

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

ПОЯСНЮВАЛЬНА ЗАПИСКА

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

на тему,

«Вебплатформа бронювання квитків у кінотеатр»

Виконав студент групи ІПЗ-22-4
СІВЧЕНКО Максим Тарасович

Керівник роботи,
ЄФРЕМОВ Юрій Миколайович

Рецензент,
ЧИЖМОТРЯ Олексій Володимирович

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

«ЗАТВЕРДЖУЮ»

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

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти, **СІВЧЕНКО Максим Тарасович**

Керівник роботи, **ЄФРЕМОВ Юрій Миколайович**

Тема роботи, **«Вебплатформа бронювання квитків у кінотеатр»**,
затверджена Наказом закладу вищої освіти від **«27» лютого 2026 р., № 92/с**

Об'єкт дослідження, процес взаємодії користувачів із кіноконтентом та механізми онлайн-резервування місць у вебсередовищі; предметом дослідження, методи та інструментальні засоби розробки вебплатформи, що забезпечує функції бронювання й продажу квитків, а також автоматизацію управління кіносеансами.

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

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

РЕФЕРАТ

Випускна кваліфікаційна робота бакалавра складається з пояснювальної записки та вебплатформи бронювання квитків у кінотеатр. Звіт містить вступ, три розділи, загальні висновки та перелік використаних джерел. Загальний обсяг звіту складає 168 сторінок, робота містить 66 рисунків, 19 таблиць та 30 джерел літератури.

У звіті розглянуто предметну галузь онлайн-бронювання квитків, процеси автоматизації продажу квитків у кінотеатрах та сучасні підходи до побудови систем взаємодії з клієнтами. Проаналізовано ключові функціональні можливості сучасних сервісів продажу квитків, зокрема інтерактивне вибору місць у залі, систему онлайн-платежів, відстеження стану сеансів у реальному часі та механізми оповіщення користувачів. Досліджено архітектурні принципи побудови вебплатформи, взаємодію між рівнями системи (клієнтським, серверним та рівнем даних), а також виявлено обмеження та виклики, притаманні наявним аналогам у сфері кінотеатрального бізнесу.

Ключові слова, ОНЛАЙН-БРОНЮВАННЯ, КІНОТЕАТР, ВЕБПЛАТФОРМА, КВИТКИ, ІНТЕРАКТИВНА СХЕМА ЗАЛУ, ПЛАТІЖНІ СИСТЕМИ, БРОНЮВАННЯ МІСЦЬ, ВЕБЗАСТОСУНОК.

					ІПЗ.КР.Б – 121 – 26 – ПЗ			
Змн.	Арк.	№ докум.	Підпис	Дата	Вебплатформа бронювання квитків у кінотеатр	Літ.	Арк.	Аркуші
Розроб.		Сівченко М.Т.						
Перевір.		Єфремов Ю.М.					3	168
Керівник						Житомирська політехніка, група ІПЗ-22-4		
Н. контр.		Тичина О.П.						
Зав. каф.		Вакалюк Т.А.						

ABSTRACT

The bachelor's final qualifying work consists of an explanatory note and a web platform for booking cinema tickets. The report contains an introduction, three sections, general conclusions and a list of sources used. The total volume of the report is 168 pages, the work contains 66 figures, 19 tables and 30 sources of literature.

The report examines the subject area of online ticket booking, ticket sales automation processes in cinemas, and modern approaches to building customer interaction systems. The key functionalities of modern ticket sales services are analyzed, including interactive seat selection in the hall, online payment system, real-time session status tracking, and user notification mechanisms. The architectural principles of building a web platform, interaction between system levels (client, server, and data levels), and limitations and challenges inherent in existing analogues in the cinema business are investigated.

Keywords, ONLINE BOOKING, CINEMA, WEB PLATFORM, TICKETS, INTERACTIVE HALL SCHEME, PAYMENT SYSTEM, SEAT RESERVATION, WEB APPLICATION.

					ІПЗ.КР.Б – 121 – 26 – ІПЗ			
Змн.	Арк.	№ докум.	Підпис	Дата				
Розроб.	Сівченко М.Т.				Вебплатформа бронювання квитків у кінотеатр	Лім.	Арк.	Аркушів
Перевір.	Єфремов Ю.М.						3	168
Керівник						Житомирська політехніка, група ІПЗ-22-4		
Н. контр.	Тичина О.П.							
Зав. каф.	Вакалюк Т.А.							

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ	5
ВСТУП	7
РОЗДІЛ 1. АНАЛІЗ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ..	9
1.1. Загальна характеристика проблеми.....	9
1.2. Аналіз аналогів програмного продукту.....	12
1.3. Вибір архітектури системи.....	19
1.4. Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення.....	22
Висновки до першого розділу.....	26
РОЗДІЛ 2. ПРОЄКТУВАННЯ СИСТЕМИ БРОНЮВАННЯ КВИТКІВ У КІНОТЕАТР	27
2.1. Визначення варіантів використання та об'єктно-орієнтованої структури системи.....	27
2.2. Розробка бази даних системи.....	40
2.3. Проєктування та реалізація алгоритмів роботи системи.....	56
2.4. Реалізація функціоналу підсистеми інтерактивного бронювання та придбання квитків.....	66
Висновки до другого розділу.....	70
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ВЕБПЛАТФОРМОЮ БРОНЮВАННЯ КВИТКІВ У КІНОТЕАТР	73
3.1. Визначення варіантів використання та об'єктно-орієнтованої структури системи.....	73
3.2. Структура інтерфейсу вебплатформи.....	77
3.3. Тестування вебплатформи.....	113
Висновки до третього розділу.....	117
ВИСНОВКИ	119
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	120
ДОДАТКИ	123

		Сівченко М.Т.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		5

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API - прикладний програмний інтерфейс

БД - база даних

ПЗ - програмне забезпечення

СУБД - система управління базами даних

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

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

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

HTTP / HTTPS - протокол передавання гіпертексту

JSON - текстовий формат обміну даними

JWT - відкритий стандарт для створення токенів доступу

		Сівченко М.Т.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		6

ВСТУП

Актуальність теми. На сьогодні сфера цифрових послуг та індустрія розваг розвиваються надзвичайно стрімко, через що кількість кіносеансів та різноманітних кінозалів суттєво зросла. Внаслідок цього користувачі часто витрачають на пошук потрібного сеансу, відповідного часу та вільного місця у залі набагато більше часу, ніж на сам перегляд фільму. Проте оперативно придбати або забронювати квиток онлайн без зайвих переplat та труднощів досі залишається проблемою. Популярні сервіси зазвичай перевантажені надокучливими рекламними банерами, мають заплутаний інтерфейс або через складну навігацію змушують клієнта проходити довгий і стомлюючий процес оформлення замовлення замість того, щоб надати можливість зробити це у кілька кліків. Саме тому зараз спостерігається гостра потреба у сучасній, швидкій та простій вебплатформі бронювання, яка б дозволяла користувачу миттєво бачити актуальний розклад сеансів, легко обирати бажану категорію місць та безпечно здійснювати оплату. На даний момент чітко простежується тенденція, де люди віддають перевагу мінімалістичним цифровим рішенням, які дозволяють розв'язати задачу купівлі квитка максимально оперативно. Отже, розробка такої вебплатформи є вкрай важливою, оскільки вона перетворює процес планування походів до кінотеатру з виснажливого та перевантаженого обов'язку на приємний, швидкий та надійний досвід, побудований на простоті, інтерактивності та якісному сервісі.

Метою випускної кваліфікаційної роботи бакалавра є розробка вебплатформи бронювання квитків у кінотеатр, що реалізується в межах випускної кваліфікаційної роботи бакалавра. Для цього досліджено механізми онлайн-замовлення посадкових місць, управління сеансами та формування розкладу на існуючих сервісах, а також визначено функціонал майбутньої системи.

Завданням проєкту став аналіз існуючих аналогів сайтів для онлайн-замовлення та бронювання квитків у кінотеатрах, виявлення їхніх переваг та недоліків для подальшого усунення мінусів у власній системі. Спроєктувати

		Сівченко М.Т.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		7

архітектуру системи та базу даних, використовуючи методи візуального моделювання. Розробити програмну реалізацію основних модулів платформи (включаючи інтерактивні схеми залів, вибір категорій місць та обробку платежів). Налаштувати підсистему підтримки, автоматичних сповіщень та інтерактивної взаємодії в реальному часі під час вибору квитків користувачами.

Об’єктом дослідження є процес взаємодії користувачів з інтерактивним розкладом сеансів та механізми онлайн-бронювання квитків у кінотеатр в інтернет-середовищі.

Предметом дослідження є застосування сучасних вебтехнологій, необхідних для створення платформи, що забезпечує онлайн-бронювання квитків, управління сеансами, автоматизацію продажів та зручну взаємодію користувачів. Для розв’язання поставлених завдань та розробки платформи застосовуються сучасні методи системного аналізу, які досліджують вимоги до продукту, принципи об’єктно-орієнтованого програмування для побудови архітектури, а також методи візуального моделювання для проєктування бази даних і логіки роботи вебзастосунку.

За темою випускної кваліфікаційної роботи бакалавра було опубліковано тези на “Міжнародної наукової конференції студентів та молодих вчених ” - 2026, 14-15 травня 2026 року [1, 2].

		Сівченко М.Т.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		8

РОЗДІЛ 1. АНАЛІЗ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

1.1. Загальна характеристика проблеми

Перед початком розробки необхідно було вивчити поточний стан ринку програмних рішень для кінотеатрів та чітко визначити вимоги до майбутнього продукту.

Аналіз сучасного стану цифровізації індустрії розваг свідчить про стрімке зростання частки онлайн-продажів квитків у загальному обсязі касових зборів кінотеатрів. За останні роки глядачі все частіше надають перевагу попередньому бронюванню місць через вебсервіси, що дозволяє уникнути черг та спланувати відвідування заздалегідь. Водночас якість цифрових інструментів, що пропонуються кінотеатрами, суттєво відстає від очікувань сучасного користувача, звиклого до зручних мобільних застосунків та миттєвого зворотного зв'язку. Більшість існуючих платформ розроблялися ще в епоху серверного рендерингу та не були переосмислені з урахуванням сучасних вимог до швидкості відгуку інтерфейсу та синхронізації даних у реальному часі [3].

Сучасні вебплатформи функціонують як глобальні інформаційні системи, що забезпечують доступ до сервісів через мережу Інтернет незалежно від географічного розташування користувача [14]. Нова вебплатформа розробляється із розрахунком на кілька груп користувачів, звичайних відвідувачів, дистриб'юторів фільмів та адміністраторів системи. На багатьох діючих сервісах інтерфейс вибору місць, процес оплати, налаштування розкладу та вивантаження фінансової звітності працюють відокремлено або реалізовані незручно. Це ускладнює роботу з платформою і сповільнює процес обслуговування клієнтів.

Тому головним завданням стало створення зручного та швидкого сервісу для людей. Платформа має надійно зберігати в одному місці всю інформацію про фільми, розклад сеансів та профілі користувачів. Також потрібно було

		Сівченко М.Т.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		9

зробити зручний вибір крісел на схемі залу, зрозумілу онлайн-оплату квитків і закласти основу для збору статистики, щоб адміністратори та дистриб'ютори могли легко бачити заповнюваність залів і касові збори.

Процеси пошуку розкладу, вибору посадкових місць, онлайн-оплати, скасування замовлень та формування звітності часто реалізуються за допомогою застарілих або розрізнених інструментів. Це значно ускладнює покупку квитків і знижує зручність використання всього сервісу кінотеатру.

Завданням випускної кваліфікаційної роботи бакалавра є розробка програмного середовища, яке забезпечить повну підтримку та автоматизацію головних процесів розкладу сеансів, продажу квитків та взаємодії з користувачами.

Окремої уваги заслуговує проблема масштабованості існуючих рішень. Під час масових прем'єр популярних фільмів онлайн-сервіси кінотеатрів часто не витримують пікового навантаження, що призводить до збоїв у роботі інтерфейсу вибору місць, помилок при проведенні платежів та втрати клієнтів. Технічною причиною таких збоїв є переважно синхронна архітектура серверної частини та відсутність механізмів кешування та оптимізації запитів до бази даних. Асинхронна подієво-орієнтована модель виконання Node.js у поєднанні з технологією WebSocket дозволяє принципово вирішити цю проблему, забезпечуючи стабільну обробку великої кількості одночасних з'єднань без блокування серверних потоків [22].

Окрім суто технічних моментів наприклад вибору місць чи оплати, на більшості сайтів кінотеатрів дуже бракує нормальної, живої підтримки та спілкування. Зазвичай такі платформи працюють як звичайні сухі автомати, людина просто купує квиток, і на цьому все закінчується. Але в реальному житті часто виникають ситуації, коли треба щось швидко уточнити щодо залу, змінити час або взагалі повернути гроші, бо змінилися плани.

Зараз у глядачів немає зручного місця, де можна без необхідності заповнення великої кількості форм оперативно зв'язатися з адміністратором платформи, розібратися з проблемою прямо в чаті й отримати адекватну

		Сівченко М.Т.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремов Ю.М.				10
Змн.	Арк.	№ докум.	Підпис	Дата		

допомогу. Метою стало поєднання чіткої роботи автоматизованої системи продажу квитків із якісним клієнтським сервісом. У цьому проєкті особливу увагу було приділено модулю чату підтримки та логіці швидкого зворотного зв'язку. Головне, щоб ні користувач, ні персонал кінотеатру не мучилися з незручним інтерфейсом, коли треба розібратися з квитками. Було важливо реалізувати код таким чином, щоб система зберігала стабільність і витримувала значне навантаження при одночасному зверненні великої кількості користувачів.

Сьогодні це може бути просто зручна покупка квитків, а завтра - велика аналітична система, яка допоможе кінотеатрам гнучко керувати сеансами та підлаштовувати розклад під те, що люди реально дивляться. Технології для платформи обиралися під ці потреби. Зв'язка Node.js та React дозволяє зробити інтерфейс дуже спритним, сторінки відкриваються миттєво, схема залу не зависає при виборі крісел, а саме замовлення оформлюється швидко. Це хороший фундамент на майбутнє, щоб сайт працював стабільно навіть під час гучних прем'єр, коли на нього одночасно звертатиметься значна кількість користувачів. Головне - щоб глядач просто купував квиток у пару кліків і не витрачав час на боротьбу з незручним інтерфейсом.

З точки зору безпеки даних користувачів, сучасна вебплатформа кінотеатру є критичною системою, оскільки обробляє персональні дані глядачів та здійснює фінансові транзакції. Відповідно до рекомендацій OWASP щодо захисту вебзастосунків, система повинна забезпечувати шифрування всіх з'єднань через протокол HTTPS, безпечне зберігання паролів з використанням криптографічного хешування, захист від міжсайтових атак та несанкціонованого доступу до API [30]. Реалізація цих вимог безпеки є невід'ємною частиною розробки будь-якого комерційного вебсервісу що працює з платіжними даними клієнтів [29].

Головне, що хотілося зробити - це поєднати чітку роботу автоматизованої каси з якісним сервісом, орієнтованим на зручність клієнта. Коли на сайті все

		Сівченко М.Т.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремов Ю.М.				11
Змн.	Арк.	№ докум.	Підпис	Дата		

працює чітко, людина не стоїть у чергах і не нервує через завислі каси, а просто спокійно планує свій відпочинок.

Це був перший крок до створення платформи, де код і технології реально працюють на користь звичайних людей, економлять їхній час і роблять усе для їхньої зручності.

