

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

ПОЯСНЮВАЛЬНА ЗАПИСКА

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

на тему:

«Вебсервіс перегляду кінофільмів»

Виконала студентка групи ЗІПЗ-21-1
ЯЦЕНКО Вероніка Павлівна

Керівник роботи:
ЧИЖМОТРЯ Олексій Володимирович

Рецензент:
ЛОБАНЧИКОВА Надія Миколаївна

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

«ЗАТВЕРДЖУЮ»

Зав. кафедри інженерії
програмного забезпечення
Тетяна ВАКАЛЮК
«14» лютого 2025 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **ЯЦЕНКО Вероніка Павлівна**

Керівник роботи: **ЧИЖМОТРЯ Олексій Володимирович**

Тема роботи: **«Вебсервіс перегляду кінофільмів»**,

затверджена Наказом закладу вищої освіти від **«14» лютого 2025 р. №62с**

Вихідні дані для роботи: вебсервіс для перегляду кінофільмів, застосування сучасних вебтехнологій для розробки зручного інтерфейсу та ефективної взаємодії з API бази даних фільмів (зокрема TMDb API) для реалізації динамічного та адаптивного вебдодатку.

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

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Чижмотря О.В,	14.02.2025	14.02.2025
2	Чижмотря О.В,	14.02.2025	14.02.2025
3	Чижмотря О.В,	14.02.2025	14.02.2025

РЕФЕРАТ

Випускна кваліфікаційна робота складається з програмної реалізації сервісу перегляду кінофільмів, а також пояснювальної записки. Пояснювальна записка до дипломної роботи містить 54 сторінки, 27 ілюстрацій, 8 таблиць, 1 додаток та 31 літературне джерело.

Метою роботи є дослідження та розробка сервісу перегляду кінофільмів, який надасть користувачам прозорий та зручний інтерфейс для перегляду кінофільмів.

В роботі описано процес аналізу технічного завдання та вимог системи, оглянуто існуюче програмне забезпечення за тематикою проєкту. Також описано процес розробки бази даних та реалізації сервісу перегляду кінофільмів. Далі зазначено порядок встановлення та налаштування сервісу, приклади тестування та опис інтерфейсу додатку.

КЛЮЧОВІ СЛОВА: REACT, МОВА ПРОГРАМУВАННЯ, КЛІЄНТ, API, БАЗА ДАНИХ, MONGODB NODE.JS.

					ІПЗ.КР.Б – 121 – 25 – ПЗ						
Змн.	Арк.	№ докум.	Підпис	Дата	«Вебсервіс перегляду кінофільмів» Пояснювальна записка			Літ.	Арк.	Аркушів	
Розроб.		Яценко В.П.									
Керівник		Чижевська О.В.								3	56
								Житомирська політехніка, група ЗІПЗ-21-1			
Рецензент		Лобанчикова Н.М.									
Зав. каф.		Вакалюк Т.А.									

ABSTRACT

The final qualification project consists of the software implementation of a movie streaming service and an explanatory report.

The explanatory report includes 54 pages, 27 illustrations, 8 tables, 1 appendices, and 31 literature sources.

The aim of the project is to research and develop a movie streaming service that provides users with a transparent and convenient interface for watching films.

The report describes the process of analyzing the technical requirements and system specifications, as well as an overview of existing software related to the project's topic. It also covers the development of the database and the implementation of the movie streaming service. Further, it outlines the steps for installation and configuration of the service, provides testing examples, and presents a description of the application interface.

KEYWORDS: REACT, PROGRAMMING LANGUAGE, CLIENT, API, DATABASE, MONGODB, NODE.JS.

		Яценко. В.П			ІІЗ.КР.Б – 121 – 25 – ІІЗ	Арк.
		Чижмотря О.В.				4
Змн.	Арк.	№ докум.	Підпис	Дата		

ЗМІСТ

РЕФЕРАТ	3
ABSTRACT	4
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ.....	6
ВСТУП	7
РОЗДІЛ 1. АНАЛІЗ ВИМОГ ТА КОНЦЕПТУАЛЬНЕ МОДЕЛЮВАННЯ СЕРВІСУ ПЕРЕГЛЯДУ КІНОФІЛЬМІВ.....	8
1.1 Технічне завдання на розробку сервісу	8
1.2 Аналіз існуючого програмного забезпечення за тематикою проєкту	9
1.3 Аналіз вимог сервісу та користувачів сервісу	13
1.4 Обґрунтування вибору інструментальних засобів та архітектури сервісу перегляду кінофільмів	14
Висновки до першого розділу.....	17
РОЗДІЛ 2. ПРОЄКТУВАННЯ СЕРВІСУ ПЕРЕГЛЯДУ КІНОФІЛЬМІВ	18
2.1 Опис акторів та алгоритмів роботи	18
2.2 Розробка бази даних	20
2.3 Комунікації і послідовність взаємодії об'єктів системи.....	24
2.4 Реалізація сервісу перегляду кінофільмів	25
Висновки до другого розділу	28
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З СЕРВІСОМ ПЕРЕГЛЯДУ КІНОФІЛЬМІВ	30
3.1 Порядок встановлення та налаштування системи.....	30
3.2 Опис інтерфейсу програмного продукту.....	31
3.3 Тестування сервісу перегляду кінофільмів	37
Висновки до третього розділу	38
ВИСНОВКИ.....	39
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	40
ДОДАТКИ.....	43

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

БД – База даних;

ПЗ – Програмне забезпечення;

СКБД – Система керування базами даних;

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

		Яценко. В.П			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Чижмотря О.В.				6
Змн.	Арк.	№ докум.	Підпис	Дата		

ВСТУП

У сучасному цифровому світі значна частина розваг, у тому числі перегляд кінофільмів, перемістилася в онлайн-простір. Швидкий розвиток технологій, зростання кількості користувачів інтернету та зручність доступу до контенту створили високий попит на якісні вебсервіси для перегляду фільмів. Такі сервіси мають забезпечувати зручний інтерфейс, ефективну навігацію, можливості персоналізації та надійність роботи, що робить їх розробку актуальною темою в галузі інформаційних технологій.

Метою цієї роботи є розробка функціонального та зручного вебсервісу для перегляду кінофільмів, який дозволить користувачам знаходити та переглядати фільми за жанрами, країнами, мовами, рейтингами, зберігати вподобане, формувати списки «переглянутого» та «бажаного до перегляду», а також залишати оцінки.

Досягнення поставленої мети потребує вирішення низки послідовних завдань. Спершу необхідно сформулювати технічне завдання на розробку вебсервісу. Далі слід проаналізувати наявні аналоги та оцінити їхній функціонал. На основі цього визначаються вимоги як до самої системи, так і до її користувачів. Наступним етапом є проектування архітектури вебзастосунку та його інтерфейсу. Після цього здійснюється безпосередня розробка сервісу із застосуванням сучасних вебтехнологій. Після реалізації важливо провести тестування, аби перевірити функціональність системи та її зручність для користувача. Завершальним кроком є налаштування сервісу для його розгортання та подальшої експлуатації.

Об'єктом дослідження є вебсервіс для перегляду кінофільмів, орієнтований на кінцевого користувача.

Предметом дослідження є застосування сучасних вебтехнологій для розробки зручного інтерфейсу та ефективної взаємодії з API бази даних фільмів (зокрема TMDb API) для реалізації динамічного та адаптивного вебдодатку.

		Яценко. В.П			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Чижомотря О.В.				7
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 1. АНАЛІЗ ВИМОГ ТА КОНЦЕПТУАЛЬНЕ МОДЕЛЮВАННЯ СЕРВІСУ ПЕРЕГЛЯДУ КІНОФІЛЬМІВ

1.1 Технічне завдання на розробку сервісу

Назва системи – вебсервіс перегляду кінофільмів.

Мета – розробка та впровадження вебсервісу для перегляду кінофільмів, який надає користувачам інструменти для зручного пошуку, фільтрації, перегляду та організації фільмів. Сервіс має забезпечити інтуїтивно зрозумілий інтерфейс, можливість створення персоналізованих списків фільмів (переглянуте, бажане до перегляду), а також інтеграцію з зовнішніми базами даних фільмів.

У сучасних умовах стрімкого розвитку цифрових технологій та зростання попиту на онлайн-розваги, вебсервіси для перегляду фільмів стали важливою частиною дозвілля. Розробка власного веб-сервісу дозволить забезпечити користувачів доступом до якісного контенту, впорядкованого за жанрами, країнами, мовами та іншими критеріями, а також створить умови для персоналізації досвіду перегляду, що є особливо актуальним у сфері сучасних вебрішень.

Сервіс має бути розроблений як веб-додаток із мінімалістичним інтерфейсом, орієнтованим на інтуїтивне користування. Інтерфейс повинен відповідати сучасним стандартам проєктування вебдодатків, а також стандартам користувацьких інтерфейсів Microsoft Windows або UNIX. Він має бути доступним цілодобово, за винятком часу обслуговування чи непередбачених ситуацій. Надійність системи передбачає середній період безперервної роботи в межах 30 робочих днів, при цьому допустимий рівень помилок не повинен перевищувати одну на кожні десять тисяч рядків коду.

Ключовими функціональними можливостями сервісу є реєстрація користувачів, авторизація та аутентифікація, перегляд списку фільмів, а також можливість фільтрації за жанрами й назвами. Адміністратор матиме окремий функціонал для додавання відео до фільмів. Усі ці функції повинні виконуватися із часом відгуку не більше п'яти секунд для типових задач і не більше двадцяти секунд для складних. Сервіс має підтримувати щонайменше

		Яценко. В.П			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Чижмотря О.В.				8
Змн.	Арк.	№ докум.	Підпис	Дата		

300 одночасно активних користувачів, із потенціалом масштабування для обслуговування ще більшої кількості при потребі. Важливо, щоб оновлення версій сервісу здійснювалося автоматично.

Система повинна бути придатною до експлуатації на комп'ютерах із мінімальною апаратною конфігурацією: 2048 МБ оперативної пам'яті, 10 ГБ вільного місця на диску, процесором із частотою 2 ГГц, операційною системою Windows або Linux та активним інтернет-з'єднанням. В якості системи керування базами даних доцільно використовувати документоорієнтовану СКБД, зокрема MongoDB.

Сервіс не повинен використовувати або зберігати інформацію, що суперечить моральним нормам, законодавству чи може завдати шкоди психіці користувачів. Бізнес-модель сервісу базується на безкоштовному доступі для користувачів із монетизацією через показ реклами, кошти на яку залучаються із власного бюджету. Кожен користувач має можливість вільно користуватися сервісом, а рекламна кампанія проводиться через зовнішні ресурси.

1.2 Аналіз існуючого програмного забезпечення за тематикою проєкту

У процесі розробки вебсервісу для перегляду кінофільмів важливо враховувати вже наявні рішення на ринку, які мають подібну функціональність та орієнтовані на українського користувача. Аналіз таких сервісів дозволяє виявити їх сильні та слабкі сторони, зрозуміти загальні підходи до реалізації функціоналу, а також визначити можливості для покращення майбутньої системи. У цьому розділі буде розглянуто три актуальні платформи, які надають можливість перегляду фільмів онлайн: Megogo, SWEET.TV та Takflix. Їхній функціонал, особливості інтерфейсу, способи взаємодії з користувачем та бізнес-моделі є важливими орієнтирами при проєктуванні власного вебсервісу.

Megogo — це провідний український онлайн-кінотеатр і потоковий відеосервіс, який надає доступ до великого обсягу мультимедійного контенту, включаючи фільми, серіали, телевізійні канали, спортивні трансляції,

		Яценко. В.П			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Чижмотря О.В.				9
Змн.	Арк.	№ докум.	Підпис	Дата		

аудіокниги та подкасти (рис.1.1). Сервіс працює за моделлю передплати, але також дозволяє орендувати окремі фільми за додаткову оплату. Його можна використовувати через веббраузери, мобільні додатки, Smart TV, а також через телевізійні приставки, що забезпечує гнучкість у виборі пристрою. Контент доступний кількома мовами, серед яких обов'язково є українська.

