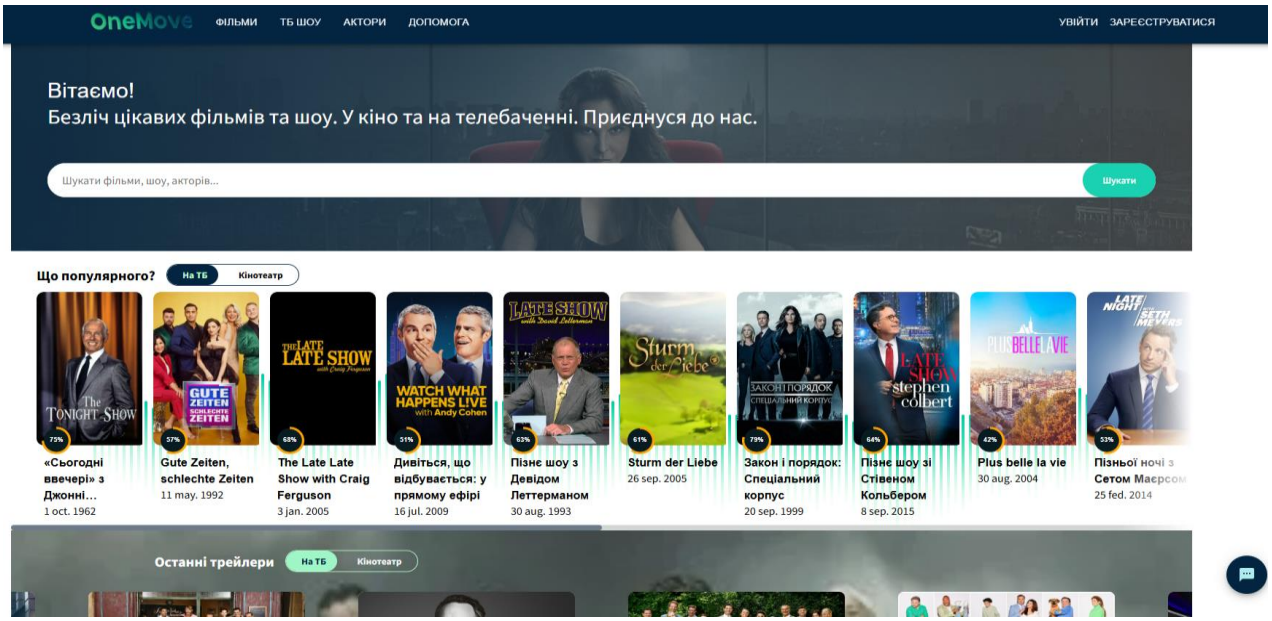

Рисунок 3.13 – Головна сторінка вебсистеми

Для легкого доступу до кожного окремого модуля вебсистеми було створено навігаційне меню, що залишається зверху під час переходу між сторінками і містить в собі посилання на всі важливі частини програми. На рисунку 3.14 продемонстровано навігаційне меню із відкритим розділом «Фільми».

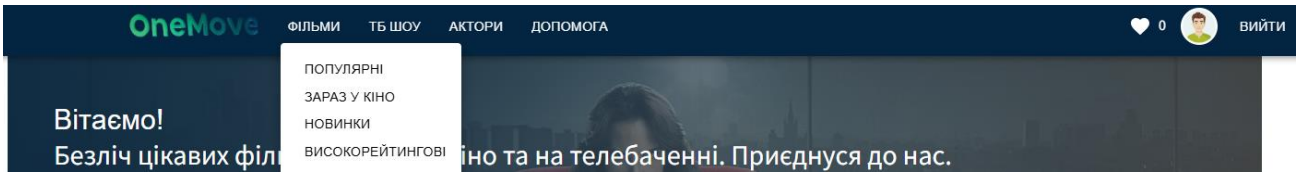

Рисунок 3.14 – Навігаційне меню

Для лаконічної демонстрації контенту було створено слайдер із перемикачем, що містить 2 кнопки: «На ТБ» та «Кінотеатр». Це було розроблено з метою економії місця на сторінці, так як кожен різновид, як телебачення так і кінотеатр містить власні популярні і рейтингові фільми та шоу, тому їх можна вмістити в один слайдер із можливістю перемикання. На рисунку 3.15 продемонстровано слайдер, що містить контент для телебачення та кінотеатру.

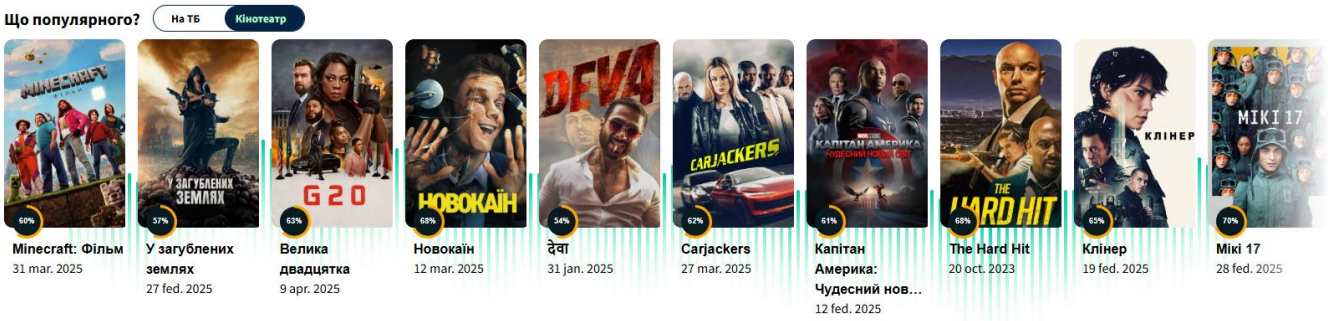

Рисунок 3.15 – Слайдер із контентом

Для повноцінного використання вебсистеми необхідно пройти процес авторизації. Форма для входу в акаунт була розроблена інтуїтивно зрозумілою, українською мовою та із функцією валідації, аби вказувати на помилки користувача. Повідомлення про помилку було вирішено відображати із червоним

фоном, так як цей колір асоціюється із помилками та небезпекою. На рисунку 3.16 продемонстровано форму для входу.

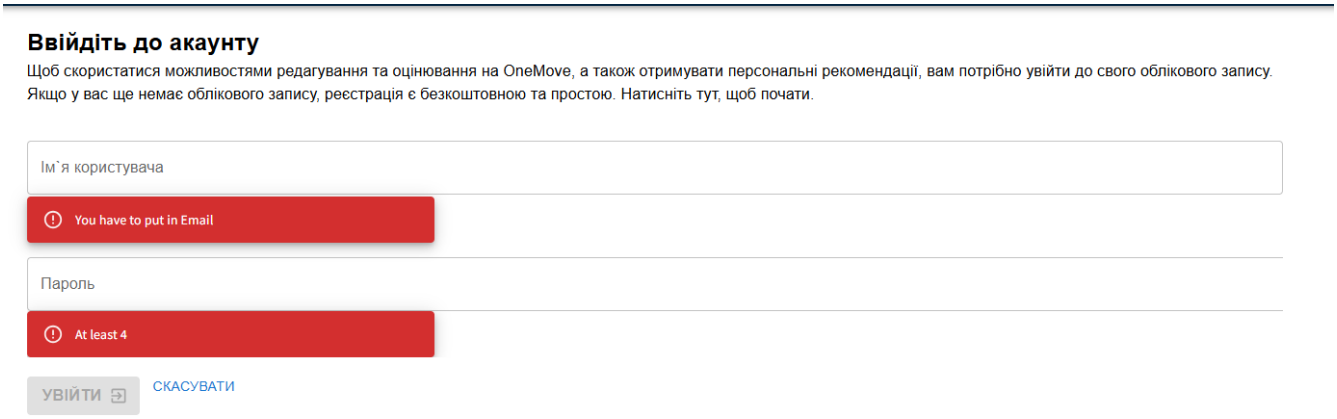

Рисунок 3.16 – Форма для входу

Після входу користувач має доступ до нового функціоналу. Одна із функцій, що доступна лише зареєстрованим користувачам – додавання до списку улюбленого. На рисунку 3.17 подемонстровано сторінку фільму, де наведено інформацію про нього та є кнопка для додавання до списку вподобаного.

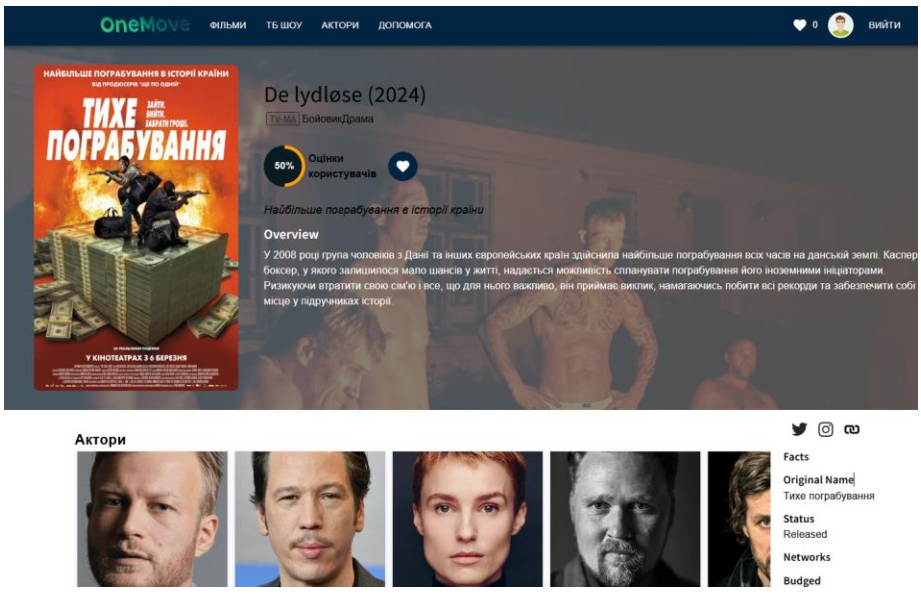

Рисунок 3.17 – Сторінка фільму

Окрім інформації про фільми, вебсистема налічує сотні персональних сторінок акторів. Для того, аби потрапити на сторінку певного актора, достатньо ввести його ім’я в пошуку, або перейти на нього через сторінку фільму. На кожній сторінці фільму або шоу є інформація про акторів, що приймали участь в процесі зйомок. Є дві опції для того, щоб отримати інформацію про кумира: сторінка фільму або пошук.

На рисунку 3.18 продемонстровано список пошуку актора.

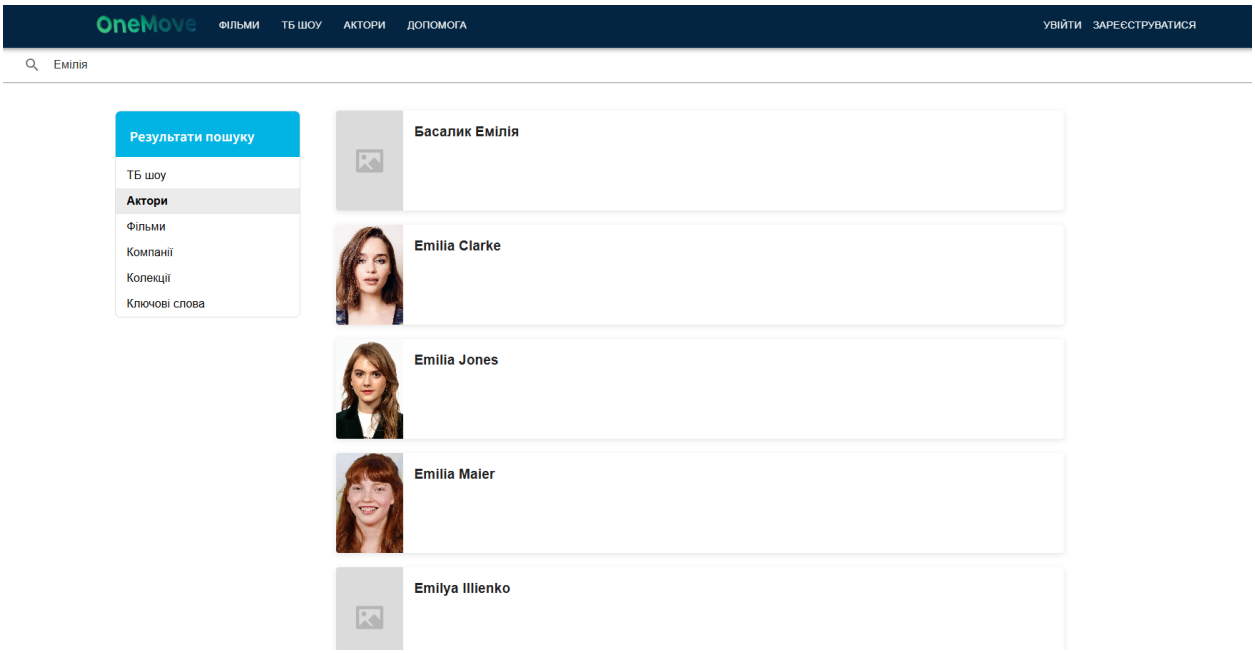

Рисунок 3.18 – Сторінка пошуку акторів

Перейшовши на актора, користувач потрапить на сторінку із повною інформацією про нього. Сторінка обов’язково демонструє фото актора у повномасштабному форматі. Персональна сторінка актора містить загальну інформацію, біографію, стать, його псевдоніми різними мовами, таблицю із кіно картинами в хронологічному порядку, в яких приймалася участь та список найвідоміших фільмів. На рисунку 3.19 продемонстровано сторінку актора українською мовою.

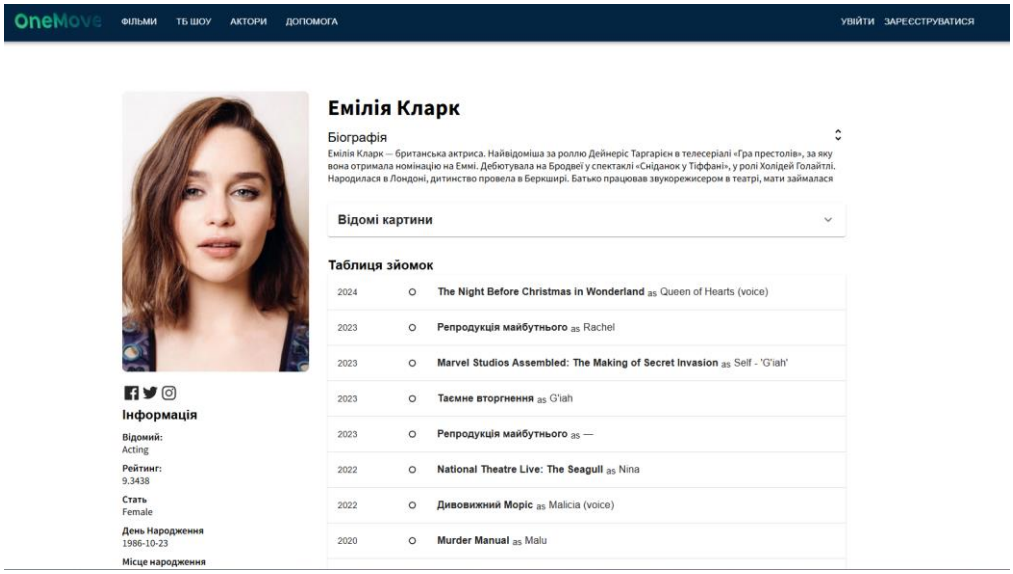

Рисунок 3.19 – Сторінка актора

Також, кожна сторінка фільму містить секцію із відео. В більшості випадків це трейлери фільму, сторінку якого переглядає користувач, щоб надати змогу отримати перше враження від контенту (див. рисунок 3.20).

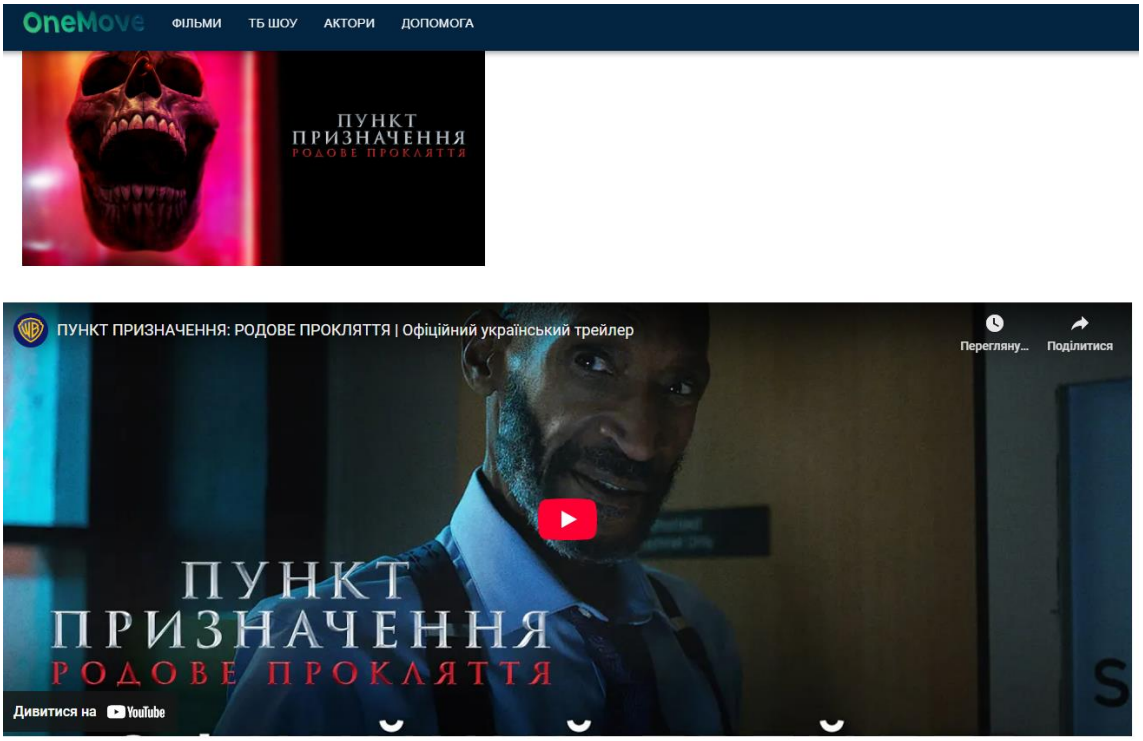

Рисунок 3.20 – Відео на сторінці фільму

Окрім відеоконтенту та іншої інформації, на сторінці обраного фільму, також присутній список із рекомендованими картинами до перегляду. Список рекомендацій базується лише на жанрах, що присутні у обраному фільмі. Тобто, жанри основного та рекомендованих фільмів обов’язково перетинатимуться. Зазвичай список рекомендацій містить від 10 до 20 елементів.

На рисунку 3.21 продемонстровано список рекомендацій для фільму «Warface», що містить такі жанри: бойовик, військовий.

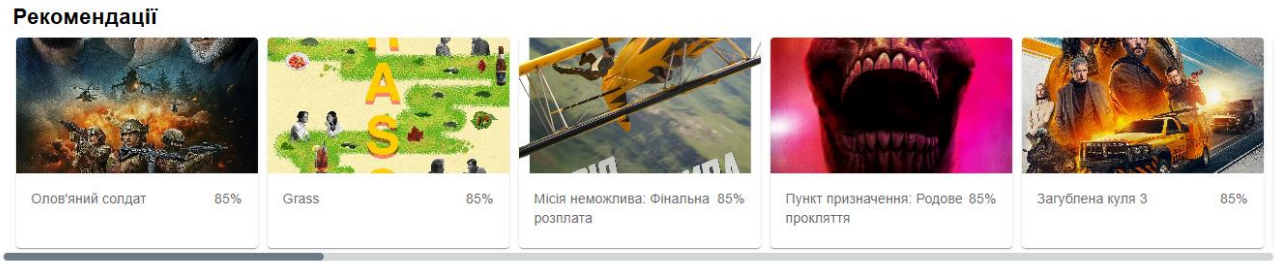

Рисунок 3.21 – Рекомендації для «Warface»

Якщо не вдалось завантажити дані для рекомендацій, то секцію буде просто приховано.

3.8 Висновки

Отже, у третьому розділі було обґрунтовано вибір технологій, фреймворків та бібліотек, що використовувались під час розробки вебсистеми кіноклубу.

Основною мовою програмування було обрано «TypeScript», що дозволить суворо типізувати увесь код та пришвидшити процес розробки. Для клієнтської частини у ролі фреймворка було обрано «NextJs», а серверної – «NestJs».

Для створення графічного інтерфейсу користувача було обрано «MaterialUI». Також щоб правильно валідувати та обробляти події форми із опитуванням, було обрано бібліотеки «Formik» та «Yup».

Під час розробки модуля для обговорення контенту в реальному часі було використано бібліотеку «SocketIO», що надає інструменти для легкої роботи із мережевим протоколом «WebSockets».

Продемонстровано алгоритм роботи модуля «Улюблене» та детально описано його розробку.

Описано розробку модуля статистики. Для візуалізації аналітичних даних було використано бібліотеку «ChartsJS».

У розділі було описано розробку модулів для підбору та обговорення контенту та продемонстровано процес реалізації графічного інтерфейсу користувача для них.

4 ТЕСТУВАННЯ ПРОГРАМИ

4.1 Тестування функціоналу вебсистеми

Тестування програмного забезпечення [26] – це процес перевірки роботи ПЗ на відповідність вимогам, специфікаціям і очікуваним результатам. Це важливий етап розробки, який допомагає виявляти помилки та дефекти, покращувати якість продукту, підвищувати безпеку та впевненість у його роботі перед початком експлуатації.

Для того, аби процес тестування приніс результати, необхідно запустити серверну частину, що містить в собі логіку по обробці даних користувача із клієнта та інші модулі, а після – клієнтську. Перед початком тестування надзвичайно важливо надати документацію, вимоги до функціоналу та загальної роботи програмного продукту, що тестуватиметься.

Для перевірки коректності роботи основних функцій програми було обрано підхід, що має назву «Smoke Testing» [27]. Цей вид тестування можна визначити, як деякий короткий цикл тестів, який здійснюються після виходу чергового білду. Виконується це для перевірки роботи основного функціоналу розроблюваної програмної системи.

Під час тестування методом «Smoke Testing» буде перевірено програму на обробку невалідних вхідних даних від користувача, поведінку при відсутності даних та загальна поведінка відповідно до заявленого функціоналу.

Першим етапом тестування вебсистеми перевіряється обробка невалідних вхідних даних від користувача під час реєстрації та авторизації. Коли користувач вводить у поле для електронної пошти рядок неправильного формату, то програма повинна повідомити про це його у такий спосіб, щоб це було помітно. На рисунку 4.1 продемонстровано повідомлення від програми про неправильний формат введених даних.

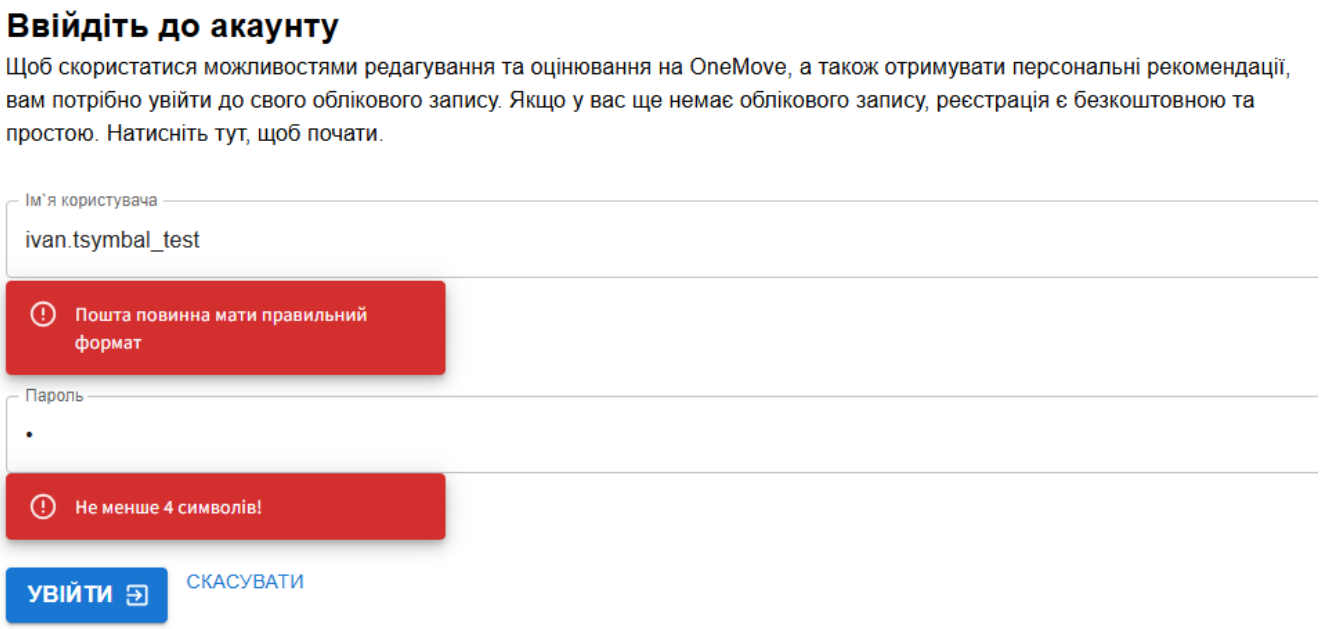

Рисунок 4.1 – Повідомлення про помилки

Наступним етапом було вирішено провести тестування реакції вебсистеми на дії неавторизованого користувача та його спроби використати заборонений функціонал. Можливість додавати контент до списку улюбленого дозволено лише тим, хто має акаунт. Відповідно, програма повинна виводити відповідне повідомлення про необхідність ввійти або ж зареєструватися, блокуючи процес виконання коду, що відповідає за взаємодію з базою даних для додавання нової інформації. На рисунку 4.2 продемонстровано повідомлення вебсистеми про заборону функціоналу.