1.2. Аналіз аналогів програмного продукту

На сьогоднішній день онлайн-сервіси для замовлення квитків та сайти кінотеатрів стали дуже популярними. Вони допомагають людям швидко знаходити потрібні сеанси, бронювати місця, купувати квитки онлайн та керувати своїми замовленнями через особистий кабінет. Такі системи сильно спрощують життя глядачам, які цінують свій час. Щоб довести, що розробка власної вебплатформи для кінотеатру у межах бакалаврської роботи є доцільною, потрібно спершу розібратися, які схожі проєкти вже є в інтернеті, та подивитися на їхні плюси й мінуси.

Вивчення аналогів дозволить краще зрозуміти їхню структуру та основні інструменти. Завдяки аналізу популярних сайтів можна визначити найбільш важливі функції, особливості реалізації інтерактивної схеми залу, роботу системи онлайн-оплати, формування розкладу сеансів та організацію зворотного зв'язку з користувачами. Це допоможе врахувати існуючий досвід і створити зручний та конкурентоспроможний продукт.

Для порівняльного аналізу було обрано три відомі платформи, які спеціалізуються на продажу квитків та автоматизації послуг кінотеатрів, українські мережі Multiplex і Планета Кіно, а також великий міжнародний сервіс Fandango.

Multiplex - це найбільша мережа кінотеатрів в Україні, яка має власну розвинену вебплатформу для клієнтів. Головна ідея цього сервісу полягає у забезпеченні швидкого доступу до актуальної афіші, детального розкладу сеансів у різних містах та можливості онлайн-купівлі квитків. Користувач може знайти інформацію про поточні фільми, тривалість сеансів, формат показу (наприклад, 2D, 3D чи IMAX), ціни на різні категорії місць, а також

		Сівченко М.Т.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремов Ю.М.				12
Змн.	Арк.	№ докум.	Підпис	Дата		

придбати супутню продукцію кінобару разом із квитками (рис. 1.1). Детальний порівняльний аналіз основних функціональних модулів вебплатформи Multiplex, а також оцінка їхньої ефективності з точки зору клієнтського досвіду, наведені у таблиці 1.1.

Таблиця 1.1.

Переваги та недоліки платформи Multiplex

Переваги	Недоліки
Актуальний розклад та велика база сеансів. Інформація про фільми, часи показів та доступні зали в усіх кінотеатрах мережі постійно оновлюється в реальному часі.	Обмежені можливості персоналізації. Система не пропонує гнучких рекомендацій фільмів на основі попередніх покупок або вподобань конкретного глядача.
Детальна інформація про умови перегляду. Наявність чітких даних про формат показу (2D, 3D, IMAX), тривалість фільму, вікові обмеження та розрахунок вартості квитків залежно від типу місця.	Перевантаженість інтерфейсу на сторінці вибору квитків. Велика кількість додаткових пропозицій (товари кінобару, акції, супутні послуги) під час оформлення замовлення може відволікати та ускладнювати процес для нових користувачів.
Зручна система вибору місць. Інтерактивна схема залів дозволяє чітко бачити розташування екрана, розділення крісел на звичайні та VIP-зони, а також зайняті й вільні місця на момент бронювання.	Відсутність живого чату підтримки на сайті. У разі виникнення проблем з оплатою або розкладом користувачеві доводиться шукати контакти або переходити в сторонні месенджери, що уповільнює розв'язання питань.
Інтеграція з платіжними системами. Можливість швидкої та безпечної оплати квитків через сучасні платіжні шлюзи (наприклад, Apple Pay або Google Pay) без необхідності щоразу вводити дані картки вручну.	Складна процедура скасування замовлення. Процес автоматичного повернення квитків через сайт у разі зміни планів глядача часто потребує додаткового звернення на гарячу лінію.

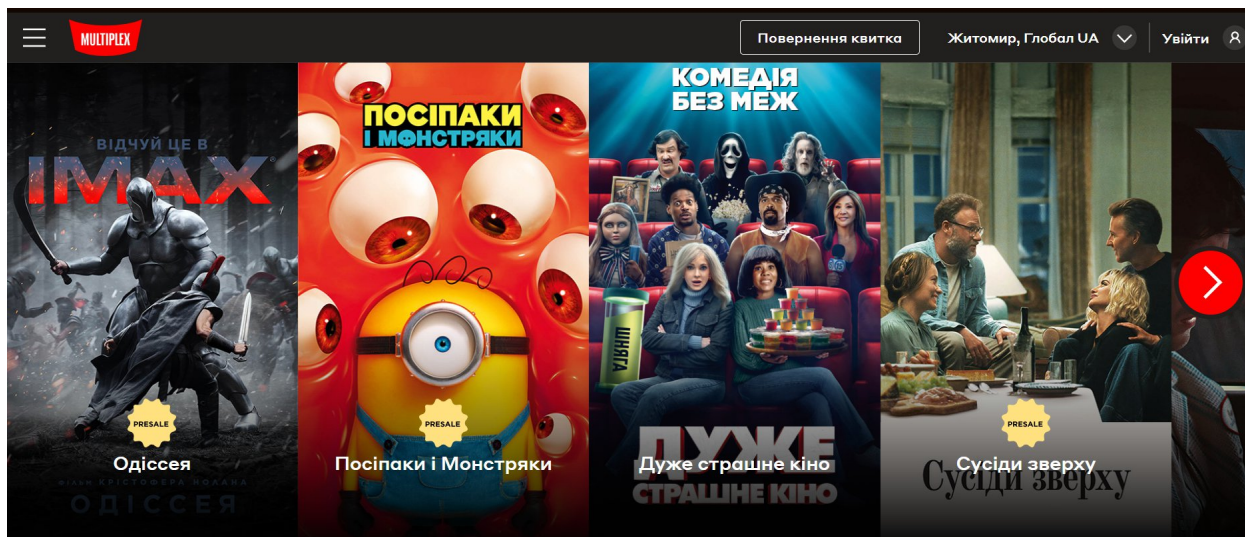


Рисунок 1.1 - Інтерфейс сайту **Multiplex**

«Планета Кіно» - це відома українська мережа кінотеатрів, яка активно розвиває свої цифрові сервіси та має сучасну вебплатформу для онлайн-операцій. Головна ідея цього ресурсу полягає в автоматизації процесу купівлі квитків, продажу продукції кінобару без черг, а також у розвитку власної екосистеми для постійних клієнтів. Система орієнтована на максимальну інтерактивність, користувач може детально вивчити особливості залів (наприклад, технології 4DX, RE'LUX або особливі крісла-реклайнери), переглянути розклад і миттєво оформити замовлення. Велику увагу тут приділено особистому кабінету, де зберігаються придбані квитки, історія візитів та бонуси. Цифрова стратегія мережі виходить далеко за межі простого бронювання. Вебплатформа виступає як повноцінний хаб для клієнтської підтримки та маркетингових комунікацій. Завдяки впровадженню алгоритмів аналізу вподобань користувачів, система демонструє персоналізовані пропозиції, що дозволяє оптимізувати заповнюваність кінозалів навіть у години низького попиту (рис. 1.2). Візуалізація архітектурного підходу до реалізації клієнтського досвіду мережі дозволяє глибше зрозуміти механізми взаємодії користувача з цифровою екосистемою. Разом із тим, для детального порівняння функціональних особливостей платформи «Планета Кіно» з іншими ключовими гравцями ринку кінотеатральних послуг, було проведено порівняльний аналіз, результати якого представлені у таблиці 1.2.

		Сівченко М.Т.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремов Ю.М.				14
Змн.	Арк.	№ докум.	Підпис	Дата		

Таблиця 1.2.

Переваги та недоліки платформи Планета Кіно

Переваги	Недоліки
Ефективна інтеграція програми лояльності. Клієнти бачать накопичені бонуси безпосередньо під час вибору квитків і можуть одразу використати їх для повної або часткової оплати місць.	Обмежені інструменти комунікації на сайті. На платформі відсутня можливість живого текстового спілкування з підтримкою або швидкого обговорення деталей сеансів між користувачами.
Детальна візуалізація технологій кінотеатру. Сайт містить вичерпну інформацію про технічні особливості кожного типу залу та специфіку преміальних місць.	Мінімальна персоналізація розкладу. Головна сторінка не підлаштовується під історію переглядів користувача, пропонуючи всім однакові загальні списки фільмів.
Зручний інтерфейс інтерактивної схеми. Процес вибору та зміни місць у кошику реалізований плавно, з чітким маркуванням різних цінових зон і контурів залу.	Складність орієнтації у специфічних форматах залів. Новим відвідувачам буває важко одразу зрозуміти різницю між різними категоріями крісел та умовами доступу до них.
Автоматизація супутніх покупок. Можливість замовити товари з кінобару одночасно із бронюванням квитків, що дозволяє отримати все за одним QR-кодом.	Жорсткі часові обмеження на повернення квитків. Скасувати замовлення через сайт без звернення до контакт-центру можна лише за певну кількість часу до початку сеансу, що не завжди зручно.

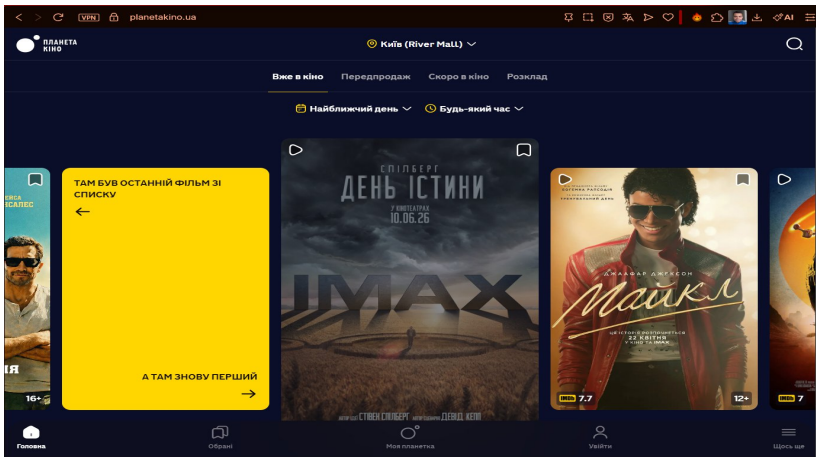


Рисунок 1.2 - Інтерфейс сайту Планета Кіно

Fandango це найбільша міжнародна онлайн-платформа та квитковий агрегатор, який об'єднує тисячі кінотеатрів різних мереж у єдину централізовану систему. Головна ідея сервісу полягає в тому, щоб надати користувачеві можливість купувати квитки в будь-який кінотеатр міста через уніфікований інтерфейс. Окрім стандартного розкладу та інтерактивних схем залів, платформа пропонує глибоку інтеграцію з сервісами перегляду трейлерів, розширену систему цифрових квитків, що не потребують друку, та уніфіковану програму лояльності, яка охоплює різні кіномережі (рис. 1.3).

Узагальнюючи підхід Fandango до агрегації медіаданих, слід зазначити, що ключовим фактором успіху такої платформи є масштабованість системи та здатність працювати з гетерогенними даними від різних партнерів. Порівняльну характеристику функціональних можливостей Fandango наведено у таблиці 1.3.

Таблиця 1.3.

Переваги та недоліки платформи Fandango

Переваги	Недоліки
Масштабність та об'єднання багатьох кіномереж. Користувач може порівнювати розклад і купувати квитки у різні кінотеатри на одному сайті без необхідності переходити на сторонні ресурси.	Складна логіка повернення квитків. Через те, що платформа є посередником між глядачем та різними кінотеатрами, правила скасування замовлення чи повернення коштів можуть сильно відрізнятися і заплутувати користувача.
Зручна система цифрових квитків. Сервіс повністю автоматизує процес проходу на сеанс за допомогою персональних QR-кодів в особистому кабінеті, які зчитуються безпосередньо на вході до зали.	Прив'язка до конкретного регіону. Більшість розширених функцій, унікальних акцій та інтеграцій працюють лише на ринку США, що ускладнює адаптацію такої моделі для інших країн.
Інтеграція з медіаконтентом. Наявність офіційних трейлерів, ексклюзивних інтерв'ю та описів фільмів безпосередньо на сторінці вибору квитків, що допомагає швидше визначитися із сеансом.	Відсутність внутрішніх інструментів для прямого спілкування. На сайті немає чатів для швидкої взаємодії з підтримкою, через що розв'язання технічних проблем під час оплати може тривати довше.

Універсальна програма лояльності. Можливість накопичувати бали та отримувати знижки, які діють у різних незалежних мережах кінотеатрів, підключених до системи.	Складність оновлення даних у реальному часі. Через роботу з багатьма різними серверами кінотеатрів іноді виникають затримки в оновленні інформації про вільні місця на схемах залів.
---	--

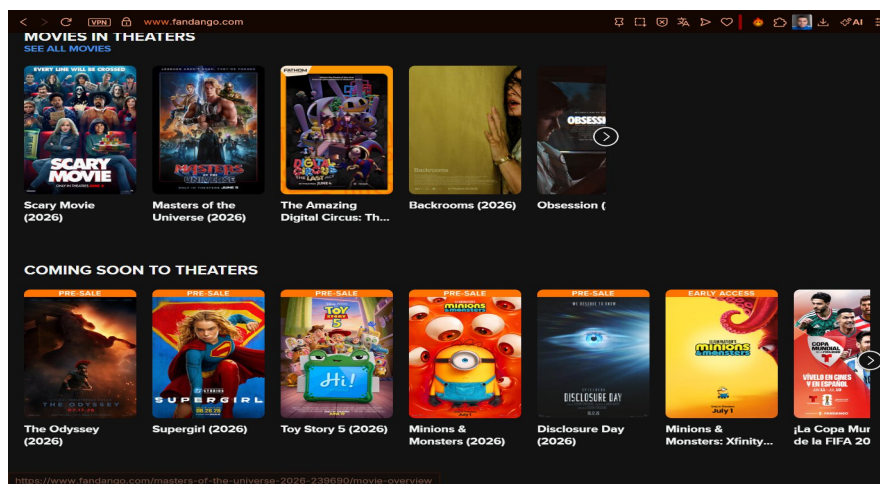


Рисунок 1.3 - Інтерфейс сайту Fandango

На основі виконаного аналізу існуючих програмних рішень було складено порівняльну таблицю ключових характеристик досліджуваних платформ у таблиці 1.4.

Таблиця 1.4.