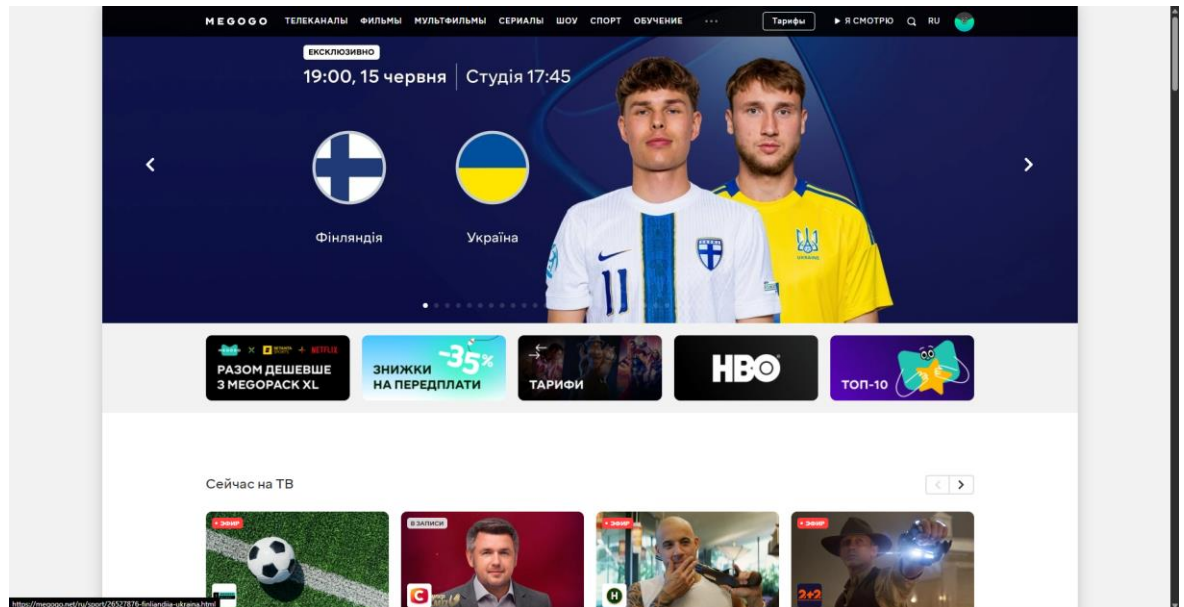


Рис. 1.1 – Інтерфейс Megogo

Однією з головних переваг Megogo є багата медіатека, що охоплює різноманітні жанри та формати. Сервіс має зручну систему підписок, яка дозволяє персоналізувати контент під вподобання кожного користувача. Додатковою перевагою є власне українське дублювання, що підкреслює локалізацію платформи. Інтерфейс Megogo вирізняється якістю та простотою, а підтримка багатьох типів пристроїв робить сервіс універсальним для будь-якого користувача.

Водночас, є й певні недоліки. Зокрема, частина контенту стає доступною лише після окремої оплати, що може створювати незручності для користувачів з базовою підпискою. У безкоштовній версії присутня реклама, яка впливає на комфорт перегляду. Крім того, система категорій іноді може здаватися заплутаною, що ускладнює навігацію по контенту.

Sweet.tv

SWEET.TV — це українська стримінгова платформа (рис.1.2), що об'єднує понад 200 телеканалів і велику колекцію фільмів і серіалів від

		Яценко. В.П			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Чижмоторя О.В.				10
Змн.	Арк.	№ докум.	Підпис	Дата		

найбільших студій (Disney, Sony, Paramount). Особливу увагу сервіс приділяє українській локалізації — SWEET.TV інвестує у створення українського дубляжу та озвучки. Сервіс працює за моделлю підписки (щомісячна оплата), також пропонує тариф з рекламою (FAST TV).

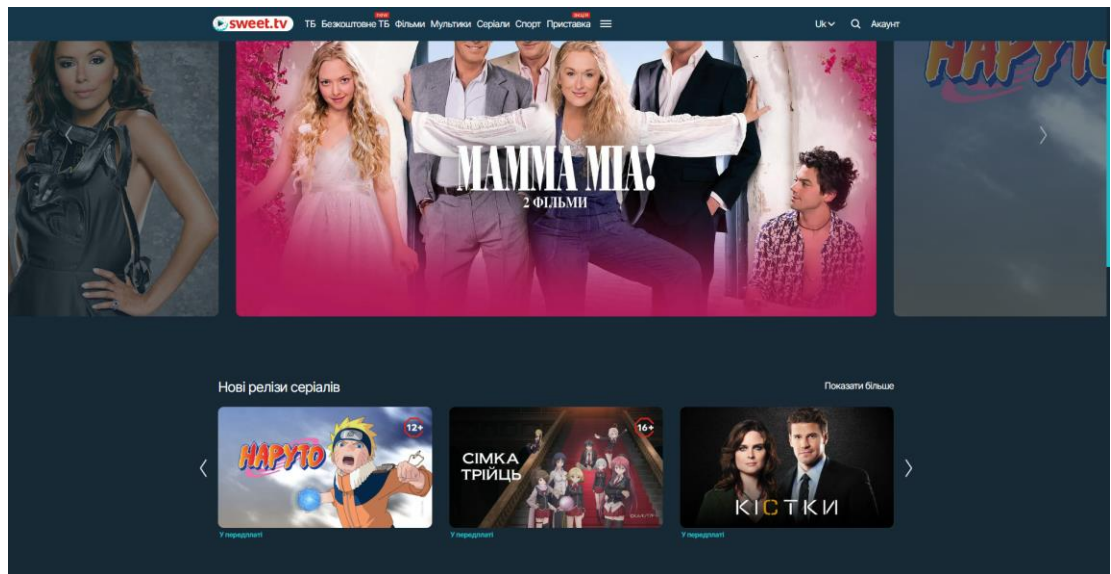


Рис. 1.2 – Інтерфейс Sweet.tv

Sweet.tv вирізняється серед українських стримінгових платформ великою кількістю контенту українською мовою, включаючи власне дублювання фільмів, що забезпечує високий рівень локалізації. Компанія має пряму співпрацю зі студіями, що дозволяє оперативно додавати новинки до каталогу. Крім того, сервіс пропонує доступні тарифні плани, які роблять його привабливим для широкої аудиторії. Однією з додаткових переваг є можливість переглядати онлайн телебачення без потреби в зовнішніх додатках.

Попри це, Sweet.tv має і свої обмеження. У порівнянні з деякими глобальними конкурентами, платформа пропонує менше ексклюзивного контенту, що може впливати на вибір користувачів, які шукають унікальні продукти. Також спостерігається нестабільна робота окремих функцій на застарілих пристроях, що може знижувати загальне враження від користування сервісом.

Takflix

		Яценко. В.П			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Чижмоторя О.В.				11
Змн.	Арк.	№ докум.	Підпис	Дата		

Takflix — це український онлайн-кінотеатр, орієнтований на показ винятково вітчизняного кіно, з особливим акцентом на незалежні та фестивальні стрічки. Однією з ключових особливостей платформи є унікальний контент, який неможливо знайти на інших сервісах. Takflix безпосередньо підтримує українських режисерів, забезпечуючи їм можливість отримувати прибуток від показу своїх фільмів. Модель монетизації побудована на оплаті за перегляд окремих фільмів, що дозволяє користувачам дивитися лише те, що їх цікавить, без обов'язкової підписки. Також платформа пропонує соціальну функцію «перегляд разом», яка дозволяє обговорювати фільм у реальному часі з друзями чи іншими глядачами (рис.1.3).

Однак Takflix має й ряд обмежень. Його каталог включає виключно українські стрічки, що значно звужує вибір для користувачів, які шукають міжнародний контент. Сервіс наразі не має мобільного додатку, що може бути незручним для тих, хто надає перевагу перегляду зі смартфонів або планшетів. Також відсутній загальнодоступний безкоштовний контент, і платформа орієнтована переважно на вузьку аудиторію поціновувачів авторського та фестивального кіно.

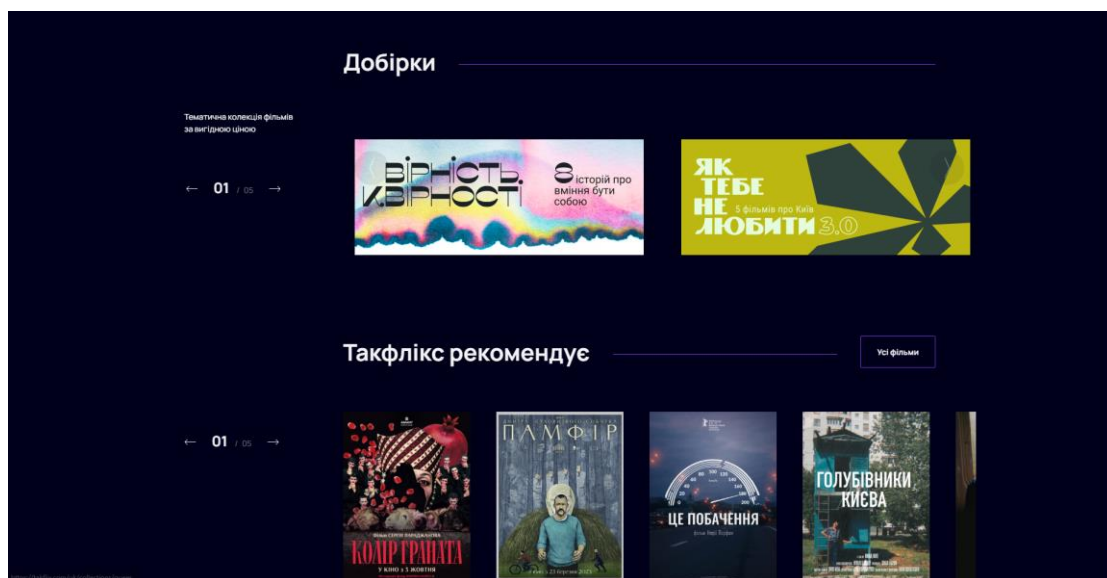


Рис. 1.3 – Інтерфейс Takflix

Для кращого розуміння розглянутих систем, порівняємо їх функціональні можливості (табл. 1.1).

		Яценко. В.П			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Чижмоторя О.В.				12
Змн.	Арк.	№ докум.	Підпис	Дата		

Функціональні можливості

Назва	Megogo	SWEET.TV	Takflix	NikaCinema
Онлайн-перегляд фільмів	+	+	+	+
Підписка	+	+	-	-
Оплата за окремий фільм	+	-	+	-
Пряма трансляція ТБ	+	+	-	-
Мобільний додаток	+	+	-	-
Безкоштовний доступ (з рекламою)	+	+	-	+

1.3 Аналіз вимог сервісу та користувачів сервісу

Мета аналізу вимог полягає у збиранні, узагальненні та формалізації потреб та очікувань майбутніх користувачів вебсервісу перегляду кінофільмів. Вимоги визначають функціональні можливості системи, необхідні різним категоріям користувачів, а також цілі, яких прагне досягти система в контексті зручного та ефективного споживання медіаконтенту.

Подальша технічна реалізація системи, що відповідатиме зазначеним вимогам, буде представлена у вигляді сценаріїв використання та технічних специфікацій, викладених у Додатку А.

Щодо бізнес-вимог, основною метою проєкту є створення зручного вебсервісу для онлайн-перегляду фільмів, який також включає базові соціальні функції. Платформа має надавати легкий доступ до контенту, зокрема і для користувачів, які не проходили реєстрацію. Утім, додаткові можливості — наприклад, збереження улюблених фільмів або історії переглядів — будуть доступні лише після створення облікового запису, що стимулює користувача до реєстрації. Система також повинна забезпечувати адміністративний контроль над контентом: адміністратор повинен мати змогу оновлювати інформацію про фільми та керувати відеоматеріалами. При цьому сервіс має бути достатньо масштабованим, щоб одночасно обслуговувати велику кількість користувачів без втрати швидкодії.

Що стосується вимог до користувачів, то сервіс передбачає три основні ролі. Гість має можливість ознайомлюватися зі списком доступних фільмів і переглядати відкритий контент без обов’язкової авторизації, а також може зареєструватися або увійти у вже створений обліковий запис. Зареєстрований користувач, окрім базових функцій, отримує доступ до особистого кабінету, може додавати фільми до списку обраного і зберігати інші персоналізовані дані. Адміністратор, у свою чергу, має всі функції звичайного користувача, але також може керувати контентом через адміністративну панель, зокрема додавати відео до фільмів.

На основі проаналізованих вимог була створена діаграма варіантів використання (рис 1.4).