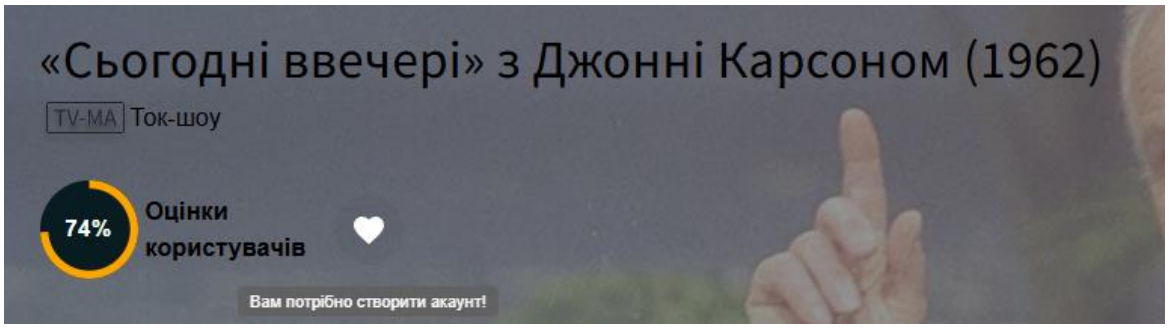

Рисунок 4.2 – Повідомлення про необхідність ввійти або зареєструватися під час спроби додати до списку «Улюблене»

Після проходження авторизації користувачеві відкривається функціонал додавання улюблених фільмів до власного списку.

На рисунку 4.3 продемонстровано повідомлення після проходження авторизації.

Рисунок 4.3 – Повідомлення про необхідність ввійти або зареєструватися під час спроби додати до списку «Улюблене»

На рисунку 4.4 продемонстровано список улюбленого користувача після додавання певного елементу.

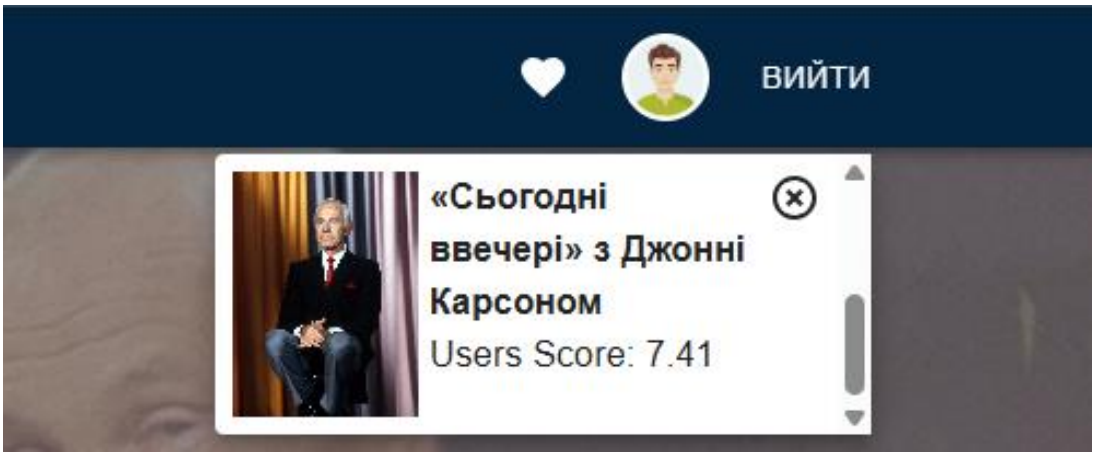

Рисунок 4.4 – Список «Улюблене» користувача

Наступним етапом було перевірено заборону на доступ до обговорень неавторизованим користувачам. Система повинна виводити повідомлення про

заборону доступу. На рисунку 4.5 продемонстровано реакцію вебсистеми на спробу увійти в обговорення без облікового запису.

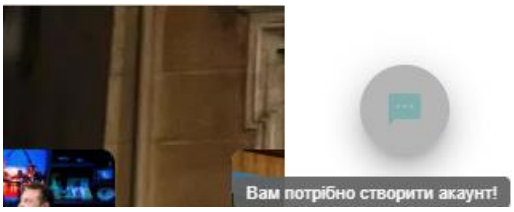

Рисунок 4.5 – Реакція вебсистеми на спробу увійти до обговорення без облікового запису

На рисунку 4.6 продемонстровано реакцію системи на спробу входу до обговорень зареєстрованим користувачем.

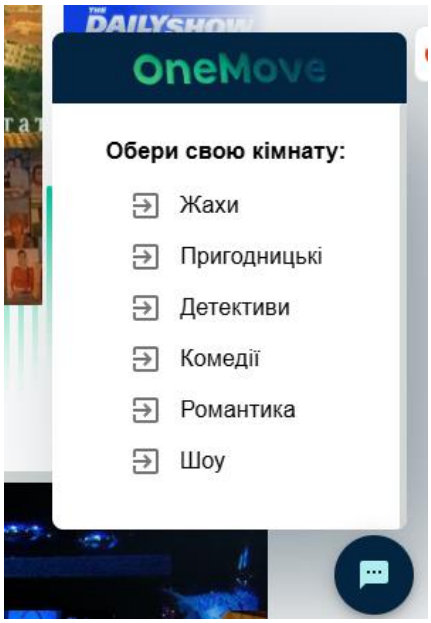

Рисунок 4.6 – Вхід до обговорення зареєстрованим користувачем

Після входу в кімнату і спробу користувача відправити щось, вебсистема повинна обробляти це правильним чином, аби інший користувач це побачив. Вебсистема повинна прослуховувати події, що передаватимуться через з’єднання з використанням протоколу вебсокетів.

Якщо користувач є автором надісланого повідомлення, то система повинна відмалювати його справа, а якщо навпаки, то зліва. Окрім основного тексту, також присутня коротка інформація: ім’я користувача відправника та час, коли було надіслано повідомлення. На рисунку 4.7 продемонстровано чат зі сторони поточного користувача.

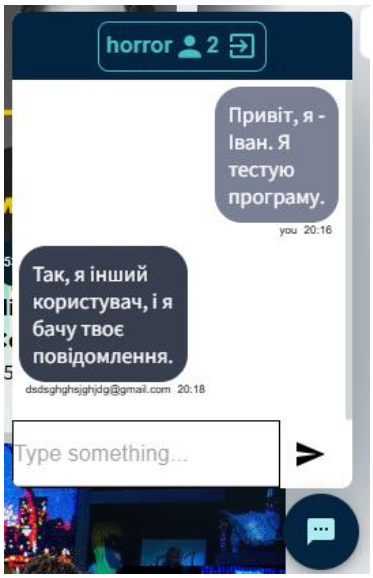

Рисунок 4.7 – Кімната з обговоренням зі сторони користувача

На рисунку 4.8 продемонстровано вигляд обговорення зі сторони співбесідника.

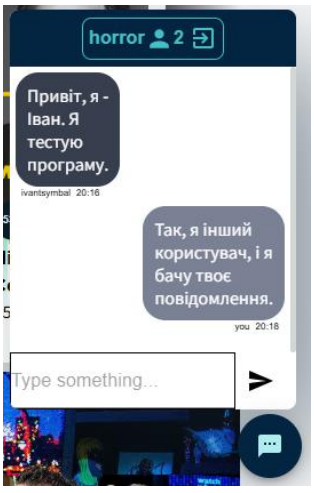

Рисунок 4.8 – Обговорення зі сторони співбесідника

Якщо один із учасників вийшов, то кількість користувачів повинна зменшитись. На рисунку 4.9 продемонстровано, що в кімнати залишився один юзер.

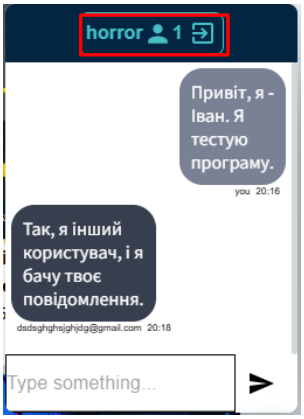

Рисунок 4.9 – Кількість користувачів у кімнаті

Наступним етапом необхідно перевірити модуль із авто підбором контенту на основі опитування, а саме регенерування результатів. На рисунку 4.10 продемонстровано початковий вигляд списку. Було дано відповідь, що користувача цікавить воєнна тематика.

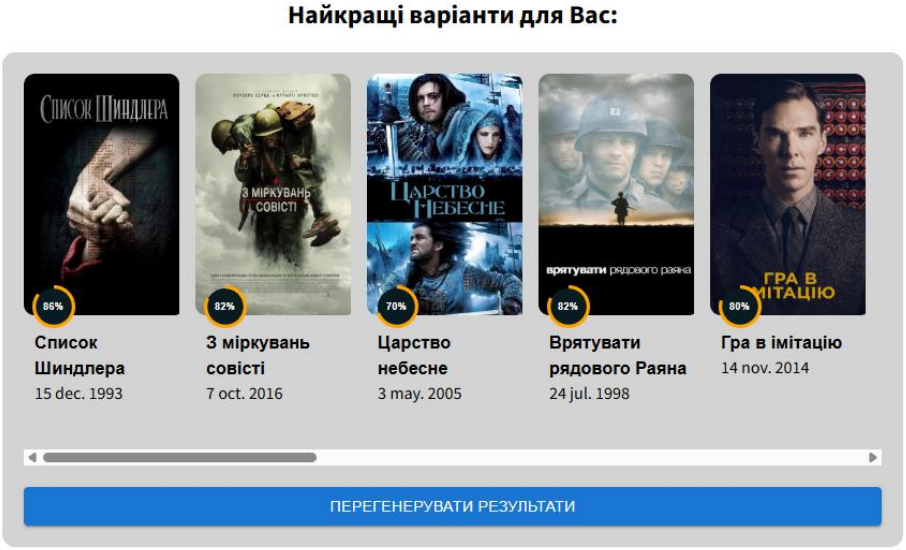

Рисунок 4.10 – Список рекомендованого контенту на воєнну тематику

Можна дійти висновку, що алгоритм підібрав контент правильно, адже у списку усі фільми, що відповідають військовій тематиці. Після натиску на кнопку «Перегенерувати результати» список повинен змінити своє наповнення. Список із рекомендованим контентом складається із 10-ти елементів. На рисунку 4.11 показано реакцію вебсистеми на натиск кнопки «Перегенерувати результати». Можна побачити, що система дійно змінила наповнення списку із запропонованим контеном.

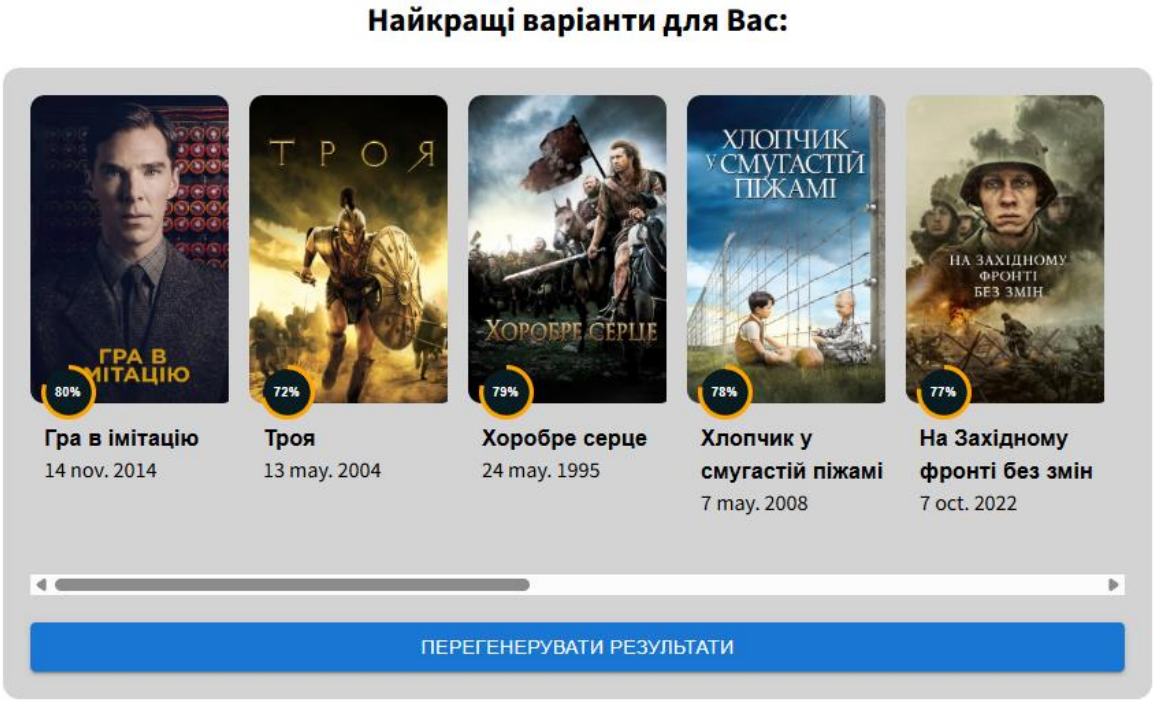

Рисунок 4.11 – Список рекомендованого контенту на воєнну тематику після нового запиту

Також додатково було перевірено функціонал пошуку контенту з використанням тегів та текстового вмісту. Для перевірки пошуку за текстовим вмістом було введено назву найвідомішого фільму «Harry Potter», таким чином система повинна вивести лише елементи, що містять у своєму описі або заголовку рядок, що ввів користувач. На рисунку 4.12 продемонстровано результати пошуку за терміном «Harry Potter».

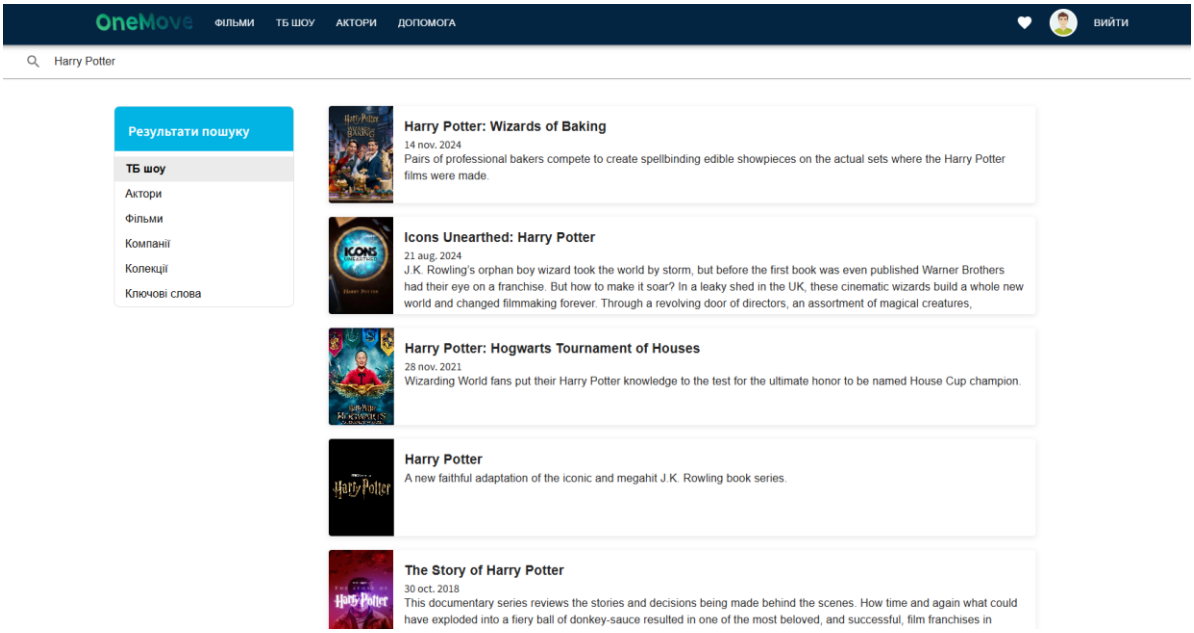

Рисунок 4.12 – Пошук за терміном «Harry Potter»

На рисунку 4.13 продемонстровано перевірку коректності роботи статистичного модуля: обрано період «День». Дата відображаєється коректно, початкова дата – сьогодні, а кінцева – відсутня. Таким чином, можна дійти висновку, що аналітичні дані відображаються коректно.

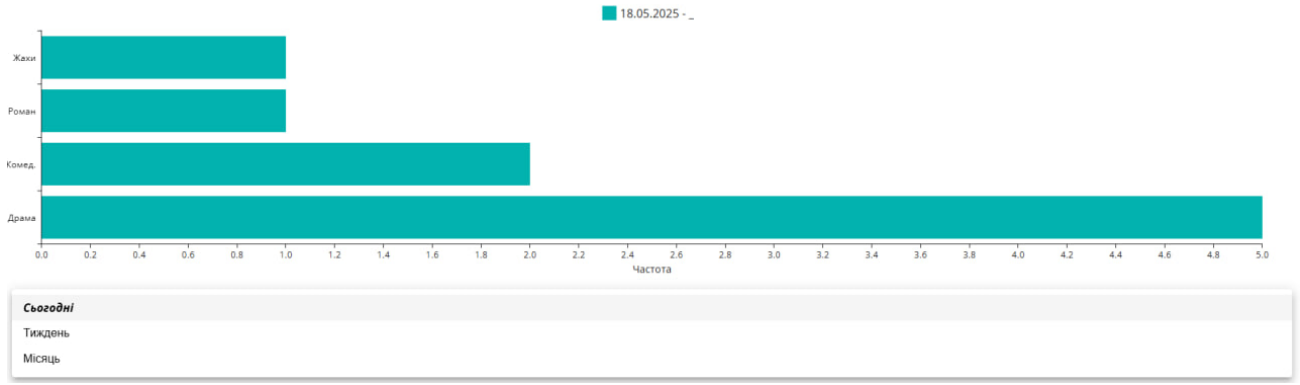

Рисунок 4.13 – Статистика за один день

Також для більш зручного процесу взаємодії із статистикою, було додано вспливаюче вікно, що з’являється під час наведення на стовпець, і яке містить розгорнуту інформацію про певний категорію (див. рисунок 4.14).

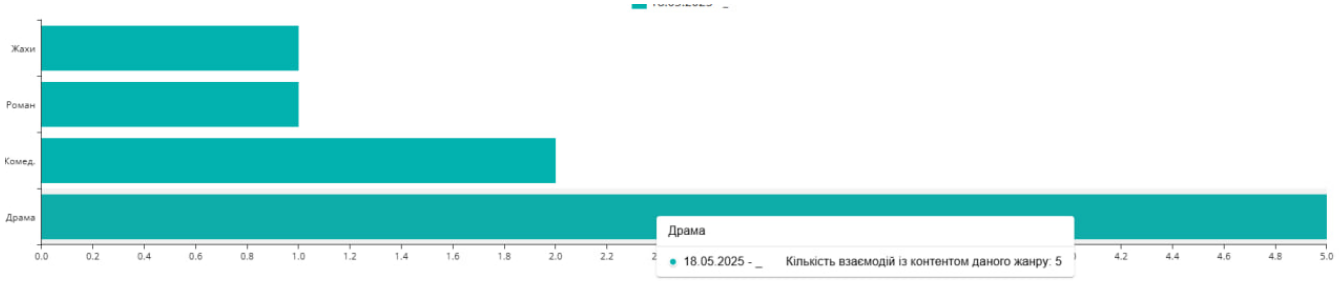

Рисунок 4.14 – Графік під час наведення курсором

Якщо користувач не авторизований, то буде продемонстровано розмитий графік із фіксованими значеннями, так як дані для побудови справжнього та динамічного – відсутні.

Надзвичайно важливо перевірити працездатність відеоплеєра, що присутній на кожній сторінці фільму. На рисунку 4.15 продемонстровано процес перемотування відео.

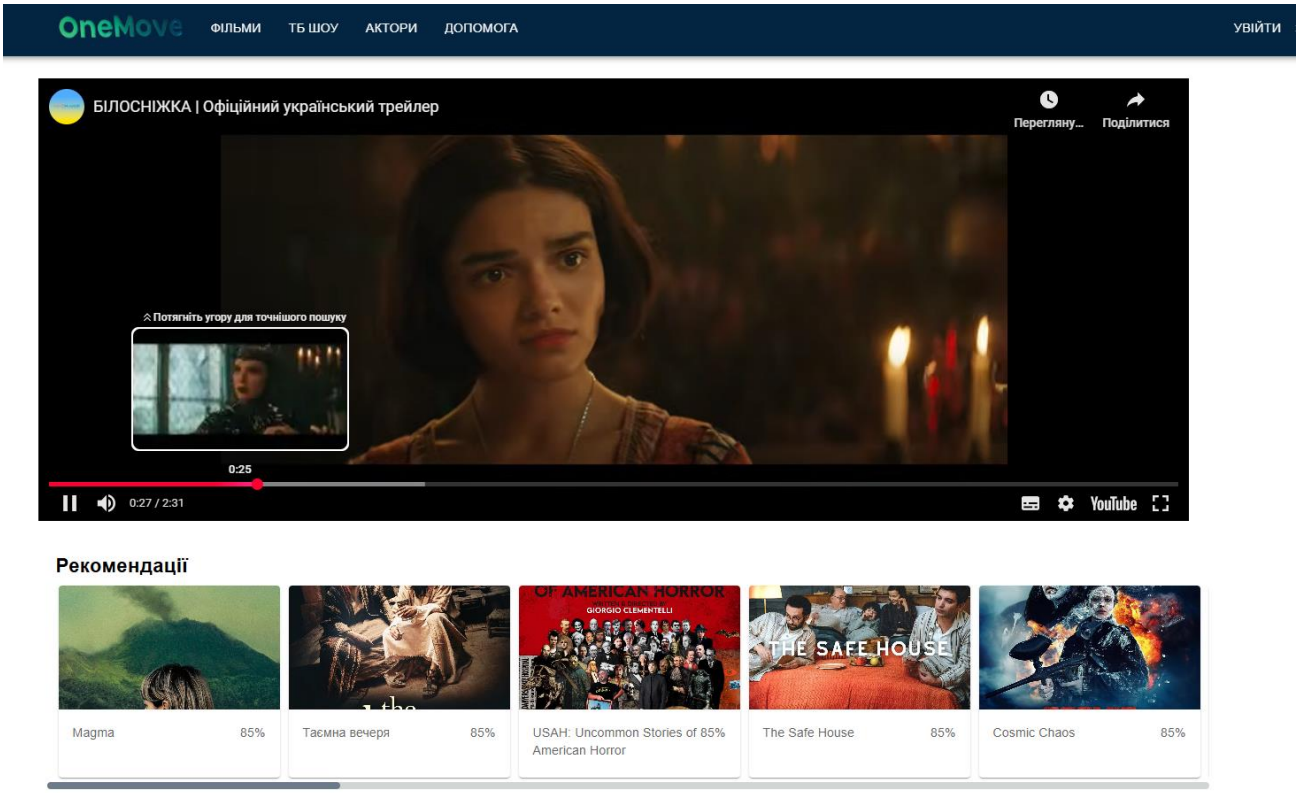

Рисунок 4.15 – Плеєр з відео

Отже, після перевірки основного функціоналу та реакції системи на певні випадки, можна зробити висновок, що все працює так, як очікується: вебсистема виводить помилки під час валідації, підбір та обговорення контенту відбуваються безперешкодно, а пошук працює коректно. Функціонал, що призначений лише для учасників системи – ніяким чином не доступний для пересічних користувачів.