Порівняльна таблиця ключових характеристик

Характеристика	AbsoluteCinema	Multiplex	Планета Кіно	Fandango
Тип платформи	Вебплатформа автоматизації та бронювання	Вебсайт мережі кінотеатрів	Вебсайт мережі кінотеатрів	Міжнародний квитковий агрегатор
Онлайн-купівля квитків	Так	Так	Так	Так
Інтерактивна схема залу	Так	Так	Так	Так
Оновлення місць у реальному часі	Так	Так	Так	Обмежено (бувають затримки)

продовж. табл.1.4

Інтеграція з платіжними шлюзами	Так	Так	Так	Частково
Зберігання квитків у кабінеті	Так	Так	Так	Так
Захист панелі керування (ролі)	Так	Так	Так	Так
Звіти для дистриб'юторів	Так	Так	Так	Обмежено
Мультимовність інтерфейсу	Ні	Так	Ні	Ні
Вхід через соцмережі	Ні	Так	Так	Так
Українська мова	Так	Так	Так	Ні
Розгортання у хмарі	Так	Так	Так	Так
Інтегрований чат підтримки	Так (внутрішній Hub)	Ні (сторонні месенджери)	Ні (соцмережі)	Ні (форма зворотного зв'язку)
Управління розкладом й сеансами	Так	Так	Так	Так
Автоматизований розрахунок зборів	Так	Так	Так	Так
Панель адміністратора	Так	Так	Так	Так
Програма лояльності / Промокоди	Так	Так	Так	Так
Повернення квитків через сайт	Так (швидке скасування)	Обмежено за часом	Обмежено за часом	Складний процес (через посередника)
Архітектурний поділ (Frontend/Backend)	Так	Так	Так	Так
Технологічний стек бекенду	Node.js	.NET / PHP	.NET / Node.js	Java / .NET
Замовлення товарів кінобару	Ні	Так	Так	Ні

Проведене зіставлення наявних рішень (табл. 1.4) демонструє, що жоден із великих ринкових продуктів не закриває всі запити аудиторії та менеджменту кінотеатрів одночасно. Розробники здебільшого концентруються на вузьких завданнях, створюють закриті внутрішні системи під конкретну мережу або будують глобальні платформи посередники, де

		Сівченко М.Т.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремов Ю.М.				18
Змн.	Арк.	№ докум.	Підпис	Дата		

втрачається оперативний зв'язок із клієнтом. У результаті відвідувачі стикаються з відсутністю швидкої технічної допомоги під час збоїв оплати, а управлінці та правовласники позбавлені зручних динамічних звітів про касові збори та завантаженість кінозалів. З огляду на це, створення альтернативної вебплатформи є цілком доцільним кроком.

Цей проєкт дає змогу реалізувати найбільш ефективні інженерні підходи в одному середовищі, інтерактивне керування посадковими місцями, синхронізацію стану залу через вебсокети без перезавантаження сторінок та пряму лінію зв'язку з оператором додатка. Водночас архітектура системи залишається максимально доступною та комфортною для звичайного користувача.

1.3. Вибір архітектури системи

Вибір архітектури є фундаментом проєкту, що визначає продуктивність системи, її стабільність під навантаженням та легкість подальшого масштабування. Враховуючи вимоги до миттєвої реакції на дії користувачів (наприклад, під час масового бронювання місць), серверна частина повинна забезпечувати мінімальні затримки.

Для вибору оптимального рішення було проаналізовано два підходи: монолітну та мікросервісну архітектуру. Модульний моноліт як архітектурний підхід забезпечує чіткий поділ відповідальності між компонентами системи при збереженні єдиного середовища розгортання [4].

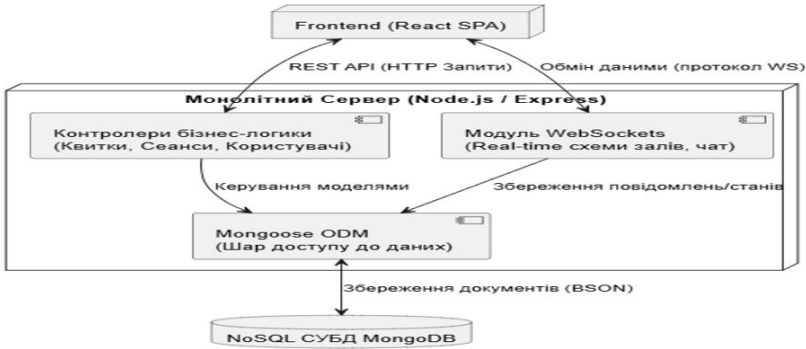


Рисунок 1.4 - Схема монолітної архітектури

Мікросервісна архітектура передбачає декомпозицію монолітного додатка на набір автономних, слабозв'язаних сервісів, візуалізацію

концептуальної моделі (рис. 1.5.). Кожен із цих компонентів зосереджений на реалізації окремого бізнес-процесу наприклад, сервіс управління розкладом, підсистема бронювання місць, модуль чат-підтримки чи система нотифікацій. Ці компоненти розгортаються незалежно один від одного та обмінюються даними через стандартизовані протоколи, найчастіше за допомогою REST API або повідомлень.

Незважаючи на суттєві переваги, мікросервісний підхід суттєво підвищує складність системи. Впровадження такої архітектури потребує складної інфраструктури для оркестрації контейнерів (наприклад, Kubernetes), налагодження надійних мережових комунікацій, а також впровадження розподілених транзакцій та складних механізмів моніторингу. Для проєктів середнього рівня, особливо на стадії MVP (мінімально життєздатного продукту), ці накладні витрати на адміністрування та складність інтеграції розподілених баз даних можуть бути економічно невиправданими, що часто призводить до уповільнення розробки та необхідності залучення великого штату DevOps-інженерів.

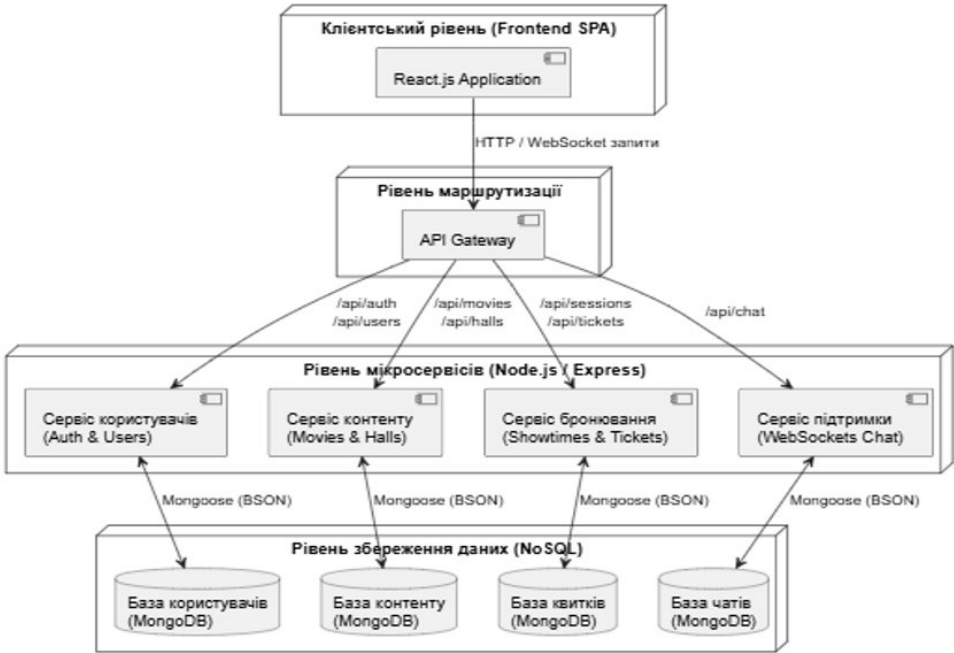


Рисунок 1.5 - Схема мікросервісної архітектури

Зваживши переваги та недоліки класичних підходів, для вебплатформи було обрано модульну монолітну архітектуру (рис. 1.6). Цей підхід дозволяє

розробляти систему як цілісний застосунок, зберігаючи простоту тестування, швидкість розгортання та ефективність розробки на початкових етапах.

Водночас усередині коду запроваджено чіткий поділ, кожна функціональна одиниця проєкту реалізована як незалежний модуль. Ці компоненти взаємодіють виключно через заздалегідь визначені інтерфейси, що приховує внутрішню логіку та запобігає перетворенню коду на заплутану структуру. Така слабка пов'язаність модулів створює надійний фундамент для масштабування: у разі зростання навантаження будь-який модуль (наприклад, сервіс бронювання) можна легко відокремити та перетворити на самостійний мікросервіс без переписування всієї системи.

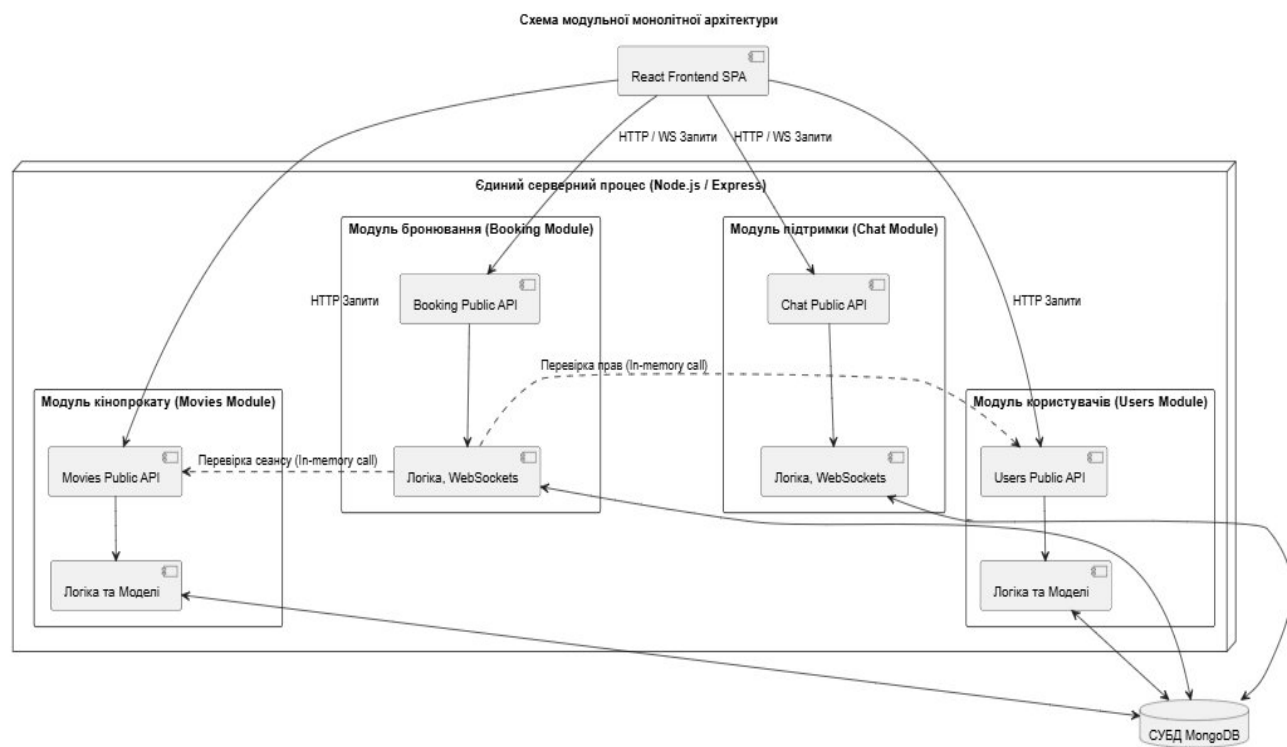


Рисунок 1.6 - Схема архітектури модульного моноліту

В основі архітектури платформи лежить звичайна клієнт-серверна модель. Клієнт-серверна взаємодія є базовим стандартом побудови сучасних вебсистем, де запити від браузера передаються на сервер через стандартизовані мережеві протоколи [3].

Клієнтська частина реалізована як SPA (Single Page Application) на базі React. Односторінковий застосунок (SPA) завантажується одноразово та динамічно оновлює вміст без перезавантаження сторінки, що забезпечує

високу швидкість відгуку інтерфейсу [19]. Після першого завантаження додаток виконується повністю в браузері користувача, динамічно оновлюючи інтерфейс без перезавантаження сторінки звертаючись до сервера лише за даними через REST API. Серверна частина (Node.js/Express) виконує функції REST API для обробки HTTP-запитів, а також підтримує постійне WebSocket-з'єднання (Socket.io) для синхронізації стану залу та роботи чату підтримки в реальному часі. Такий розподіл завдань значно спрощує написання коду, полегшує тестування окремих частин сайту та підвищує безпеку даних.

Поєднання клієнт-серверної моделі з модульним монолітом на сервері стало найкращим рішенням для проєкту. Завдяки такому розділенню React-клієнт забезпечує миттєву реакцію інтерфейсу (зокрема під час вибору місць у залі в реальному часі), оскільки не залежить від рендерингу на сервері - сервер відповідає лише за обробку даних та бізнес-логіку. Водночас сервер спокійно й автономно обробляє замовлення, створює квитки та забезпечує роботу чату підтримки.

1.4. Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення

Вебплатформа AbsoluteCinema поєднує в собі базу даних кіноконтенту, можливість вибору місць у залі, бронювання квитків, а також чати підтримки глядачів. Реалізація цих функцій вимагає швидкої обробки запитів, гнучкої структури даних та створення легкого інтерфейсу. Тому для розробки системи за принципом модульного моноліту було обрано збалансований стек технологій.

Основою серверної частини стала платформа Node.js та мова програмування JavaScript. Використання однієї мови як для фронтенду, так і для бекенду дозволяє розробляти систему в межах єдиної екосистеми, що значно прискорює написання коду. Платформа Node.js працює на базі рушія V8 та використовує асинхронну модель обробки подій. Завдяки цьому сервер

		Сівченко М.Т.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		22

може стабільно опрацьовувати велику кількість одночасних запитів від користувачів без блокування потоків [13, 15].

Для обміну даними в реальному часі (наприклад, щоб миттєво оновлювати зайняті крісла на схемі залу під час відкриття продажів або передавати повідомлення в чаті) було обрано технологію WebSockets. Організація мережевої взаємодії між клієнтом та сервером у режимі реального часу базується на протоколах транспортного рівня, що забезпечують надійну передачу даних [22]. Вона забезпечує постійне відкрите з'єднання між браузером і сервером, протокол WebSocket стандартизовано специфікацією RFC 6455, яка визначає механізм встановлення постійного дуплексного з'єднання між браузером та сервером [23], що позбавляє від необхідності постійно надсилати повторні HTTP-запити для оновлення сторінки [2, 21].

Робота з даними. Для збереження інформації використовується документ-орієнтована NoSQL база даних MongoDB. Такий вибір зумовлений гнучкістю структури документів, що дозволяє легко адаптувати схеми даних під потреби системи. Для отримання пов'язаних даних між колекціями використовується метод `populate()` бібліотеки Mongoose, який трансліюється в операцію агрегації `$lookup` на рівні MongoDB, забезпечуючи ефективну вибірку без надлишкового дублювання даних. У базі даних зберігаються всі основні сутності платформи, профілі користувачів, каталоги фільмів, сеанси, замовлені квитки та історія чатів. MongoDB забезпечує високу швидкість читання та запису даних, що є критично важливим для стабільної роботи системи під час масового бронювання квитків [25, 27].

Взаємодія сервера з базою даних реалізована через Mongoose ODM. Ця бібліотека дозволяє будувати чіткі об'єктно-орієнтовані схеми для документів безпосередньо в коді програми, забезпечує автоматичну валідацію полів перед збереженням і надає зручний інтерфейс для виконання запитів.

Клієнтську частину побудовано на бібліотеці React. Головним плюсом є компонентний підхід, який дозволяє створити елементи інтерфейсу (наприклад, картку фільму чи інтерактивне крісло в залі) один раз, а далі

		Сівченко М.Т.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		23

використовувати їх на різних сторінках проєкту. Це значно спрощує підтримку коду та прискорює розробку. Завдяки технології Virtual DOM, React оновлює інтерфейс динамічно, без повного перезавантаження сторінок у браузері, що забезпечує плавну роботу сайту [20].

Для розгортання платформи та налаштування середовища використовується технологія контейнеризації Docker. Вона дозволяє запакувати вебзастосунок разом з усіма його залежностями, системними бібліотеками та конфігураційними файлами в один ізольований контейнер. Це повністю вирішує проблему сумісності середовищ, гарантуючи, що система працюватиме однаково стабільно як на локальному комп'ютері розробника, так і на віддаленому сервері [26].