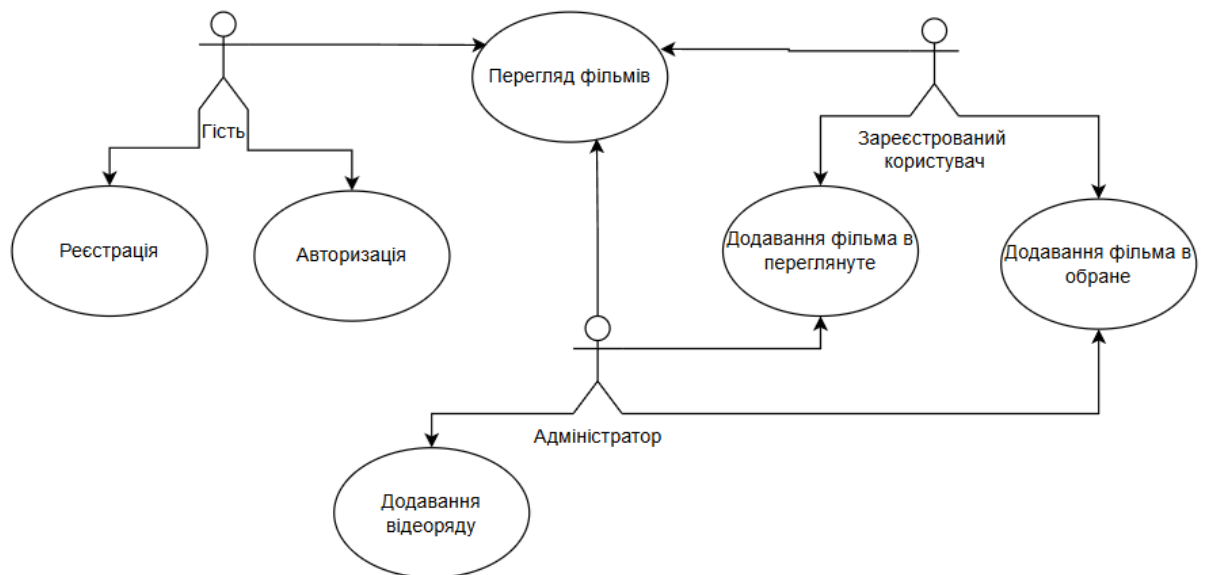


Рис. 1.4 – Діаграма варіантів використання

Діаграма варіантів використання відображає у собі проаналізовані вимоги користувачів

1.4 Обґрунтування вибору інструментальних засобів та архітектури сервісу перегляду кінофільмів

Для розробки вебсервісу перегляду фільмів було обрано клієнт-серверну архітектуру, оскільки саме цей підхід найкраще відповідає вимогам сучасних вебзастосунків, що повинні забезпечувати стабільну роботу з великою кількістю користувачів та централізоване управління даними.

Архітектура дозволяє чітко розмежувати обов'язки між клієнтською та серверною частинами. Інтерфейс користувача, або frontend, відповідає за відображення контенту, взаємодію з користувачем, обробку подій на кшталт реєстрації, додавання улюблених фільмів чи перегляду відео. Серверна частина, або backend, займається обробкою запитів, автентифікацією, керуванням даними про фільми, відеофайлами, а також зберіганням персоналізованих налаштувань користувачів, зокрема історії переглядів і списків обраного.

Однією з вагомих переваг цієї моделі є її масштабованість: серверну частину можна легко адаптувати до зростання кількості користувачів, використовуючи хмарні технології чи системи балансування навантаження. Важливою є й безпека: централізоване зберігання даних дозволяє ефективно реалізувати механізми авторизації, шифрування та контроль доступу.

Клієнт-серверний підхід також значно спрощує процес супроводу та оновлення сервісу — зміни, внесені на сервері, автоматично відображаються у всіх користувачів, без необхідності оновлення застосунку з їхнього боку. Завдяки сумісності з такими технологіями, як REST API або GraphQL, дана архітектура легко дозволяє реалізувати доступ до сервісу з різних платформ — веббраузера, мобільного застосунку чи Smart TV. Крім того, вона підтримує асинхронну роботу, що дає змогу ефективно реалізувати потокове відтворення відео, кешування даних і фонове завантаження.

Таким чином, клієнт-серверна модель дозволяє створити гнучкий, масштабований і зручний у користуванні вебсервіс, який відповідає вимогам сучасного медіаконтенту і забезпечує високу якість користувацького досвіду.

Для розробки сервісу було обрано стек MERN. Даний стек складається з ReactJS для розробки клієнтської частини, базу даних MongoDB, та мову програмування NodeJS з використанням Express для розробки серверної частини.

У процесі розробки серверної частини використовувався фреймворк Express.js — легкий та мінімалістичний фреймворк для Node.js, що забезпечує

		Яценко. В.П			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Чижмотря О.В.				15
Змн.	Арк.	№ докум.	Підпис	Дата		

зручне створення REST API та маршрутизацію запитів. Завдяки Express структура серверного застосунку стала логічною, зрозумілою та легко масштабованою. Express також дозволив ефективно обробляти запити з клієнта, реалізовувати middleware-функціонал для автентифікації, логування та обробки помилок. Додатково використовувалися jsonwebtoken для генерації і перевірки токенів доступу. Для завантаження відеоряду фільмів використовується Multer. Для забезпечення безпечного збереження паролів користувачів використовується бібліотека Bcrypt. Mongoose використовується як драйвер доступу до БД MongoDB.

У клієнтській частині застосунку використано сучасні інструменти та бібліотеки, щоб забезпечити зручний, швидкий і функціональний інтерфейс. Базовою бібліотекою UI стала Material UI (MUI) — вона реалізує компоненти за стандартами Google Material Design, дозволяючи швидко зібрати адаптивний інтерфейс із кнопок, карток, панелей навігації та іншого. Завдяки MUI отримано можливість легко розширювати теми дизайну, забезпечувати доступність для користувачів та користуватися підтримкою активної спільноти.

Для перевірки форм введення застосовано бібліотеку Yup — вона забезпечує централізовану валідацію даних (наприклад, email, паролі, інші поля) в реальному часі та надає зрозумілі повідомлення про помилки, що значно покращує користувацький досвід.

Навігацію в межах SPA-додатку реалізовано засобами React-router-dom. Це дозволяє перемикатися між сторінками (список фільмів, детальна інформація, реєстрація, особистий кабінет тощо) без перезавантаження сторінки, що забезпечує швидкий та плавний UX.

Для зв'язку із сервером використовується бібліотека Axios — вона відповідає за HTTP-запити до REST API (отримання фільмів, автентифікацію, збереження обраного, історії переглядів тощо), забезпечуючи стабільність та простоту взаємодії з бекендом.

		Яценко. В.П			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Чижмотря О.В.				16
Змн.	Арк.	№ докум.	Підпис	Дата		

Загалом, цей стек забезпечує швидку розробку інтерфейсу завдяки готовим UI-елементам та централізованій валідації, прозору маршрутизацію та зручну роботу з HTTP-запитами. Клієнтська частина додатку вийшла не лише привабливою зовні, а й функціонально правильною, зручною для користувача та простою в підтримці.

Висновки до першого розділу

У першому розділі кваліфікаційної роботи проведено всебічний аналіз та підготовку технічної бази для розробки вебсервісу перегляду кінофільмів. Спочатку визначено завдання, ціль і мету проєкту, а також сформовано технічне завдання. Далі було виконано огляд і порівняння аналогів програмних продуктів, що дозволило визначити ключові функції для реалізації власного рішення. Проведено аналіз вимог до системи та розроблено діаграму випадків використання, яка надала чітке уявлення про взаємодію користувачів і систему. Завершальним етапом першого розділу стало обґрунтування вибору клієнт-серверної архітектури та підбору інструментальних засобів — Express.js, jsonwebtoken, Multer, Bcrypt, Mongoose на сервері та React, Material UI, React-router-dom, Yup, Axios на клієнті. Це створило міцний фундамент для подальшого проєктування та впровадження системи.

		Яценко. В.П			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Чижмотря О.В.				17
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 2. ПРОЄКТУВАННЯ СЕРВІСУ ПЕРЕГЛЯДУ КІНОФІЛЬМІВ

2.1 Опис акторів та алгоритмів роботи

Наступним кроком розробки системи сервісу перегляду кінофільмів є аналіз алгоритму роботи та створення діаграм активності.

Основними акторами є користувачі з наступними ролями: гість, зареєстрований користувач та адміністратор (табл. 2.1).

Таблиця 2.1

Користувачі

Актор	Короткий опис
Гість	Має можливість перегляду кінофільмів, реєструвати новий обліковий запис та авторизуватися в уже існуючий
Зареєстрований користувач	Має можливість перегляду кінофільмів, додавати фільми до списку переглянутих та до списку бажаних
Адміністратор	Має доступ до усіх можливостей решти акторів і може додавати відеоряд до фільмів

Актори будуть взаємодіяти з один з іншим за допомогою WEB - інтерфейсу. Розберемо основні алгоритми роботи за допомогою діаграм активності [19].

Процес перегляду фільму може розпочати будь-який актор. Для цього він натискає на іконку потрібного фільму в списку, клієнт робить запит на API, отримує дані та відображає їх на сторінці. Користувач може переглянути фільм натиснувши на плеєр. (рис 2.1).



Рис. 2.1 – Діаграма активності для перегляду фільму

Процес реєстрації може запустити гість. Він натискає відповідну клавішу та вводить дані для реєстрації у відповідні поля, після чого сервіс обробляє дані та записує дані користувача до бази даних. (рис 2.2).



Рис. 2.2 – Діаграма активності для реєстрації користувача

Процес додавання відеоряду може запустити лише адміністратор. Він переходить на сторінку фільму та натискає кнопку «Додати відео», після чого обирає потрібний відеоряд та завантажує його на сервер (рис. 2.3).



Рис. 2.3 – Діаграма активності для додавання відеоряду

2.2 Розробка бази даних

Наступним важливим кроком розробки сервісу перегляду кінофільмів є опис та створення бази даних. База даних складається з 4 таблиць, розберемо їх (табл. 2.2).

		Яценко. В.П			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Чижмотря О.В.				20
Змн.	Арк.	№ докум.	Підпис	Дата		

Опис таблиць

Назва таблиці	Короткий опис
User	Містить у собі користувачів та їх права доступу
Wishlist	Містить у собі список обраних до перегляду фільмів користувачів
Watched	Містить у собі список переглянутих користувачем фільмів
Video	Містить у собі посилання на відеоряди фільмів

Усі колекції бази даних описані за допомогою бібліотеки Mongoose, що є об'єктно-документним мапером (ODM) для MongoDB у середовищі Node.js. Це дозволяє описувати структуру бази даних у коді на мові JavaScript, що пришвидшує розробку та надає зручний синтаксис для подальшого редагування та змін. Завдяки паттерну MVC, усі моделі зберігаються окремо, описані у вигляді класів, які наслідують об'єкт з бібліотеки Mongoose.

Розберемо зв'язки таблиць за допомогою діаграми візуалізації бази даних (рис 2.4).

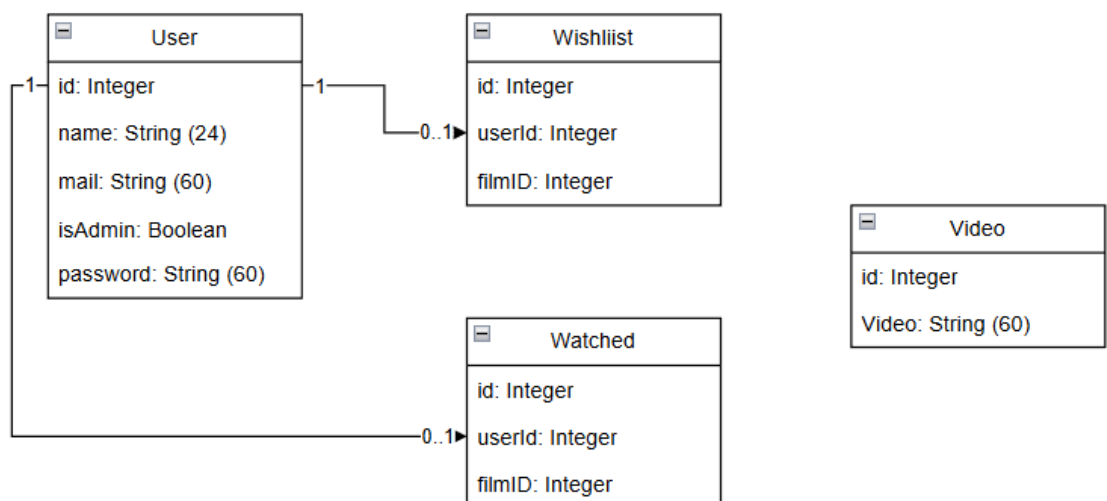


Рис. 2.4 – Візуалізація бази даних

Для кращого розуміння діаграми розберемо таблиці окремо.