4.2 Розробка інструкції користувача

Інструкція користувача повинна визначати технічні вимоги, як до середовища запуску так і до обладнання для коректної роботи програми, що вводиться в експлуатацію. Мінімальні та рекомендовані вимоги до обладнання наведено в таблиці 4.1.

Таблиця 4.1 – Вимоги до апаратного забезпечення

Вимоги до конфігурування	Рекомендовано	Мінімально
Оперативна пам'ять	8 ГБ	2 ГБ
Процесор	Intel Core i5	Intel Core i2
Вільний простір на диску	4 ГБ	2 ГБ
Операційна система	Windows 8 та вище	Windows XP або macOS
Монітор	Роздільна здатність не менше 1920x1080 пікселів	Роздільна здатність не менше 1280x720 пікселів
Швидкість Інтернету	100 МБіт/сек	50 МБіт/сек
Відеокарта	Intel Iris Xe G7	Intel Iris

Також, окрім технічних вимог повинні бути описані загальні рекомендації щодо процесу експлуатації програмного забезпечення: швидкість з’єднання з мережею Інтернет, випадки, коли потрібно пройти процес авторизації. Варто описати той функціонал, що доступний лише авторизованим користувачам вебсистеми.

Найбільший вплив на роботу вебсистеми має швидкість Інтернету, так як уся інформація відображається завдяки отриманню контенту із серверної частини з використанням запитів. Таким чином, для зменшення часу очікування відгуку інтерфейсу, користувач повинен мати швидкість Інтернет з‘єднання не менше 100МБіт/сек.

Вебсистема нормально функціонує і без особистих даних користувача, але для більш точного, наприклад, процесу побудови графіків – необхідно пройти процес авторизації, аби почався запис аналітичних даних у зовнішнє сховище, так як система не здійснює відслідковування пересічних відвідувачів з метою економії місця у “бакеті”.

Також список функціоналу, що відкритий лише для авторизованих користувачів:

1. Обговорення контенту в реальному часі. Доступ до листування заборонено неавторизованим користувачам, адже система не підтримує анонімності.

2. Вподобання контенту. Функціонал, що дозволяє створювати персональні списки із вподобаним контентом також відкритий лише для постійних користувачів – це зроблено з метою економії дискового простору.

3. Аналітика. Алгоритм відслідковування активності користувача починає працювати після входу в акаунт, записуючи у зовнішнє сховище дії, корелюючи їх із ідентифікатором (“id”) користувача. Якщо користувач не авторизувався, то замість графіка буде відображено розмита статична діаграма.

4. Автоматизований підбір контенту. Даний модуль здатен працювати і без вимоги авторизації користувача, але підібраний контент базуватиметься лише на відповідях на питання, тобто підхід не буде комплексним. Після входу в профіль і проходження опитування, система отримає із сховища список улюбленого та активність і опрацює їх відповідним чином.

Після входу в вебсистему користувач потрапляє на головну сторінку, де продемонстровано найактуальніший контент як для телебачення так і кінотеатру. Користувач має можливість безперешкодно здійснювати перегляд інформації та контенту, здійснювати пошук за текстом та з використанням фільтрів, підбір контенту шляхом опитування, а також створювати обліковий запис.

Якщо користувач хоче використати додатковий функціонал, то він буде вимушений створити обліковий запис або, при наявності, ввійти до нього. Після авторизації йому буде доступна можливість додавати фільми до списку улюбленого, а також участь в обговореннях разом з іншими учасниками платформи на рівних умовах.

4.4 Висновки

Отже, у четвертому розділі було проведено тестування роботи вебсистеми кіноклубу для підбору та обговорення контенту з використанням методу «Smoke Testing». Протестовано наявний функціонал та підтверджено правильність його роботи. Перевірено реакцію програми на введення невалідних даних зі сторони користувача.

Перевірено правильність роботи модуля аналітики та коректність відображення даних у вигляді графіків.

Розроблено інструкцію користувача та описано рекомендовані та мінімальні показники обладнання, на якому використовуватиметься вебсистема.

ВИСНОВКИ

У бакалаврській кваліфікаційній роботі були розроблені модулі вебсистеми кіноклубу для підбору і обговорення контенту. Для написання кодової бази було використано середовище розробки «Visual Studio Code», що пропонує безліч плагінів і полегшує процес розробки.

Бакалаврську кваліфікаційну роботу було розроблено відповідно до рекомендацій, наведених в методичних вказівках [28].

Було проведено аналіз сучасного стану проблеми на ринку кінопоказу, розглянуто аналоги розробленої системи, а саме їх переваги та недоліки. Базуючись на проведеному аналізі аналогів розробленої вебсистеми було сформовано основні завдання бакалаврської кваліфікаційної роботи.

В ході аналізу технологій було обґрунтовано вибір мови програмування «TypeScript», як основної і «NextJs» та «NestJs» - для клієнтської та серверної частин, що також базується на «TypeScript». «TypeScript» є типізованою версією мови програмування «JavaScript».

Перелік завдань та цілей, що було досягнуто в результаті виконання бакалаврської кваліфікаційної роботи:

– проведено аналіз стану розвитку застосунків кіноклубів;

– розроблено загальну модель та алгоритм роботи вебсистеми;

– розроблено метод автоматизованого підбору контенту;

– розроблено метод збору інформації про користувача та формування аналітичних графіків;

– імплементовано увесь заявлений функціонал;

– розроблено інтуїтивний інтерфейс користувача;

– протестовано коректність роботи усіх модулів та імплементованих алгоритмів вебсистеми кіноклубу з використанням методу «Smoke Testing».

Було розроблено блок-схеми загального алгоритму роботи вебсистеми та модуля автоматизованого підбору контенту. Розроблено діаграми станів усіх розроблюваних модулів та схеми інтерфейсу для них.

Для більш точних пропозицій стосовно контенту було розроблено метод підбору контенту, що базується не лише на відповідях користувача, але і його активності та списку улюбленого. Кожна складова методу має пріоритетність, найвищий – у даних про активність користувача.

Також було розроблено метод для збору інформації про користувача та формування аналітичних графіків, що також аналізує усі складові системи: активність, список вподобаних, нещодавно переглянуті.

Тестування програми довело, що вебсистема повністю акумулює увесь заявлений функціонал у повному обсязі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. BBC. Чому кінематограф набирає поплуряності в Інтернеті [Електронний ресурс] – Режим доступу до ресурсу: https://www.bbc.com/ukrainian/features-45761876 (дата звернення: 03.05.2025).

2. SMM Bussiness [Електронний ресурс] – Режим доступу до ресурсу: https://bazarmedia.info/2023/09/27/96e9vfzgj6/ (дата звернення: 25.03.2025).

3. Автоматизація процесів у бізнесі [Електронний ресурс] – Режим доступу до ресурсу: https://marketer.ua/ua/how-ai-will-revolutionize-your-business-360-degrees/ (дата звернення: 03.05.2025).

4. Цимбал І.Ю. Development of a film club web system for discussing video content in real time / Матеріали LIV Всеукраїнська науково-технічна конференція факультету будівництва, цивільної та екологічної інженерії (2025): збірник доповідей [Електронний ресурс] – Режим доступу до ресурсу: https://conferences.vntu.edu.ua/index.php/all-fbtegp/all-fbtegp-2025/paper/view/24010/19832 (дата звернення: 03.05.2025).

5. Войтко В.В. Розробка методу і засобів реалізації вебсистеми для підбору й обговорення відео контенту / В.В. Войтко, А.В. Денисюк, О.В. Гаврилюк, Н.Є. Барчук, І.Ю. Цимбал // Матеріали LIV Всеукраїнської науково-технічної конференції підрозділів Вінницького національного технічного університету (НТКП ВНТУ–2025) : збірник доповідей [Електронний ресурс]. – Режим доступу до ресурсу:https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2025/paper/view/24223/20165 (дата звернення: 03.05.2025).

6. Бевз С. В. Розробка методу автоматизованого пріоритетного підбору персоналізованого контенту / С. В. Бевз, С. М. Бурбело, І. Ю. Цимбал // Матеріали Міжнародної науково-практичної інтернет-конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2025)»: збірник доповідей [Електронний ресурс]. – Режим доступу до ресурсу:

https://conferences.vntu.edu.ua/index.php/mn/mn2025/paper/view/25396/20958 (дата звернення: 03.05.2025).

7. Why app is essential for business [Електронний ресурс] – Режим доступу до ресурсу: https://radixweb.com/guides/web-application-guide-for-business-success (дата звернення: 04.05.2025).

8. Як працюють алгоритми мереж [Електронний ресурс] – Режим доступу до ресурсу: https://speka.media/n22807-9826nm (дата звернення: 04.05.2025).

9. IMDb [Електронний ресурс] – Режим доступу до ресурсу: https://www.imdb.com/ (дата звернення: 07.05.2025).

10. MyVue [Електронний ресурс] – Режим доступу до ресурсу: https://www.myvue.com/ (дата звернення: 07.05.2025).

11. FanDango [Електронний ресурс] – Режим доступу до ресурсу: https://www.fandango.com/ (дата звернення: 07.05.2025).

12. SPA у програмуванні [Електронний ресурс] – Режим доступу до ресурсу: https://foxminded.ua/spa-u-prohramuvanni/ (дата звернення: 07.05.2025).

13. Односторінкові та багатосторінкові сайти [Електронний ресурс] – Режим доступу до ресурсу: https://dou.ua/forums/topic/25444/ (дата звернення: 29.03.2025).

14. Types of page generation [Електронний ресурс] – Режим доступу до ресурсу: https://dou.ua/forums/topic/41585/ (дата звернення: 10.05.2025).

15. What are cookie files for? [Електронний ресурс] – Режим доступу до ресурсу: https://theinweb.media/sho-take-fajli-cookies/ (дата звернення: 30.03.2025).

16. Google bucket storage [Електронний ресурс] – Режим доступу до ресурсу: https://cloud.google.com/storage/docs/json_api/v1/buckets (дата звернення: 10.05.2025).

17. Argon2 алгоритм [Електронний ресурс] – Режим доступу до ресурсу: https://www.npmjs.com/package/argon2 (дата звернення: 10.05.2025).

18. Sanders W. (2017). Learning PHP Design Patterns (pp. 8-10). The USA: Oreilly, 2021 (дата звернення: 15.05.2025).

19. Visio [Електронний ресурс] – Режим доступу до ресурсу: https://www.guru99.com/uk/microsoft-visio-tutorial.html?gpp (дата звернення: 15.05.2025).

20. Material UI [Електронний ресурс] – Режим доступу до ресурсу: https://mui.com/material-ui/ (дата звернення: 15.05.2025).

21. ChartsJs [Електронний ресурс] – Режим доступу до ресурсу: https://www.chartjs.org/ (дата звернення: 16.05.2025).

22. SocketIO [Електронний ресурс] – Режим доступу до ресурсу: https://www.chartjs.org/ (дата звернення: 16.05.2025).

23. SWR [Електронний ресурс] – Режим доступу до ресурсу: https://swr.vercel.app (дата звернення: 18.05.2025).

24. Firebase [Електронний ресурс] – Режим доступу до ресурсу: https://firebase.google.com/ (дата звернення: 18.05.2025).

25. VSCode [Електронний ресурс] – Режим доступу до ресурсу: https://code.visualstudio.com/ (дата звернення: 20.05.2025).

26. Сіддікі С. Learning Test-Driven Development: A Polyglot Guide to Writing Uncluttered Code. The USA: Oreilly, 2021 (дата звернення: 24.05.2025).

27. Smoke Testing [Електронний ресурс] – Режим доступу до ресурсу: https://www.techtarget.com/searchsoftwarequality/definition/smoke-testing (дата звернення: 24.05.2025).

28. Методичні вказівки до виконання бакалаврських кваліфікаційних робіт для студентів студентів спеціальності 121 "Інженерія програмного забезпечення" / Уклад. О. Н. Романюк, Г. О. Черноволик – Вінниця: ВНТУ, 2022. – 52 с (дата звернення: 27.05.2025).

Додатки

Додаток А. Технічне завдання

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
Романюк О. Н.
«21» березня 2025 р.

Технічне завдання
на бакалаврську кваліфікаційну роботу
«Розробка методу та засобів реалізації вебсистеми кіноклубу для підбору й обговорення відеоконтенту»
за спеціальністю 121 – Інженерія програмного забезпечення

Керівник бакалаврської кваліфікаційної роботи:
к.т.н., доц. Войтко В. В.
«21» березня 2025 р.

Виконав:
студент гр. ЗПІ-21б Цимбал І. Ю.
«21» березня 2025 р.

Вінниця – 2025 року

1.

Найменування та галузь застосування

Бакалаврська кваліфікаційна робота: «Розробка методу та засобів реалізації вебсистеми кіноклубу для підбору й обговорення відеоконтенту».

Галузь застосування – сфера надання послуг: заклади кінопоказу.

2. Підстава для розробки.

Підставою для виконання бакалаврської кваліфікаційної роботи (БКР) є індивідуальне завдання на БКР та наказ №97 від 20 березня 2025 року про закріплення тем БКР.

3. Мета та призначення розробки.

Метою дослідження є підвищення відвідуваності у сфері кінопоказу за рахунок використання вебсистеми для обговорення та підбору контенту, що дозволить залучати нових відвідувачів та аудиторію в мережі Інтернет, відслідковувати і розуміти їх вподобання, тим самим підвищуючи рівень зацікавленості та прибуток.

Призначення роботи – розробка вебсистеми кіноклубу для підбору та обговорення відеоконтенту.

4. Вихідні дані для проведення НДР

1. NextJS [Електронний ресурс] – Режим доступу до ресурсу: https://nextjs.org/
2. NestJS [Електронний ресурс] – Режим доступу до ресурсу: https://nestjs.com/
3. Алгоритми мереж [Електронний ресурс] – Режим доступу до ресурсу: https://speka.media/n22807-9826nm

5. Технічні вимоги

Комп’ютер з 64-розрядним (x64) або ж 32-розрядним процесором та тактовою

частотою 2 ГГц. Достатньо всього 4 ГБ оперативної пам’яті, а місця на жорсткому диску – 10 ГБ. Операційна система Windows 10 або нижче (включно до Windows 8). З клієнтського боку: комп’ютер з підключенням до мережі «Інтернет» та сучасний браузер.

6. Конструктивні вимоги.

Вебсистема повинна повністю відповідати загальноприйнятим вимогам щодо розробки front-end застосунків.

Інтерфейс користувача повинен бути інтуїтивно зрозумілим, так як вебсистема може бути використана користувачами з будь-яким рівнем володіння комп’ютером.

7. Перелік технічної документації, що пред’являється по закінченню робіт:

- пояснювальна записка до бакалаврської кваліфікаційної роботи;
- технічне завдання;
- лістинги програми.

8. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

9. Стадії та етапи розробки

№ з/п	Назва етапів бакалаврської кваліфікаційної роботи	Строк виконання етапів роботи
1	Аналіз розвитку вебсистем для кіноклубів та постановка задач дослідження	25.03.25 – 29.03.25
2	Розробка методів та алгоритмів роботи системи	30.03.25 – 10.04.25
3	Розробка програмного забезпечення системи	11.04.25 – 10.05.25
4	Тестування програми	11.05.25 – 14.05.25

5	Оформлення матеріалів до захисту БКР	15.05.25 – 30.05.25

10. Порядок контролю та прийняття

Всі етапи бакалаврської кваліфікаційної роботи контролюються науковим керівником згідно плану по виконанню роботи. Прийняття бакалаврської кваліфікаційної роботи здійснюється ДЕК, затвердженою зав. кафедрою згідно з графіком захисту.

77

Додаток Б. Протокол перевірки бакалаврської кваліфікаційної роботи на наявність текстових запозичень

ПРОТОКОЛ
ПЕРЕВІРКИ БАКАЛАВРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: «Розробка методу та програмних засобів для комунікацій в системах телемедицини»

Тип роботи: БКР

Підрозділ: кафедра програмного забезпечення, ФІТКІ

Науковий керівник: Войтко В. В.

Оригінальність	97 %
Схожість	3 %

Аналіз звіту подібності

☐ Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.
☐ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її автора. Роботу направити на доопрацювання.
☐ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ Черноволик Г. О.

Ознайомлені з повним звітом подібності, який був згенерований системою «Strike Plagiarism».

Автор роботи _____ Цимбал І. Ю.

Керівник роботи _____ Войтко В. В.

Додаток В. Лістинг програми

// Chat.tsx

```
import { Typography } from '@mui/material';
import { Box } from '@mui/system';
import { BodyChatBox, BoxChat, HeaderChatBox, HeaderImage } from
'components/Chat/styles';
import { LOGO } from 'components/constants';
import socket from 'socket';
import { changeRoomInfo } from '_redux/Chat/slice';
import { selectChat } from '_redux/Chat/selectors';

import { useEffect, useState } from 'react';
import { useAppDispatch } from 'hooks/typedRedux';
import Rooms from 'components/Chat/components/Rooms';
import FooterChat from 'components/Chat/components/FooterChat';
import Message from 'components/Chat/components/Message';
import HeaderChat from 'components/Chat/components/HeaderChat';
import OneMove from './oneMove.png';

const Chat = () => {
    const dispatch = useAppDispatch();
    const chat = selectChat();

    useEffect(() => {
        socket.on('CHAT:JOINED', users => dispatch(changeRoomInfo({ ...chat, users
})));
        socket.on('CHAT:LEAVE', users => dispatch(changeRoomInfo({ ...chat, users
})));
        socket.on('CHAT:MESSAGE:RECEIVE', messages => dispatch(changeRoomInfo({
...chat, messages })));
    }, []);

    return (
        <Box sx={BoxChat}>
            <Box sx={HeaderChatBox}>
                {chat.isAuth ? (
                    <HeaderChat />
                ) : (
                    <Box component="img" src={OneMove} sx={{ width: '143px', height:
'30px' }} />
                )}
            </Box>
            <Box sx={BodyChatBox}>
```

```
                {chat.isAuth ? (
                    <>
                        {chat.messages.map(item => (
                            <Message item={item} />
                        ))}
                    </>
                ) : (
                    <Rooms />
                )}
            </Box>
            {chat.isAuth && <FooterChat />}
        </Box>
    );
};
export default Chat;
```

// Message.tsx

```
import { Typography } from '@mui/material';
import { Box } from '@mui/system';
import { FC } from 'react';
import { MessageProps } from '_redux/Chat/types';
import { selectUser } from '_redux/User/selectors';
import { BoxMessageItem } from './styles';

interface MessagesProps {
    item: MessageProps;
}
const Message: FC<MessagesProps> = ({ item }) => {
    const { username } = selectUser();
    const ownMessage = item.username === username;
    return (
        <>
            <Box
                sx={{
                    ...BoxMessageItem,
                    alignSelf: ownMessage ? 'flex-end' : 'flex-start',
                    background: ownMessage ? '#7A8194' : '#373E4E',
                }}
            >
                {item.message}
            </Box>
            <Box
                sx={{ alignSelf: ownMessage ? 'flex-end' : 'flex-start', display:
'flex', margin: '0px 10px 0px 10px' }}
            >
                <Typography sx={{ fontSize: '8px', marginRight: '5px' }}>
```

```
                    {ownMessage ? 'you' : item.username}
                </Typography>
                <Typography sx={{ fontSize: '8px' }}>{item.time}</Typography>
            </Box>
        </>
    );
};
export default Message;
```

// Server.ts

```
const express = require('express');

const app = express();
const http = require('http');
const cors = require('cors');
const { Server } = require('socket.io');

app.use(cors());
app.use(express.json());
app.use(express.urlencoded({ extended: true }));

const server = http.createServer(app);
const io = new Server(server, {
    cors: {
        origin: 'http://localhost:3000',
        methods: ['GET', 'POST'],
    },
});

const rooms = new Map();

app.post('/rooms', (req, res) => {
    const { roomId } = req.body;
    if (!rooms.has(roomId)) {
        rooms.set(
            roomId,
            new Map([
                ['users', new Map()],
                ['messages', []],
            ])
        );
    }
    res.send();
});

app.get('/rooms/:id', (req, res) => {
    const { id } = req.params;
```

```
    const obj = rooms.has(id)
        ? {
                users: [...rooms.get(id).get('users').values()],
                messages: [...rooms.get(id).get('messages').values()],
          }
        : { users: [], messages: [] };
    res.json(obj);
});

app.get('/rooms/:id/messages', (req, res) => {
    const { id } = req.params;
    const messages = rooms.get(id).get('messages');
    return res.json(messages);
});

io.on('connection', socket => {
    socket.on('CHAT:CONNECT', data => {
        socket.join(data.roomId);
        rooms.get(data.roomId).get('users').set(socket.id, data.username);
        const users = [...rooms.get(data.roomId).get('users').values()];
        socket.to(data.roomId).emit('CHAT:JOINED', users);
    });
    socket.on('CHAT:SEND_MESSAGE', data => {
        rooms.get(data.roomId).get('messages').push(data);
        const messages = rooms.get(data.roomId).get('messages');
        socket.to(data.roomId).emit('CHAT:MESSAGE:RECEIVE', messages);
    });
    socket.on('CHAT:EXIT', data => {
        rooms.forEach((value, roomId) => {
            value.get('users').delete(socket.id);
            const users = [...value.get('users').values()];
            socket.to(roomId).emit('CHAT:LEAVE', users);
            socket.leave(roomId);
        });
    });
    socket.on('disconnect', () => {
        console.log('DISCONNECTED');
    });
});

server.listen(3001, () => console.log('SERVER RUN'));
```