Для забезпечення стабільної роботи вебплатформи AbsoluteCinema у розгорнутому вигляді було визначено такі системні вимоги до апаратного та програмного забезпечення:

Мінімальні вимоги для розгортання системи:

- Оперативна пам'ять: 8 ГБ (рекомендовано 16 ГБ)
- Процесор: 4-ядерний (рекомендовано 8 ядерний)
- Мінімальний обсяг у 50 ГБ включає: сховище медіафайлів каталогу (постери, трейлери), системні логи PM2, резервні копії mongodump та тимчасові шари Docker-контейнерів. Рекомендований обсяг 100 ГБ на SSD забезпечує запас для масштабування медіабібліотеки протягом першого року експлуатації.
- Мережа: Стабільне з'єднання 100 Мбіт/с (рекомендовано виділений канал 1 Гбіт/с)
- Операційна система: Linux (Ubuntu 22.04 LTS) або Windows Server 2019+ (рекомендовано Docker Compose, Nginx (як Reverse Proxy))

Обраний стек технологій є оптимальним для бакалаврського проєкту, оскільки він забезпечує високу продуктивність, швидку розробку та дозволяє без зайвих складнощів масштабувати платформу в майбутньому. Вибір інструментального стеку для розробки вебплатформи визначається

		Сівченко М.Т.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремов Ю.М.				24
Змн.	Арк.	№ докум.	Підпис	Дата		

функціональними вимогами, масштабованістю та підтримкою сучасних стандартів вебтехнологій [16].

Окрім суто технічних переваг, цей набір інструментів значно спрощує підтримку та розвиток програми по мірі її зростання. Завдяки використанню документ-орієнтованої бази даних MongoDB та моделюванню схем через Mongoose, структуру даних можна гнучко змінювати та розширювати без зупинки роботи всієї вебплатформи. Це дозволяє впроваджувати новий функціонал та повністю виключає ризик втрати чи пошкодження персональних даних користувачів під час оновлення системи.

Для фронтенду було обрано бібліотеку React, яка у зв'язці з інструментом збірки Vite забезпечує миттєве оновлення інтерфейсу в процесі розробки (Hot Module Replacement) та швидку генерацію фінальних бандлів. Це робить процес тестування інтерфейсу та внесення змін максимально оперативним.

Описаний комплексний підхід від вибору технологічної платформи Node.js до контейнеризації за допомогою Docker робить архітектуру AbsoluteCinema гнучкою та відмовостійкою. Завдяки цьому вебплатформу можна легко масштабувати, швидко оновлювати й розгортати на новому серверному обладнанні без тривалого ручного налаштування середовища, що є критично важливою умовою для сучасних вебсервісів.

Чіткий поділ обов'язків між модулями системи гарантує стабільність платформи при зростанні кількості активних користувачів. Оскільки кожен компонент моноліту відповідає за власну ізольовану частину роботи, розробник може оптимізувати продуктивність окремих сервісів (наприклад, модуля бронювання або чату), не зачіпаючи при цьому інший код програми. Така прозора та зрозуміла структура проєкту значно спрощує подальший розвиток системи.

У перспективі це дозволить повністю зосередитися на розширенні функціоналу платформи наприклад, на інтеграції нових платіжних шлюзів, удосконаленні системи персональних рекомендацій чи розширенні

		Сівченко М.Т.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремов Ю.М.				25
Змн.	Арк.	№ докум.	Підпис	Дата		

інтерактивних можливостей чату, не витрачаючи час на боротьбу з технічними обмеженнями чи конфліктами залежностей. У підсумку, такий збалансований архітектурний та технологічний вибір дозволив створити якісний, цілісний та готовий до реального використання програмний продукт у межах випускної кваліфікаційної роботи бакалавра.

Висновки до першого розділу

У межах першого розділу було окреслено ключову мету дослідження та деталізовано план розробки вебплатформи AbsoluteCinema. Дослідження специфіки обраної ніші дозволило чітко сформулювати функціональні межі майбутнього продукту, де основними опціями визначено ведення актуального розкладу кіносеансів, інтерактивну взаємодію з мапою глядацького залу, безпосередню генерацію квитків під час їх придбання, а також оперативну комунікацію користувачів із технічною підтримкою.

Шляхом порівняльного аналізу діючих вебсервісів-аналогів вдалося виявити їхні технологічні обмеження та переваги. Систематизація цих даних допомогла знайти оптимальні інструменти для підвищення конкурентоспроможності власної розробки.

Окреме місце в роботі посідає теоретичне обґрунтування вибору системної архітектури. З огляду на специфіку навантажень, пріоритет було віддано модульному моноліту. Такий підхід поєднує простоту відлагодження, швидкість релізу та низькі затримки при міжмодульній взаємодії, але водночас створює надійну базу для подальшого масштабування системи шляхом потенційного відокремлення автономних сервісів.

На основі сформованих архітектурних вимог було затверджено фінальний інструментарій розробника, середовище виконання Node.js, клієнтська бібліотека React, NoSQL-рішення MongoDB для гнучкого збереження документів, а також протокол WebSockets для миттєвого оновлення стану системи.

		Сівченко М.Т.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		26

РОЗДІЛ 2. ПРОЄКТУВАННЯ СИСТЕМИ БРОНЮВАННЯ КВИТКІВ У КІНОТЕАТР

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

Ключовим етапом проєктування вебплатформи AbsoluteCinema є побудова архітектури, здатної витримувати пікові навантаження під час масового продажу квитків та оперативно реагувати на дії відвідувачів. Для забезпечення безпеки, розмежування рівнів доступу до даних та ефективного адміністрування контенту в системі реалізовано гнучку модель керування правами користувачів [5].

Загалом у структурі взаємодії із системою виділено чотири базові функціональні ролі: Гість, зареєстрований користувач, адміністратор, дистриб'ютор. Для анонімних відвідувачів та постійних клієнтів сервісу платформа є центральним інформаційним простором для взаємодії з кінотеатром. До моменту авторизації в системі гостям залишається доступним широкий перелік ознайомчих та пошукових опцій. Усі базові можливості, які не потребують створення облікового запису, деталізовано у таблиці 2.1. Таким чином, реалізований функціонал забезпечує гостям безперешкодний доступ до актуального репертуару та інформації про умови надання послуг, що сприяє залученню нових відвідувачів до користування сервісом

Таблиця 2.1 Доступний функціонал платформи для неавторизованих відвідувачів.

Таблиця 2.1.

Доступний функціонал платформи для неавторизованих відвідувачів

Найменування функції	Опис та технічні особливості реалізації
Оперативний пошук контенту	Знаходження потрібних кінострічок за назвою, акторським складом або ключовими словами.

продовж. табл.2.1

Багаторівневе сортування	Фільтрація списку фільмів за категоріями, жанрами, роком виробництва та глядацьким рейтингом.
Ознайомлення з медіакарткою	Перегляд детальної інформації про фільм, тривалість, опис сюжету та діючий розклад сеансів.
Інтерактивний перегляд медіа	Відтворення офіційних відеоінформаційних трейлерів у модальних вікнах без перезавантаження сторінки.

Успішне проходження процедури автентифікації або реєстрація нового облікового запису відкриває клієнту доступ до персоналізованих сервісів платформи. Авторизований користувач отримує можливість повноцінно взаємодіяти з інтерактивними інструментами кінотеатру та керувати власними замовленнями [1, 2]. Повний перелік розширених можливостей користувача наведено в таблиці 2.2.

Таблиця 2.2.

Додатковий функціонал для авторизованих клієнтів

Компонент системи	Опис функціональних можливостей користувача
Керування квитками	Перегляд інтерактивної схеми залу, вибір доступних місць, онлайн-бронювання та купівля квитків на обрані сеанси.
Особистий кабінет	Доступ до електронних квитків, збереження унікальних кодів для сканування на вході та перегляд історії попередніх замовлень.
Зворотний зв'язок	Можливість виставляти оцінки переглянутим кінострічкам.

		Сівченко М.Т.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремов Ю.М.				28
Змн.	Арк.	№ докум.	Підпис	Дата		

продовж. табл.2.2

Зв'язок з адміністрацією	Надсилання офіційних звернень, запитів або скарг до служби підтримки кінотеатру та ведення діалогу у чаті в режимі реального часу.
--------------------------	--

Роль Дистриб'ютора є спеціалізованим рівнем доступу, створеним виключно для офіційних партнерів кінотеатру представників кінопрокатних компаній та правовласників. Цей статус надається адміністрацією платформи для забезпечення прозорості у відносинах із постачальниками контенту. На відміну від внутрішнього персоналу, дистриб'ютор не має прав на редагування каталогу чи втручання в налаштування сеансів, а виконує суто контролюючу та аналітичну функцію. Повний перелік його можливостей наведено в таблиці 2.3.

Таблиця 2.3.

Функціональні можливості для дистриб'юторів

Напрямок діяльності	Опис функцій та інструментів дистриб'ютора
Аналітика продажів	Перегляд детальної статистики щодо кількості реалізованих квитків на фільми власного виробництва або прокату.
Моніторинг у реальному часі	Відстеження заповнюваності глядацьких залів під час проведення сеансів, де демонструються його кінострічки.
Фінансова звітність	Отримання автоматично згенерованих звітів про касові збори (Box Office) за визначені проміжки часу (день, тиждень, весь період прокату).
Оцінка популярності	Перегляд узагальнених показників комерційного успіху стрічки для планування подальшої маркетингової стратегії та умов співпраці.

Головними вимогами до клієнтського інтерфейсу вебплатформи є візуальний мінімалізм, висока швидкість завантаження графічних елементів

		Сівченко М.Т.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремов Ю.М.				29
Змн.	Арк.	№ докум.	Підпис	Дата		

та інтуїтивно зрозуміла навігація, яка дозволяє придбати квиток за мінімальну кількість кроків.

Для забезпечення повноцінної життєдіяльності сервісу, оперативного наповнення бази даних актуальним кіноконтентом та підтримки порядку в комунікаційних шлюзах розроблено єдину Адміністративну панель. Оскільки в системі керування не передбачено поділу на молодший персонал, наявний адміністратор володіє абсолютними правами та виконує повний спектр менеджерських і контролюючих функцій [5]. Структуру повноважень адміністратора платформи систематизовано в таблиці 2.4.

Таблиця 2.4.

Функціональні обов'язки адміністратора системи

Напрям адміністрування	Конкретні завдання та зони відповідальності
Контент-менеджмент	Повне наповнення каталогу кінокартин, додавання нових фільмів, описів, завантаження постерів та трейлерів, оновлення поточного розкладу сеансів.
Керування інфраструктурою залів	Конфігурування інтерактивних схем глядацьких залів, встановлення вартості квитків залежно від сектору або часу показу.
Модерація спільноти	Контроль за текстовим наповненням сайту, перевірка та видалення користувацьких відгуків чи коментарів, які порушують правила, очищення системи від спаму.
Робота з користувачами	Керування обліковими записами, аналіз активності відвідувачів, блокування користувачів, що систематично порушують правила та надання спеціальних статусів дистриб'юторам.

Служба підтримки	Обробка прямих звернень і скарг від клієнтів, ведення діалогу в адміністративному чаті для вирішення технічних чи організаційних питань [2].
------------------	--

Модуль керування для адміністратора спроектовано за принципами мінімалізму, де за зовнішньою лаконічністю елементів приховано високу функціональність панелі моніторингу. Інтерфейс навмисно звільнено від складних і перевантажених корпоративних таблиць, що дозволяє менеджеру фокусуватися на виконанні поточних завдань. Особливу увагу при розробці було приділено забезпеченню швидкого доступу до ключових метрик системи, системних логів та інструментів оперативного редагування контенту.

Така організація робочого простору дозволяє адміністратору миттєво реагувати на критичні сповіщення, виявляти технічні збої на ранніх етапах за допомогою аналізу журналів помилок, а також оперативно опрацьовувати вхідні запити від відвідувачів платформи. У підсумку, спрощений візуальний стиль у поєднанні з централізованим збором аналітики підвищує швидкість адміністрування та гарантує безперебійну роботу всього вебзастосунку.

Характеристика об'єкта комп'ютеризації

Об'єктом автоматизації у межах випускної кваліфікаційної роботи бакалавра є операційні процеси розкладу сеансів, інтерактивного вибору глядацьких місць, онлайн-продажу квитків, а також комунікаційна взаємодія відвідувачів із адміністрацією кінотеатру.

Вебплатформа AbsoluteCinema вирішує проблему розрізненості сервісів завдяки повній централізації всіх операційних процесів у межах єдиної інформаційної системи. Вона консолідує в собі актуальну афішу, інтерактивні інструменти вибору місць, модулі онлайн-оплати, інтерфейси для збору комерційної аналітики дистриб'юторів, а також цифрові канали прямого зв'язку клієнтів з адміністрацією.

		Сівченко М.Т.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		31

Для забезпечення коректної взаємодії користувачів та автоматизації логіки кінотеатру в базі даних проєкту моделюються ключові сутності предметної області, детальна специфікація яких (типи полів, зв'язки, обмеження) наведена у таблиці 2.5:

Користувач зберігає облікові дані відвідувачів, їхні ролі, профілі та налаштування доступу.

Фільм містить повну медіаінформацію про кінострічки, постери, ліцензійні умови та трейлери.

Сеанс пов'язує конкретний фільм із часом показу, кінозалом та поточною матрицею зайнятих і вільних крісел.

Квиток фіксує факт транзакції, обране місце, унікальний код для перевірки на вході та статус оплати.

Таблиця 2.5.

Функціональні та нефункціональні вимоги

Категорія вимог	Підкатегорія / Напрямок	Зміст технічної вимоги до вебплатформи
Функціональні вимоги	Взаємодія з профілем клієнта	Реєстрація нових користувачів, автентифікація та відновлення доступу.
	Операції з квитками та залами	Перегляд інтерактивної схеми кінозалу в реальному часі. Вибір посадочних місць та онлайн-бронювання квитків на сеанси. Електронна оплата квитків із генерацією унікального коду контролю.
	Робота з каталогом (афішею)	Пошук кінострічок за ключовими текстовими запитами. Фільтрація фільмів за жанровими категоріями, датами та рейтингом. Перегляд карток фільмів, описів, постерів та відеотрейлерів.

продовж. табл.2.5.

	Зворотний зв'язок та комунікація	Виставлення оцінок фільмам. Миттєвий обмін повідомленнями в чаті технічної підтримки. Надсилення клієнтських запитів чи скарг до адміністратора.
	Адміністрування та аналітика	Додавання, оновлення параметрів та видалення фільмів чи сеансів. Модерація призначеного для користувачів контенту. Керування користувацькими обліковими записами та їх блокування. Перегляд дистриб'юторами аналітики касових зборів і заповнюваності залів.
Нефункціональні вимоги	Архітектурні та технологічні рішення	Backend побудовано на базі платформи Node.js та фреймворку Express. Frontend реалізовано за допомогою бібліотеки React та збирача Vite. Робота в реальному часі (чати, схема залу) базується на WebSockets. Збереження інформації забезпечує NoSQL СУБД MongoDB через Mongoose ODM.
	Інфраструктура та розгортання	Контейнеризація всіх сервісів додатка засобами Docker та Docker Compose. Використання вебсервера Nginx як зворотного проксі (Reverse Proxy). Ведення журналів логування системних подій та критичних помилок.
	Швидкодія та надійність	Асинхронна архітектура серверної частини для обробки багатьох запитів. Реалізація постраничного виведення (пагінації) для списків даних. Автоматична перевірка валідності схем даних на рівні ODM.