Таблиця User це, по суті, найважливіша частина сервісу, оскільки дана таблиця зберігає користувачів, без яких немає сенсу у роботі проєкту. В даній таблиці збережені ідентифікатор користувача, що дозволяє отримувати швидкий доступ до його даних, або ж до даних в інших таблицях, які пов'язані з користувачем. Також тут зберігаються унікальні мейли користувачів, які використовуються при авторизації, оскільки нікнейми повторюватися можуть, а пошти – ні. Паролі користувачів зберігаються у хешованому вигляді для збереження їх секретності навіть у випадку витоку даних. Також в таблиці описані права користувача, чи є він адміністратором, чи це звичайний зареєстрований користувач без додаткових прав (табл. 2.3).

Таблиця 2.3

Користувачі

User		
Назва поля	Тип даних	PK/FK
ID	Integer	PK
name	String(24)	-
mail	String(60)	-
isAdmin	Boolean	-
password	String(24)	-

Таблиця зі списком бажаного є таблицею, доступ до якої мають лише зареєстровані користувачі сервісу. Вона дає змогу користувачу натискати в інтерфейсі відповідну кнопку, що записує ідентифікатор фільму до таблиці, разом з ідентифікатором користувача. Завдяки цьому користувач отримує можливість на спеціальній сторінці побачити усі фільми, які він додавав до списку. У разі ж, якщо користувач додасть фільм до переглянутих, фільм буде видалено з даної таблиці (табл. 2.4)

Таблиця 2.4

Список бажаних

Wishlist		
Назва поля	Тип даних	PK/FK
ID	Integer	PK
userId	Integer	FK
filmId	Integer	FK

Таблиця зі списком переглянутих фільмів (табл. 2.5) концептуально схожа зі списком доданих в бажане, оскільки має ті ж самі таблиці. В даній таблиці зберігаються усі фільми, які користувач переглянув і відмітив як переглянуті. Якщо користувач захоче переглянути фільм знову, то він може додати його до списку бажаних, що автоматично видалить фільм із переглянутих.

Таблиця 2.5

Список переглянутих

Watched		
Назва поля	Тип даних	PK/FK
ID	Integer	PK
userId	Integer	FK
filmId	Integer	FK

В таблиці Video збережені відеоряди, які адміністратори додали до фільмів. Це найпростіша, але одна з найважливіших таблиць (табл. 2.6).

Таблиця 2.6

Структура курсу

Video		
Назва поля	Тип даних	PK/FK
ID	Integer	PK
Video	String (60)	-

2.3 Комунікації і послідовність взаємодії об'єктів системи

Наступним кроком моделювання сервісу перегляду кінофільмів є створення діаграм послідовностей, що допоможе покращити розуміння про комунікації компонентів у системі.

Переглянувши діаграму створення облікового запису можна (рис.2.5) бачити, що створення ОЗ починається з користувача, який заповнює відповідну форму. Далі дані відправляються на сервер, де валідуються і у разі, якщо проблем не виникає, то вони відправляються до БД. БД, в свою чергу, зберігає ці дані та віддає про це сигнал серверу, який в свою чергу віддає дані клієнту.

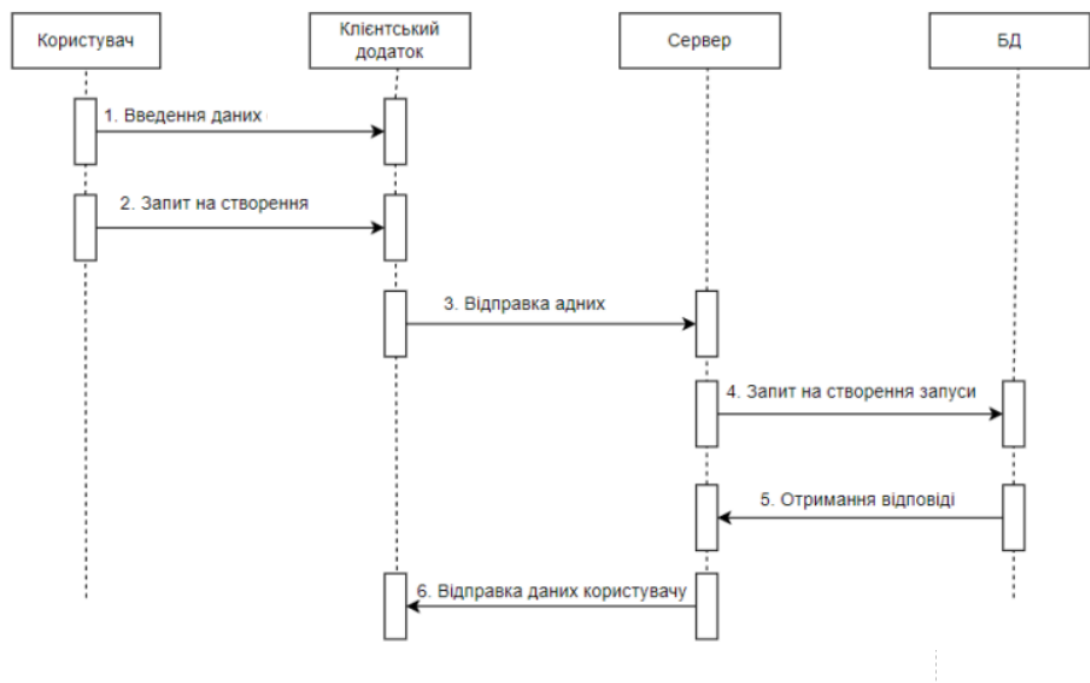


Рис. 2.5 – Діаграма послідовності створення облікового запису

Звертаючи увагу на діаграму (рис.2.6) можна бачити, що для відображення відфільтрованих по назві фільмів система передає дані, що отримує від користувача, до віддаленого API сервісу IMDb, отримує від нього список фільмів і відображає користувачу. Варто розуміти, що цей процес відбувається кожного разу, коли користувач вводить новий символ в поле пошуку.



Рис. 2.6 – Діаграма послідовності пошуку фільму

2.4 Реалізація сервісу перегляду кінофільмів

Далі розглянемо реалізацію деяких частин програмного коду проєкту.

```

import { createApi, fetchBaseQuery } from '@reduxjs/toolkit/query/react';

const apiKey = 'fa0cbbaf31b0830daf5e8683fdc0648d';

export const tmdbApi = createApi({
  reducerPath: 'tmdbApi',
  baseQuery: fetchBaseQuery({ baseUrl: 'https://api.themoviedb.org/3' }),
  endpoints: builder => ({
    getType: builder.query({
      query: ({ language, page, type }) =>
        `/${type}/popular?api_key=${apiKey}&language=${language}&page=${page}`,
    }),
    getTrendingAll: builder.query({
      query: ({ language, page }) =>
        `/trending/all/week?api_key=${apiKey}&language=${language}&page=${page}`,
    }),
    getMovieGenre: builder.query({
      query: ({ language, genre, page }) =>
        `/discover/movie?api_key=${apiKey}&language=${language}&page=${page}&with_genres=${genre}`,
    }),
    getPopularMovie: builder.query({
      query: ({ language, page }) =>
        `/movie/popular?api_key=${apiKey}&language=${language}&page=${page}`,
    }),
    getGenres: builder.query({
      query: () => `/genre/movie/list?language=uk&api_key=${apiKey}`,
    }),
    getFilm: builder.query({
      query: id => `/movie/${id}?api_key=${apiKey}&language=uk`,
    }),
    getActors: builder.query({
      query: id => `/movie/${id}/credits?api_key=${apiKey}&language=uk`,
    }),
    getMovieVideos: builder.query({
      query: id => `/movie/${id}/videos?api_key=${apiKey}`,
    }),
  })
});

```

```

    }},
  });

export const {
  useGetTypeQuery,
  useGetTrendingAllQuery,
  useGetMovieGenreQuery,
  useGetGenresQuery,
  useGetActorsQuery,
  useGetFilmQuery,
  useGetMovieVideosQuery,
} = tmdbApi;

```

Цей фрагмент коду реалізує підключення до TMDB API, використовуючи можливості Redux Toolkit Query у React. Спочатку імпортується createApi і fetchBaseQuery, після чого визначається apiKey, необхідний для авторизації запитів до TMDB.

Далі створюється tmdbApi з createApi, де reducerPath встановлює ім'я у Redux store, а baseQuery конфігурує основу для HTTP-запитів до базового URL <https://api.themoviedb.org/3>. У блоці endpoints задаються методи (endpoints), які автоматизують формування запитів. Наприклад, getType будує URL для отримання популярного контенту певного типу (фільму або серіалу) із вказаними параметрами мовою, сторінкою й типом. Інші методи дозволяють запитувати список трендів, жанри фільмів, конкретні фільми, акторський склад та трейлери. Кожен endpoint використовує визначену функцію query, яка повертає рядок шляху з необхідними параметрами api_key, мовою й іншими опціями.

Після створення API експортовано хук-функції (useGetTypeQuery, useGetTrendingAllQuery, useGetMovieGenreQuery, useGetGenresQuery, useGetActorsQuery, useGetFilmQuery, useGetMovieVideosQuery), які надають готові функції реагування на запити з компонентів React. Використання цих хук-функцій дозволяє компонентам отримувати дані з TMDB, обробляти статуси завантаження та кешувати результати, мінімізуючи писання шаблонного коду для взаємодії із сервером.

```

import React from 'react';
import { useSelector } from 'react-redux';
import {
  useGetGenresQuery,
  useGetMovieGenreQuery,
  useGetTrendingAllQuery,

```

		Яценко. В.П			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Чижмотря О.В.				26
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    useGetTypeQuery,
  } from '../services/tmdbApi';
export default function useMoviesQuery() {
  const responsePopularFilms = useGetTypeQuery({
    language: 'uk',
    page: 1,
    type: 'movie',
  });
  const responsePopularTV = useGetTypeQuery({
    language: 'uk',
    page: 1,
    type: 'tv',
  });
  const responsePopularCartoons = useGetMovieGenreQuery({
    language: 'uk',
    genre: 16,
    page: 1,
  });
  const responsePopular = useGetTrendingAllQuery({
    language: 'uk',
    page: 1,
  });
  const responseGenre = useGetGenresQuery();
  const isLoading =
    responsePopularCartoons.isFetching ||
    responsePopular.isFetching ||
    responsePopularFilms.isFetching ||
    responsePopularTV.isFetching;
  const hasError =
    responsePopularCartoons.error ||
    responsePopular.error ||
    responsePopularFilms.error ||
    responsePopularTV.error;
  return {
    isLoading,
    hasError,
    responsePopularCartoons,
    responsePopular,
    responsePopularFilms,
    responsePopularTV,
    responseGenre,
  };
}

```

Цей спеціальний React-хук `useMoviesQuery` централізує одночасне виконання кількох RTK Query запитів до TMDb-про API: популярних фільмів (`useGetTypeQuery`), популярних серіалів, мультфільмів за жанром, трендових сюжетів та списку жанрів. Він збирає їх в один логічний блок і обчислює `isLoading` (якщо будь-який запит обробляється) і `hasError` (якщо є помилка хоча б у одному запиті). Повернуті дані розділені за контентом — мультфільми, тренди, популярні фільми і серіали, жанри — що дозволяє компонентам отримувати все необхідне в єдиному виклику хука.

		Яценко. В.П			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Чижмотря О.В.				27
Змн.	Арк.	№ докум.	Підпис	Дата		

Такий підхід відповідає рекомендаціям RTK Query щодо `render-optimized hooks`, які автоматично стартують запити, кешують результати і відслідковують стани завантаження (`isFetching`) чи помилок. Він також демонструє практику виклику кількох query-хуків в одному компоненті, що обговорюється в ком'юніті як оптимальний спосіб синхронного отримання різного контенту .

У результаті `useMoviesQuery` забезпечує стандартизовану обробку множинних запитів, їх узгодження, об'єднані стани завантаження та помилок, а також чистий і зручний інтерфейс для компонентів React.