// Home.tsx

```
import React, { useEffect, useState } from 'react';
```

```
import { useAppDispatch } from 'hooks/typedRedux';
import { getMovies } from '_redux/Home/asyncActions';
import { selectHome } from '_redux/Home/selectors';
import { Box } from '@mui/system';
import { selectUser } from '_redux/User/selectors';
import TextsmsIcon from '@mui/icons-material/Textsms';
import Chat from 'components/Chat';
import { Fab, Tooltip } from '@mui/material';
import SearchWindow from './components/SearchWindow';
import FilmSection from './components/FilmSection';
import TrailerSection from './components/TrailerSection';
import './styles.scss';

const Home = () => {
    const dispatch = useAppDispatch();
    const items = selectHome();
    const [isVisible, setIsVisible] = useState(false);
    const { username } = selectUser();

    const onClickChat = () => {
        if (!username) {
            return null;
        }
        setIsVisible(!isVisible);
    };

    useEffect(() => {
        dispatch(getMovies());
    }, []);
    return (
        <Box sx={{ position: 'relative' }}>
            <SearchWindow />
            {items.popular.tv && (
                <FilmSection
                    contentMaterials={{ Cinema: items.popular.movie, TV:
items.popular.tv }}
                    movies={items.popular.tv}
                    title="Що популярного?"
                />
            )}
            {items.popular.movie && (
                <TrailerSection
                    materialContent={{ Cinema: items.popular.movie, TV:
items.popular.tv }}
                    movie={items.popular.tv}
                    title="Останні трейлери"
```

```
                />
            )}
            {items.rated.tv && (
                <FilmSection
                    contentMaterials={{ Cinema: items.rated.movie, TV: items.rated.tv
}}
                    movies={items.rated.tv}
                    title="У тренді"
                />
            )}
            <Tooltip title={username ? 'Додати до улюбленого' : 'Вам потрібно створити
акаунт!'}>
                <Fab
                    onClick={onClickChat}
                    sx={{ backgroundColor: '#032541', position: 'fixed', bottom:
'40px', right: '40px' }}
                    aria-label="add"
                >
                    <TextsmsIcon sx={{ color: '#afeeee' }} />
                </Fab>
            </Tooltip>

            {isVisible && <Chat />}
        </Box>
    );
};
export default Home;

// FilmSection.tsx

/* eslint-disable no-unused-expressions */
import './styles.scss';
import React, { FC, useEffect, useState } from 'react';
import { movieItem } from '_redux/Home/types';
import { useAppDispatch } from 'hooks/typedRedux';
import CardFilm from 'components/CardFilm';
import { setQueryParams } from '_redux/Catalogue/slice';
import { selectCatalogue } from '_redux/Catalogue/selectors';
import backgroun from 'assets/bgMain.svg';

interface FilmSectionProps {
    title: string;
    movies: movieItem[];
    contentMaterials: any;
}
const FilmSection: FC<FilmSectionProps> = ({ title, movies, contentMaterials }) => {
```

```
    const dispatch = useAppDispatch();
    const [width, setWidth] = useState<string>('');
    const [content, setContent] = useState([]);
    const [left, setLeft] = useState(0);
    const { params } = selectCatalogue();

    const onClickType = (event: any, w: number, l: number, cont: any) => {
        setLeft(l);
        setWidth(event.target.offsetWidth + w);
        !left ? event.target.classList.add('active') :
event.target.classList.remove('active');
        setContent(cont);
    };

    useEffect(() => {
        dispatch(setQueryParams({ ...params, currentPlace: !left ? 'tv' : 'movie' }));
    }, [left]);

    return (
        <div className="section">
            <div className="section__header">
                <div className="section__title">{title}</div>
                <div className="section__switch">
                    <div
                        className="section__switch_color"
                        style={{
                            width: `${width}px`,
                            left: `${left}px`,
                            background: '#032541',
                        }}
                    />
                    <button
                        onClick={event => onClickType(event, 20, 0,
contentMaterials.TV)}
                        type="button"
                        className={`section__btn_toggle ${!left ? 'active' : ''}`}
                    >
                        На ТБ
                    </button>
                    <button
                        onClick={event => onClickType(event, 40, 85,
contentMaterials.Cinema)}
                        type="button"
                        className={`section__btn_toggle ${left === 85 ? 'active' :
''}`}
                    >
```

```
                        Кінотеатр
                    </button>
                </div>
            </div>
            <div className="section__main" style={{ background: `url(${backgroun}) 50%
100px no-repeat` }}>
                {(!content.length ? movies : content)?.map(item => (
                    <CardFilm
                        key={item.id}
                        id={item.id}
                        dateString={item.release_date ?? item.first_air_date}
                        title={item.name || item.original_name || item.title}
                        imgPath={item.poster_path}
                        mark={item.vote_average}
                        place={!left ? 'tv' : 'movie'}
                    />
                ))}
            </div>
        </div>
    );
};
export default FilmSection;

// CardFilm.tsx


/* eslint-disable no-unused-vars */
import * as React from 'react';
import DateConvert from 'components/DateConvertor';
import CircularProgressBar from 'components/CircularProgressBar';
import { BASE_URL, toNavigate } from 'pages/constants';
import { Box, Typography } from '@mui/material';
import { NO_COVER_FILM } from 'components/constants';
import { useNavigate } from 'react-router-dom';
import { BoxWrappStyles, MainImageStyles, TypoTitle } from
'components/CardFilm/styles';
import './CardFilm.scss';

interface FilmItemProps {
    title: string;
    imgPath: string;
    dateString: string;
    mark: number;
    id: number;
    place: string;
}
```

```
const CardFilm: React.FC<FilmItemProps> = ({ title, imgPath, dateString, mark, id,
place }) => {
    const navigate = useNavigate();
    return (
        <Box sx={BoxWrappStyles}>
            <Box sx={{ borderRadius: '8px' }} onClick={() => toNavigate(navigate,
'/item', `/${place}`, `/${id}`)}>
                <Box
                    component="img"
                    src={imgPath === null ? NO_COVER_FILM : `${BASE_URL}${imgPath}`}
                    sx={{ ...MainImageStyles }}
                />
            </Box>

            <CircularProgressBar sizeFont="9px" sizeCircle={40} name="circle__film"
value={mark * 10} />
            <Box sx={{ paddingLeft: '10px' }}>
                <Typography sx={TypoTitle} className="title">
                    {title}
                </Typography>
                <Box>
                    <DateConvert data={dateString} />
                </Box>
            </Box>
        </Box>
    );
};
export default CardFilm;

// ContentBot.tsx
import { ChangeEvent, useCallback, useState } from 'react';
import { Field, Formik } from 'formik';
import { Button, Chip, FormLabel, Radio, Slider } from '@mui/material';
import './styles.scss';
import { GENRE_CODE_BY_FIELD_NAME, SORTING_QUERY } from 'pages/ContentBot/consts';
import axios from 'axios';
import { API_KEY, URL } from '_redux/constants';
import { movieItem } from '_redux/Home/types';
import CardFilm from 'components/CardFilm';
import { QuestionWrapper } from './components/QuestionWrapper';

const INITIAL_VALUES = {
    comedian: false,
    cruel: false,
    horror: 0,
```

```
    historical: false,
    favouriteGenre: '',
    favouriteMovie: '',
    military: false,
    onlyHighRating: false,
    isHappy: false,
};

const ContentBot = () => {
    const [isLoading, setIsLoading] = useState(false);
    const [results, setResults] = useState<null | movieItem[]>(null);

    const handleSubmitForm = useCallback(async (values: any) => {
        setIsLoading(true);

        const genresArray = Object.keys(values).reduce((acc: number[], item: string)
=> {
            if (
                item &&
                typeof GENRE_CODE_BY_FIELD_NAME[item as keyof typeof
GENRE_CODE_BY_FIELD_NAME] !== 'undefined'
            ) {
                return values[item]
                    ? [...acc, GENRE_CODE_BY_FIELD_NAME[item as keyof typeof
GENRE_CODE_BY_FIELD_NAME]]
                    : acc;
            }
            return acc;
        }, []);

        let query = `?with_genres=${genresArray.join(',')}`;
        const sortBy = SORTING_QUERY[Math.ceil(Math.floor(Math.random() * 6))];

        query += values.onlyHighRating ? '&sort_by=popularity.asc' : '';
        query += query.includes('&sort_by') ? `,${sortBy}` : `&sort_by=${sortBy}`;

        const { data } = await
axios.get(`${URL}discover/movie${query}&${API_KEY}&language=uk`);

        const slicedResults = data.results.slice(Math.ceil(Math.floor(Math.random() *
5)), 15);
        setResults(slicedResults);
        setIsLoading(false);
    }, []);

    return (
```

```
        <div className="wrapper">
            <h2 className="title">
                {results?.length ? 'Найкращі варіанти для Вас:' : 'З тебе відповіді, з
нас - фільми!'}
            </h2>
            <div className="formWrapper">
                <Formik onSubmit={() => {}} initialValues={INITIAL_VALUES}
validateOnBlur>
                    {({ setFieldValue, values }) => {
                        const handleSelectNo = (e: ChangeEvent<HTMLInputElement>) => {
                            setFieldValue(e.target.name, false);
                        };

                        const handleSelectYes = (e: ChangeEvent<HTMLInputElement>) =>
{
                            setFieldValue(e.target.name, true);
                        };

                        const handleSelectTag = (value: string) => {
                            setFieldValue('favouriteGenre', value);
                        };

                        return results?.length ? (
                            <div className="resultsContainer">
                                <div className="resultsWrapper">
                                    {results.map(item => (
                                        <CardFilm
                                            key={item.id}
                                            id={item.id}
                                            dateString={item.release_date ??
item.first_air_date}
                                            title={item.title ?? item.original_name}
                                            imgPath={item.poster_path}
                                            mark={item.vote_average}
                                            place="movie"
                                        />
                                    ))}
                                </div>
                                <Button
                                    variant="contained"
                                    color="primary"
                                    onClick={() => handleSubmitForm(values)}
                                    className="button"
                                >
                                    Перегенерувати результати
                                </Button>
```

```
                            </div>
                        ) : (
                            <>
                                <QuestionWrapper title="1. Чи подобаються Вам фільми
та шоу з розважальним контекстом?">
                                    <Radio
                                        checked={values.comedian}
                                        id="yes"
                                        name="comedian"
                                        onChange={handleSelectYes}
                                    />
                                    <FormLabel id="yes">Так, подобається</FormLabel>
                                    <Radio
                                        checked={!values.comedian}
                                        id="no"
                                        name="comedian"
                                        onChange={handleSelectNo}
                                    />
                                    <FormLabel id="no">Ні, не моє</FormLabel>
                                </QuestionWrapper>
                                <QuestionWrapper title="2. Фільми із жорстокими
сценами - моя слабкість.">
                                    <Radio checked={values.cruel} id="yes"
name="cruel" onChange={handleSelectYes} />
                                    <FormLabel id="yes">Так, обожнюю!</FormLabel>
                                    <Radio checked={!values.cruel} id="no"
name="cruel" onChange={handleSelectNo} />
                                    <FormLabel id="no">Ні, я добра людина</FormLabel>
                                </QuestionWrapper>
                                <QuestionWrapper title="3. Оцініть власну прихильність
до фільмів жахів від 0 до 100">
                                    <Field
                                        as={Slider}
                                        defaultValue={50}
                                        aria-label="Default"
                                        valueLabelDisplay="auto"
                                        value={values.horror}
                                        name="horror"
                                    />
                                </QuestionWrapper>
                                <QuestionWrapper title="4. Мені більше подобаються
фільми історичного напрямку, з глибоким сенсом. Я люблю міркувати.">
                                    <Radio
                                        checked={values.historical}
                                        id="yes"
                                        name="historical"
```

```
                                    onChange={handleSelectYes}
                                />
                                <FormLabel id="yes">Так, я прагматик</FormLabel>
                                <Radio
                                    checked={!values.historical}
                                    id="no"
                                    name="historical"
                                    onChange={handleSelectNo}
                                />
                                <FormLabel id="no">Ні, я люблю відпочивати під час
перегляду</FormLabel>
                            </QuestionWrapper>
                            <QuestionWrapper title="5. Виберіть лише ОДИН жанр, що
найбільше описує Вас і Ваші вподобання">
                                <div className="tagsWrapper">
                                    <Chip label="Сімейні" onClick={() =>
handleSelectTag('10751')} />
                                    <Chip label="Пригоди" onClick={() =>
handleSelectTag('12')} />
                                    <Chip label="Кримінал" onClick={() =>
handleSelectTag('80')} />
                                    <Chip label="Мелодрама" onClick={() =>
handleSelectTag('18')} />
                                    <Chip label="Мультфільми" onClick={() =>
handleSelectTag('16')} />
                                </div>
                            </QuestionWrapper>
                            <QuestionWrapper title="6. Виберіть ОДИН фільм зі
списку, зо хотіли б переглянути.">
                                <Radio checked={false} id="yes"
name="favouriteMovie" />
                                <FormLabel id="yes">Зелена миля</FormLabel>
                                <Radio checked={false} id="no"
name="favouriteMovie" />
                                <FormLabel id="no">Парубвечір</FormLabel>
                                <Radio checked={false} id="no"
name="favouriteMovie" />
                                <FormLabel id="no">Скажене весілля</FormLabel>
                                <Radio checked={false} id="no"
name="favouriteMovie" />
                                <FormLabel id="no">Механік</FormLabel>
                            </QuestionWrapper>
                            <QuestionWrapper title="7. Я прихильник воєнної
тематики">
                                <Radio
                                    checked={values.military}
```

```
                                    id="yes"
                                    name="military"
                                    onChange={handleSelectYes}
                                />
                                <FormLabel id="yes">Так, я фанат</FormLabel>
                                <Radio
                                    checked={!values.military}
                                    id="no"
                                    name="military"
                                    onChange={handleSelectNo}
                                />
                                <FormLabel id="no">Ні, це не моє</FormLabel>
                            </QuestionWrapper>
                            <QuestionWrapper title="8. Хочу бачити лише відомі
фільми із високими рейтингами">
                                <Radio
                                    checked={values.onlyHighRating}
                                    id="yes"
                                    name="onlyHighRating"
                                    onChange={handleSelectYes}
                                />
                                <FormLabel id="yes">Так, я люблю лише відомі
витвори</FormLabel>
                                <Radio
                                    checked={!values.onlyHighRating}
                                    id="no"
                                    name="onlyHighRating"
                                    onChange={handleSelectNo}
                                />
                                <FormLabel id="no">Ні, мені подобаються
маловідомі</FormLabel>
                            </QuestionWrapper>
                            <QuestionWrapper title="9. Чи щаслива Ви людина?">
                                <Radio
                                    checked={values.isHappy}
                                    id="yes"
                                    name="isHappy"
                                    onChange={handleSelectYes}
                                />
                                <FormLabel id="yes">Так</FormLabel>
                                <Radio checked={!values.isHappy} id="no"
name="isHappy" onChange={handleSelectNo} />
                                <FormLabel id="no">Ні, це не про мене</FormLabel>
                            </QuestionWrapper>
                            <Button variant="contained" color="primary"
onClick={() => handleSubmitForm(values)}>
```

```
                                Отримати рекомендації
                            </Button>
                        </>
                    );
                }}
            </Formik>
        </div>
    </div>
    );
};
export default ContentBot;
```

// SignUp.tsx

```
import AuthForm from 'components/AuthForm';
import { FC, forwardRef, useState } from 'react';
import { Formik } from 'formik';
import { Box } from '@mui/system';
import { Button, FormControl, Input, InputLabel, OutlinedInput, Snackbar, TextField,
Typography } from '@mui/material';
import MuiAlert, { AlertProps } from '@mui/material/Alert';
import { ButtonSign, inputStyle } from 'components/AuthForm/styles';
import ExitToAppIcon from '@mui/icons-material/ExitToApp';
import { useNavigate } from 'react-router-dom';
import { useAuth } from 'hooks/useAuth';
import { BUTTON_TEXT, TITLE, UNDER_FORM, UNDER_TITLE, VALIDATE_SCHEME } from
'./constants';

const SignUp: FC = () => {
    const [avatarFile, setAvatarFile] = useState<any>({});
    const navigate = useNavigate();
    const Alert = forwardRef<HTMLDivElement, AlertProps>(function Alert(props, ref) {
        // eslint-disable-next-line react/jsx-props-no-spreading
        return <MuiAlert elevation={6} ref={ref} variant="filled" {...props} />;
    });
    return (
        <AuthForm title={TITLE} underTitle={UNDER_TITLE} underFormText={UNDER_FORM}>
            <Formik
                initialValues={{
                    username: '',
                    password: '',
                    confirmPassword: '',
                    email: '',
```

```
                }}
                onSubmit={values => useAuth(true, values, avatarFile, navigate)}
                validateOnBlur
                validationSchema={VALIDATE_SCHEME}
            >
                {({ values, errors, touched, handleChange, handleBlur, handleSubmit,
isValid, dirty }) => {
                    return (
                        <Box>
                            <Box>
                                <TextField
                                    id="standard-multiline-flexible"
                                    label="Username"
                                    multiline
                                    maxRows={4}
                                    variant="outlined"
                                    value={values.username}
                                    onChange={handleChange}
                                    name="username"
                                    onBlur={handleBlur}
                                    sx={inputStyle}
                                />
                                {errors.username && touched.username && (
                                    <Snackbar open autoHideDuration={6000} sx={{
position: 'static' }}>
                                        <Alert severity="error" sx={{ width: '30%' }}>
                                            {errors.username}
                                        </Alert>
                                    </Snackbar>
                                )}
                            </Box>
                            <Box>
                                <FormControl sx={{ m: 0, width: '25ch' }}
variant="outlined">
                                    <InputLabel sx={{ margin: '30px 0 0 0' }}
htmlFor="outlined-adornment-password">
                                        Password
                                    </InputLabel>
                                    <OutlinedInput
                                        id="outlined-adornment-password"
                                        type="password"
                                        label="Password"
                                        value={values.password}
                                        onChange={handleChange}
                                        name="password"
                                        onBlur={handleBlur}
```

```
                                    sx={{ width: '1335px', margin: '30px 0 0 0' }}
                                />
                            </FormControl>
                            {errors.password && touched.password && (
                                <Snackbar open autoHideDuration={6000} sx={{
position: 'static' }}>
                                    <Alert severity="error" sx={{ width: '30%' }}>
                                        {errors.password}
                                    </Alert>
                                </Snackbar>
                            )}
                        </Box>
                        <Box>
                            <FormControl sx={{ m: 0, width: '25ch' }}
variant="outlined">
                                <InputLabel sx={{ margin: '30px 0 0 0' }}
htmlFor="outlined-adornment-password">
                                    Confirm
                                </InputLabel>
                                <OutlinedInput
                                    id="outlined-adornment-password"
                                    type="password"
                                    label="Password"
                                    value={values.confirmPassword}
                                    onChange={handleChange}
                                    name="confirmPassword"
                                    onBlur={handleBlur}
                                    sx={{ width: '1335px', margin: '30px 0 0 0' }}
                                />
                            </FormControl>
                            {errors.confirmPassword && touched.confirmPassword &&
(
                                <Snackbar open autoHideDuration={6000} sx={{
position: 'static' }}>
                                    <Alert severity="error" sx={{ width: '30%' }}>
                                        {errors.confirmPassword}
                                    </Alert>
                                </Snackbar>
                            )}
                        </Box>
                        <Box>
                            <TextField
                                id="standard-multiline-flexible"
                                label="Email"
                                multiline
                                maxRows={4}
```

```
                                    variant="outlined"
                                    value={values.email}
                                    onChange={handleChange}
                                    name="email"
                                    onBlur={handleBlur}
                                    sx={inputStyle}
                                />
                                {errors.email && touched.email && (
                                    <Snackbar open autoHideDuration={6000} sx={{
position: 'static' }}>
                                        <Alert severity="error" sx={{ width: '30%' }}>
                                            {errors.email}
                                        </Alert>
                                    </Snackbar>
                                )}
                            </Box>
                            <Box sx={{ dispaly: 'flex', flexDirection: 'row' }}>
                                <Typography>Load your avatar</Typography>
                                <Input
                                    sx={{ ...inputStyle, margin: '0', padding: 0 }}
                                    name="avatar"
                                    id="avatar"
                                    onChange={(event: any) =>
setAvatarFile(event.target.files[0])}
                                    type="file"
                                />
                            </Box>
                            <Button
                                disabled={!isValid && !dirty}
                                sx={ButtonSign}
                                variant="contained"
                                endIcon={<ExitToAppIcon />}
                                onClick={() => handleSubmit()}
                                type="submit"
                            >
                                {BUTTON_TEXT}
                            </Button>
                            <Button variant="text" sx={{ marginTop: '20px' }}>
                                Cancel
                            </Button>
                        </Box>
                    );
                }}
            </Formik>
        </AuthForm>
    );
```

```
};
export default SignUp;

// MovieActions.ts
import { createAsyncThunk } from '@reduxjs/toolkit';
import axios from 'axios';
import { API_KEY, URL } from '../constants';
import {
    CreditsProps,
    FilmProps,
    ItemImagesProps,
    keywordsProps,
    RecommendationObj,
    Review,
    TVItemProps,
} from './types';

export const getInfoCurrent = createAsyncThunk('getInfo', async ({ id, place }: { id:
number; place: string }) => {
    const { data: currentMov } = await axios.get<FilmProps & TVItemProps>(
        `${URL}${place}/${id}?${API_KEY}&language=uk`
    );
    const { data: currentActors } = await axios.get<CreditsProps>(
        `${URL}${place}/${id}/credits?${API_KEY}&language=uk`
    );
    const { data: reviews } = await
axios.get<Review>(`${URL}${place}/${id}/reviews?${API_KEY}&language=uk`);
    const { data: currentImages } = await axios.get<ItemImagesProps>(
        `${URL}${place}/${id}/images?${API_KEY}&language=uk`
    );

    const recommendations = await axios
        .get<RecommendationObj>(`${URL}${place}/${id}/recommendations?${API_KEY}&langu
age=uk`)
        .then(res => res.data);
    const { data: keywords } = await
axios.get<keywordsProps>(`${URL}${place}/${id}/keywords?${API_KEY}&language=uk`);
    return {
        currentMov,
        currentActors: [...currentActors.cast, ...currentActors.crew],
        reviews,
        currentImages,
        recommendations,
        keywords,
    };
});
```

```
// ActorActions.ts
import { createAsyncThunk } from '@reduxjs/toolkit';
import axios from 'axios';
import { URL, API_KEY } from '../constants';

export const getActorInfo = createAsyncThunk('ActorInfo', async (id: number) => {
    const { facebook_id, twitter_id, instagram_id } = await axios
        .get(`${URL}person/${id}/external_ids?${API_KEY}&language=uk`)
        .then(resp => resp.data);
    const credits = await
axios.get(`${URL}person/${id}/combined_credits?${API_KEY}&language=uk`).then(resp => {
        const res = [...resp.data.cast, ...resp.data.crew];
        return res.sort(
            (a, b) =>
                Number((b.first_air_date ?? b.release_date)?.slice(0, 4)) -
                Number((a.first_air_date ?? a.release_date)?.slice(0, 4))
        );
    });
    const { data: actor } = await
axios.get(`${URL}person/${id}?${API_KEY}&language=uk`);
    return {
        actor,
        credits,
        networks: [
            facebook_id && `https://www.facebook.com/${facebook_id}`,
            twitter_id && `https://twitter.com/${twitter_id}`,
            instagram_id && `https://www.instagram.com/${instagram_id}/`,
        ],
    };
});


// CatalogueActions.ts
import { createAsyncThunk } from '@reduxjs/toolkit';
import axios from 'axios';
import { FilterQueryProps, GenresToSearch } from '_redux/Catalogue/types';
import { URL, API_KEY } from '../constants';

export const getFilters = createAsyncThunk('getFilters', async () => {
    const { data } = await axios.get(`${URL}genre/movie/list?${API_KEY}&language=uk`);
    const results = data.genres.map((item: GenresToSearch) => ({ ...item, value:
`&with_genres=${item.id}` }));
    return results;
});
```

```
export const getItemsCatalogue = createAsyncThunk(
    'getPopular',
    async ({ place, page, type }: { place: string; page: number; type: string }) => {
        const { data } = await axios.get<any,
any>(`${URL}${place}/${type}?${API_KEY}&page=${page}&language=uk`);
        return data;
    }
);

export const getSearchWithFilters = createAsyncThunk(
    'searchWithFilters',
    async ({ value, place, page }: { value: FilterQueryProps; place: string; page:
number }) => {
        const queries = Object.values(value).join('');
        const { data } = await
axios.get(`${URL}discover/${place}?${API_KEY}${queries}&page=${page}&language=uk`);
        return data;
    }
);
```

// HomeActions.ts

```
import { createAsyncThunk } from '@reduxjs/toolkit';
import axios from 'axios';
import { URL, API_KEY } from '../constants';

export const getMovies = createAsyncThunk('getAllMovies', async () => {
    const { data: popularTV } = await
axios.get(`${URL}tv/popular?${API_KEY}&language=uk`);
    const { data: popularMov } = await
axios.get(`${URL}movie/popular?${API_KEY}&language=uk`);
    const { data: ratedTV } = await
axios.get(`${URL}tv/top_rated?${API_KEY}&language=uk`);
    const { data: ratedMov } = await
axios.get(`${URL}movie/top_rated?${API_KEY}&language=uk`);
    return {
        popular: {
            tv: popularTV.results,
            movie: popularMov.results,
        },
        rated: {
            tv: ratedTV.results,
            movie: ratedMov.results,
        },
    };
```

```
});

// App.tsx
import './App.scss';
import { Route, Routes } from 'react-router-dom';
import ActorCard from 'pages/ActorPage';
import Footer from 'components/Footer';
import { HOME_PAGE, ITEM_PAGE, SEARCH_PAGE, CATALOGUE_PAGE, ACTOR_PAGE, SIGN_IN,
SIGN_UP, CONTENT_BOT } from 'routes';
import SignIn from 'pages/SignIn';
import SignUp from 'pages/SignUp';
import ContentBot from 'pages/ContentBot';
import Home from './pages/Home/Home';
import Navigation from './components/Navigation';
import ItemPage from './pages/Item';
import Search from './pages/Search';
import Catalogue from './pages/CataloguePage';

const App = () => {
    return (
        <div className="App">
            <Navigation />
            <Routes>
                <Route path={HOME_PAGE} element={<Home />} />
                <Route path={CONTENT_BOT} element={<ContentBot />} />
                <Route path={ITEM_PAGE} element={<ItemPage />} />
                <Route path={SEARCH_PAGE} element={<Search />} />
                <Route path={CATALOGUE_PAGE} element={<Catalogue />} />
                <Route path={ACTOR_PAGE} element={<ActorCard />} />
                <Route path={SIGN_IN} element={<SignIn />} />
                <Route path={SIGN_UP} element={<SignUp />} />
            </Routes>
            <Footer />
        </div>
    );
};

export default App;
```