продовж. табл.2.5.

	Забезпечення безпеки	Авторизація сесій користувачів за допомогою токенів JWT. Криптографічне хешування паролів користувачів перед збереженням у БД. Захист форм авторизації та реєстрації через Google reCAPTCHA. Налаштування HTTPS-шифрування, політик CORS та обмеження прав за ролями.
	Інтерфейсні стандарти	Повна підтримка української мови інтерфейсу та кодування Unicode. Обмін даними між клієнтом і сервером у структурованому форматі JSON. Проектування архітектури API за стандартами RESTful із коректними HTTP-статусами.

Процеси реєстрації та конфігурування облікового запису користувача. Нові відвідувачі платформи мають можливість створити персональний профіль за допомогою класичної реєстрації через електронну пошту [17]. Успішний вхід до системи знімає обмеження гостьового режиму та відкриває повний доступ до транзакційного й комунікаційного функціоналу сайту. Головним елементом кабінету є розділ придбаних квитків, де зберігаються активні замовлення з унікальними кодами для контролю на вході до кінозалу, а також архів попередніх відвідувань кінотеатру. Загальну логіку взаємодії користувача із цим модулем відображено на діаграмі варіантів використання (див. рис. 2.1).

Взаємодія з каталогом та навігація. Пошук необхідних кінострічок реалізовано через інтерактивний рядок запитів або за допомогою багаторівневої системи фільтрації, яка сортує діючу афішу за жанрами, датами сеансів або поточним рейтингом. Картка кожного фільму містить вичерпну інформацію про тривалість, акторський склад, опис сюжету та відеотрейлер.

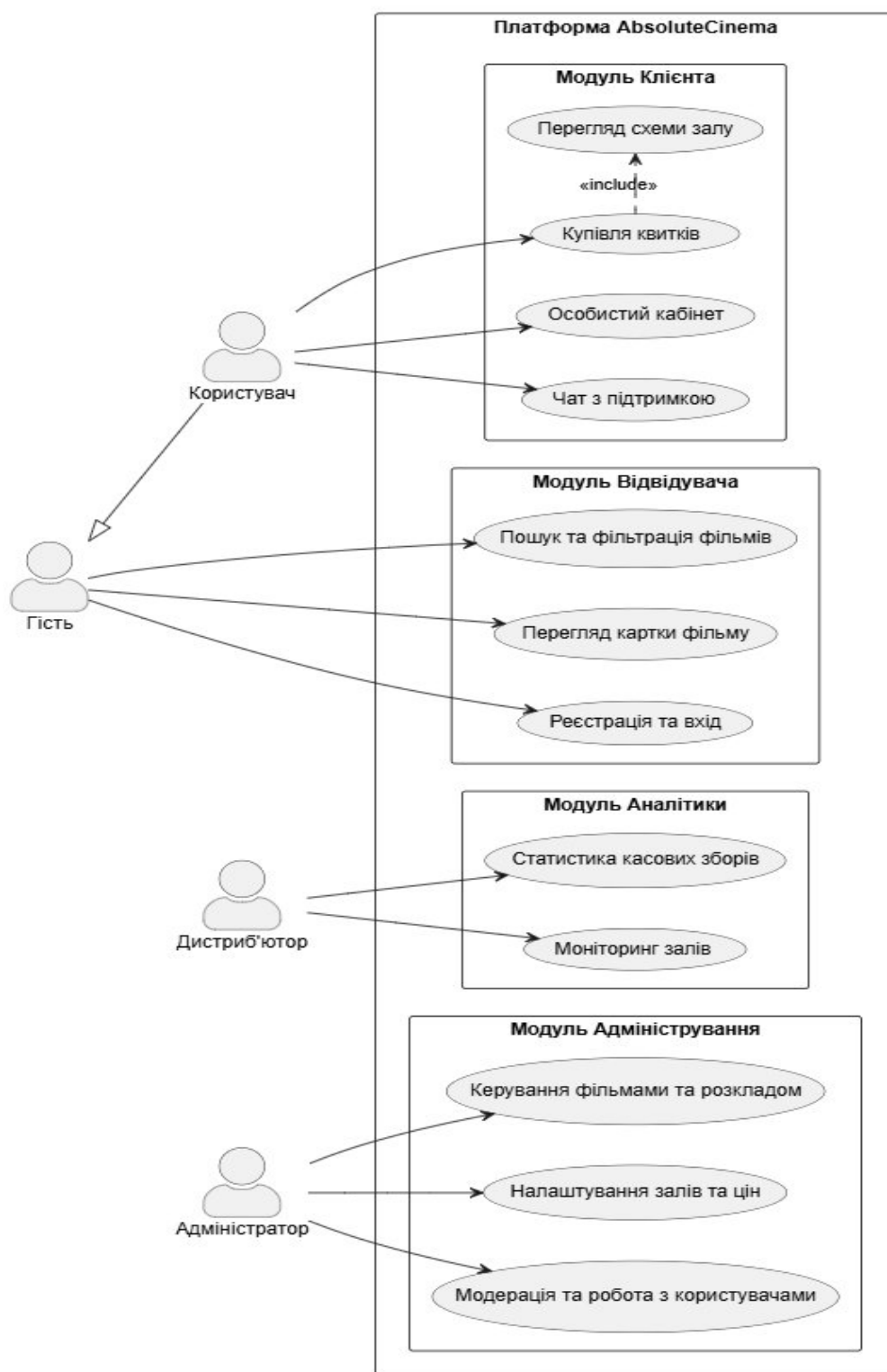


Рисунок 2.1 - Діаграма варіантів використання системи

Транзакційний та комунікаційний функціонал. Після вибору фільму та сеансу користувач переходить до інтерактивної карти залу, де в режимі реального часу відображається поточний стан посадочних місць. Система дозволяє забронювати або відразу оплатити обрані крісла, мінімізуючи час на оформлення замовлення. Окрім транзакційної складової, авторизовані клієнти

формують загальний глядацький рейтинг кінострічок за допомогою виставлення оцінок. Для розв'язання організаційних або технічних питань користувачі можуть оперативно зв'язатися з адміністрацією кінотеатру через вбудований текстовий чат підтримки, який працює без затримок.

Інструменти контролю за користувацьким контентом. У межах розробленої архітектури адміністратор платформи виконує функції нагляду за інформаційним наповненням. Окремим блоком роботи є опрацювання прямих клієнтських скарг та модерація текстових повідомлень в онлайн-чаті підтримки для забезпечення конструктивного діалогу та дотримання правил безпеки у вебзастосунку.

Модуль керування та конфігурування бази даних. Адміністратору надано повний набір інструментів для оперативного ведення та актуалізації інформаційних масивів у СУБД MongoDB. Використовуючи інтерфейс адміністративної панелі, менеджер самостійно створює, редагує або видаляє картки фільмів, завантажує графічні постери, оновлює посилання на офіційні відеотрейлери, а також формує розклад кіносеансів. Логіка взаємодії адміністратора з базою даних розроблена таким чином, щоб зміни миттєво відображалися на стороні клієнтського інтерфейсу, мінімізуючи ризик виникнення помилок при бронюванні глядачами місць у кінозалах.

Система контролю облікових записів та рольового доступу. Адміністративна панель надає керівнику платформи абсолютні повноваження щодо моніторингу та моделювання профілів користувачів. Менеджер має можливість оперативно коригувати рівні доступу (наприклад, надавати верифікованим партнерам спеціалізований статус дистриб'ютора), переглядати поточний стан реєстрацій, а також застосовувати санкції у вигляді тимчасового або повного блокування облікових записів тих відвідувачів, які порушують правила кінотеатру.

Наступним логічним етапом проєктування, після деталізації функціональних ролей та побудови сценаріїв використання, є розробка об'єктно-орієнтованої моделі вебплатформи AbsoluteCinema. Ця структура

		Сівченко М.Т.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Сфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		36

описується за допомогою діаграми класів (Class Diagram) за стандартом UML. Вона дозволяє наочно продемонструвати внутрішній устрій програмного забезпечення, визначити склад базових сутностей, їхні атрибути, методи, а також логічні та архітектурні зв'язки, які забезпечують взаємодію між компонентами системи.

База даних вебплатформи AbsoluteCinema містить чотири основні блоки для інформації. Перший блок відповідає за автентифікацію, авторизацію та комунікацію, забезпечуючи роботу користувацьких профілів на основі схеми користувачів, безпечний розподіл прав доступу за ролями, а також функціонування чатів підтримки через систему тікетів, де повідомлення зберігаються у вигляді вбудованих субдокументів. Другий блок формує каталог медіаконтенту та інформаційного наповнення, акумулюючи вичерпні відомості про кінострічки, включаючи їхні технічні параметри, описи, накопичені метрики глядацького рейтингу, посилання на постери й трейлери, а також загальну стрічку новин платформи. Третя категорія описує планування сеансів та інфраструктуру кінотеатру, де схеми залів виступають шаблонами для конфігурування геометрії приміщень, а колекція сеансів у реальному часі оперує динамічною матрицею глядацьких місць із фіксацією їхнього поточного статусу, цінових категорій та часових лімітів на бронювання. Завершальним блоком призначено фіксацію транзакцій та збір зворотного зв'язку, який містить інформацію про куплені квитки з унікальними кодами замовлень, записи виставлених оцінок для формування медіареєтингу, а також підписки користувачів на повідомлення про старт показів очікуваних стрічок.

Усі ці дані пов'язані між собою трьома способами на рівні об'єктно-орієнтованого проектування схем. Вкладений масив або жорсткий зв'язок означає, що залежна частина є вкладеним субдокументом і не може існувати без цілого. Тому при видаленні сеансу або тікета чату з бази даних автоматично зникнуть усі вбудовані в них посадочні місця чи текстові повідомлення та увесь їхній внутрішній вміст. Агрегація чи м'який зв'язок за посиланнями об'єднує незалежні об'єкти за допомогою ідентифікаторів, тому

		Сівченко М.Т.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		37

якщо термін прокату фільму завершиться і адміністратор змінить його статус, дані про саму кінокартину не зникнуть, а видалення або перенесення сеансу ніяк не вплине на збережені в базі даних квитки, що критично для фінансової звітності. Останній тип зв'язку це асоціація, яка вказує на авторство або операційне володіння ресурсом. Вона визначає, чи є конкретний користувач творцем повідомлень у тікеті, власником придбаних квитків, автором виставлених оцінок у рейтингу або правовласником, закріпленим за карткою фільму.

Рольові моделі та поточні статуси об'єктів у межах вебплатформи AbsoluteCinema мають чітке розмежування та безпосередньо керують логікою виконання операцій на сайті. Користувачі системи поділяються на чотири базові категорії, кожна з яких володіє власним рівнем доступу до колекцій бази даних, анонімні гості, авторизовані покупці, верифіковані дистриб'ютори контенту та системні адміністратори. Кожна медіакартка в інформаційному каталозі містить булевий прапорець прокат Status, який визначає, чи демонструється стрічка в кінотеатрі на даний момент, чи її показ уже завершено.

Для транзакційного модулю купівлі квитків передбачено динамічну систему статусів посадочних місць у залі, де кожне крісло може перебувати у стані вільного, заблокованого на час оформлення транзакції або успішно заброньованого іншим клієнтом. Самі згенеровані фінансові талони оперують станами, що вказують на успішну оплату, резервування без проведення платежу або повернення коштів у разі скасування сеансу. Надіслані клієнтами електронні квитки на рівні контролю доступу маркуються як використані або активні.

Схожий принцип поетапної зміни станів реалізовано в системі зворотного зв'язку та технічної підтримки через чат-тікети. Створювані користувачами запити до адміністрації проходять життєвий цикл від відкритого звернення, яке очікує на реакцію персоналу, до стану перебування

		Сівченко М.Т.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		38

в процесі обробки конкретним менеджером та фінального закриття після вирішення питання.

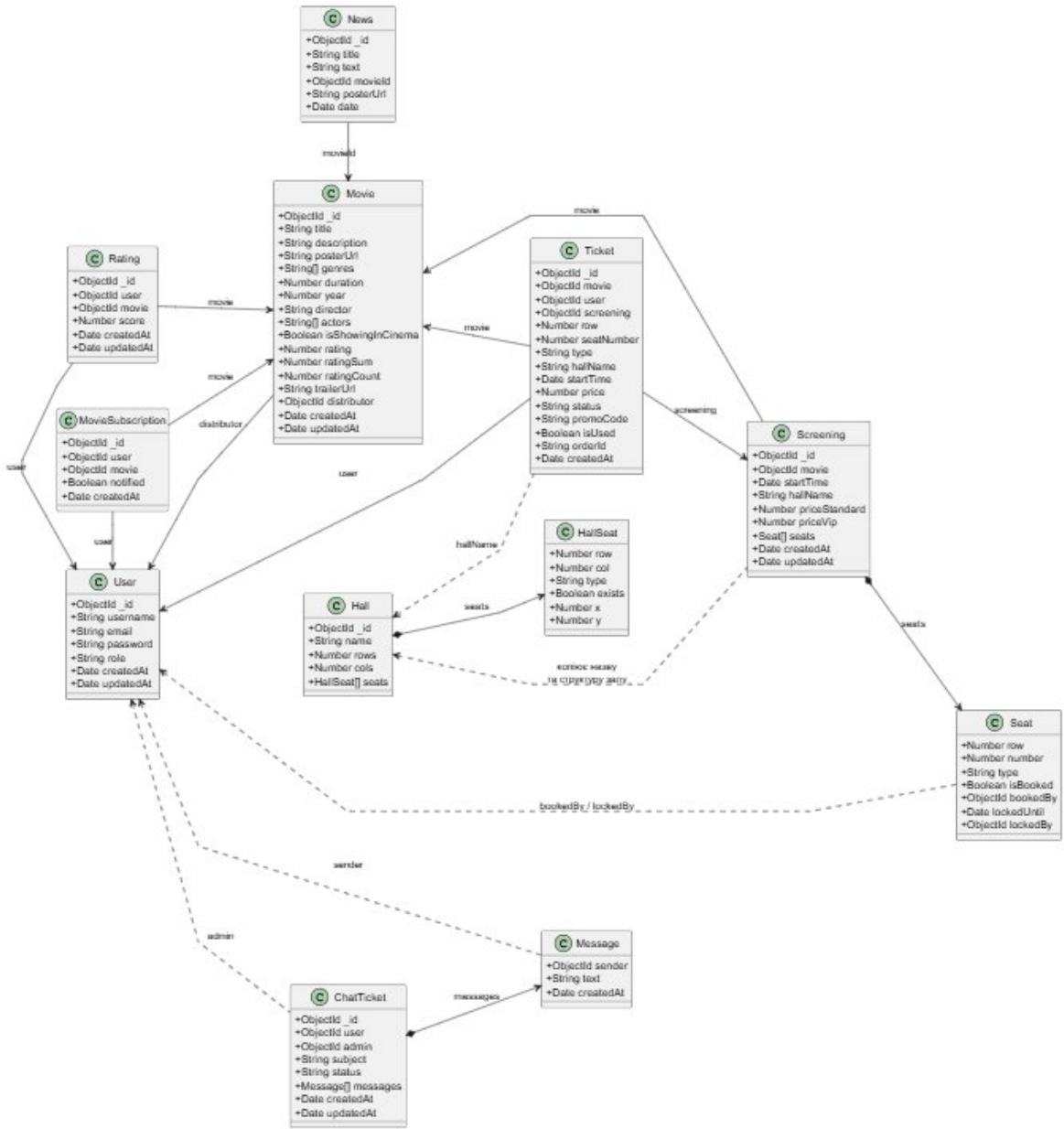


Рисунок 2.2 - Діаграма класів

Для детального відображення внутрішньої структури даних та об'єктної архітектури розроблюваної системи було побудовано об'єктно-орієнтовану модель. На рис. 2.2 наведено детальну діаграму класів доменних моделей вебплатформи AbsoluteCinema. Ця схема містить вичерпний перелік усіх ключових сутностей бази даних, їхні внутрішні поля, типи даних, перелічення (enum), а також архітектурні зв'язки, що визначають логіку взаємодії компонентів при обробці транзакцій, бронюванні місць та веденні чатів

підтримки. З метою забезпечення високої читабельності та спрощення візуального сприйняття загальної концепції, взаємозв'язки між основними колекціями MongoDB розширено за допомогою логічних залежностей, які наочно демонструють процеси денормалізації та копіювання шаблонів записів під час генерації активних кіносеансів.