```
import { createSlice } from '@reduxjs/toolkit';
const initialState = {
  title: "",
};
export const currentQuery = createSlice({
  name: 'currentQuery',
  initialState,
  reducers: {},
});
export default currentQuery.reducer;
```

Наведений код — це slice стану в Redux Toolkit, створений за допомогою `createSlice`. Він визначає початковий (`initialState`), який містить поле `title` як порожній рядок, та не містить жодних редьюсерів у полі `reducers`. Це означає, що в цьому slice наразі немає жодної логіки для зміни стану, лише порожня структура.

Хоча `createSlice` автоматично створює екшн-крейтори й типи дій на основі визначених редьюсерів, у цьому випадку через відсутність редьюсерів такі екшни не згенеровано. Для подальших змін стану цього slice можна додати редьюсери або використовувати зворотні дії (`extraReducers`).

Отже, slice називається `'currentQuery'`, його формат зручний для розширення в майбутньому. Якщо не передбачається додавати власних редьюсерів, можна замінити `createSlice` на `createReducer`, але `createSlice` залишає можливість легко додавати логіку в майбутньому.

Висновки до другого розділу

У цьому розділі було здійснено повний цикл проектування сервісу. Спочатку проведено аналіз усіх користувачів, які взаємодіятимуть із

		Яценко. В.П			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Чижмотря О.В.				28
Змн.	Арк.	№ докум.	Підпис	Дата		

системою, та побудовано діаграми використання, що відображають основні сценарії роботи з сервісом. Після цього було спроектовано структуру бази даних, визначено необхідні таблиці та поля. Далі розглянуто логіку взаємодії між компонентами системи за допомогою діаграм послідовностей. Завершальним етапом стала безпосередня реалізація сервісу у вигляді програмного коду. Отже, у межах цього розділу реалізовано усі ключові етапи проєктування та розробки, що забезпечили повну функціональність вебзастосунку.

		Яценко. В.П			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Чижмотря О.В.				29
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З СЕРВІСОМ ПЕРЕГЛЯДУ КІНОФІЛЬМІВ

3.1 Порядок встановлення та налаштування системи

Для розуміння як відбувається взаємодія компонентів (рис. 3.1) було розроблену діаграму компонентів, що описує взаємодію між різними компонентами.

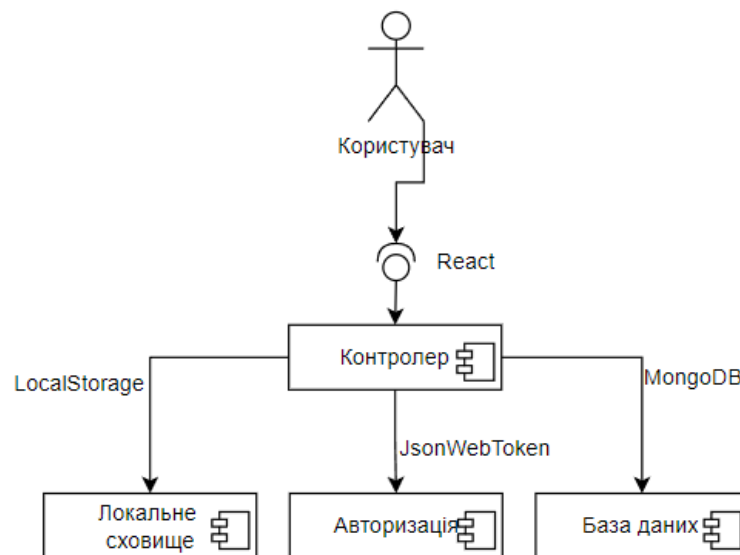


Рис. 3.1 – Діаграма компонентів

Сервіс отримує всі дані безпосередньо від користувачів: запити до сервера надходять через React-клієнт, де користувач ініціює дію. Ці запити обробляються контролером, який визначає, чи потрібна взаємодія з базою даних — у такому випадку звертається до відповідного модуля. Якщо йдеться про авторизацію, контролер взаємодіє з компонентом, що працює з JSONWebToken. У випадку необхідності збереження файлів (наприклад, зображень), запити направляються до компоненту локального сховища, який виконує обробку та збереження файлів. Таким чином, забезпечується чітка і логічна взаємодія між усіма компонентами сервісу — клієнтом, контролерами, базою даних, авторизацією та файловим сховищем.

Також було розроблено діаграму розгортання, яка надає інформацію про фізичну будову системи (рис. 3.2).

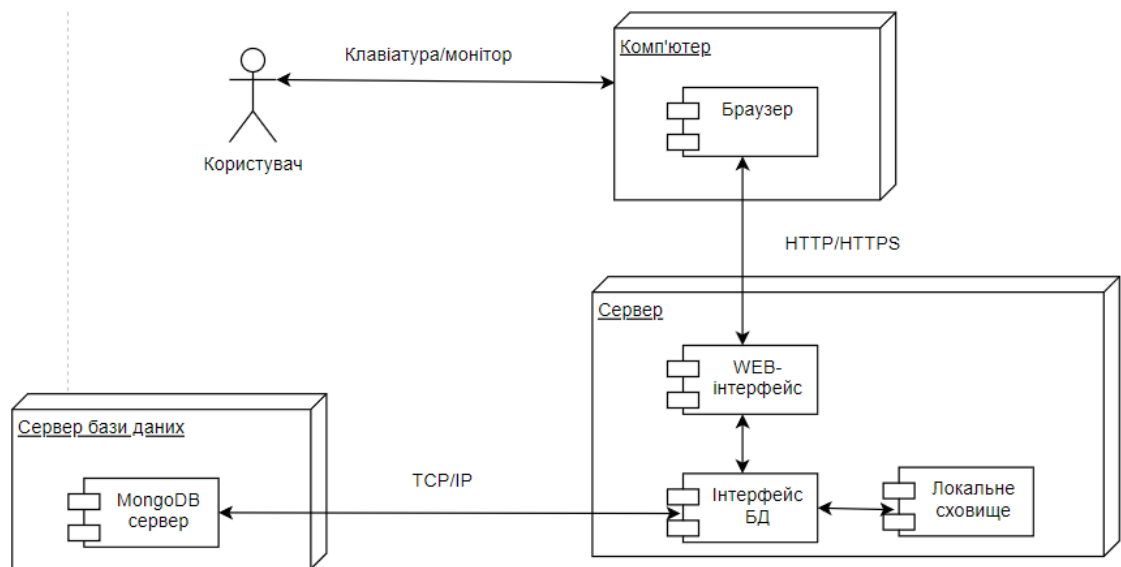


Рис. 3.2 – Діаграма розгортання

Аналіз діаграми розгортання показує, що для запуску сервісу потрібен хостинг із можливістю зберігання файлів безпосередньо на сервері. При цьому база даних розміщується віддалено, тому її не треба запускати в комплекті із сервером. Проте необхідно забезпечити, щоб хостинг підтримував можливість серверу здійснювати запити до зовнішніх ресурсів — як для взаємодії з удаленою базою, так і для обміну файлами. Таким чином, важливо обрати хостинг, який надає здатність виконувати HTTP-запити до сторонніх сервісів і забезпечує дисковий простір або сумісні рішення для зберігання даних.

3.2 Опис інтерфейсу програмного продукту

При переході на головну сторінку користувач може побачити декілька списків фільмів, які він має можливість відкрити до перегляду прямо з головного екрану (рис. 3.3)

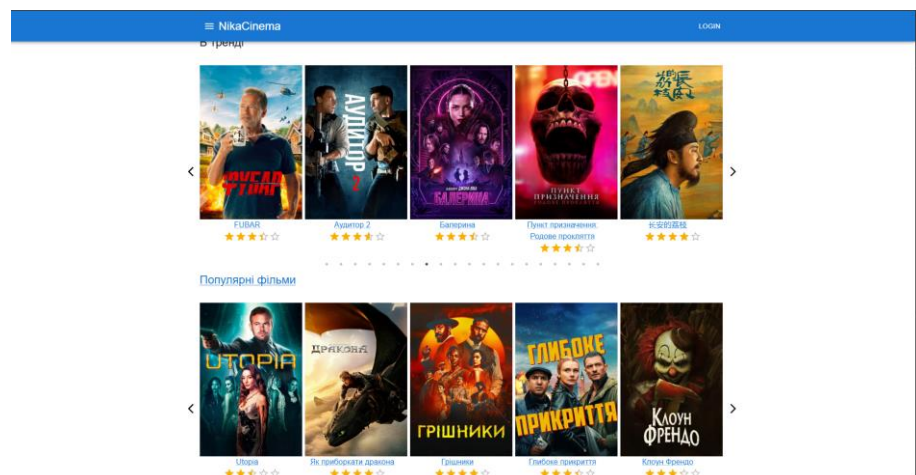


Рис. 3.3 – Головна сторінка

		Яценко. В.П			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Чижмотря О.В.				31
Змн.	Арк.	№ докум.	Підпис	Дата		

Якщо ж користувач має бажання відфільтрувати фільми по назві, тобто він знає який саме фільм його цікавить, то він може ввести назву в відповідне поле на головній сторінці (рис.3.4)

Пошук фільмів

Рис. 3.4 – Поле пошуку

Ввівши назву користувач може побачити, що відображення контенту на екрані змінилося відображається лише той контент, що відповідає запиту (рис.3.5). Тут контент відображається уже без слайдерів, а звичайною плиткою. Варто звернути увагу, що оновлення сторінки відбувається динамічно разом з введенням даних в поле.

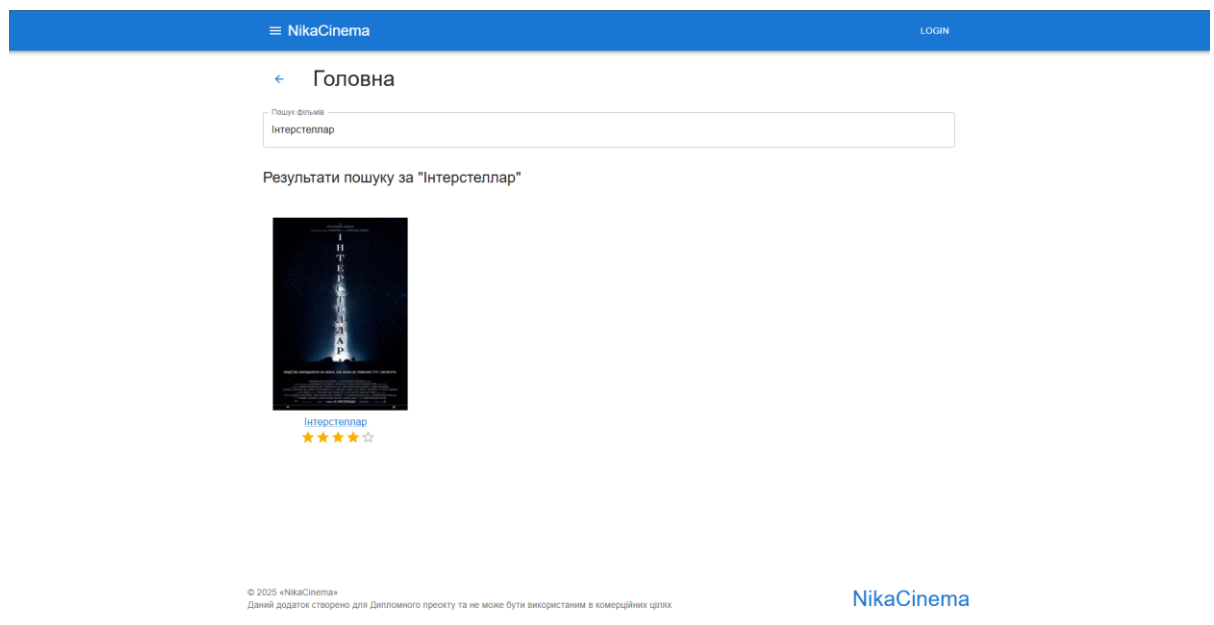


Рис. 3.5 – Результати пошуку

Відкривши бургер-меню (рис.3.6), що знаходиться зліва, користувач має можливість побачити найпопулярніші жанри і відкрити сторінки, які наповнені контентом відповідного жанру.

Наприклад відкривши розділ сімейних фільмів користувач може бачити усі фільми, що відповідають цьому критерію (рис. 3.7).

		Яценко. В.П			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Чижмотря О.В.				32
Змн.	Арк.	№ докум.	Підпис	Дата		

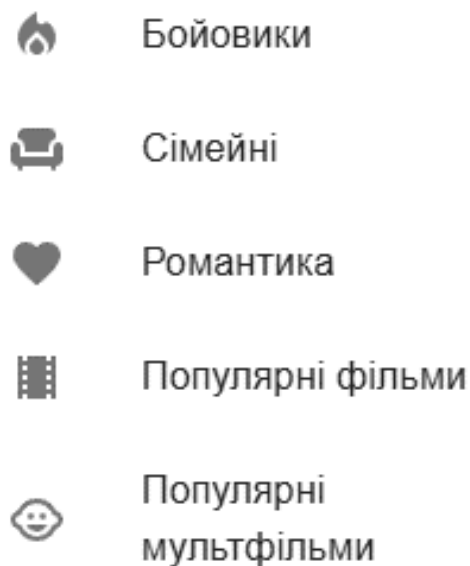


Рис. 3.6 – Рекомендовані жанри

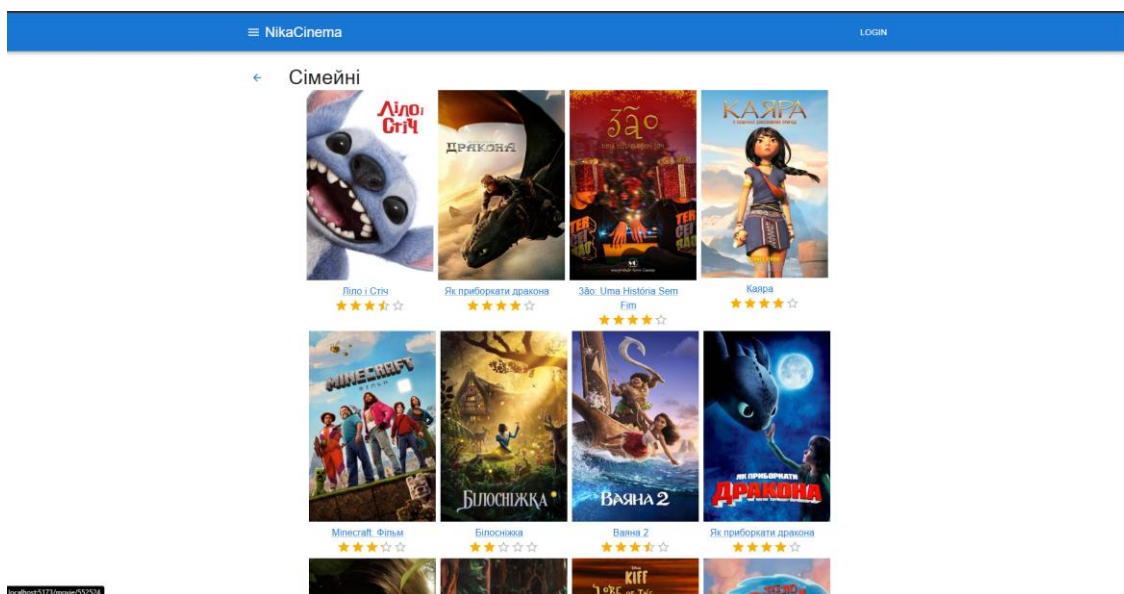


Рис. 3.7 – Сторінка жанру

Внизу сторінки користувач може бачити пагінацію, завдяки якій на сторінці одночасно відображається задана кількість фільмів і не відбувається перегрузки (рис. 3.8).



Рис. 3.8 – Пагінація

		Яценко. В.П			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Чижмотря О.В.				33
Змн.	Арк.	№ докум.	Підпис	Дата		

Перейшовши на сторінку фільму користувач може побачити постер, а також деяку кількість інформації, що відповідає фільму (рис.3.9)

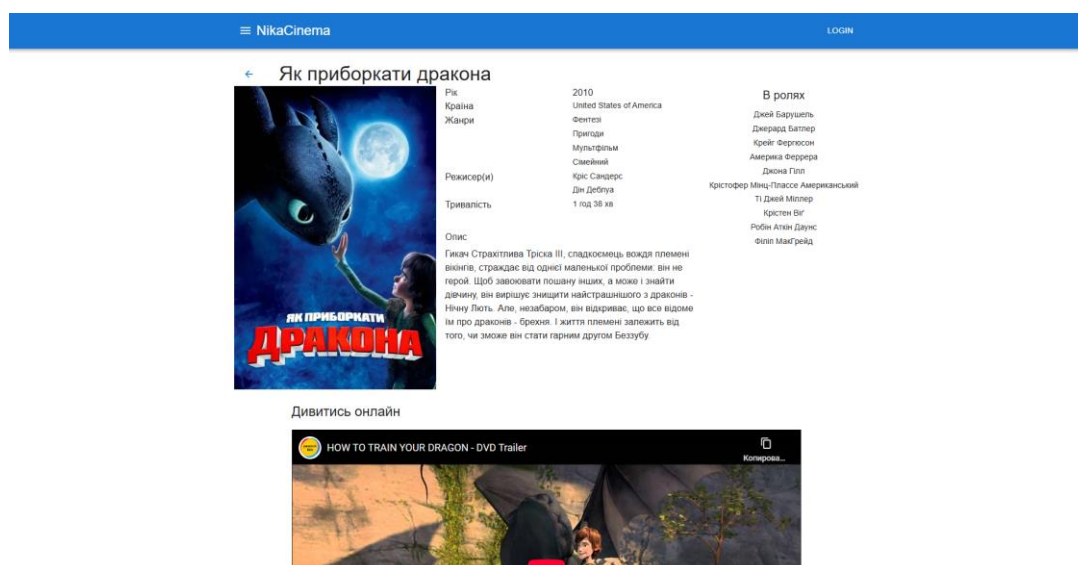


Рис. 3.9 – Сторінка фільму

В наступній частині екрану користувач може побачити трейлер фільму (рис.3.10), оскільки сайт не є ліцензійним і не має права пропонувати користувачам відеоряд, адже це буде вважатися піратським контентом. На майбутнє реалізовано функціонал додавання відеоряду для фільмів, на які буде отримано ліцензійні права.

Дивитись онлайн

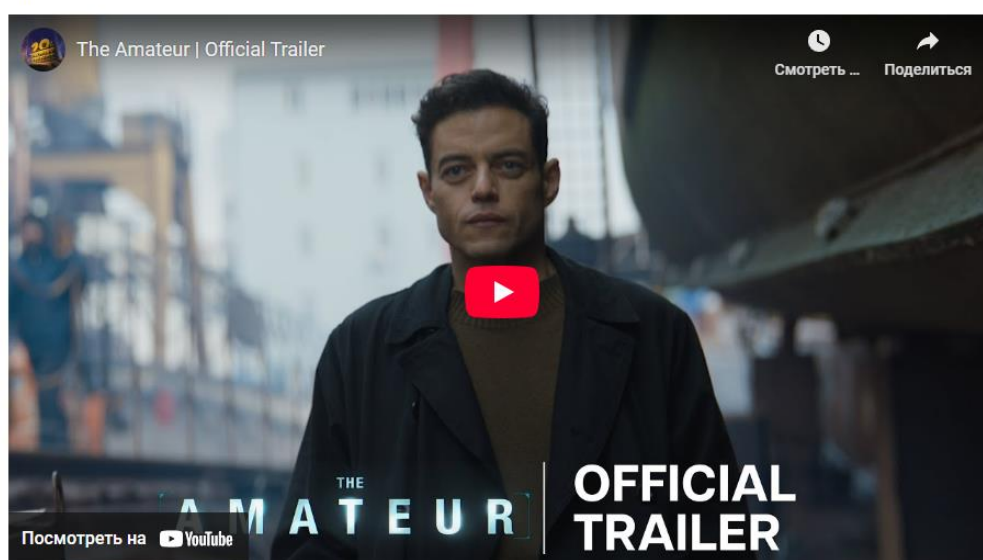


Рис. 3.10 – Трейлер фільму

Якщо у фільма є приквели та сіквели, то під відеорядом можна побачити список усіх фільмів, що відносяться до заданої колекції (рис.3.11)

		Яценко. В.П			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Чижмоторя О.В.				34
Змн.	Арк.	№ докум.	Підпис	Дата		

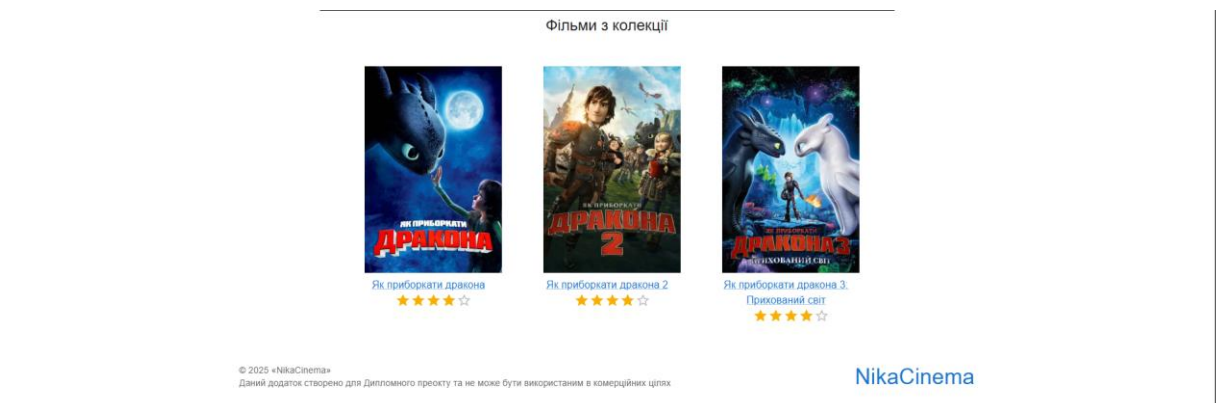


Рис. 3.11 – Колекції

Щоб отримати додатковий доступ користувач має зареєструватися. Для цього використовується відповідна кнопка (рис.3.12)



Рис. 3.12 – Кнопка авторизації

Натиснувши користувач бачить модальне вікно, де може обрати авторизацію (рис.3.13) або авторизацію (рис.3.14).

Вхід у акаунт

ВХІД

РЕЄСТРАЦІЯ

Email

Пароль

СКАСУВАТИ

УВІЙТИ

Рис. 3.13 – Вікно авторизації

Реєстрація

ВХІД	РЕЄСТРАЦІЯ
Ім'я	
Email	
Пароль	
СКАСУВАТИ	ЗАРЕЄСТРУВАТИСЯ

Рис. 3.14 – Вікно реєстрації

Увійшовши користувач може побачити привітання та кнопку виходу із облікового запису (рис.3.15).

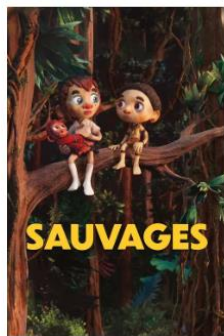


Рис. 3.15 – Відображення користувача

Натиснувши на нікнейм можна відкрити модальне вікно, де будуть відображені списки бажаного (рис. 3.16) та переглянутого (рис.3.17)

Мій профіль

ВІШЛІСТ ПЕРЕГЛЯНУТІ



Sauvages

Рис. 3.16 – Список бажаного

		Яценко. В.П			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Чижмотря О.В.				36
Змн.	Арк.	№ докум.	Підпис	Дата		



Крапля



Аматор



Еліо

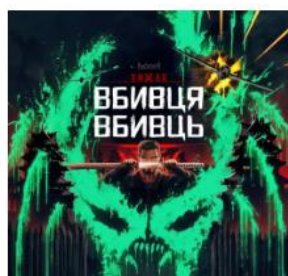


Рис. 3.17 – Список переглянутого

3.3 Тестування сервісу перегляду кінофільмів

Тестування є невід'ємною складовою розробки будь-якої програмної системи, особливо коли йдеться про вебсервіси, які повинні забезпечувати надійний та безперебійний доступ ресурсів. Враження від перегляду фільмів значною мірою залежить від якості роботи сервісу, на якому користувач вирішує ознайомитися з фільмом. Тому тестування сервісу перегляду кінофільмів має ключове значення для досягнення поставлених цілей.