Додаток Г. Ілюстративна частина

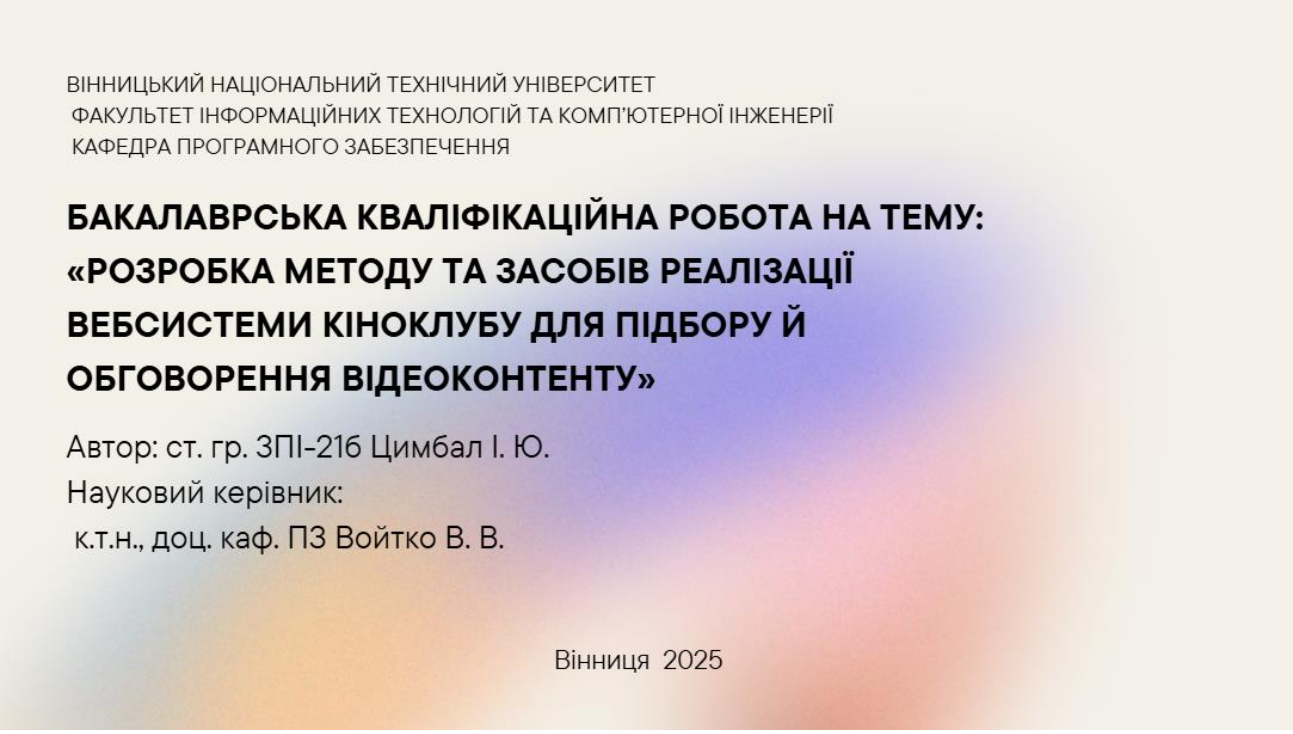

Рисунок Г.1 – Титульний слайд

Рисунок Г.2 – Актуальність розробки

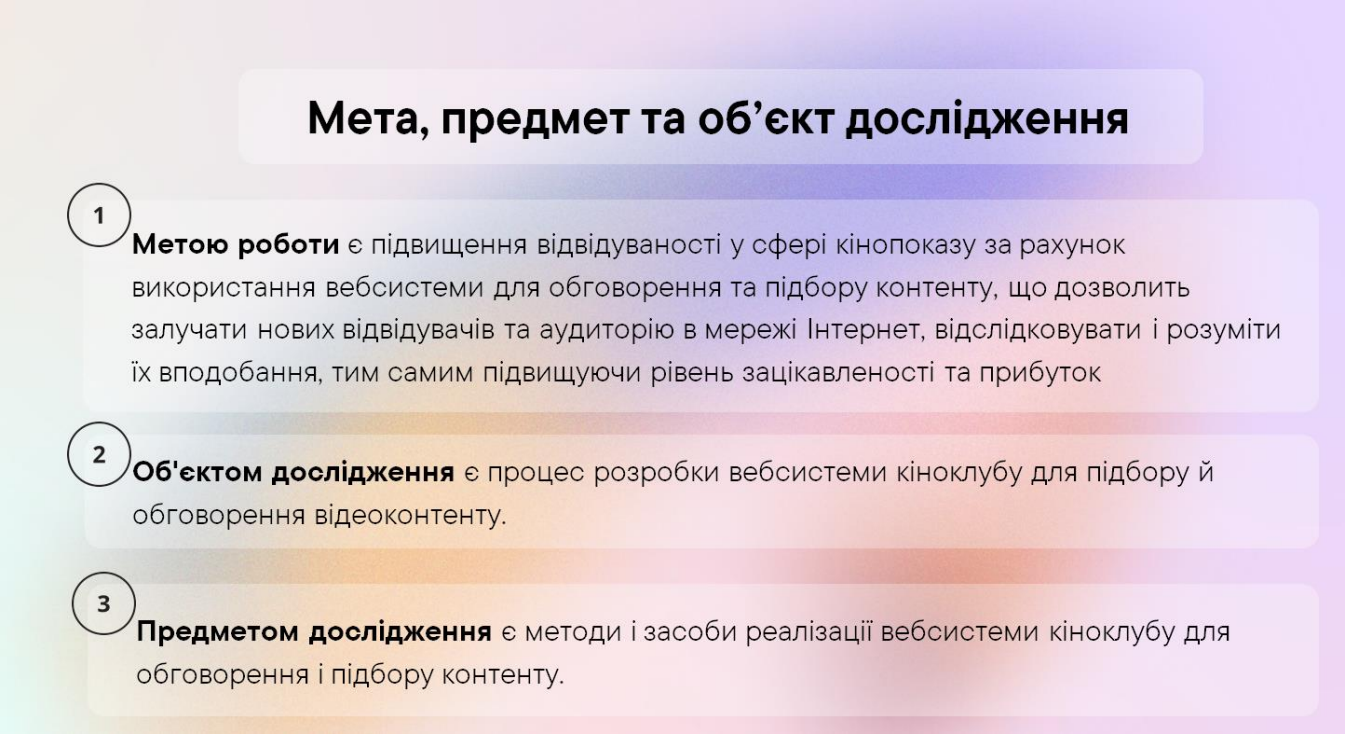

Рисунок Г.3 – Мета, предмет та об’єкт дослідження

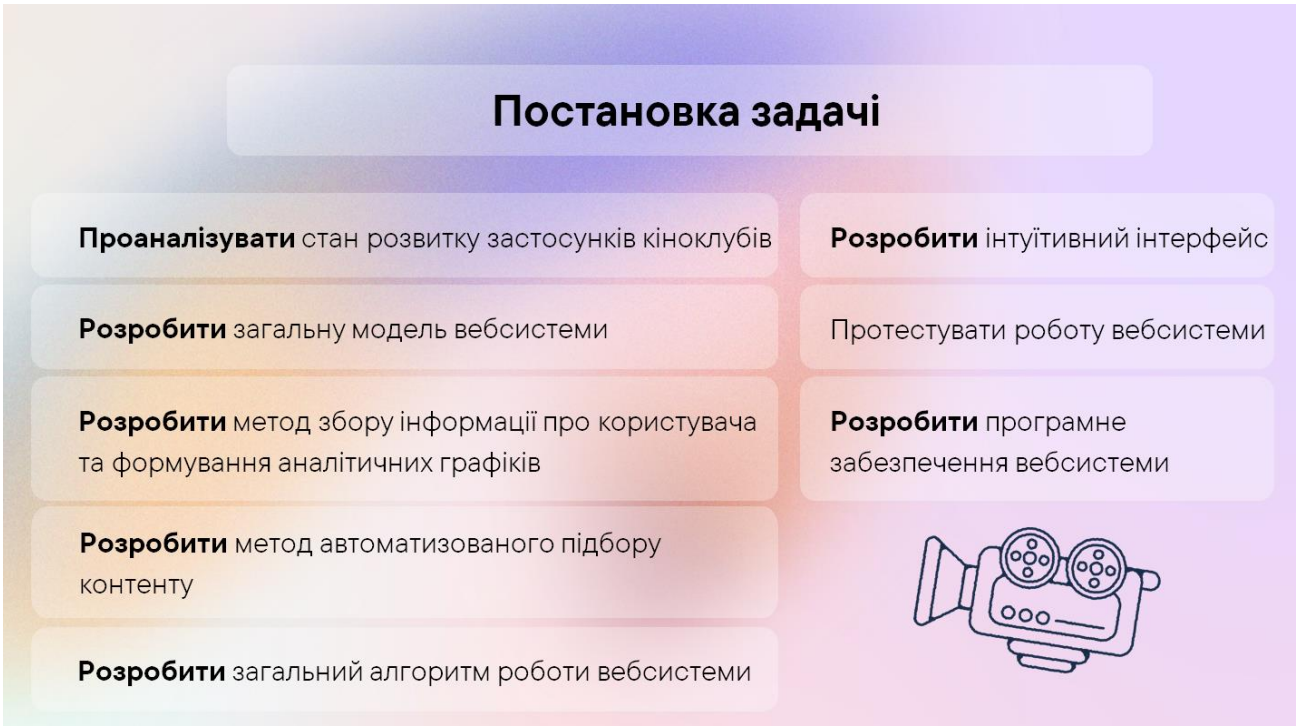

Рисунок Г.4 – Постановка задачі

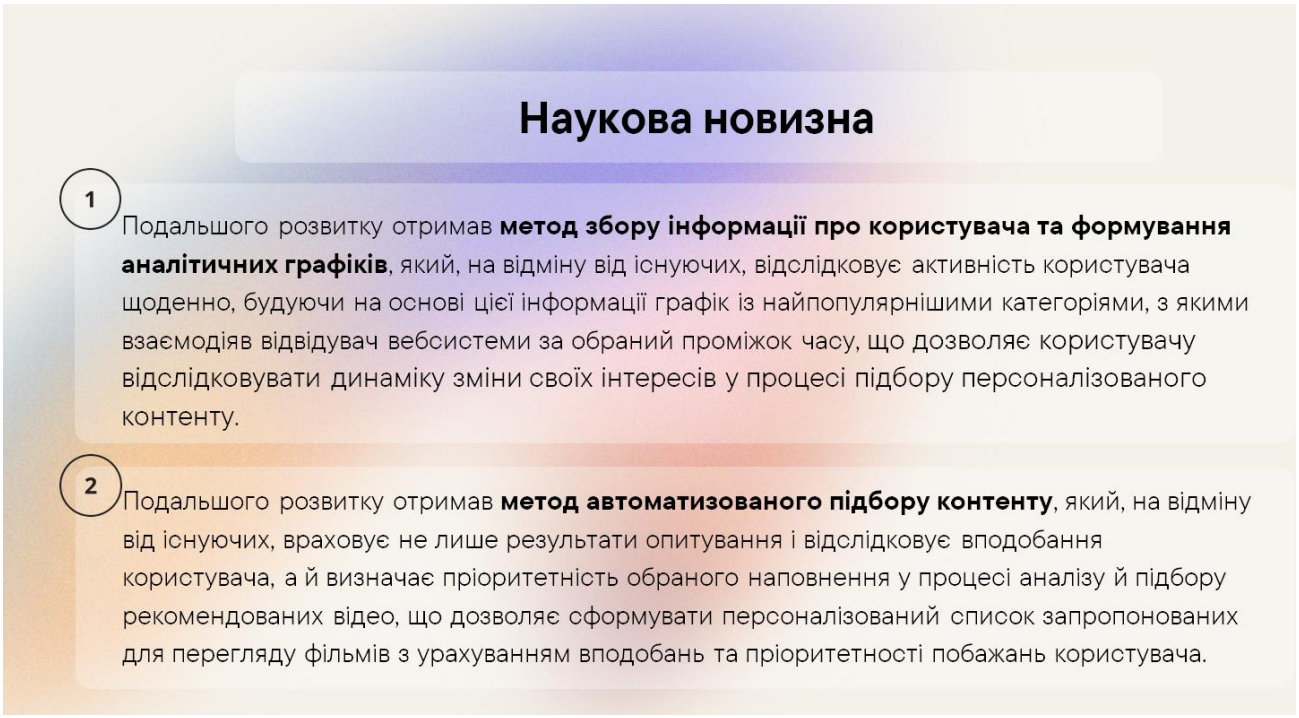

Рисунок Г.5 – Наукова новизна

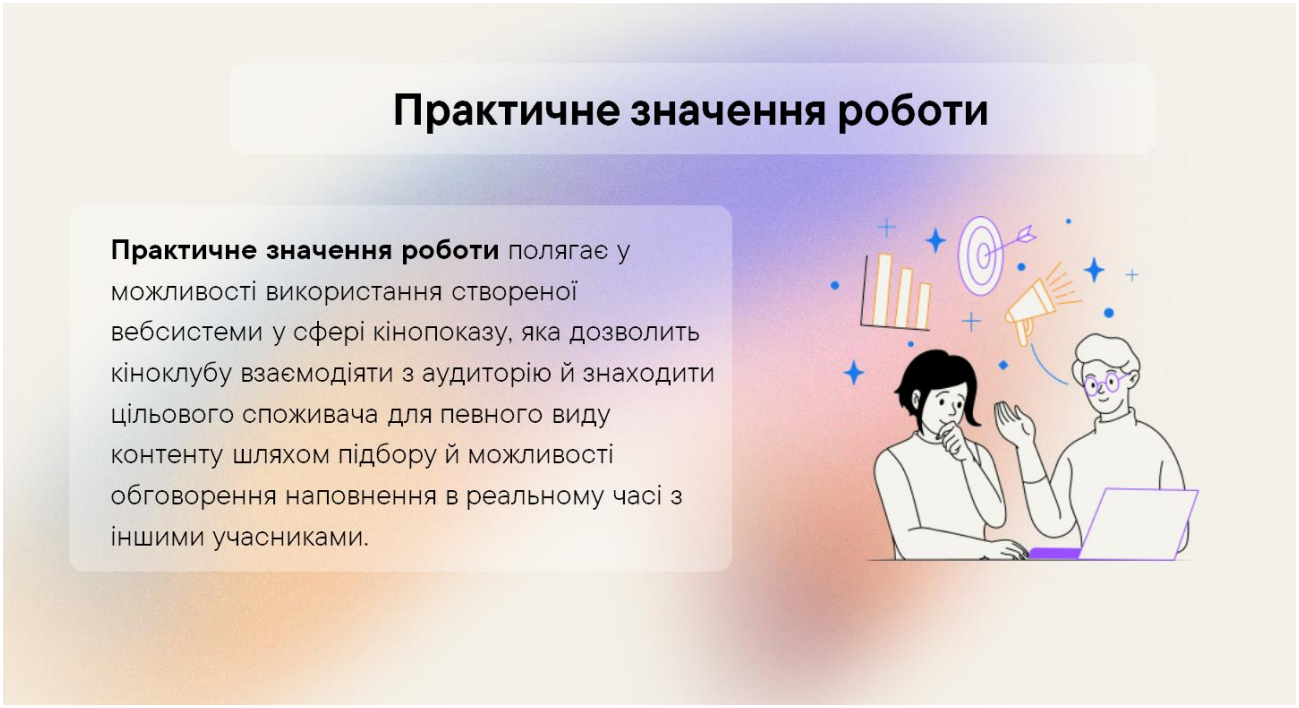

Рисунок Г.6 – Практичне значення

Аналіз аналогів вебсистеми

Критерії оцінювання	FandonGo	MyVue	IMDb	Власна розробка
Комунікації в реальному часі із учасниками системи	0	0	0	1
Автопідбір контенту	1	1	1	1
Точність підбору контенту	1	1	1	1
Створення списку «Улюблене»	1	0	1	1
Статистика користувача	0	0	0	1
Загальний коефіцієнт	**3**	**2**	**3**	**5**

Рисунок Г.7 – Порівняння аналогів

Метод автоматизованого підбору контенту (Ч. 1)

1. Вебсистема приймає відповіді користувача на запитання, що передаються через форму, що створено з використанням бібліотеки «Formik».

2. Після підтвердження форми викликається метод onSubmit(), що приймає відповіді в ролі аргументу функції.

3. Так як відповіді на питання є об'єктом, де кожен ключ – назва поля у формі. Таким чином, для обробки використовується метод Object.keys() та Object.values() що дають можливість отримати ключі та їх значення. Після отримання массиву з назвами ключів форми використовується forEach() метод, що на вхід приймає ключ форми і витягує назву жанру із статичного списку, залежно від відповіді користувача.

Рисунок Г.8 – Метод автоматизованого підбору контенту ч.1

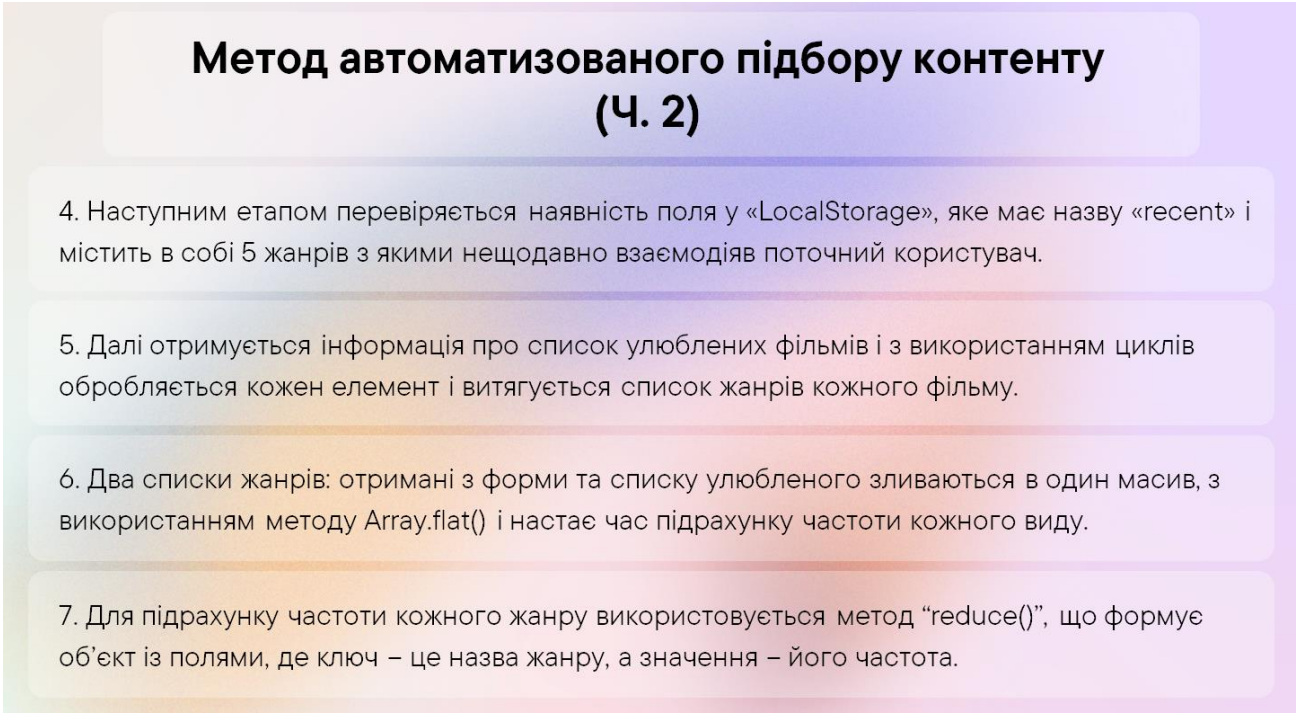

Рисунок Г.9 – Метод автоматизованого підбору контенту ч.2

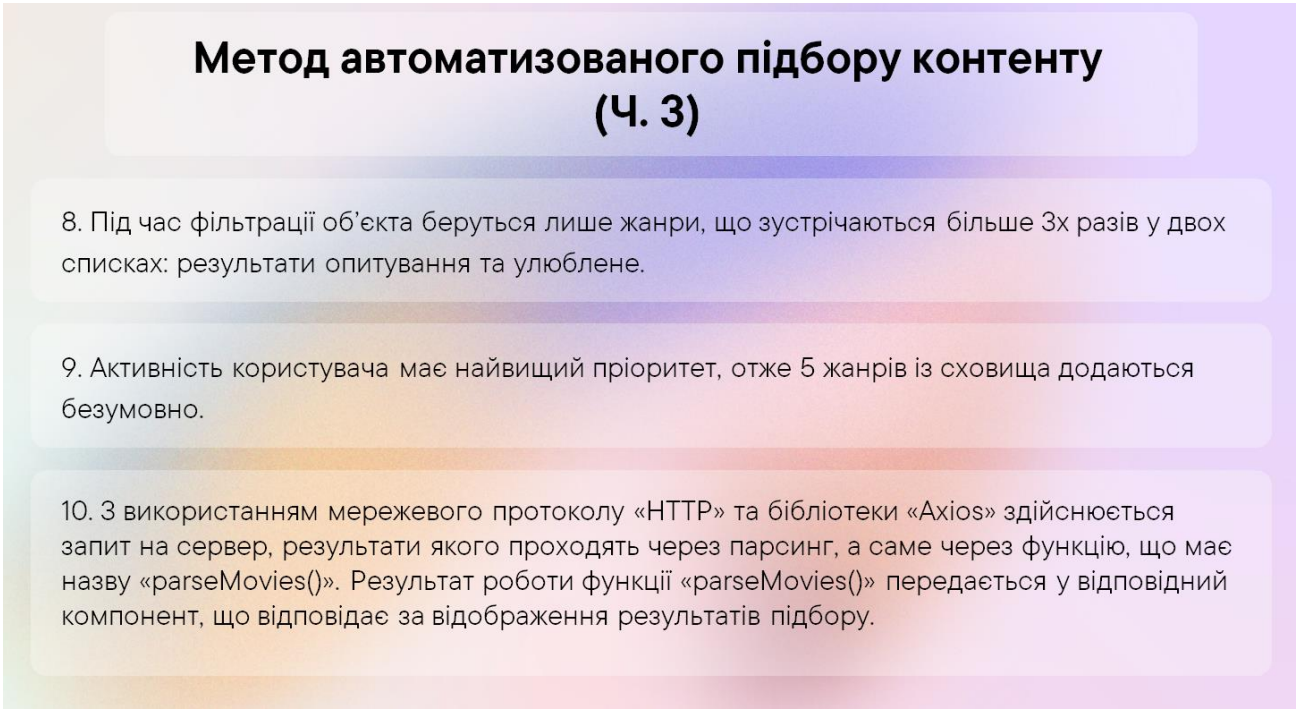

Рисунок Г.10 – Метод автоматизованого підбору контенту ч.3

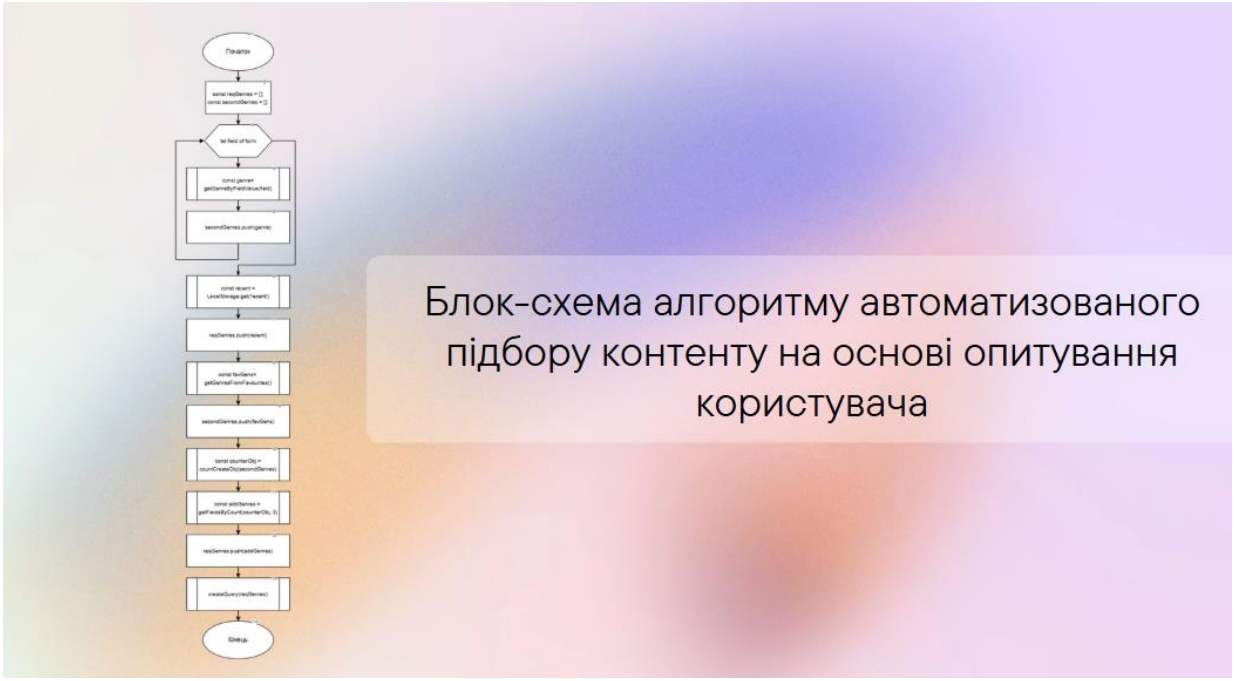

Рисунок Г.11 – Блок-схема алгоритму автоматизованого підбору контенту на основі опитування користувача