2.2. Розробка бази даних системи

Організація колекцій NoSQL-бази даних та визначення зв'язків між документами є ключовим етапом проєктування інформаційної системи [6]. Логічний устрій бази даних розроблявся на основі раніше затверджених Mongoose-моделей і повністю охоплює весь функціонал онлайн-кінотеатру.

Під час формування схем колекцій окремий акцент аналізу було зміщено на організацію взаємодії між документами MongoDB. Для цього використовувалися два підходи, інтеграція вкладених субдокументів та перехресні посилання за допомогою ObjectId. Жорсткі композиційні зв'язки (наприклад, масиви кірсел у сеансах) реалізовані всередині єдиного батьківського документа. Такий крок унеможливорює появу інформаційного сміття, оскільки очищення базової сутності автоматично видаляє всі пов'язані з нею дані на рівні диска. З іншого боку, транзакційні блоки та метрики зворотного зв'язку (куплені квитки, оцінки фільмів) використовують незалежний тип зв'язку, агрегацію. Якщо адміністратор змінить картку кінокартини чи оновить профіль відвідувача, фінансові показники та статистика зборів для дистриб'юторів не зазнають викривлень, що гарантує точність комерційного обліку [10].

Увесь масив інформації вебплатформи AbsoluteCinema акумулюється всередині єдиного NoSQL-середовища. Воно об'єднує облікові профілі відвідувачів, детальні медіакартки кінострічок, розклад сеансів із геометрією залів, а також зворотний зв'язок від аудиторії у вигляді балів рейтингу. Організація колекцій MongoDB дозволяє без затримок перевіряти цифрові токени під час входу на сайт, розмежовувати повноваження між покупцями,

		Сівченко М.Т.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремов Ю.М.				40
Змн.	Арк.	№ докум.	Підпис	Дата		

адмінами й дистриб'юторами за допомогою рольових маркерів, а також фіксувати унікальні ідентифікатори куплених квитків для контролю глядачів на вході. Внутрішня архітектура змонтованих інформаційних потоків відображена на схемі, що розташована нижче. Зважаючи на велику кількість специфічних полів Mongoose-моделей, внутрішніх переліків (enum) та перехресних індексів, розгорнуту графічну інтерпретацію взаємодії колекцій з усіма технічними атрибутами перенесено до фінальної частини пояснювальної записки (див. рис. додаток А).

Фундаментом архітектури збереження даних у проєкті є колекція користувачів. У межах єдиного документа зберігаються автентифікаційні параметри, захешовані захисні ключі та базові ідентифікаційні налаштування облікового запису. Консолідація цих атрибутів всередині однієї моделі дозволяє вебзастосунку миттєво проводити верифікацію клієнта під час авторизації без виконання високовартісних операцій злиття даних із суміжних джерел. Рівні доступу до функцій адміністрування чи касової аналітики визначаються за допомогою спеціального текстового індикатора обов'язків. Усі інші процеси взаємодії такі як генерація куплених квитків, фіксація глядацьких оцінок чи створення діалогів у техпідтримці реалізовані через окремі сутності, які містять прямі посилання на унікальні ідентифікатори профілів. Завдяки такій структурі бекенд безпомилково визначає ініціатора кожної операції на сайті, що унаочнює загальна схема логічного розподілу даних у додатках до роботи.

Каталог доступного відеоматеріалу формується навколо колекції кінострічок. Схема цієї моделі спроектована універсальною, що дозволяє зберігати базовий опис, тривалість, жанрову приналежність та посилання на медіафайли безпосередньо в одному документі. Паралельно з інформаційним описом фільмів функціонують пов'язані транзакційні підсистеми сеансів та бронювання місць, які відповідають за інтерактивне відображення вільних крісел у залі. Також у системі передбачено окремі колекції для публікації новин, фіксації глядацьких балів та оформлення підписок на заплановані

		Сівченко М.Т.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремов Ю.М.				41
Змн.	Арк.	№ докум.	Підпис	Дата		

прем'єри, що забезпечує незалежне масштабування кожного інформаційного блоку.

Для фіксації зворотного зв'язку та формування медіареєтингу кінострічок використовуються незалежні колекції оцінок. Кожен виставлений бал реєструється як окремий документ, який одночасно посилається на унікальний ідентифікатор автора та конкретну картку фільму. Складені унікальні індекси на рівні СУБД унеможливають повторне голосування з одного профілю, індексування полів колекцій є стандартним механізмом прискорення запитів у системах управління базами даних [24], що захищає статистику від штучних накруток. Комунікаційні процеси підтримуються інтерактивним текстовим модулем, інтегрованим безпосередньо у схему звернень. Накопичена у базі даних інформація про куповані квитки, підписки на майбутні релізи та глядацькі оцінки дозволяє системі формувати точні аналітичні масиви даних, які надалі можуть використовуватися для побудови персоналізованих рекомендаційних добірок для відвідувачів [12].

Для технічного супроводу та контент-менеджменту в базі даних виділено окремий функціональний сегмент. Він містить колекції новинних публікацій та тікет-систему підтримки, що відстежує стадії обробки запитів. Взаємодія між доменними схемами Mongoose реалізована через перехресні посилання ObjectId. Проектування схем даних передбачає визначення типів полів, обмежень унікальності та індексів для забезпечення цілісності інформації [9]. Для прискорення вибірки активних звернень користувачів у колекції ChatTicket встановлено складений індекс { user: 1, status: 1 }. Індексування полів колекцій є стандартним механізмом оптимізації запитів у системах управління базами даних [24]. Використання NoSQL-архітектури, де динамічні структури (наприклад, статус бронювання місць) вбудовані безпосередньо в документи сеансів, забезпечує високу гнучкість масштабування та легку інтеграцію майбутніх модулів без порушення логіки наявної системи. Детальна структура цих допоміжних колекцій наведена у таблиці 2.6 – 2.14.

		Сівченко М.Т.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремов Ю.М.				42
Змн.	Арк.	№ докум.	Підпис	Дата		

Таблиця 2.6

Структура колекції сутності User

Назва поля	Тип даних	Тип зв'язку	Опис атрибута
_id	ObjectId	ID	Генеративний системний ідентифікатор документа в межах СУБД.
username	String		Текстове ім'я або нікнейм для автентифікації відвідувача на сайті.
email	String	Унікальний індекс	Електронна поштова адреса клієнта, що слугує логіном для авторизації.
password	String		Криптографічний хеш пароля, створений для безпечного входу.
role	String (Enum)		Повноваження облікового запису (рівні користувача, адміна чи дистриб'ютора).
createdAt	Date		Часова мітка автоматичної фіксації моменту реєстрації профілю.
updatedAt	Date		Штамп часу, який реєструє дату останнього оновлення даних користувача.
isBannedBooking	Boolean		Логічний прапорець блокування доступу користувача до бронювання квитків.
isBannedSupport	Boolean		Логічний прапорець блокування доступу користувача до чату технічної підтримки.
abandonedPaymentsCount	Number		Лічильник кількості незавершених платіжних сесій для виявлення зловживань.
paymentClickTimestamps	Array (Date)		Масив часових міток натискань кнопки оплати для аналізу частоти флуд-активності.

Фундаментом архітектури збереження даних у проєкті є колекція користувачів (User). Вона акумулює всі необхідні відомості для ідентифікації відвідувача, розмежування його прав доступу та моніторингу поведінкової активності на платформі. Структуру колекції детально описано у таблиці 2.6.

Кожен документ колекції містить автентифікаційні параметри (username, email, password) та рольовий індикатор (role), який визначає рівень доступу облікового запису - звичайний користувач, адміністратор або дистриб'ютор. Окрему групу полів складає антифлуд-підсистема: лічильник покинутих платіжних сесій (abandonedPaymentsCount) та масив часових міток (paymentClickTimestamps) дозволяють адміністратору виявляти зловживання та застосовувати точкові санкції - блокування бронювання (isBannedBooking) або доступу до чату підтримки (isBannedSupport) - без повного видалення облікового запису. Часові мітки createdAt та updatedAt генеруються автоматично на рівні Mongoose завдяки опції { timestamps: true }.

Таблиця 2.7

Структура колекції сутності Ticket

Назва	Тип даних	Тип зв'язку	Опис
_id	ObjectId	ID	Унікальний системний маркер конкретної транзакції в базі даних.
movie	ObjectId	Ref	Ідентифікатор-посилання на документ кінострічки з колекції фільмів.
user	ObjectId	Ref	Посилання на обліковий запис клієнта, який здійснив покупку.
screening	ObjectId	Ref	Посилання на конкретний часовий сеанс у розкладі кінотеатру.
seatNumber	Number		Номер посадочного місця на інтерактивній схемі.
row	Number		Числове значення, що визначає ряд обраного крісла в залі.
type	String (Enum)		Категорія комфортності глядацького місця (standard або vip).

продовж. табл.2.7.

hallName	String		Текстова назва кінозалу, де відбуватиметься показ стрічки.
startTime	Date		Зафіксовані дата та точний час початку обраного кіносеансу.
price	Number		Фінансова вартість придбаного квитка.
status	String (Enum)		Поточний стан оплати або резерву (booked, paid, refunded).
promoCode	String		Застосований код для отримання знижки; null якщо не використовувався.
isUsed	Boolean		Логічний маркер проходження контролю на вході до кінозалу.
orderId	String		Автоматично згенерований унікальний код замовлення для швидкого пошуку.
createdAt	Date		Штамп часу, який фіксує точний момент проведення операції.

Транзакційну діяльність платформи фіксує колекція квитків (Ticket). Кожен документ цієї колекції є цифровим підтвердженням факту придбання місця на конкретний сеанс і містить повний набір даних, необхідних як для контролю глядача на вході до залу, так і для формування фінансової звітності дистриб'юторів. Структуру колекції наведено у таблиці 2.7.

Документ квитка пов'язує три незалежні колекції через поля-референції: movie вказує на картку фільму, user - на профіль покупця, screening - на конкретний сеанс із його розкладом та конфігурацією залу. Координати обраного місця (row, seatNumber) та його категорія (type) денормалізовані та збережені безпосередньо в документі квитка це виключає необхідність повторних запитів до колекції сеансів при формуванні звітів. Поле status відстежує життєвий цикл квитка від резервування (booked) через успішну оплату (paid) до можливого повернення (refunded). Унікальний код замовлення (orderId) генерується автоматично на момент створення документа та слугує

		Сівченко М.Т.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Сфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		45

для швидкого пошуку транзакції як адміністратором, так і самим користувачем. Поле promoCode фіксує застосований компенсаційний код знижки, якщо він був використаний під час оформлення замовлення.

Таблиця 2.8

Структура колекції сутності Screening

Назва	Тип даних	Тип зв'язку	Опис
_id	ObjectId	ID	Системний унікальний ідентифікатор конкретного кіносеансу в базі даних.
movie	ObjectId	Ref	Посилання на документ фільму з медіакаталогу, який транслюється.
startTime	Date		Зафіксовані дата та точний час початку демонстрації стрічки.
hallName	String		Текстова назва кінозалу, у якому зарезервовано час для показу.
priceStandard	Number		Базова фінансова вартість квитка на звичайне глядацьке місце.
priceVip	Number		Встановлена ціна квитка на місця підвищеного комфорту (VIP).
seats	Array (Object)	Вкладений масив	Вкладений масив субдокументів, що містить поточний стан усіх крісел залу.
createdAt	Date		Часовий штамп автоматичної реєстрації моменту створення сеансу.
updatedAt	Date		Штамп часу, який відображає дату останнього редагування параметрів показу.

Планування кіносеансів та управління станом глядацького залу в реальному часі забезпечує колекція сеансів (Screening). Вона є центральною операційною сутністю платформи, оскільки саме навколо неї будується весь

процес інтерактивного бронювання місць. Структуру колекції наведено у таблиці 2.8.

Документ сеансу через поле-референцію movie пов'язується з карткою фільму з медіакаталогу, визначаючи яка саме кінострічка демонструватиметься. Часова мітка startTime та назва залу hallName формують унікальний розклад показу. Цінова політика сеансу задається двома полями - priceStandard для звичайних місць та priceVip для місць підвищеного комфорту, що дозволяє гнучко керувати вартістю квитків для різних категорій глядачів.

Ключовим архітектурним рішенням є поле seats - вкладений масив субдокументів, який зберігає повну матрицю крісел залу безпосередньо всередині документа сеансу. Кожен субдокумент містить координати крісла, його тип, статус бронювання (isBooked), ідентифікатор покупця (bookedBy) та часовий ліміт тимчасового блокування (lockedUntil). Завдяки такій структурі сервер отримує повний знімок стану залу за один запит до бази даних без виконання додаткових операцій об'єднання колекцій, що є критично важливим для швидкодії системи під час пікового навантаження при масовому продажу квитків.

Таблиця 2.9

Структура колекції сутності Rating

Назва	Тип даних	Тип зв'язку	Опис
_id	ObjectId	ID	Системний унікальний маркер документа оцінки в межах СУБД.
user	ObjectId	Ref	Ідентифікатор профілю клієнта, який виставив бал фільму.
movie	ObjectId	Ref	Посилання на медіакартку кінокартини, що отримала оцінку.
score	Number		Числове значення виставленого балу в межах встановленого ліміту (від 1 до 10).

продовж. табл.2.9.

createdAt	Date		Штамп часу, який автоматично реєструє момент надсилання голосу.
updatedAt	Date		Дата та точний час внесення останніх змін у виставлену оцінку.

Для формування об'єктивного глядацького рейтингу кінострічок у системі передбачено окрему колекцію оцінок (Rating). Архітектурне рішення винести кожен виставлений бал в окремий документ, а не зберігати оцінки як вкладений масив у картці фільму, зумовлено необхідністю забезпечити унікальність голосування та можливість його подальшого редагування. Структуру колекції наведено у таблиці 2.9.

Кожен документ колекції одночасно посилається на профіль автора оцінки (user) та на картку кінострічки (movie) через поля-референції типу ObjectId. На рівні СУБД для цих двох полів встановлено складений унікальний індекс { user: 1, movie: 1 }, який архітектурно унеможливорює повторне голосування одного користувача за один і той самий фільм. Числове поле score приймає значення від 1 до 10 відповідно до валідаційних обмежень схеми Mongoose. Для оптимізації швидкодії інтерфейсу агреговані показники рейтингу (ratingSum, ratingCount) зберігаються денормалізовано безпосередньо в документі фільму та оновлюються атомарно у момент збереження нової оцінки, що перетворює ресурсомістку агрегацію на просте зчитування двох числових полів [8].