Для перевірки роботи сервісу перегляду кінофільмів були розроблені тестові випадки, які описують предмет тестування, кроки, вхідні дані та очікуваний результат, ось деякі з них (табл. 3.1).

Очікується, що завдяки проведенню тестування сервісу буде відсіяно значну частину помилок, що можуть призвести до отримання користувачем негативного досвіду.

		Яценко. В.П			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Чижмотря О.В.				37
Змн.	Арк.	№ докум.	Підпис	Дата		

Тестові випадки

Опис тестового випадку	Кроки	Вхідні дані	Очікуваний результат	Фактичний результат	Пройшло/ Не пройшло
Успішна авторизація з правильними даними	1. Відкрити сторінку авторизації. 2. Ввести логін та пароль. 3. Натиснути кнопку «Увійти»	Логін: Nika Пароль: 1111	Успішна авторизація користувача	Успішна авторизація користувача	Пройшло
Авторизація з неправильними даними	1. Відкрити сторінку авторизації. 2. Ввести логін та пароль. 3. Натиснути кнопку «Увійти»	Логін: Nika Пароль: 1234	Повідомлення про помилку	Повідомлення про помилку	Пройшло
Авторизація з пустими полями	1. Відкрити сторінку авторизації. 2. Натиснути кнопку «Увійти»	Логін: Пароль:	Повідомлення про помилку	Повідомлення про помилку	Пройшло
Перегляд фільму	1. Відкрити сторінку з фільмами 2. Перехід на сторінку з фільмом 3. Натиснути кнопку «Play»	-	Успішний запуск фільму	Успішний запуск фільму	Пройшло

Проведене тестування дозволило виявити та усунути недоліки, що значно знижує ризики під час експлуатації системи. Це, в свою чергу, сприяє підвищенню задоволеності користувачів та забезпеченню стабільної роботи.

Висновки до третього розділу

У цьому розділі було спроектовано та деталізовано фізичну архітектуру вебсервісу для перегляду фільмів за допомогою діаграм компонентів та розгортання. Далі виконано підготовку та конфігурацію середовища для розгортання сервісу на сервері. Було описано та протестовано основні функціональні модулі системи, а також розглянуто користувацький інтерфейс, включно з його структурою та алгоритмом взаємодії користувача із сервісом.

		Яценко. В.П			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Чижмотря О.В.				38
Змн.	Арк.	№ докум.	Підпис	Дата		

ВИСНОВКИ

У першому розділі було досліджено технічне завдання для вебсервісу перегляду фільмів та сформульовано цілі і задачі проєкту. Проведено аналіз існуючих аналогів, що дозволило виділити ключові функціональні рішення. Створено діаграму варіантів використання для опису взаємодії користувачів із системою і уточнено вимоги до продукту.

У другому розділі наведено діаграми активності та послідовності, які демонструють поведінку системи під різні сценарії роботи та його акторів. Розроблено структуру бази даних та описано її реалізацію. Також описані процеси розробки та реалізації системи, включаючи бекенд і фронтенд компоненти.

У третьому розділі спроектовано систему з точки зору компонентів і розгортання, що показує, як сервіси взаємодіють між собою. Описано налаштування середовища для розгортання на сервері, проведено тестування основних функцій із прикладами тестових кейсів. Представлено інтерфейс користувача, пояснено механізми роботи з ним.

Таким чином, у процесі виконання роботи було проведено повний цикл: аналіз, проєктування, розробка та перевірка вебсервісу для перегляду фільмів, повністю відповідного функціональним вимогам. В результаті я опанував нові підходи до моделювання та побудови програмних систем, що суттєво покращило мої навички як розробника.

		Яценко. В.П			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Чижмотря О.В.				39
Змн.	Арк.	№ докум.	Підпис	Дата		

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. MacDonald M. HTML5: The Missing Manual, 2nd Edition. The Book That Should Have Been in the Box / Matthew MacDonald. – 2014. – 518 с.
2. Gascón U. Node.js for Beginners: A comprehensive guide to building efficient, full-featured web applications with Node.js / Ulises Gascón. – 2024. – 382 с.
3. Shelley S. Learning Node: Moving to the Server-Side, 2nd Edition / Shelley Shelley. – 2016. – 288 с.
4. React-router-dom tutorial [Електронний ресурс] – Режим доступу до ресурсу: <https://reactrouter.com/en/main/start/tutorial>.
5. Types of UML Diagrams [Електронний ресурс] – Режим доступу до ресурсу: <https://www.lucidchart.com/blog/types-of-UML-diagrams>.
6. NPM bcrypt [Електронний ресурс] – Режим доступу до ресурсу: <https://www.npmjs.com/package/bcrypt>.
7. Code Quickly. Learn JavaScript Quickly: A Complete Beginner's Guide to Learning JavaScript, Even If You're New to Programming (Crash Course With Hands-On Project) / Code Quickly. – 2020. – 174 с.
8. Express documentation [Електронний ресурс] – Режим доступу до ресурсу: <https://expressjs.com/>.
9. React.js documentation [Електронний ресурс] – Режим доступу до ресурсу: <https://uk.legacy.reactjs.org/>.
10. Banks A., Porcello E. Learning React: Modern Patterns for Developing React Apps / A. Banks, E. Porcello. – 2020. – 307 с.
11. NPM mui [Електронний ресурс] – Режим доступу до ресурсу: <https://www.npmjs.com/package/@mui/material>
12. Flanagan D. JavaScript: The Definitive Guide: Master the World's Most-Used Programming Language / David Flanagan. – 2020. – 704 с.
13. Node.js documentation [Електронний ресурс] – Режим доступу до ресурсу: <https://nodejs.org/docs/latest/api/>.
14. Діаграма розгортання [Електронний ресурс] – Режим доступу до ресурсу: <https://studfile.net/preview/5010027/page:6/>.

		Яценко. В.П			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Чижмоторя О.В.				40
Змн.	Арк.	№ докум.	Підпис	Дата		

15. Lim G. Beginning Node.js, Express & MongoDB Development / Greg Lim. – 2019. – 154 с.
16. Robin W. The Road to React: Your journey to master plain yet pragmatic React.js / Wieruch Robin. – 2018. – 286 с.
17. NPM nodemon [Електронний ресурс] – Режим доступу до ресурсу: <https://www.npmjs.com/package/nodemon>.
18. JavaScript Tutorial [Електронний ресурс] – Режим доступу до ресурсу: <https://www.w3schools.com/js/>.
19. NPM jsonwebtoken [Електронний ресурс] – Режим доступу до ресурсу: <https://www.npmjs.com/package/jsonwebtoken>.
20. Як будувати UML-діаграми [Електронний ресурс] – Режим доступу до ресурсу: <https://dou.ua/forums/topic/40575/>.
21. Subramanian V. Pro MERN Stack: Full Stack Web App Development with Mongo, Express, React, and Node / Vasan Subramanian. – 2019. – 536 с.
22. NPM yup [Електронний ресурс] – Режим доступу до ресурсу: <https://www.npmjs.com/package/yup>.
23. NPM Axios [Електронний ресурс] – Режим доступу до ресурсу: <https://www.npmjs.com/package/axios>.
24. MERN Stack Explained [Електронний ресурс] – Режим доступу до ресурсу: <https://www.mongodb.com/resources/languages/mern-stack>.
25. Діаграма послідовності [Електронний ресурс] – Режим доступу до ресурсу: <https://www.maxzosim.com/sequence-diagrams/>.
26. Документація MIUI [Електронний ресурс] – Режим доступу до ресурсу: <https://mui.com/material-ui/getting-started/>
27. TMDb документація [Електронний ресурс] – Режим доступу до ресурсу: <https://developer.themoviedb.org/docs/getting-started>
28. How to Build a Video Streaming App: A Complete 2025 Guide [Електронний ресурс] – Режим доступу до ресурсу: <https://www.scnsoft.com/blog/how-to-build-a-video-streaming-app>

		Яценко. В.П			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Чижмоторя О.В.				41
Змн.	Арк.	№ докум.	Підпис	Дата		

29. Architecture for a Video Streaming Service [Електронний ресурс] – Режим доступу до ресурсу: <https://hygraph.com/blog/video-streaming-architecture>
30. The Ultimate Guide to Video Streaming API: Build, Integrate & Scale in 2025 [Електронний ресурс] – Режим доступу до ресурсу: <https://www.integrate.io/blog/ultimate-guide-to-api-integration-solutions/>
31. NPM multer [Електронний ресурс] – Режим доступу до ресурсу: <https://www.npmjs.com/package/multer>

		Яценко. В.П			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Чижмотря О.В.				42
Змн.	Арк.	№ докум.	Підпис	Дата		

ДОДАТКИ

		Яценко. В.П			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Чижмотря О.В.				43
Змн.	Арк.	№ докум.	Підпис	Дата		

Лістинг файлу authController.js