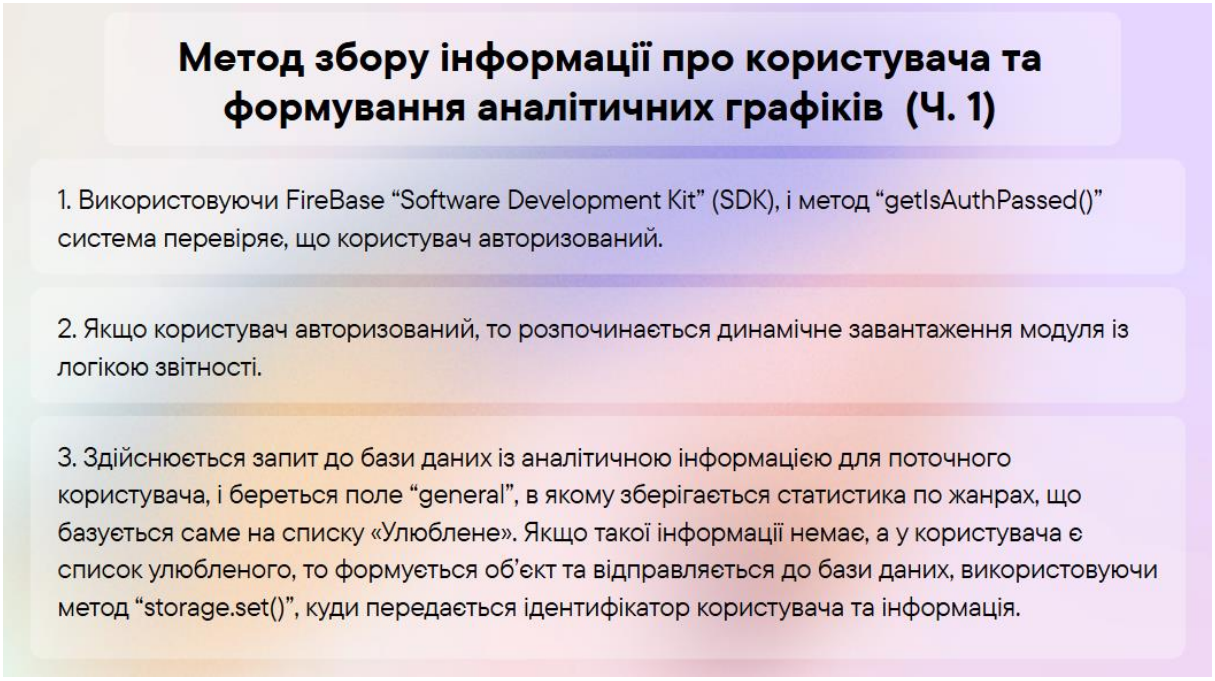

Рисунок Г.12 – Метод збору інформації про користувача та формування аналітичних графіків ч.1

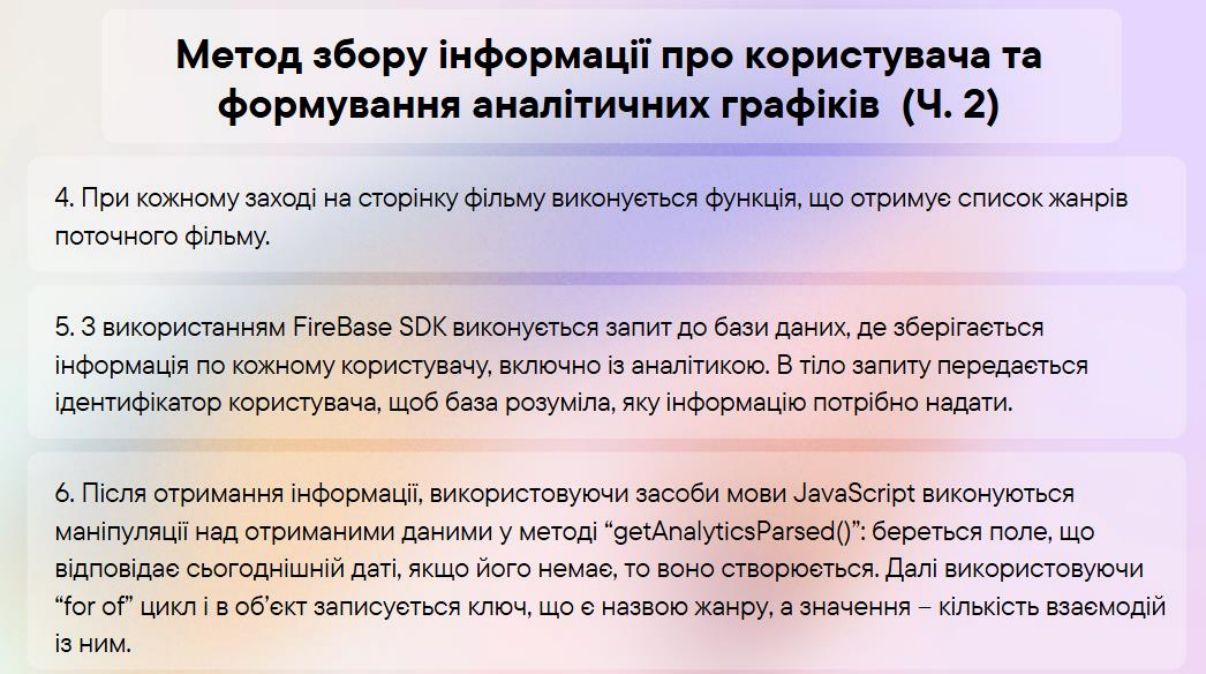

Рисунок Г.13 – Метод збору інформації про користувача та формування аналітичних графіків ч.2

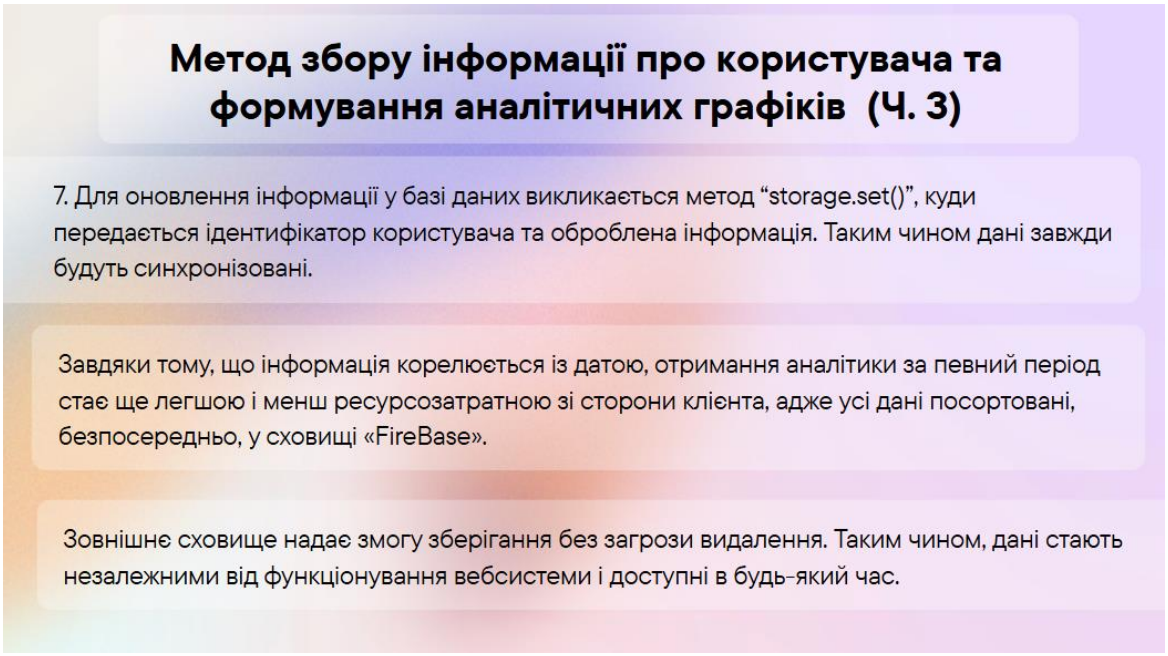

Рисунок Г.14 – Метод збору інформації про користувача та формування аналітичних графіків ч.3

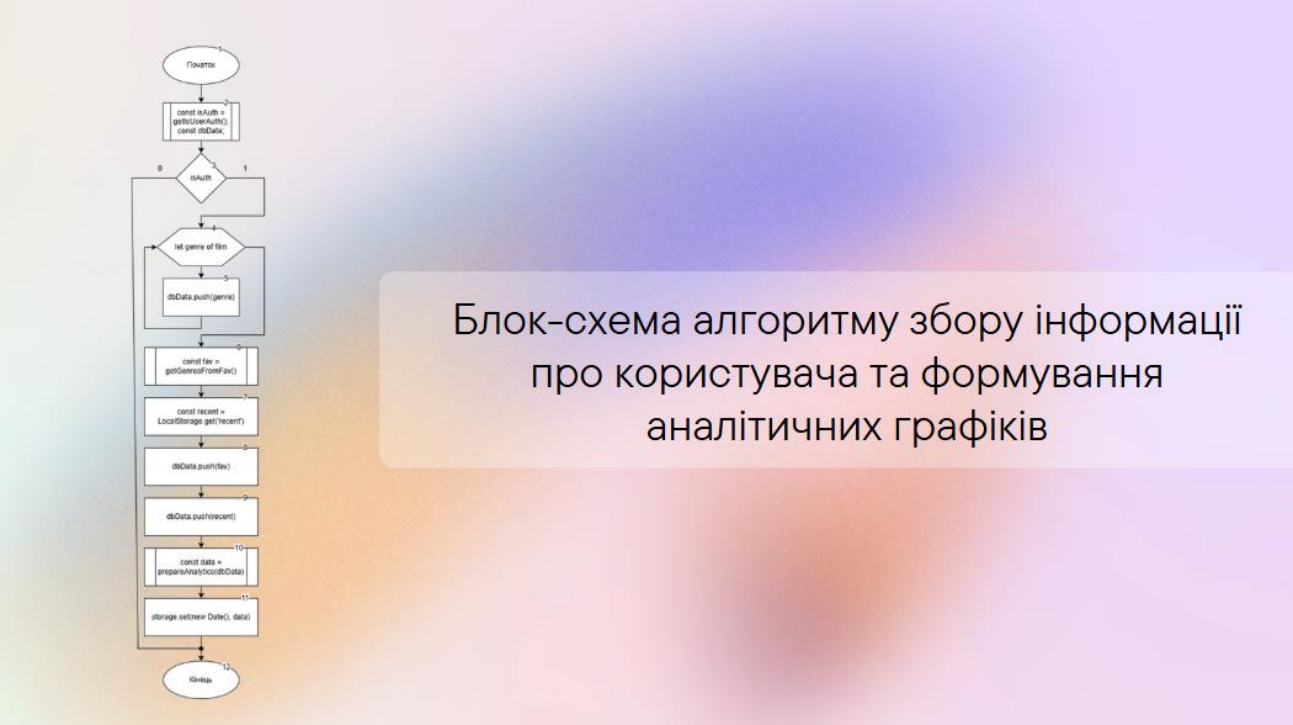

Рисунок Г.15 – Блок-схема алгоритму збору інформації про користувача та формування аналітичних графіків