Таблиця 2.10

Структура колекції сутності News

Назва	Тип даних	Тип зв'язку	Опис
_id	ObjectId	ID	Унікальний генеративний маркер новинного документа всередині СУБД.

продовж. табл.2.10.

title	String		Текстовий заголовок публікації, що відображається у новинній стрічці.
text	String		Основне інформаційне наповнення (тіло новини) з детальним описом події.
movieId	ObjectId	Ref	Опціональне посилання на картку фільму, якого стосується анонс.
posterUrl	String		Відносний шлях до графічного файлу обкладинки або прев'ю новини на сервері.
date	Date		Часовий штамп автоматичної фіксації моменту виходу публікації на сайті.
isAlert	Boolean		Логічний прапорець системного сповіщення - визначає чи є новина цільовим повідомленням для конкретних користувачів.
targetUsers	Array (String)		Масив ідентифікаторів користувачів, яким адресовано цільове сповіщення.

Інформаційне наповнення платформи та система персоналізованих сповіщень реалізовані через колекцію новин (News). На відміну від стандартних новинних стрічок, ця колекція виконує подвійну функцію: публікацію загальних маркетингових матеріалів для всіх відвідувачів та адресну доставку системних сповіщень конкретним користувачам. Структуру колекції наведено у таблиці 2.10.

		Сівченко М.Т.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремов Ю.М.				49
Змн.	Арк.	№ докум.	Підпис	Дата		

Кожен документ колекції містить заголовок (title) та основний текст публікації (text), які відображаються у новинній стрічці платформи. Поле movieId є опціональною референцією на картку фільму - воно заповнюється лише для новин, що стосуються конкретної кінострічки, наприклад анонсів прем'єр або змін у розкладі. Поле posterUrl зберігає шлях до графічного файлу обкладинки новини на сервері. Логічний прапорець isAlert визначає режим публікації: якщо його значення true, новина є системним сповіщенням і відображається лише для користувачів, чиї ідентифікатори перелічені у масиві targetUsers. Цей механізм використовується, зокрема, для автоматичного інформування глядачів про скасування сеансу на який вони придбали квитки, або про старт продажу квитків на очікуваний фільм за підпискою.

Таблиця 2.11

Структура колекції сутності MovieSubscription

Назва	Тип даних	Тип зв'язку	Опис
_id	ObjectId	ID	Системний унікальний маркер документа підписки в межах СУБД.
user	ObjectId	Ref	Ідентифікатор профілю відвідувача, який підписався на оновлення стрічки.
movie	ObjectId	Ref	Посилання на картку конкретного фільму, за релізом якого стежить клієнт.
notified	Boolean		Логічний індикатор, який вказує, чи було надіслано сповіщення користувачу.
createdAt	Date		Часовий штамп, що фіксує точний момент створення запиту на підписку.

Для забезпечення пріоритетного інформування глядачів про початок продажу квитків на очікувані прем'єри у системі реалізовано механізм підписок на фільми через колекцію (MovieSubscription). Ця колекція дозволяє

		Сівченко М.Т.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Сфремов Ю.М.				50
Змн.	Арк.	№ докум.	Підпис	Дата		

відвідувачам стежити за анонсованими кінострічками зі статусом «Скоро у прокаті» та автоматично отримувати сповіщення у момент відкриття продажу. Структуру колекції наведено у таблиці 2.11.

Кожен документ колекції фіксує зв'язок між конкретним користувачем (user) та фільмом (movie) через поля-референції типу ObjectId. Логічний індикатор notified є ключовим полем керування життєвим циклом підписки: початкове значення false означає що сповіщення ще не надсилалось, а після успішної доставки системного повідомлення значення змінюється на true, що запобігає повторному надсиланню дублюючих сповіщень одному користувачу. Часова мітка createdAt фіксує момент оформлення підписки та використовується для сортування черги сповіщень у хронологічному порядку. При відкритті продажу квитків на фільм сервер виконує вибірку всіх документів колекції де movie збігається з відповідним ідентифікатором та notified має значення false, після чого формує персоналізовані записи в колекції News з масивом targetUsers для кожного підписника.

Таблиця 2.12

Структура колекції сутності Movie

Назва	Тип даних	Тип зв'язку	Опис
_id	ObjectId	ID	Унікальний системний ідентифікатор картки фільму в базі даних.
title	String		Текстова офіційна назва кінострічки для відображення в каталозі.
description	String		Детальний синопсис або текстовий опис сюжетної лінії фільму.
posterUrl	String		Шлях до графічного файлу офіційного постера на сервері зберігання.

продовж. табл.2.12.

genres	Array (String)		Масив текстових міток, що визначають жанрову приналежність контенту.
duration	Number		Хронометраж кінокартини, зафіксований у хвилинах.
year	Number		Календарний рік виходу кінострічки у прокат.
director	String		Ім'я та прізвище головного режисера-постановника.
isShowingInCinema	Boolean		Логічний прапорець наявності стрічки в поточному прокаті.
rating	Number		Поточний середній бал фільму, розрахований на основі голосів глядачів.
ratingSum	Number		Акумульована сума всіх виставлених балів для розрахунку рейтингу.
ratingCount	Number		Загальна кількість користувачів, які залишили оцінку для цієї стрічки.
trailerUrl	String		Посилання на офіційний відеотрейлер кінокартини.
distributor	ObjectId	Ref	Посилання на профіль правовласника (користувача з роллю дистриб'ютора).
createdAt	Date		Часова мітка автоматичної реєстрації моменту внесення фільму до бази.
updatedAt	Date		Штамп часу останнього редагування картки фільму.

Інформаційний каталог кінострічок платформи формується навколо колекції фільмів (Movie). Вона є центральною медіасутністю системи, до якої звертаються більшість інших колекцій через поля-референції. Структуру колекції наведено у таблиці 2.12.

		Сівченко М.Т.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремов Ю.М.				52
Змн.	Арк.	№ докум.	Підпис	Дата		

Документ фільму акумулює повний набір метаданих кінострічки: описову інформацію (title, description, director), технічні параметри (duration, year), жанрову класифікацію у вигляді масиву текстових міток (genres) та посилання на медіафайли (posterUrl, trailerUrl). Логічний прапорець isShowingInCinema визначає поточний статус прокату стрічки та керує її відображенням в афіші платформи. Поле distributor через референцію ObjectId прив'язує картку фільму до профілю правовласника з роллю дистриб'ютора, що забезпечує коректну фільтрацію контенту в аналітичному кабінеті партнера.

Окрему групу складають поля агрегованого рейтингу: rating зберігає поточний середній бал, тоді як ratingSum та ratingCount є денормалізованими лічильниками, що оновлюються атомарно при кожному новому голосуванні. Такий підхід усуває необхідність виконання ресурсомістких агрегаційних запитів до колекції Rating при завантаженні сторінки фільму, замінюючи їх простим зчитуванням трьох числових полів. Часові мітки createdAt та updatedAt генеруються автоматично на рівні Mongoose завдяки опції { timestamps: true }.

Таблиця 2.13

Структура колекції сутності Halls

Назва	Тип даних	Посилання	Опис
_id	ObjectId	ID	Унікальний системний маркер документа конкретного залу всередині СУБД.
name	String		Текстова унікальна назва кінозалу для відображення в інтерфейсі сайту.
rows	Number		Загальна кількість рядів у глядацькому залі.
cols	Number		Загальна кількість місць у кожному ряду залі.
seats	Array (Object)	Вкладений масив	Вкладений масив субдокументів, який задає сітку, координати та типи фізичних місць.
seats.row	Number		Номер ряду, до якого належить конкретне крісло.

продовж. табл.2.13.

seats.col	Number		Номер колонки (позиція крісла в ряду).
seats.type	String (Enum)		Категорія місця (standard або vip).
seats.exists	Boolean		Логічний прапорець фізичної наявності крісла (для залів з нестандартною геометрією).
seats.x	Number		Координата X для візуального відображення крісла на схемі залу.
seats.y	Number		Координата Y для візуального відображення крісла на схемі залу.

Фізичну інфраструктуру кінотеатру в базі даних представляє колекція залів (Hall). Вона виконує роль шаблонного сховища конфігурацій глядацьких приміщень і слугує основою для генерації динамічних схем місць при створенні нових сеансів. Структуру колекції наведено у таблиці 2.13.

Кожен документ колекції описує конкретний кінозал через його унікальну назву (name) та геометричні параметри сітки (rows, cols), які визначають загальну місткість приміщення. Ключовим елементом схеми є вкладений масив субдокументів seats, що містить повну матрицю фізичних крісел залу. Кожен субдокумент зберігає координати місця в сітці (row, col), його категорію (type), піксельні координати для відображення на інтерактивній схемі (x, y) та логічний прапорець фізичної наявності крісла (exists). Останнє поле є особливо важливим для залів з нестандартною геометрією - наприклад, з проходами або відсутніми місцями в окремих рядах - де exists: false виключає крісло з відображення на схемі без порушення нумерації сусідніх місць.

При створенні нового сеансу адміністратором бекенд копіює масив seats з відповідного документа колекції Hall та додає до кожного субдокумента операційні поля (isBooked, bookedBy, lockedUntil, lockedBy), формуючи динамічну матрицю стану місць для конкретного показу в колекції Screening.

		Сівченко М.Т.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Сфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		54

Таблиця 2.14

Структура колекції сутності Chatticket

Назва	Тип даних	Посилання	Опис
_id	ObjectId	ID	Генеративний системний ідентифікатор документа конкретного звернення в базі даних.
user	ObjectId	Ref	Посилання на обліковий запис клієнта, який створив запит на підтримку.
admin	ObjectId	Ref	Посилання на профіль адміністратора, який взяв чат в обробку.
subject	String		Текстова тема звернення, яка визначає категорію проблеми користувача.
status	String (Enum)		Поточний стан життєвого циклу тикета (open, in_progress, closed).
messages	Array (Object)	Вкладений масив	Вкладений масив субдокументів, який містить хронологічну історію реплік чату.
messages.sender	ObjectId	Ref	Посилання на профіль користувача або адміністратора, який надіслав повідомлення.
messages.text	String		Текстовий вміст повідомлення у діалозі.
messages.createdAt	Date		Часовий штамп моменту надсилання повідомлення.
createdAt	Date		Часовий штамп автоматичної реєстрації моменту відкриття запиту.
updatedAt	Date		Штамп часу, який фіксує дату останнього повідомлення або зміни статусу чату.

Комунікаційну взаємодію між відвідувачами платформи та адміністрацією кінотеатру забезпечує колекція тикетів підтримки (ChatTicket). Кожен документ колекції представляє окреме звернення користувача та містить повну історію діалогу від моменту відкриття запиту до його закриття. Структуру колекції наведено у таблиці 2.14.

		Сівченко М.Т.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		55

Документ тікета через поля-референції пов'язує двох учасників діалогу: користувача (user), який ініціював звернення, та адміністратора (admin), який взяв запит в обробку. Поле admin за замовчуванням має значення null та заповнюється лише після того, як конкретний менеджер натискає кнопку «ВЗЯТИ В РОБОТУ» в адміністративній панелі. Тема звернення (subject) визначає категорію проблеми користувача та за замовчуванням має значення «Загальне питання по роботі сайту». Поле status відстежує життєвий цикл тікета через три стани: open - звернення очікує на відповідь адміністратора, in_progress - діалог активно ведеться, closed - питання вирішено та тікет переведено до архіву.

Хронологічна історія переписки зберігається у вкладеному масиві субдокументів messages. Кожне повідомлення містить референцію на відправника (sender), текстовий вміст (text) та часову мітку (createdAt). Завдяки збереженню повідомлень як вбудованих субдокументів уся історія діалогу отримується за один запит до бази даних без додаткових операцій об'єднання колекцій. На рівні СУБД встановлено складений індекс { user: 1, status: 1 }, що прискорює вибірку активних звернень конкретного користувача [24].

Таким чином, наведені вище таблиці 2.6 – 2.14 детально описують інфологічну модель даних вебплатформи AbsoluteCinema. Використання NoSQL-підходу дозволило гнучко поєднати сувору типізацію ключових бізнес-суттєвостей із динамічними структурами.

2.3. Проєктування та реалізація алгоритмів роботи системи

Логіка функціонування інтерактивних модулів вебплатформи AbsoluteCinema формалізована за допомогою поведінкових моделей. На рис. 2.3 продемонстровано діаграму активності, що описує процес взаємодії відвідувача з медіакаталогом та підсистемою оцінювання. Ключовим етапом алгоритму є верифікація сесії, оскільки доступ до публічних даних є вільним,

		Сівченко М.Т.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		56

тоді як транзакційні дії та формування рейтингу вимагають обов'язкової перевірки прав на рівні бекенду [11].

Алгоритм розпочинається із завантаження головної сторінки застосунку. Користувач використовує форму фільтрації, вказуючи параметри пошуку, такі як назва, рік або жанр. Серверна частина опрацьовує запит, формує вибірку з бази даних MongoDB та повертає фронтенду релевантний перелік фільмів. Після цього відвідувач обирає фільм, що ініціює динамічне завантаження його детального синопсису, технічних характеристик та поточних статистичних показників.

Спроба оцінити фільм активує внутрішній тригер безпеки, який розділяє виконання алгоритму на два сценарії. Якщо JWT-токен відсутній або неактивний, система блокує інтерфейс оцінювання та перенаправляє користувача на сторінку авторизації. Якщо клієнт вже успішно пройшов автентифікацію, доступ до надсилання даних надається миттєво.

Перед записом у NoSQL-сховище сервер виконує технічну валідацію: перевіряє відповідність балу діапазону від 1 до 10 та контролює унікальність голосу для кожного облікового запису. Після валідації дані записуються до колекції ratings, що запускає тригер оновлення медіакаталогу, де здійснюється перерахунок середнього рейтингу кінострічки. Оновлені показники автоматично відображаються в інтерфейсі користувача в реальному часі, завершуючи цикл роботи алгоритму платформи.

Паралельно із процесом оцінювання платформа забезпечує функціонування підсистеми підписок на анонсовані фільми. Якщо для обраної кінострічки відсутні активні сеанси у колекції Screening, користувачеві пропонується оформити підписку через відповідну кнопку на сторінці фільму. При натисканні авторизований клієнт ініціює POST-запит, що створює документ у колекції MovieSubscription із прапорцем notified: false. У момент, коли адміністратор додає новий сеанс для цього фільму, бекенд автоматично виконує вибірку всіх активних підписок та генерує персоналізовані записи у колекції News із масивом targetUsers, що забезпечує доставку push-сповіщення

		Сівченко М.Т.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		57

до особистої новинної стрічки кожного підписника — таким чином цикл взаємодії користувача з медіакаталогом замикається не лише на поточному контенті, а й на майбутніх прем'єрах платформи.