```

import bcrypt from 'bcrypt';
import jwt from 'jsonwebtoken';

import User from '../models/User.js';

export const register = async (req, res) => {
  const { email, password, name } = req.body;
  try {
    const exist = await User.findOne({ email });
    if (exist)
      return res.status(400).json({ message: 'Email вже використовується' });

    const hashedPassword = await bcrypt.hash(password, 10);
    const user = await User.create({ email, password: hashedPassword, name });

    res.status(201).json({ message: 'Реєстрація успішна' });
  } catch (err) {
    res.status(500).json({ message: 'Помилка сервера' });
  }
};

export const login = async (req, res) => {
  const { email, password } = req.body;
  try {
    const user = await User.findOne({ email });
    if (!user)
      return res.status(400).json({ message: 'Невірний email або пароль' });

    const isMatch = await bcrypt.compare(password, user.password);
    if (!isMatch)
      return res.status(400).json({ message: 'Невірний email або пароль' });

    const token = jwt.sign({ id: user._id }, process.env.JWT_SECRET, {
      expiresIn: '7d',
    });
    res.json({ token, user });
  } catch (err) {
    res.status(500).json({ message: 'Помилка входу' });
  }
};

```

Лістинг файлу watchedController.js

```

import Watched from '../models/Watched.js';
import Wishlist from '../models/Wishlist.js';

export const addToWatched = async (req, res) => {
  const { id, name, poster_path } = req.body;
  const userId = req.user.id;

  if (!id || !name || !poster_path) {
    return res.status(400).json({ message: "Всі поля обов'язкові" });
  }

```

		Яценко. В.П			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Чижмотря О.В.				44
Змн.	Арк.	№ докум.	Підпис	Дата		

```

try {
  await Wishlist.deleteOne({ userId, id });
  const existing = await Watched.findOne({ userId, id });
  if (existing) {
    return res
      .status(400)
      .json({ message: 'Фільм вже доданий до переглянутих' });
  }

  const newItem = new Watched({ userId, id, name, poster_path });
  await newItem.save();

  res
    .status(201)
    .json({ message: 'Фільм додано до переглянутих', item: newItem });
} catch (error) {
  res.status(500).json({ message: 'Помилка сервера' });
}
};

```

```

export const getWatched = async (req, res) => {
  const userId = req.user.id;

  try {
    const list = await Watched.find({ userId });
    res.json(list);
  } catch (error) {
    res.status(500).json({ message: 'Помилка сервера' });
  }
};

```

Лістинг файлу wishlistController.js

```

import Watched from '../models/Watched.js';
import Wishlist from '../models/Wishlist.js';

```

```

export const addToWishlist = async (req, res) => {
  const { id, name, poster_path } = req.body;
  const userId = req.user.id;

  if (!id || !name || !poster_path) {
    return res.status(400).json({ message: "Всі поля обов'язкові" });
  }

  try {
    await Watched.deleteOne({ userId, id });

    const existing = await Wishlist.findOne({ userId, id });
    if (existing) {
      return res.status(400).json({ message: 'Фільм вже є у вішлісті' });
    }

    const newItem = new Wishlist({ userId, id, name, poster_path });
    await newItem.save();

    res
      .status(201)
      .json({ message: 'Фільм додано до вішлісту', item: newItem });
  } catch (error) {
    res
      .status(500)
      .json({ message: 'Помилка сервера' });
  }
};

```

		Яценко. В.П			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Чижмотря О.В.				45
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    res.status(500).json({ message: 'Помилка сервера' });
  }
};

export const getWishlist = async (req, res) => {
  const userId = req.user.id;

  try {
    const list = await Wishlist.find({ userId });

    res.json(list);
  } catch (error) {
    res.status(500).json({ message: 'Помилка сервера' });
  }
};

```

Лістинг файлу authMiddleware.js

```

import jwt from 'jsonwebtoken';

const authMiddleware = (req, res, next) => {
  const authHeader = req.headers.authorization;

  if (!authHeader || !authHeader.startsWith('Bearer ')) {
    return res
      .status(401)
      .json({ message: 'Немає токена. Доступ заборонено.' });
  }

  const token = authHeader.split(' ')[1];

  try {
    const decoded = jwt.verify(token, process.env.JWT_SECRET);
    req.user = decoded;
    next();
  } catch (err) {
    return res
      .status(401)
      .json({ message: 'Недійсний токен. Доступ заборонено.' });
  }
};

export default authMiddleware;

```

Лістинг файлу User.js

```

import mongoose from 'mongoose';

const userSchema = new mongoose.Schema({
  name: { type: String, required: true },
  email: { type: String, required: true, unique: true },
  password: { type: String, required: true },
  isAdmin: { type: Boolean, default: false },
});

export default mongoose.model('User', userSchema);