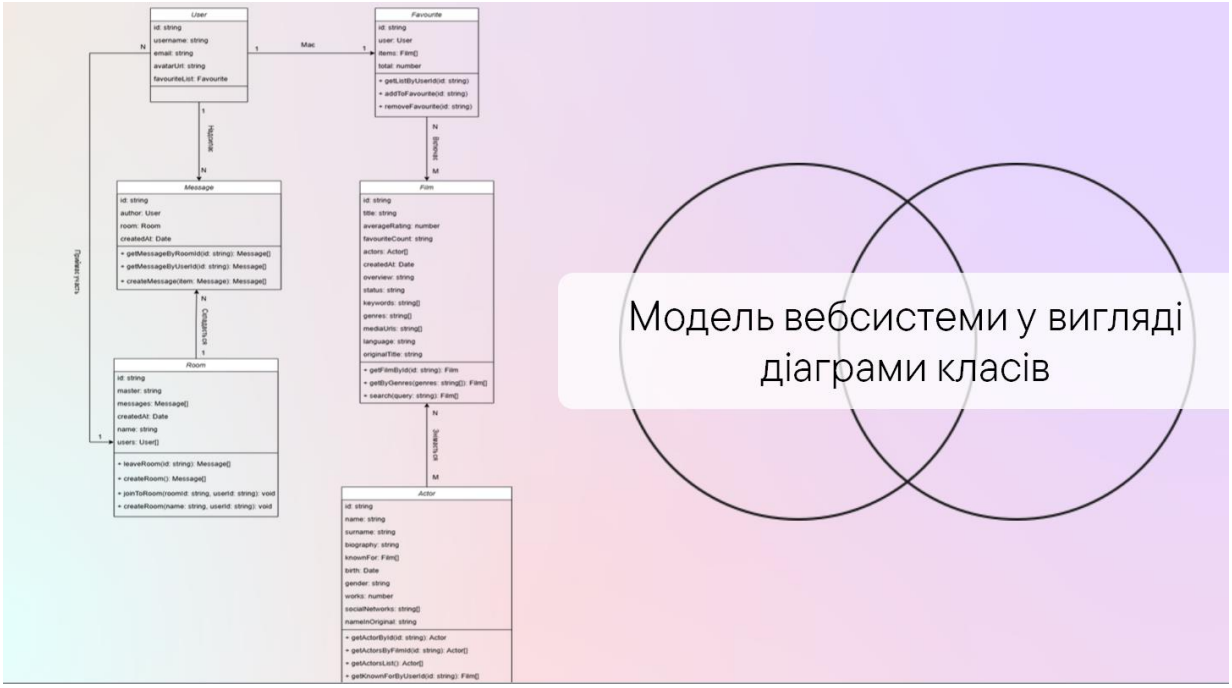

Рисунок Г.16 – Модель системи у вигляді діаграми класів

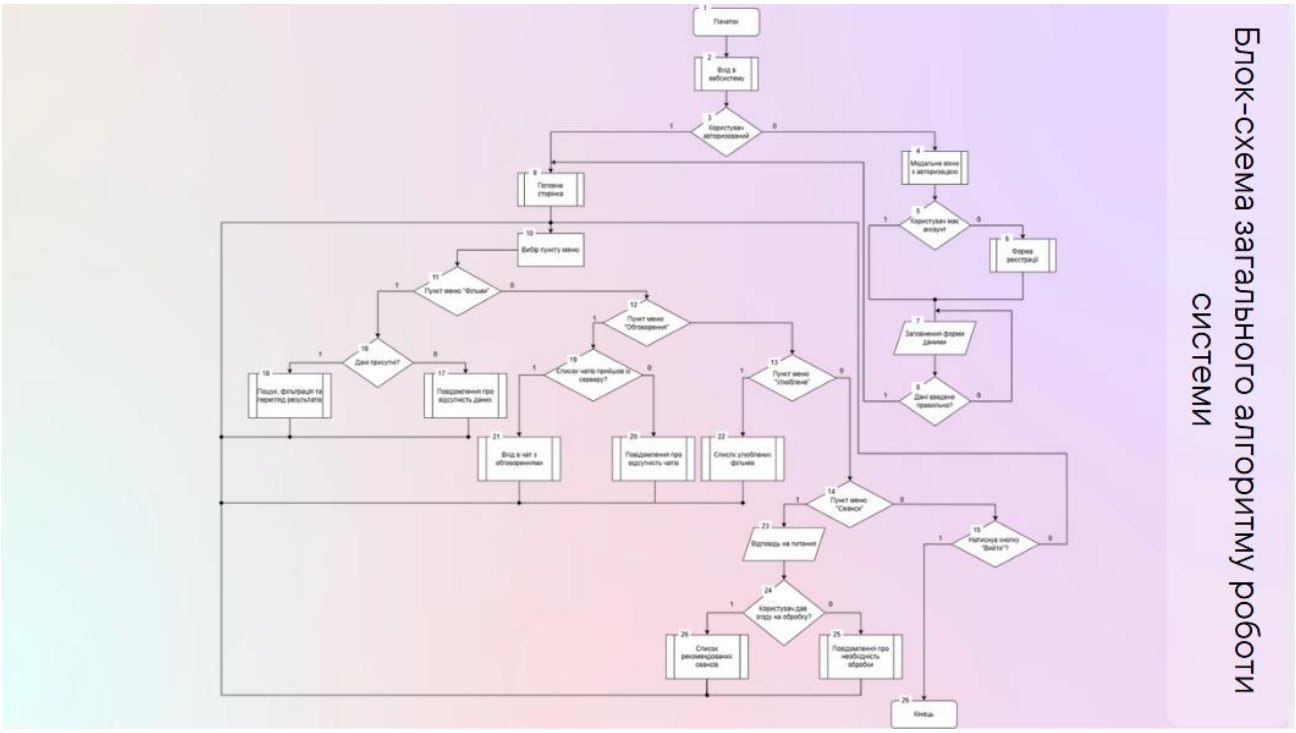

Рисунок Г.17 – Блок-схема загального алгоритму роботи системи

Рисунок Г.18 – Схеми інтерфейсу

Рисунок Г.19 – Діаграма станів модуля обговорення контенту

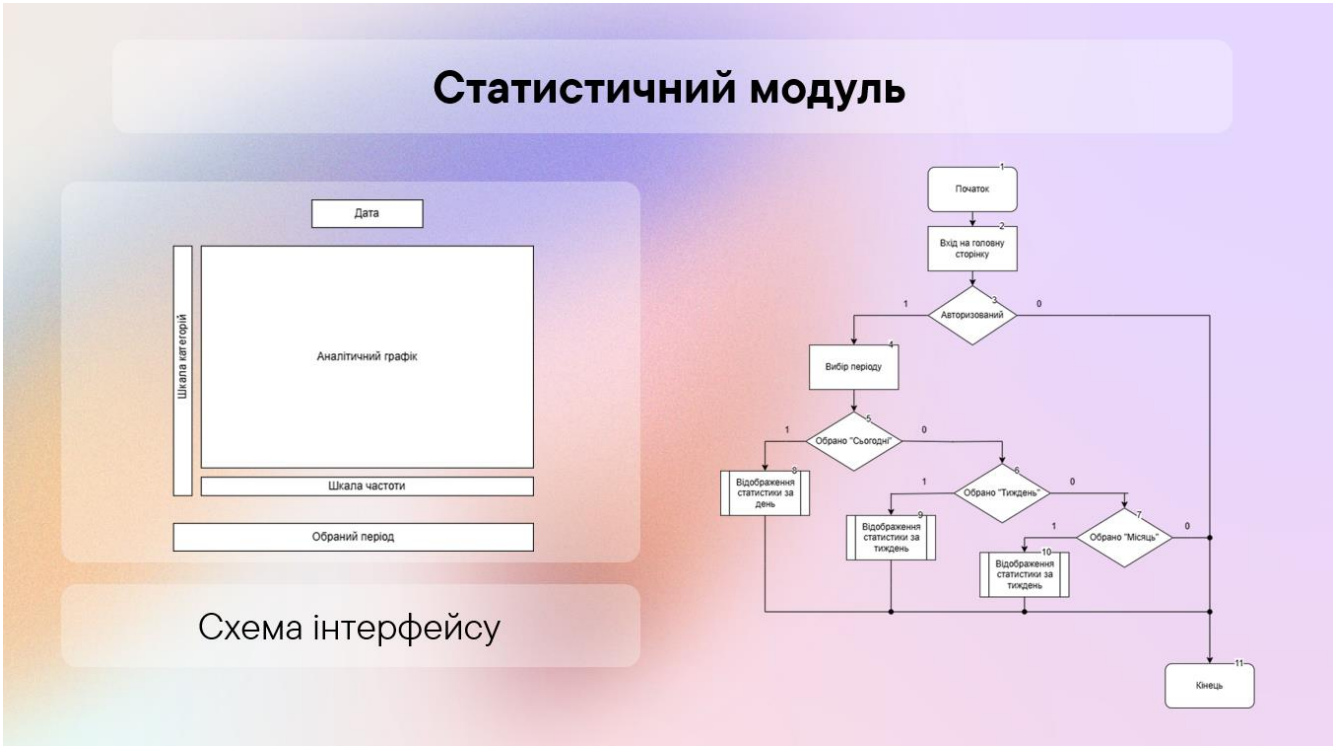

Рисунок Г.20 – Схема інтерфейсу та загальний алгоритм статистичного модуля

Рисунок Г.21 – Використані технології

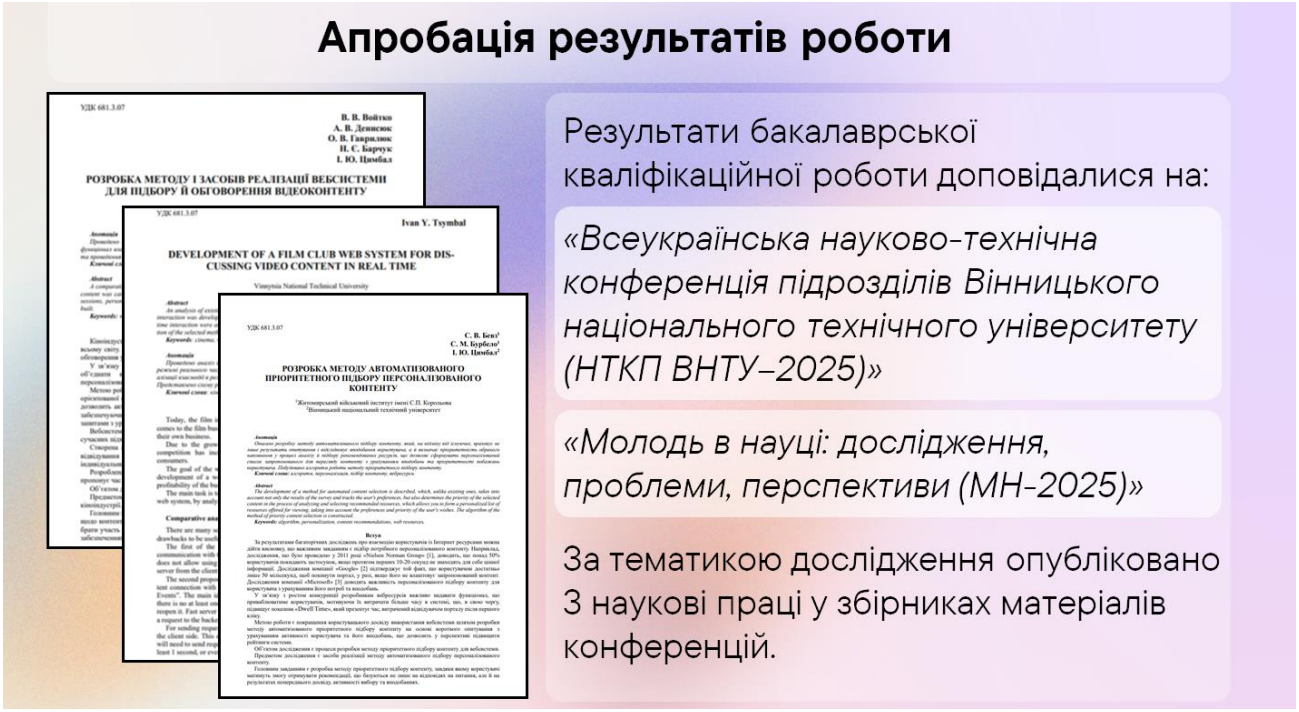

Рисунок Г.22 – Апробація та публікація результатів роботи

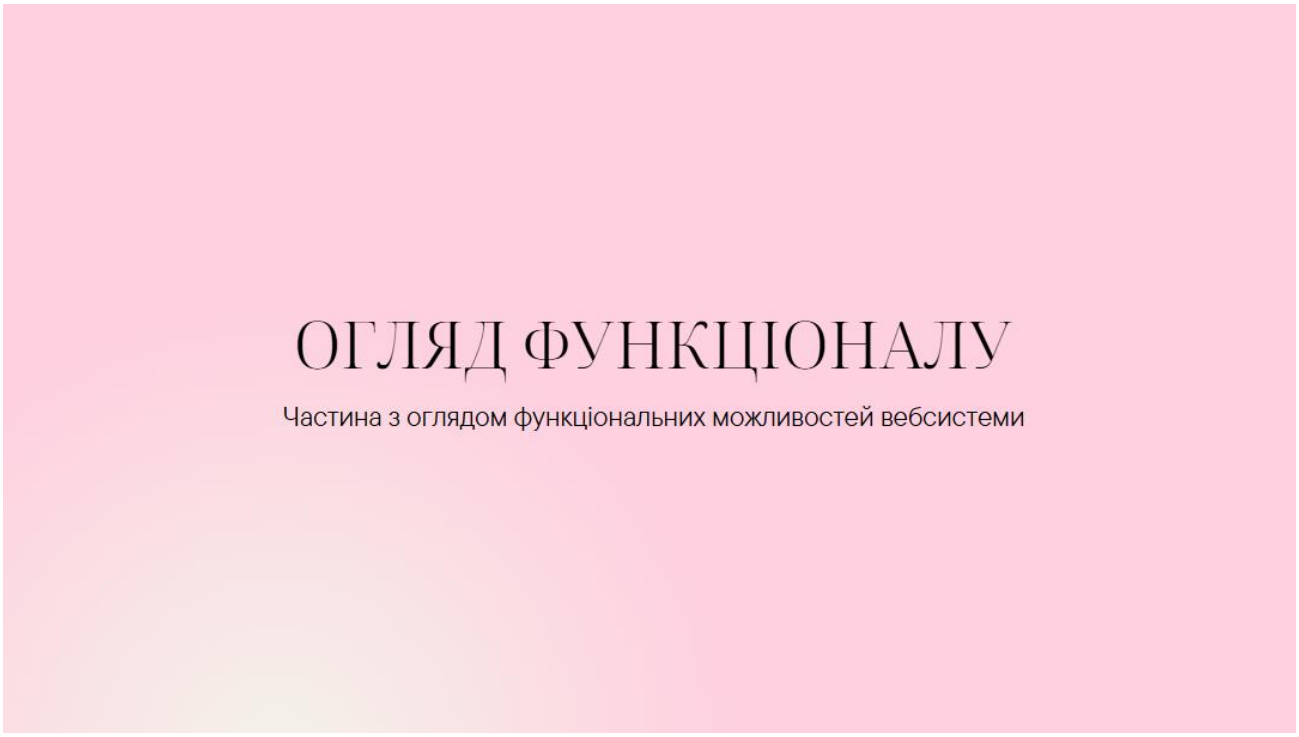

Рисунок Г.23 – Вступний слайд в огляд функціоналу

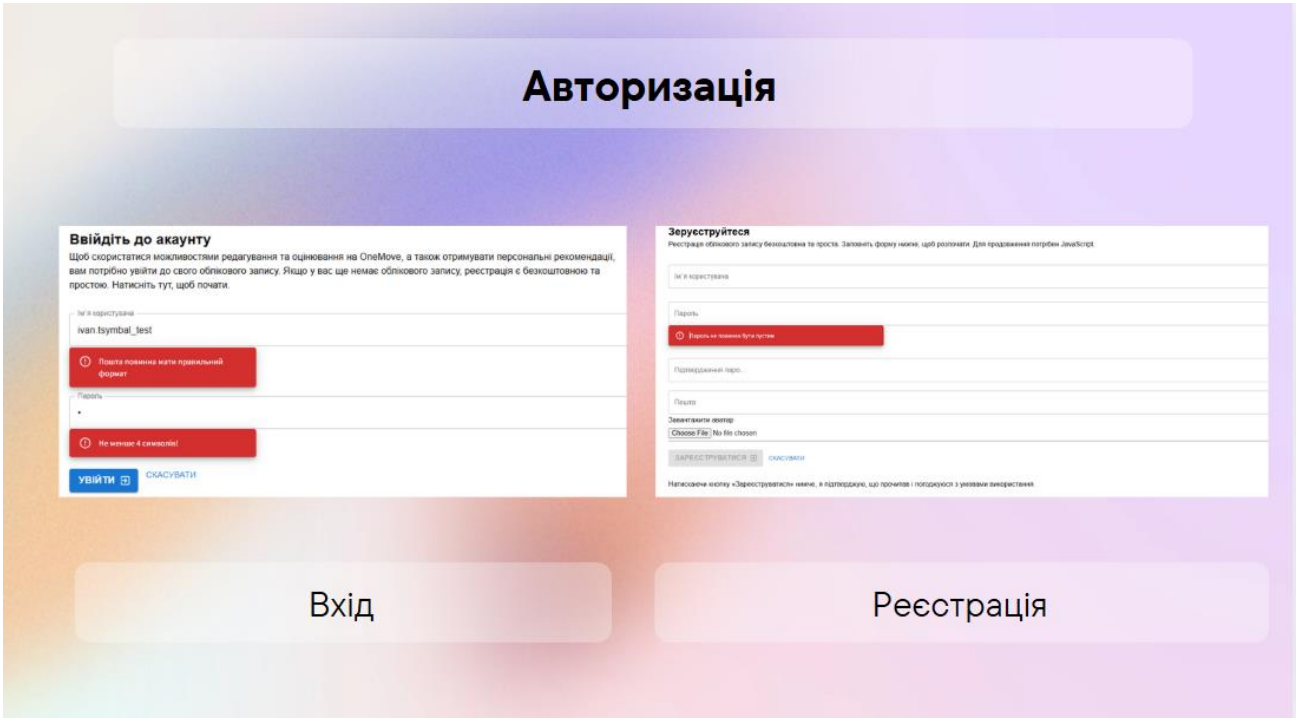

Рисунок Г.24 – Огляд функціоналу авторизації

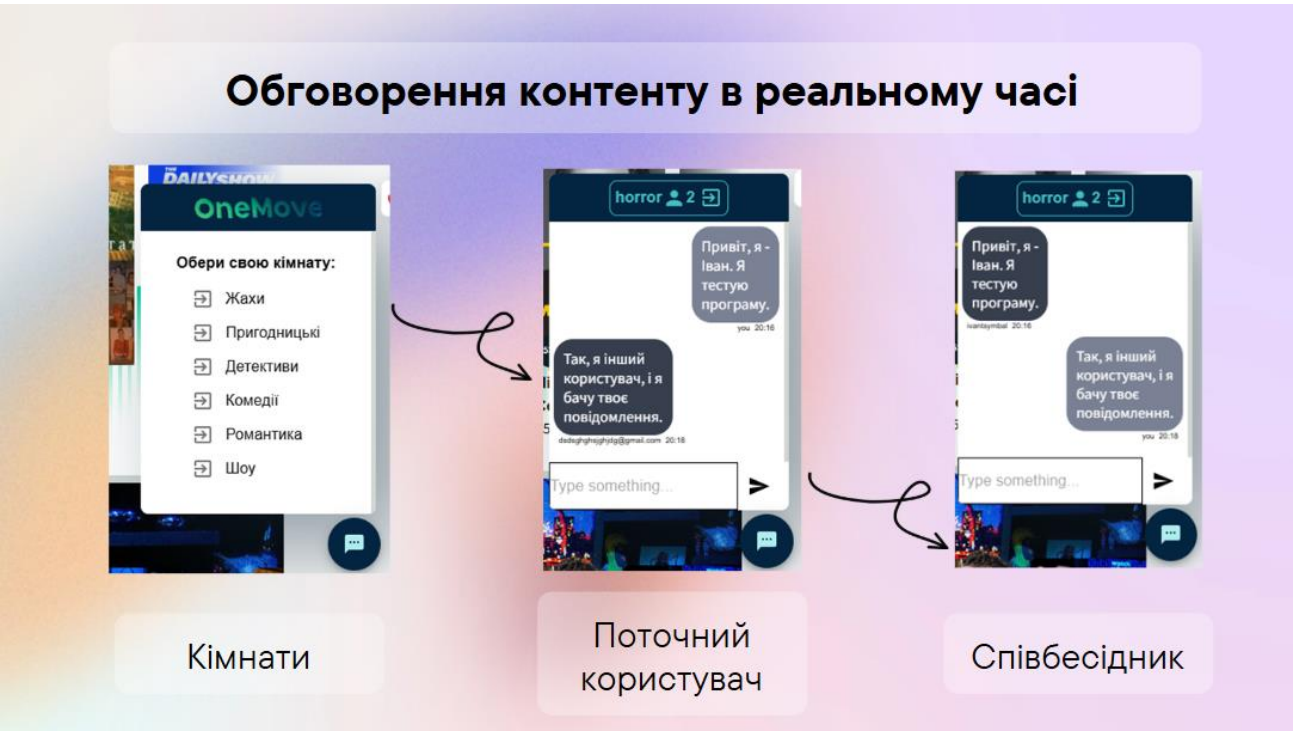

Рисунок Г.25 – Огляд функціоналу обговорення контенту в реальному часі

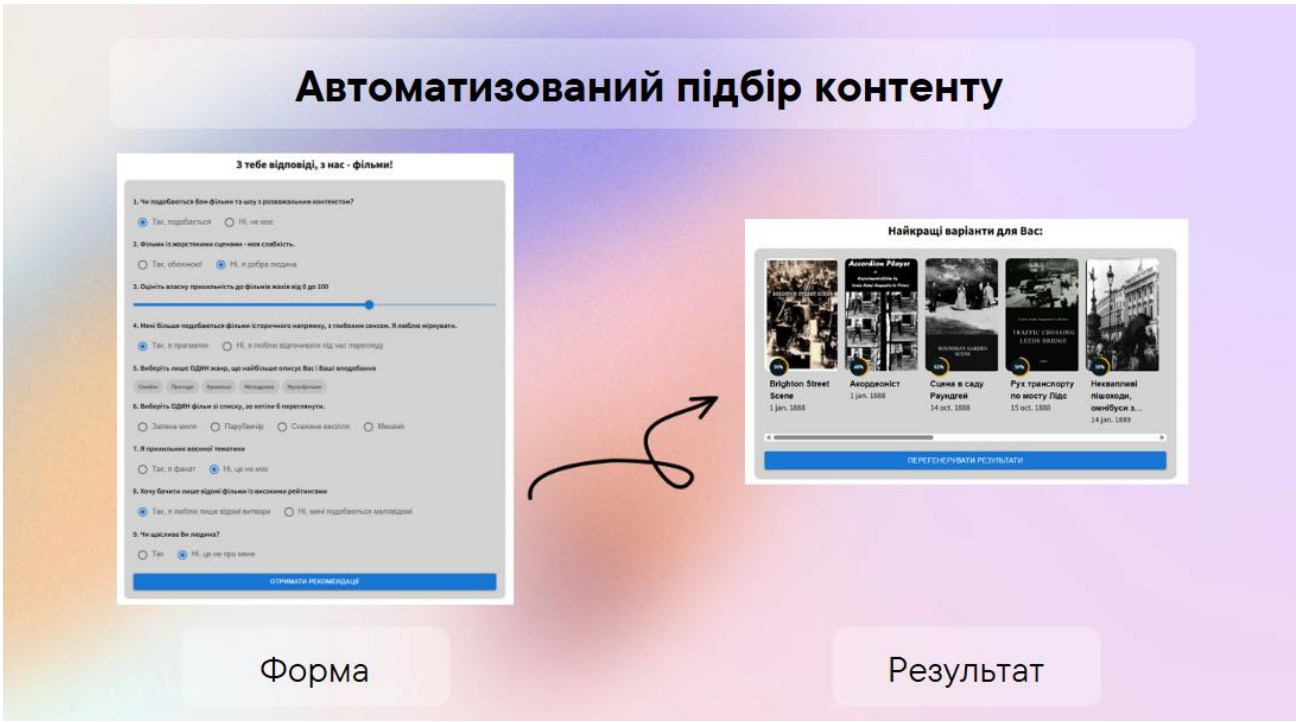

Рисунок Г.26 – Огляд функціоналу автоматизованого підбору контенту

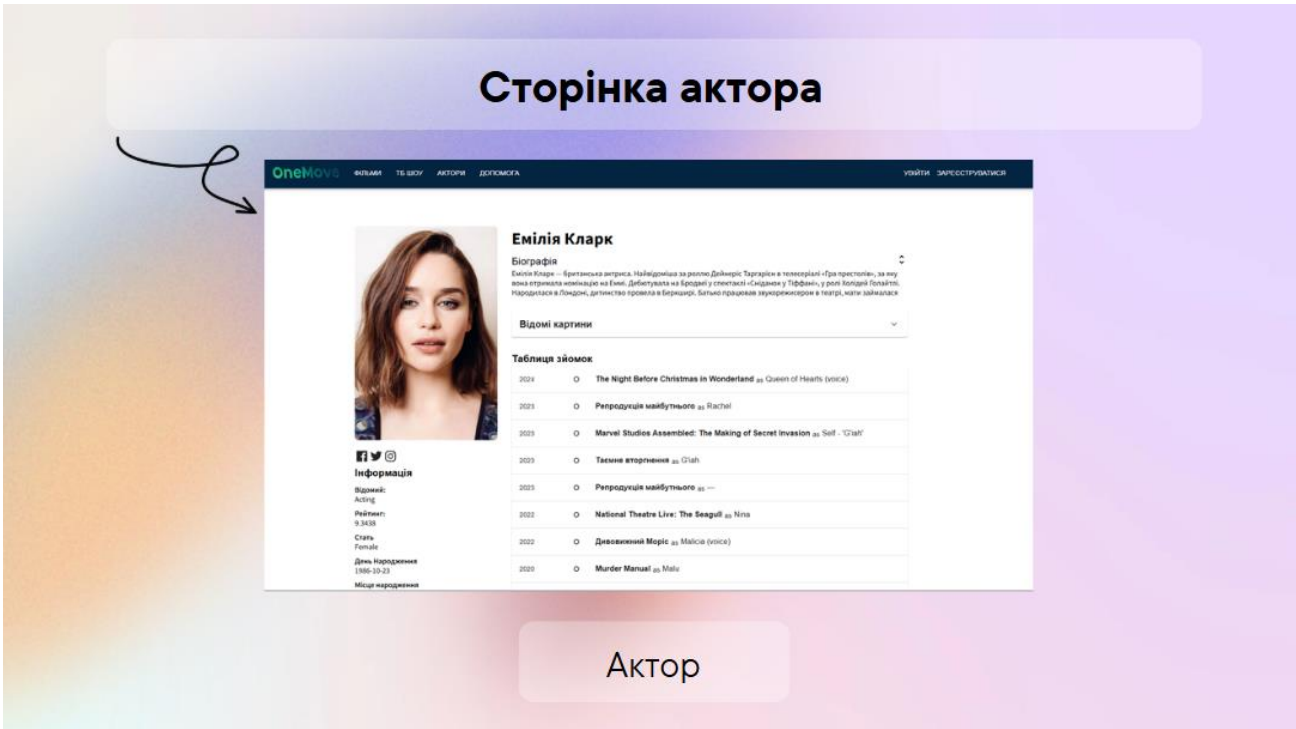

Рисунок Г.27 – Огляд функціоналу сторінки актора

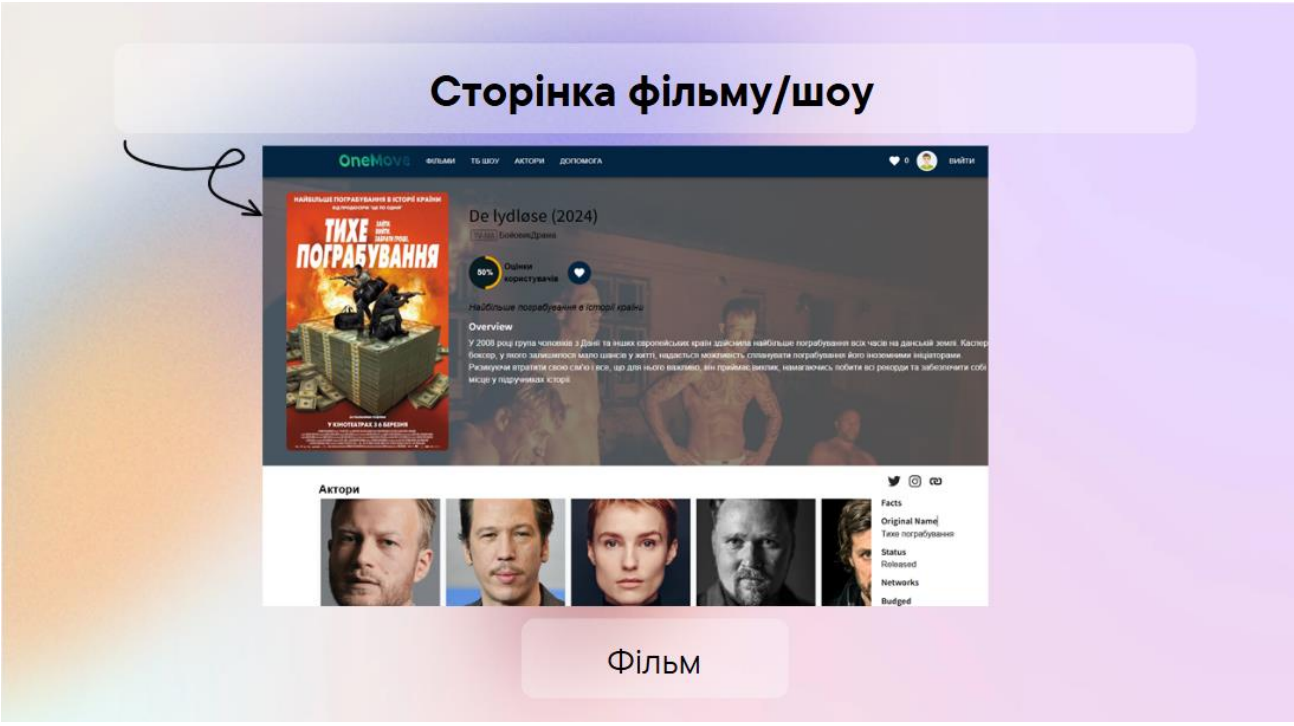

Рисунок Г.28 – Огляд сторінки фільму

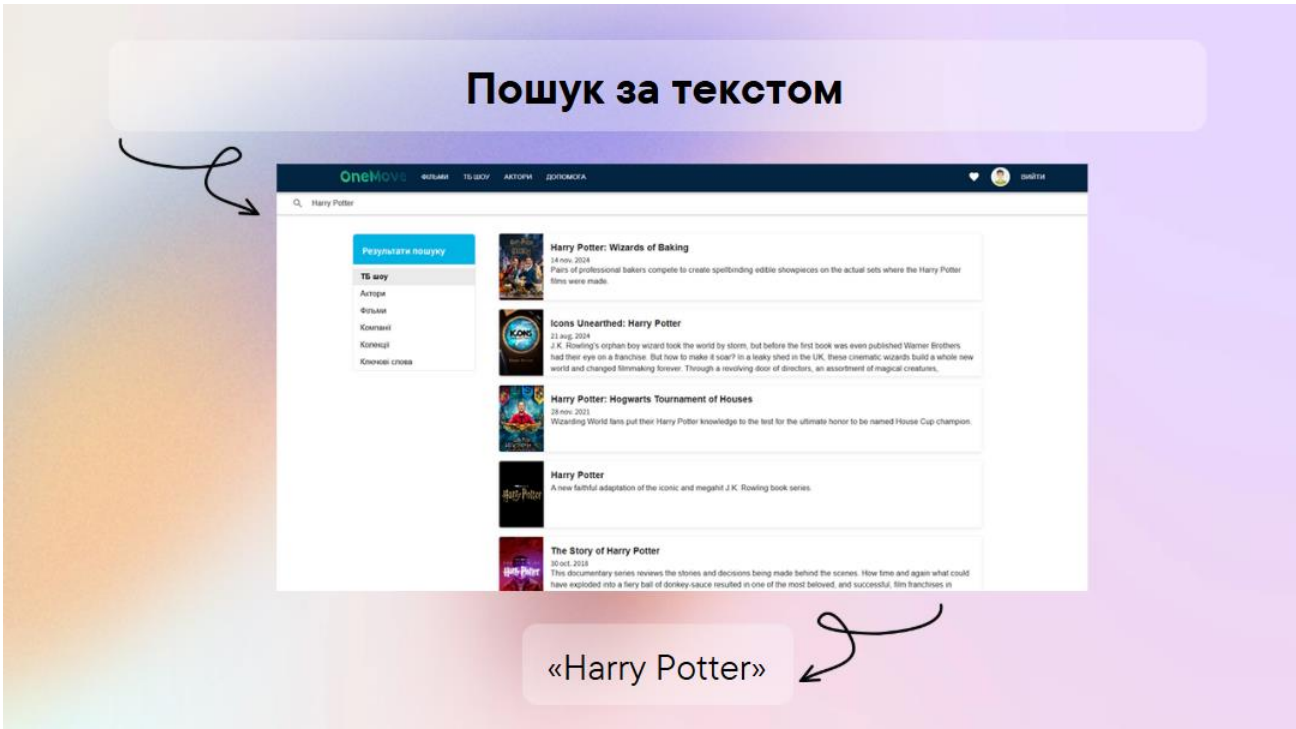

Рисунок Г.29 – Огляд функціоналу пошуку за текстом

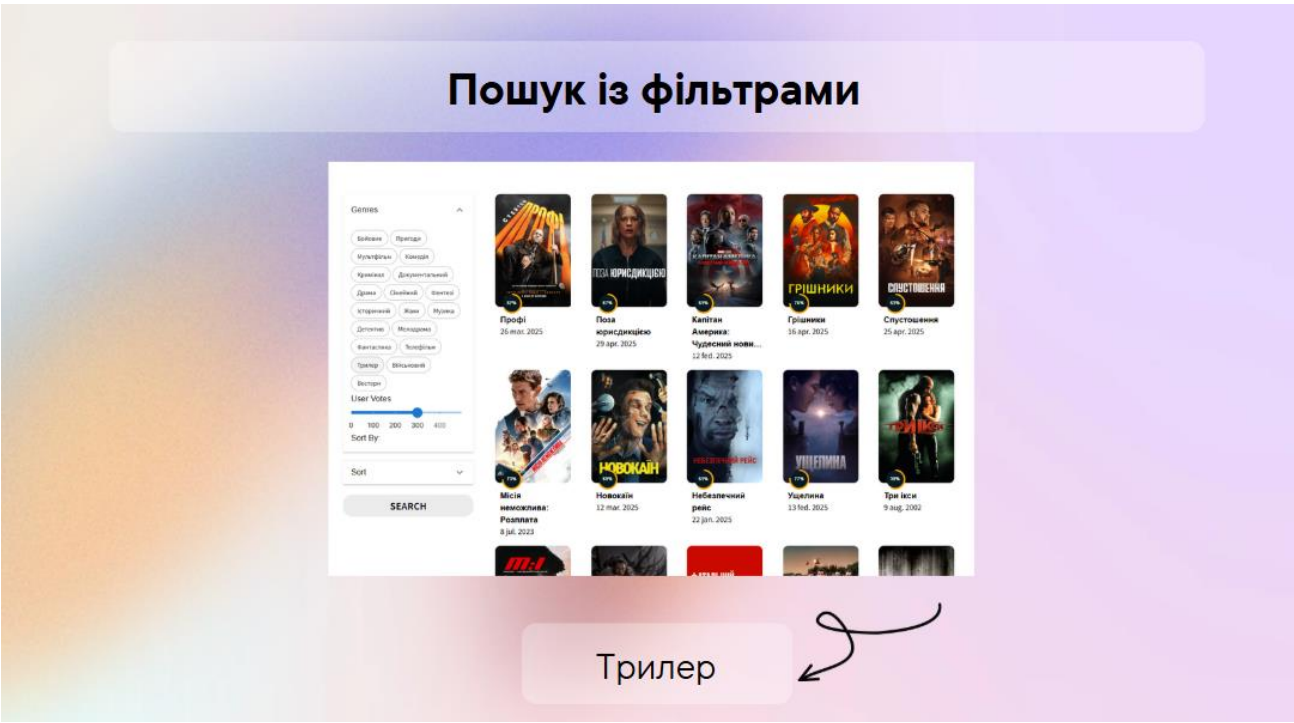

Рисунок Г.30 – Огляд функціоналу пошуку з фільтрацією

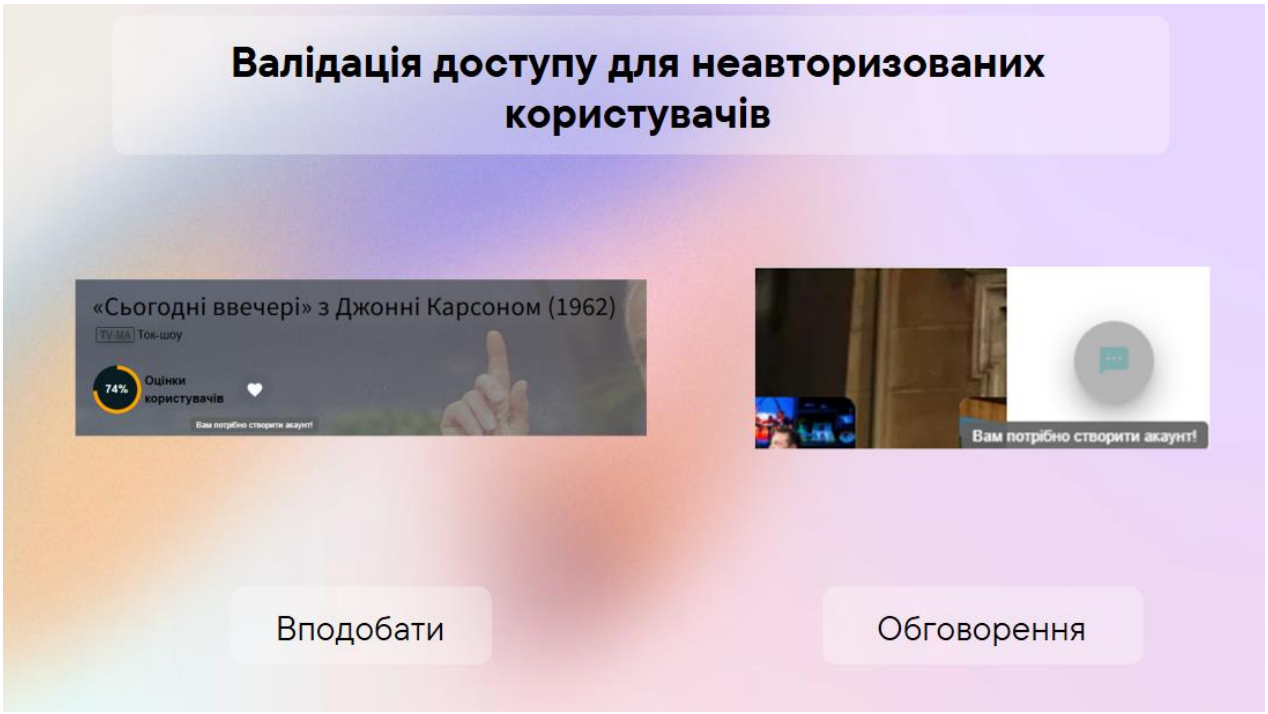

Рисунок Г.31 – Огляд функціоналу з доступом для авторизованих користувачів

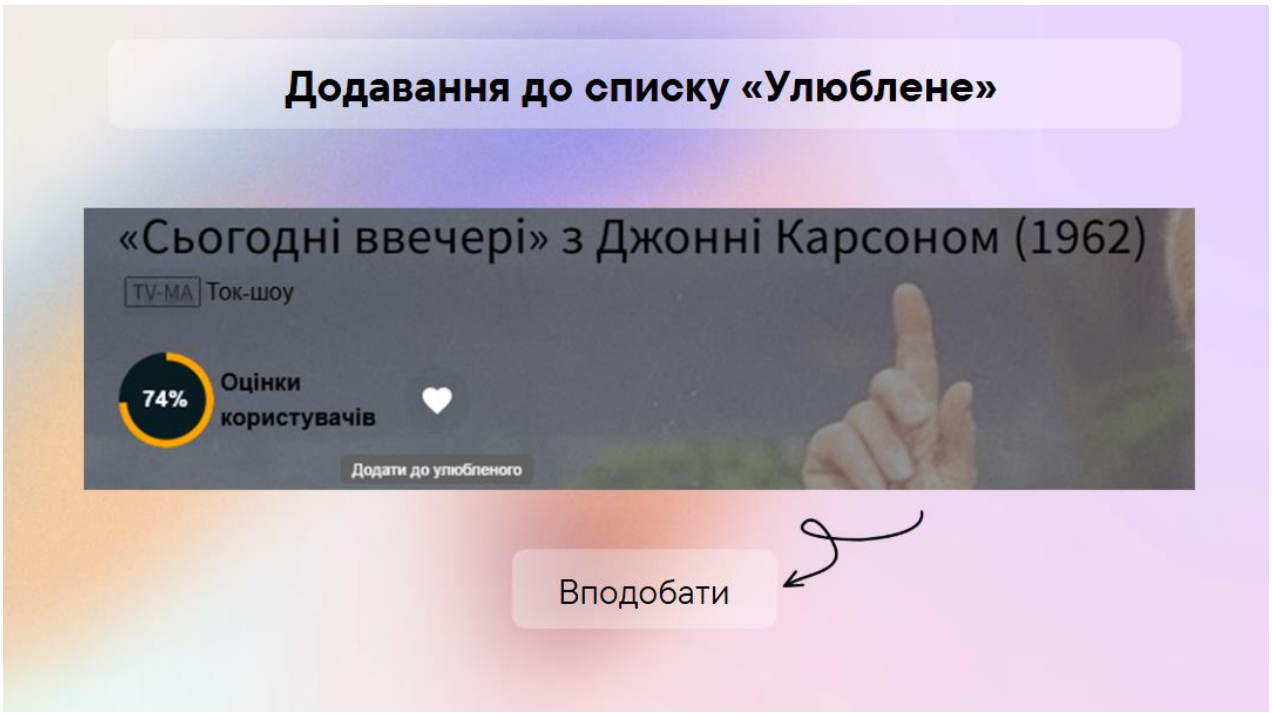

Рисунок Г.32 – Огляд функціоналу додавання до улюбленого

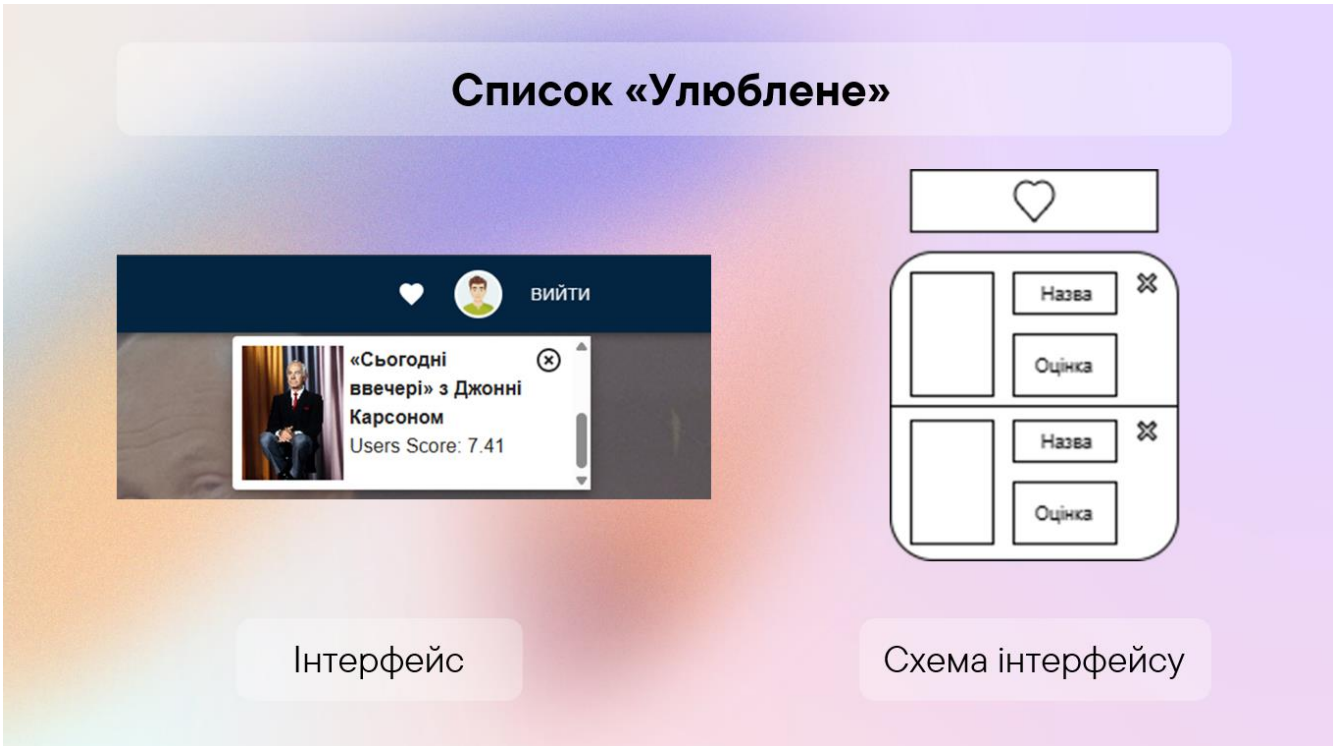

Рисунок Г.33 – Огляд списку улюбленого

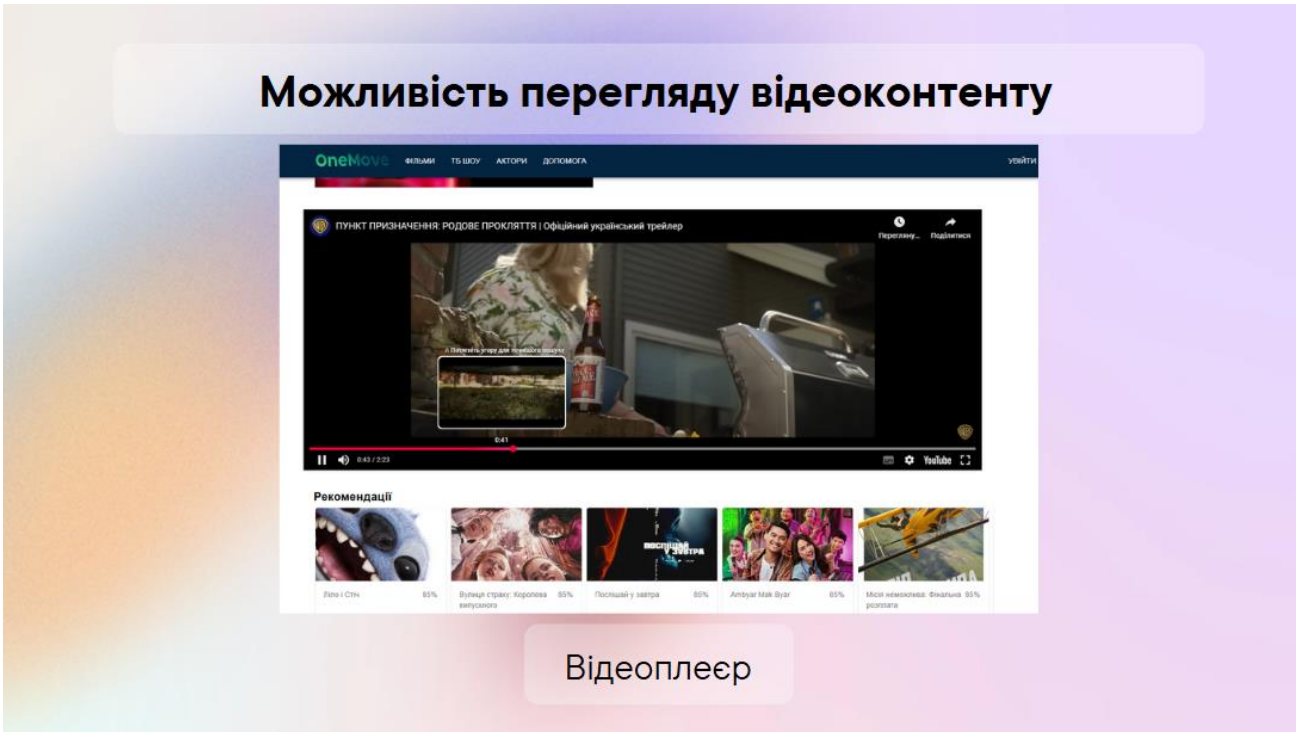

Рисунок Г.34 – Огляд функціоналу перегляду відеоконтенту

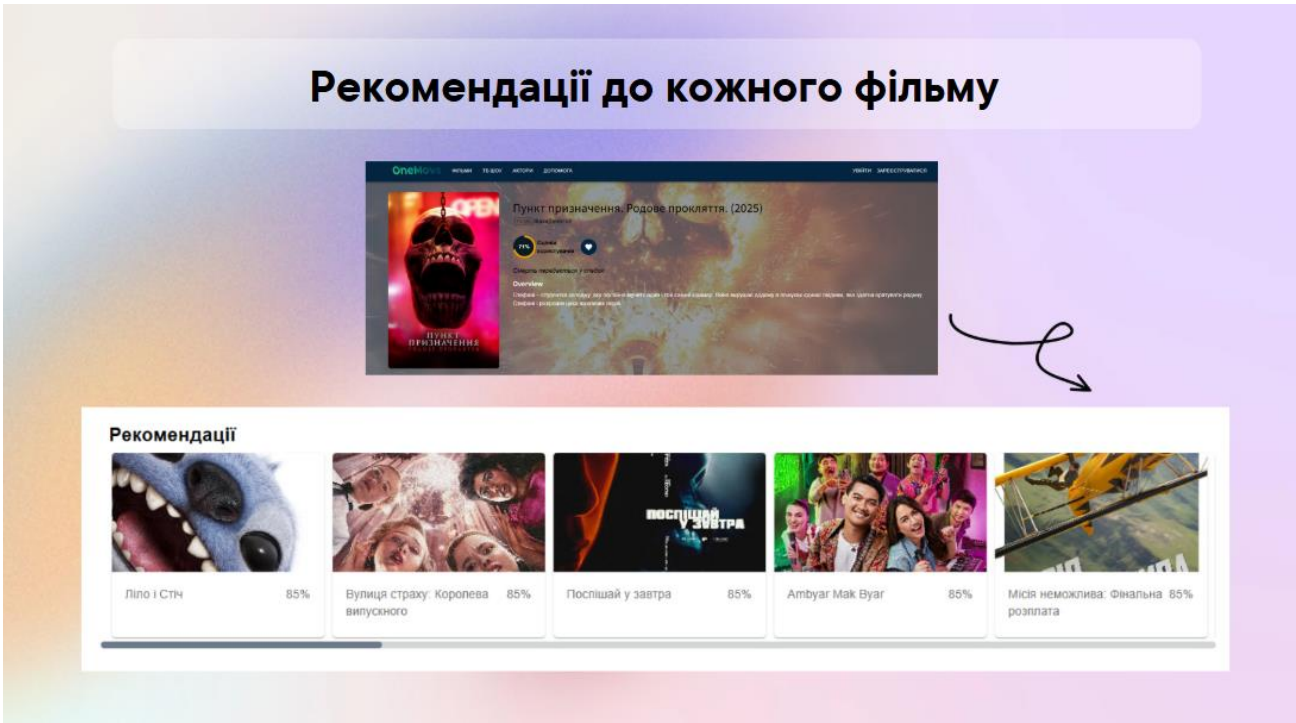

Рисунок Г.35 – Огляд рекомендацій до фільму

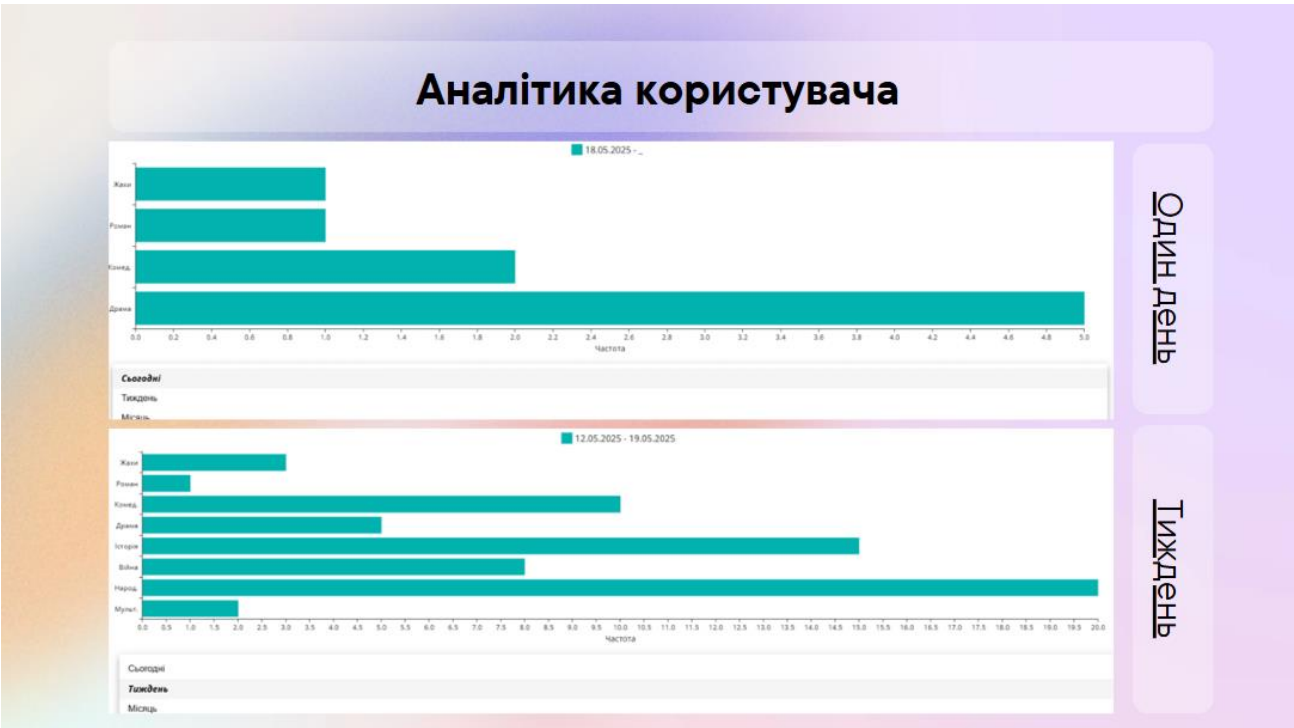

Рисунок Г.36 – Огляд функціоналу аналітики

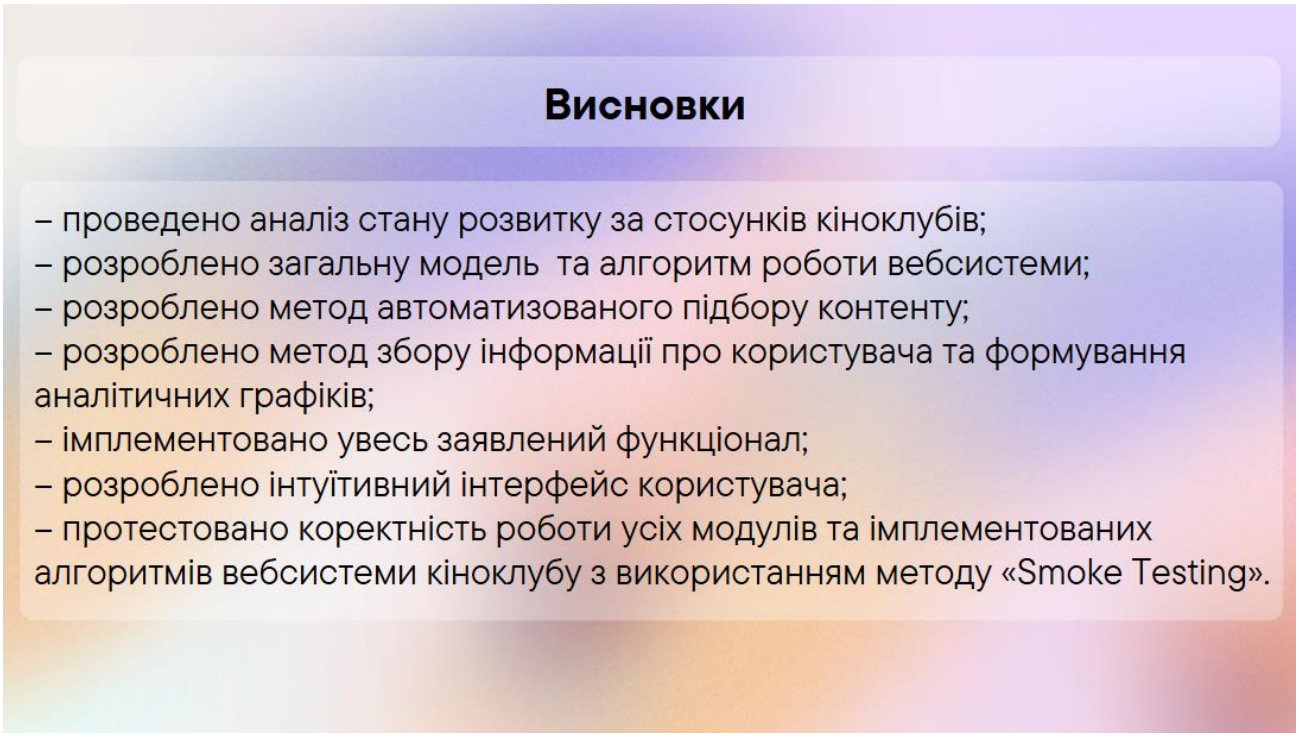

Рисунок Г.37 – Висновки

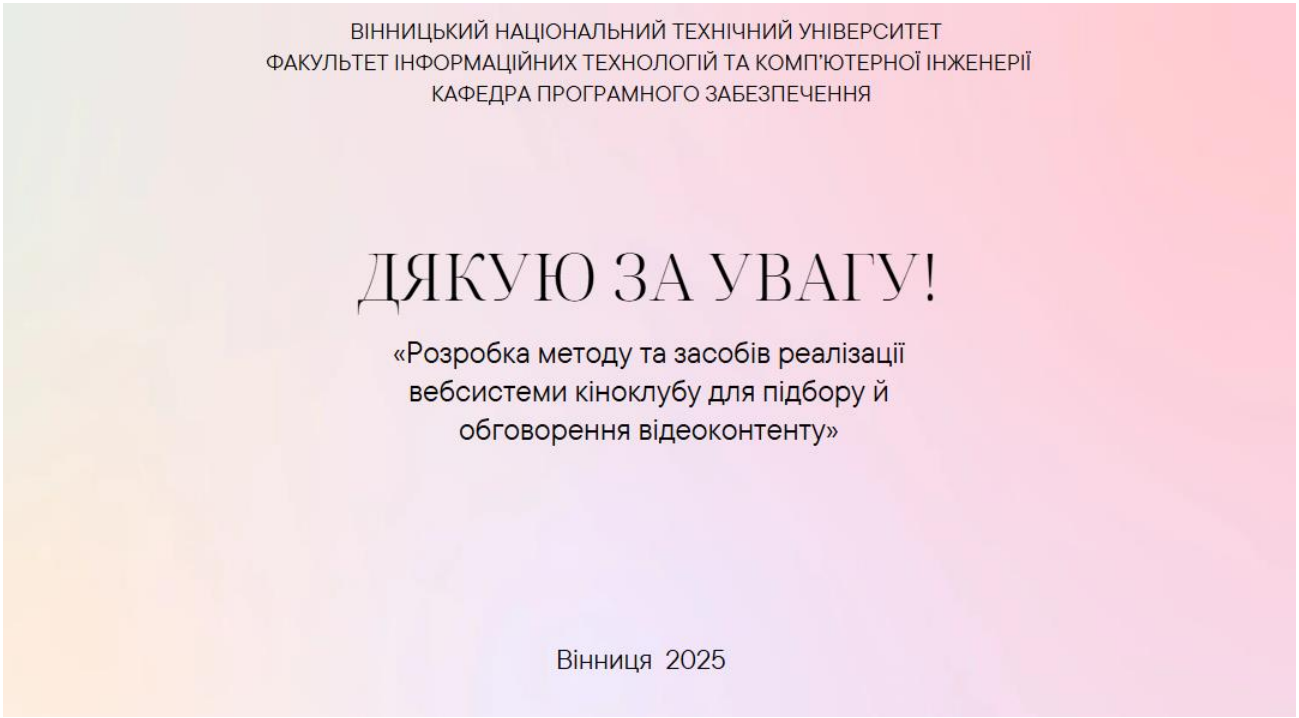

Рисунок Г.38 – Кінцевий слайд