```

Лістинг файлу AuthModal.jsx

```

import {
  Alert,
  Button,

```

		Яценко. В.П			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Чижмотря О.В.				46
Змн.	Арк.	№ докум.	Підпис	Дата		

```

Dialog,
DialogActions,
DialogContent,
DialogTitle,
Stack,
TextField,
ToggleButton,
ToggleButtonGroup,
} from '@mui/material';
import { useFormik } from 'formik';
import PropTypes from 'prop-types';
import React, { useState } from 'react';
import * as Yup from 'yup';

import api from '../services/api';

export default function AuthModal({ open, onClose, onLogin }) {
  const [mode, setMode] = useState('login');
  const [serverError, setServerError] = useState("");

  const handleChangeMode = (_, newMode) => {
    if (newMode) {
      setMode(newMode);
      setServerError("");
      formik.resetForm();
    }
  };

  const formik = useFormik({
    initialValues: {
      email: "",
      password: "",
      name: "",
    },
    validationSchema: Yup.object({
      email: Yup.string().email('Невірний email').required('Обов'язково'),
      password: Yup.string()
        .min(6, 'Мінімум 6 символів')
        .required('Обов'язково'),
      name:
        mode === 'register'
          ? Yup.string().min(2, 'Мінімум 2 символи').required('Обов'язково')
          : Yup.string(),
    }),
    onSubmit: async (values, { setSubmitting }) => {
      setServerError("");
      try {
        const url = mode === 'login' ? '/auth/login' : '/auth/register';
        const payload =
          mode === 'login'
            ? { email: values.email, password: values.password }
            : values;

        const response = await api.post(url, payload);

        if (mode === 'login') {
          localStorage.setItem('token', response.data.token);
          localStorage.setItem('username', response.data.user.name);
        }
      } catch (error) {
        setServerError(error.message);
      }
    },
  });
}

```

		Яценко. В.П			ІІЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Чижмотря О.В.				47
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    if (onLogin) {
      onLogin({
        token: response.data.token,
        name: response.data.user.name,
      });
    }
  }

  onClose();
} catch (err) {
  const message =
    err.response?.data?.message || 'Помилка з'єднання з сервером';
  setServerError(message);
} finally {
  setSubmitting(false);
}
},
});

return (
  <Dialog open={open} onClose={onClose}>
    <DialogTitle>
      {mode === 'login' ? 'Вхід у акаунт' : 'Реєстрація'}
    </DialogTitle>

    <DialogContent>
      <Stack spacing={2} sx={{ mt: 1 }}>
        <ToggleButtonGroup
          value={mode}
          exclusive
          onChange={handleChangeMode}
          fullWidth
        >
          <ToggleButton value="login">Вхід</ToggleButton>
          <ToggleButton value="register">Реєстрація</ToggleButton>
        </ToggleButtonGroup>

        {serverError && <Alert severity="error">{serverError}</Alert>}

        {mode === 'register' && (
          <TextField
            label="Ім'я"
            name="name"
            value={formik.values.name}
            onChange={formik.handleChange}
            onBlur={formik.handleBlur}
            error={formik.touched.name && !!formik.errors.name}
            helperText={formik.touched.name && formik.errors.name}
            fullWidth
          />
        )}

        <TextField
          label="Email"
          name="email"
          value={formik.values.email}
          onChange={formik.handleChange}

```

		Яценко. В.П			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Чижмоторя О.В.				48
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        onBlur={formik.handleBlur}
        error={formik.touched.email && !!formik.errors.email}
        helperText={formik.touched.email && formik.errors.email}
        fullWidth
      />

      <TextField
        label="Пароль"
        type="password"
        name="password"
        value={formik.values.password}
        onChange={formik.handleChange}
        onBlur={formik.handleBlur}
        error={formik.touched.password && !!formik.errors.password}
        helperText={formik.touched.password && formik.errors.password}
        fullWidth
      />
    </Stack>
  </DialogContent>

  <DialogActions>
    <Button onClick={onClose}>Скасувати</Button>
    <Button
      onClick={formik.handleSubmit}
      disabled={formik.isSubmitting}
      variant="contained"
    >
      {mode === 'login' ? 'Увійти' : 'Зареєструватися'}
    </Button>
  </DialogActions>
</Dialog>
);
}

AuthModal.propTypes = {
  open: PropTypes.bool.isRequired,
  onClose: PropTypes.func.isRequired,
  onLogin: PropTypes.func,
};

```

Лістинг файлу MovieCard.jsx

```

import {
  Box,
  Button,
  Link,
  Rating,
  Stack,
  Tooltip,
  Typography,
} from '@mui/material';
import React from 'react';
import { Link as RouterLink } from 'react-router-dom';

import styles from './MovieCard.module.css';

export default function MovieCard({ movie }) {
  return (
    <Stack>

```

		Яценко. В.П			ІІЗ.КР.Б – 121 – 25 – ІІЗ	Арк.
		Чижмотря О.В.				49
Змн.	Арк.	№ докум.	Підпис	Дата		

```

<RouterLink to={` /movie/${movie.id}`}>
  <img
    src={`https://image.tmdb.org/t/p/w500${movie.poster_path}`}
    alt={movie.title}
    className={styles.img}
  />
</RouterLink>
<Link
  component={RouterLink}
  to={` /movie/${movie.id}`}
  textAlign="center"
  sx={{ width: '200px' }}
>
  {movie.title
    ? movie.title
    : movie.name
    ? movie.name
    : movie.original_title}
</Link>

{movie.vote_average && (
  <Stack alignItems="center">
    <Tooltip title={` ${movie.vote_average} /10`} >
      <Box>
        <Rating
          name="read-only"
          value={movie.vote_average / 2}
          readOnly
          precision={0.1}
        />
      </Box>
    </Tooltip>
  </Stack>
)}
</Stack>
);
}

```

Лістинг фалу Navbar.jsx

```

import MenuIcon from '@mui/icons-material/Menu';
import {
  AppBar,
  Box,
  Button,
  Container,
  Drawer,
  IconButton,
  Link,
  List,
  ListItem,
  ListItemButton,
  ListItemIcon,
  ListItemText,
  Slide,
  Toolbar,
  Typography,
  useScrollTrigger,
} from '@mui/material';
import React, { useEffect, useState } from 'react';

```

		Яценко. В.П			ІІЗ.КР.Б – 121 – 25 – ІЗ	Арк.
		Чижмотря О.В.				50
Змн.	Арк.	№ докум.	Підпис	Дата		

```

import { Link as RouterLink, useNavigate } from 'react-router-dom';

import { iconComponents, LIST } from '../././constants';
import AuthModal from '../AuthModal/AuthModal';
import ProfileModal from '../profileModal/ProfileModal';

const Icon = ({ iconName }) => {
  const IconComponent = iconComponents[iconName];
  return <IconComponent />;
};

export default function Navbar() {
  const [isOpen, setIsOpen] = useState(false);
  const [isProfileOpen, setProfileOpen] = useState(false);
  const [isAuthOpen, setAuthOpen] = useState(false);
  const [username, setUsername] = useState(null);
  const navigate = useNavigate();
  const trigger = useScrollTrigger({ target: window });

  useEffect(() => {
    const name = localStorage.getItem('username');
    if (name) setUsername(name);
  }, []);

  const handleDrawerToggle = () => setIsOpen(prev => !prev);

  const handleLogout = () => {
    localStorage.removeItem('token');
    localStorage.removeItem('username');
    setUsername(null);
    navigate('/');
  };

  // callback для оновлення стану після логіну
  const handleLogin = ({ token, name }) => {
    localStorage.setItem('token', token);
    localStorage.setItem('username', name);
    setUsername(name);
    setAuthOpen(false);
  };

  return (
    <>
      <Slide appear={false} direction="down" in={!trigger}>
        <AppBar>
          <Container maxWidth="lg">
            <Toolbar>
              <IconButton color="inherit" onClick={handleDrawerToggle}>
                <MenuIcon />
              </IconButton>
              <Drawer open={isOpen} onClose={handleDrawerToggle}>
                <Box sx={{ width: 250 }} onClick={handleDrawerToggle}>
                  <List>
                    {LIST.map(item => (
                      <Link
                        key={item.title}
                        component={RouterLink}
                        to={item.url}

```

		Яценко. В.П			ІІЗ.КР.Б – 121 – 25 – ІІЗ	Арк.
		Чижмоторя О.В.				51
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        underline="none"
        color="inherit"
      >
      <ListItem disablePadding>
        <ListItemButton>
          <ListItemIcon>
            <Icon iconName={item.icon} />
          </ListItemIcon>
          <ListItemText primary={item.title} />
        </ListItemButton>
      </ListItem>
    </Link>
  )}
</List>
</Box>
</Drawer>

<Typography
  sx={{ color: 'white', textDecoratation: 'none' }}
  component={RouterLink}
  variant="h5"
  to="/"
>
  NikaCinema
</Typography>

<Box sx={{ flexGrow: 1 }} />

{username ? (
  <>
    <Typography
      variant="body1"
      sx={{ mr: 2 }}
      onClick={() => setProfileOpen(true)}
    >
      Привіт, {username}
    </Typography>
    <Button color="inherit" onClick={handleLogout}>
      Logout
    </Button>
  </>
) : (
  <Button color="inherit" onClick={() => setAuthOpen(true)}>
    LogIn
  </Button>
)}
</Toolbar>
</Container>
</AppBar>
</Slide>

<AuthModal
  open={isAuthOpen}
  onClose={() => setAuthOpen(false)}
  onLogin={handleLogin}
/>

<ProfileModal
  open={isProfileOpen}

```

		Яценко. В.П			ІІЗ.КР.Б – 121 – 25 – ІІЗ	Арк.
		Чижмотря О.В.				52
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    onClose={() => setProfileOpen(false)}
  />
</>
);
}

```

Лістинг файлу tmdbApi.js

```

import { createApi, fetchBaseQuery } from '@reduxjs/toolkit/query/react';

const apiKey = 'fa0cbbaf31b0830daf5e8683fdc0648d';

export const tmdbApi = createApi({
  reducerPath: 'tmdbApi',
  baseQuery: fetchBaseQuery({ baseUrl: 'https://api.themoviedb.org/3' }),
  endpoints: builder => ({
    getType: builder.query({
      query: ({ language, page, type }) =>
        `${type}/popular?api_key=${apiKey}&language=${language}&page=${page}`,
    }),
    getTrendingAll: builder.query({
      query: ({ language, page }) =>
        `/trending/all/week?api_key=${apiKey}&language=${language}&page=${page}`,
    }),
    getMovieGenre: builder.query({
      query: ({ language, genre, page }) =>
        `/discover/movie?api_key=${apiKey}&language=${language}&page=${page}&with_genres=${genre}`,
    }),
    getPopularMovie: builder.query({
      query: ({ language, page }) =>
        `/movie/popular?api_key=${apiKey}&language=${language}&page=${page}`,
    }),
    getGenres: builder.query({
      query: () => `/genre/movie/list?language=uk&api_key=${apiKey}`,
    }),
    getFilm: builder.query({
      query: id => `/movie/${id}?api_key=${apiKey}&language=uk`,
    }),
    getActors: builder.query({
      query: id => `/movie/${id}/credits?api_key=${apiKey}&language=uk`,
    }),
    getMovieVideos: builder.query({
      query: id => `/movie/${id}/videos?api_key=${apiKey}`,
    }),
    getCollection: builder.query({
      query: collectionId =>
        `/collection/${collectionId}?api_key=${apiKey}&language=uk`,
    }),
    getSearchMovies: builder.query({
      query: ({ query, language = 'uk', page = 1 }) =>
        `/search/movie?api_key=${apiKey}&language=${language}&query=${encodeURIComponent(query)}&page=${page}`,
    }),
  }),
});

```

		Яценко. В.П			ІІЗ.КР.Б – 121 – 25 – ІІЗ	Арк.
		Чижмотря О.В.				53
Змн.	Арк.	№ докум.	Підпис	Дата		

```

export const {
  useGetTypeQuery,
  useGetTrendingAllQuery,
  useGetMovieGenreQuery,
  useGetGenresQuery,
  useGetActorsQuery,
  useGetFilmQuery,
  useGetMovieVideosQuery,
  useGetCollectionQuery,
  useGetSearchMoviesQuery
} = tmdbApi;

```

		Яценко. В.П			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Чижмотря О.В.				54
Змн.	Арк.	№ докум.	Підпис	Дата		