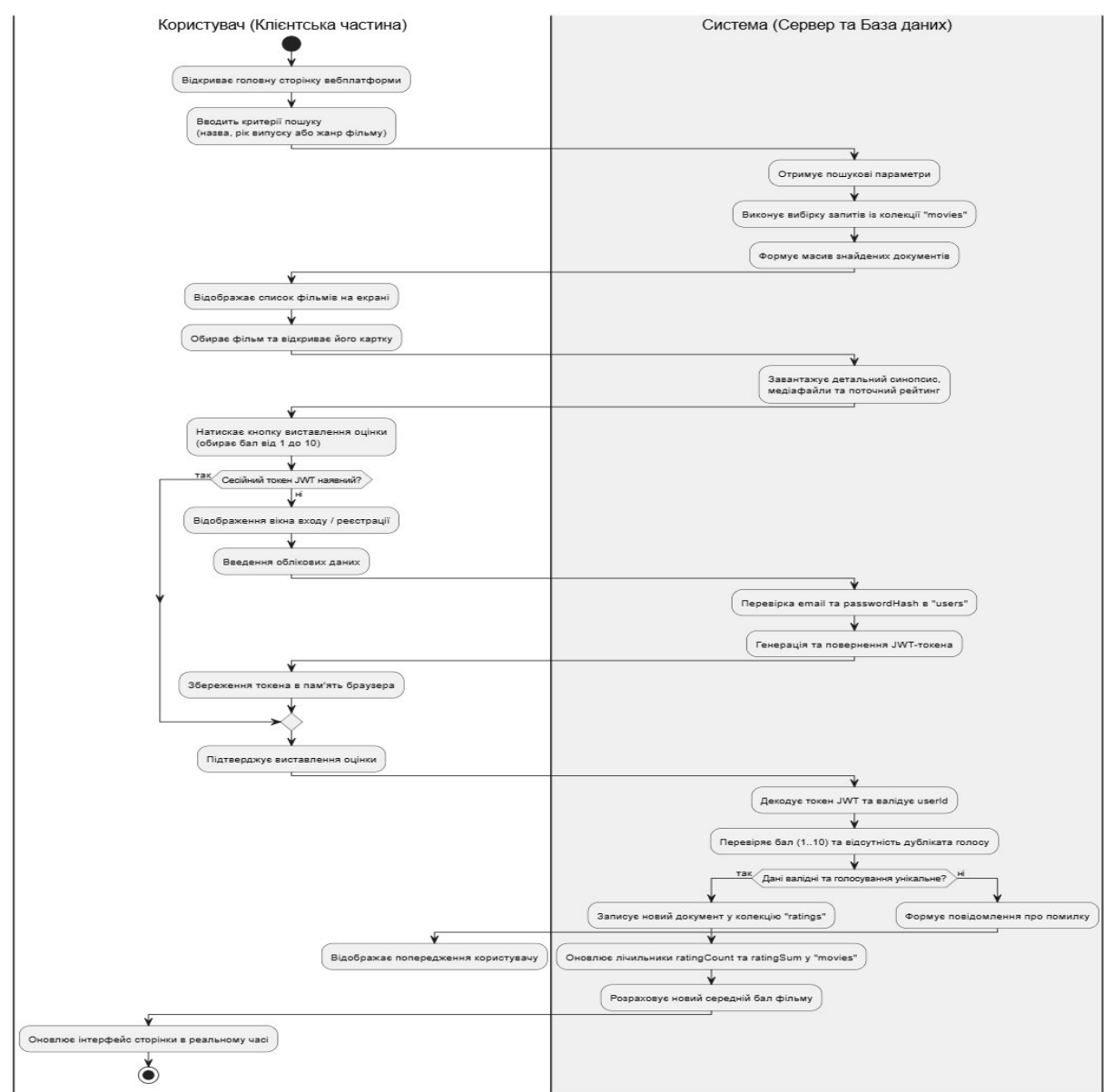


Рисунок 2.3 - Діаграма активності для пошуку фільму та виставлення йому оцінки.

Динаміку взаємодії користувачів із модулем бронювання ілюструє поведінкова модель (рис. 2.4). Процес починається з вибору фільму та переходу до розкладу. Вебзастосунок ініціює асинхронний запит до сервера, який аналізує колекції halls та tickets, вилучаючи структуру залу та вільні місця. Отримані метадані дозволяють фронтенду динамічно відобразити

інтерактивну карту. Хоча перегляд схеми доступний усім, перехід до оплати вимагає верифікації профілю.

Після рендерингу форм користувач отримує доступ до пулу незалежних операцій, що обробляються через ізольовані REST API ендпоінти. Це забезпечує високу швидкість відгуку інтерфейсу при виконанні таких дій, як зміна параметрів підписки або додавання фільмів до списку очікування.

Обов'язковим компонентом безпеки перед виконанням будь-якої мутації даних є серверна верифікація сесії. Система декодує JWT-токен, вилучає ідентифікатор користувача та порівнює його з даними в базі. Якщо ідентифікатори не збігаються, а роль клієнта не передбачає адмін-повноважень, сервер блокує операцію. Успішне коригування статусів, як-от проведення повернення коштів у колекції tickets, виконується атомарно, гарантуючи цілісність інформаційного середовища AbsoluteCinema.

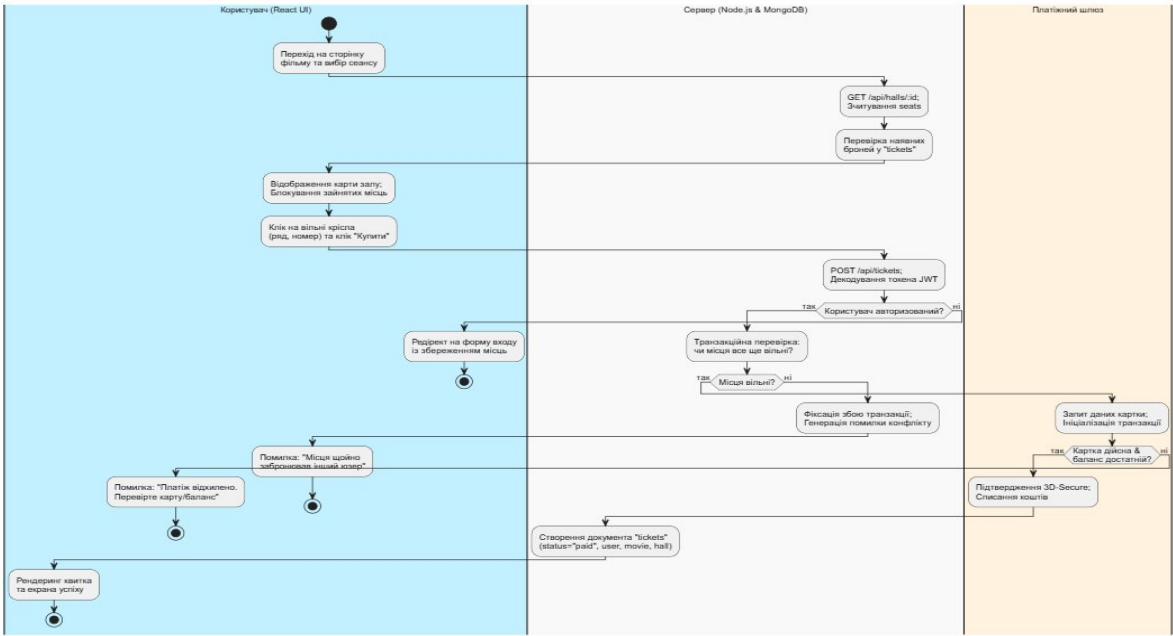


Рисунок 2.4 - Діаграма активності процесу бронювання та придбання квитків

Специфіку та логіку процесу розширення інформаційного каталогу платформи шляхом внесення нових медіаданих ілюструє поведінкова модель, наведена на рис. 2.5. Архітектурою вебплатформи AbsoluteCinema не передбачено жодних механізмів або інтерфейсних рішень для ініціалізації чи додавання кінострічок звичайними відвідувачами сайту. Процес розширення медіакаталогу є суворо привілейованим внутрішнім процесом,

який виконується виключно адміністратором системи через захищену адміністративну панель та ізольовану REST API кінцеву точку. Офіційне позиціонування та поява нового контенту в системи відбувається лише після ручного введення даних, технічної валідації та фінального схвалення з боку адміністрації проєкту.

Процедура генерації медіаконтенту адміністратором передбачає покрокове виконання серії обов'язкових перевірок надійності даних. На початковому етапі в адмін-панелі заповнюються ключові параметри фільму, після чого надсилається POST-запит на сервер Node.js. За допомогою Mongoose ODM система автоматично ініціює комплексний запит на унікальність об'єкта. Перевірка здійснюється одночасно за трьома критичними критеріями, назвою кінострічки, роком її випуску та ім'ям режисера. Такий підхід дозволяє архітектурно виключити ризик виникнення колізій у базі даних MongoDB, адже в кінематографі існує ймовірність релізу абсолютно різних за змістом фільмів, які мають ідентичне найменування та виходять в один і той самий рік.

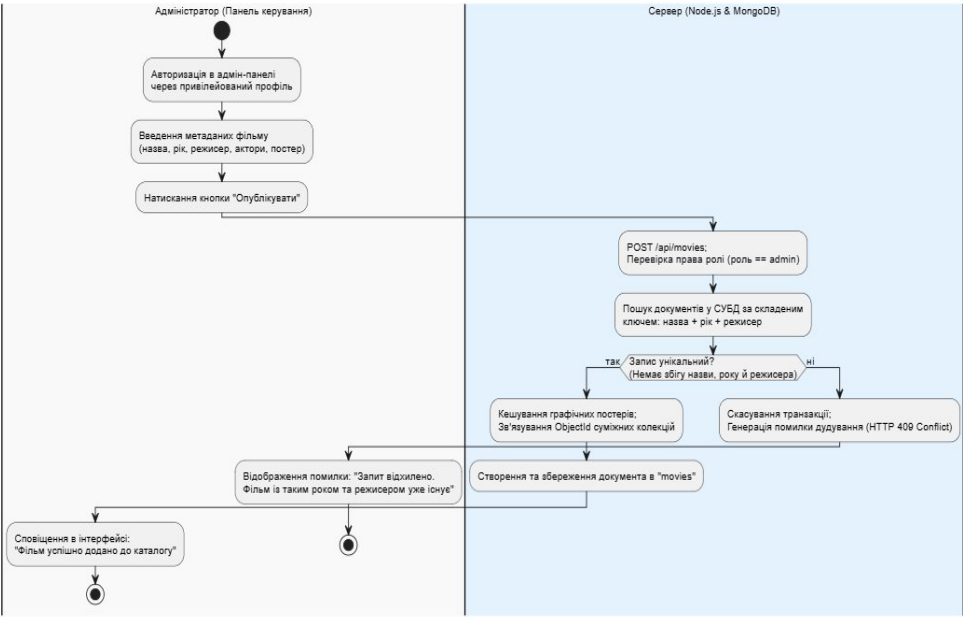


Рисунок 2.5 - Діаграма активності для медіаконтенту

Діаграма послідовності на рис. 2.6 описує взаємодію користувача з каталогом та модулем оцінювання. Процес починається із запиту від React UI до Node.js API з передачею JWT-токена. Сервер перевіряє статус профілю в

		Сівченко М.Т.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремов Ю.М.				60
Змн.	Арк.	№ докум.	Підпис	Дата		

MongoDB: при виявленні блокування (isBlocked) запит відхиляється з помилкою 403 Forbidden, обмежуючи доступ до контенту.

Після верифікації сервер повертає метадані фільму та його рейтинг на фронтенд. Авторизований користувач може оцінити стрічку за шкалою від 1 до 10. При надсиланні оцінки контролер перевіряє унікальність голосу, запобігаючи накручуванню.

Якщо оцінка від цього користувача для обраного фільму відсутня, запис фіксується в колекції ratings. За допомогою транзакцій MongoDB сервер виконує атомарне оновлення лічильників суми балів та кількості голосів у колекції movies. Після успішного збереження сервер повертає статус 201 Created, а оновлена статистика в реальному часі відображається в інтерфейсі. Спроба повторного голосування відхиляється з помилкою 409 Conflict, яка інформує користувача про те, що його голос вже враховано.

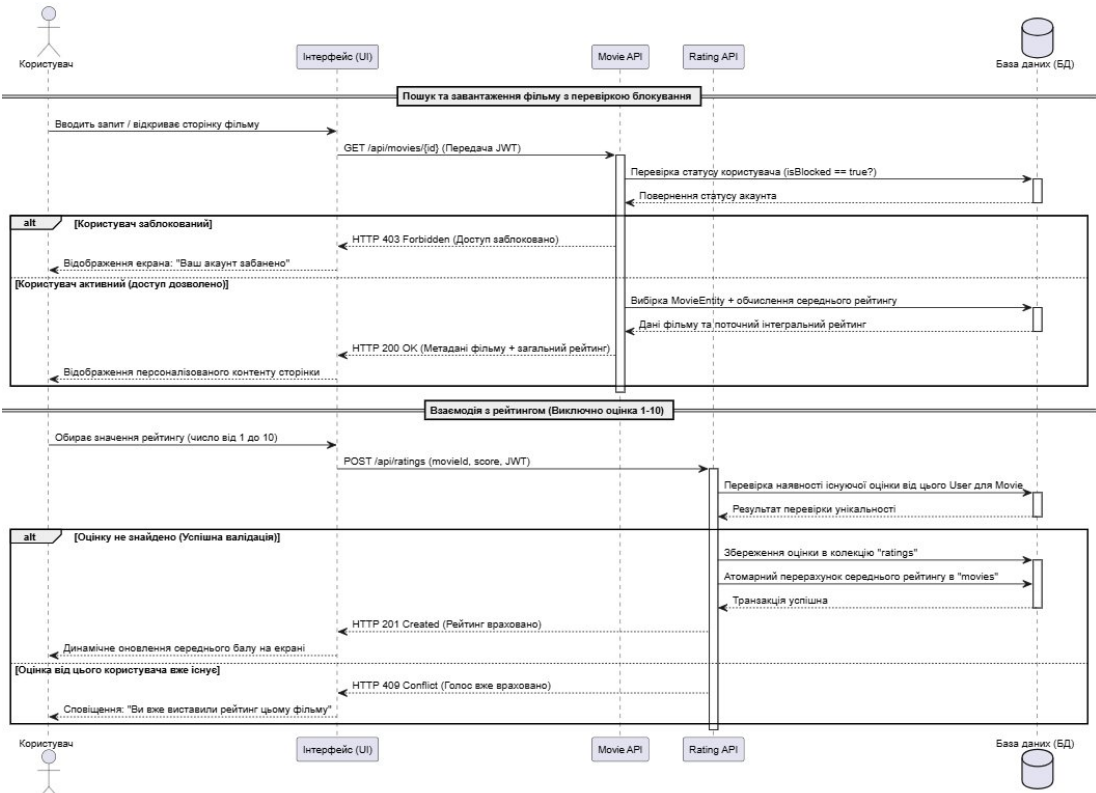


Рисунок 2.6 - Діаграма послідовності для додавання оцінки

Взаємодію користувачів із підтримкою ілюструє діаграма послідовності (рис. 2.7–2.10). Процес стартує, коли клієнт через React UI надсилає повідомлення на /api/chat/send. Сервер декодує JWT-токен і перевіряє

наявність відкритого діалогу: якщо його немає, Mongoose автоматично створює новий документ у колекції chattickets зі статусом «open», додаючи запит у чергу адміністратора.

Панель Support Hub отримує актуальну чергу через асинхронні GET-запити на /api/chat/admin/queue. При виборі тикета адміністратором виконується POST-запит, що прив'язує до документа admin ObjectId, змінює статус на «in_progress» та відправляє користувачеві привітання.

Ізольований модуль компенсації (Refund) дозволяє адміністратору переглянути всі квитки користувача та ініціювати повернення через /api/chat/admin/refund-ticket. Серверна транзакція змінює статус квитка на «refunded», генерує 90-денний промокод для клієнта та звільняє місце в колекції screenings (isBooked = false). Через Socket.io актуальна схема залу миттєво оновлюється для всіх глядачів. Алгоритм завершується автоматичним повідомленням у чат та переведенням тикета в статус «closed» через PATCH-запит.

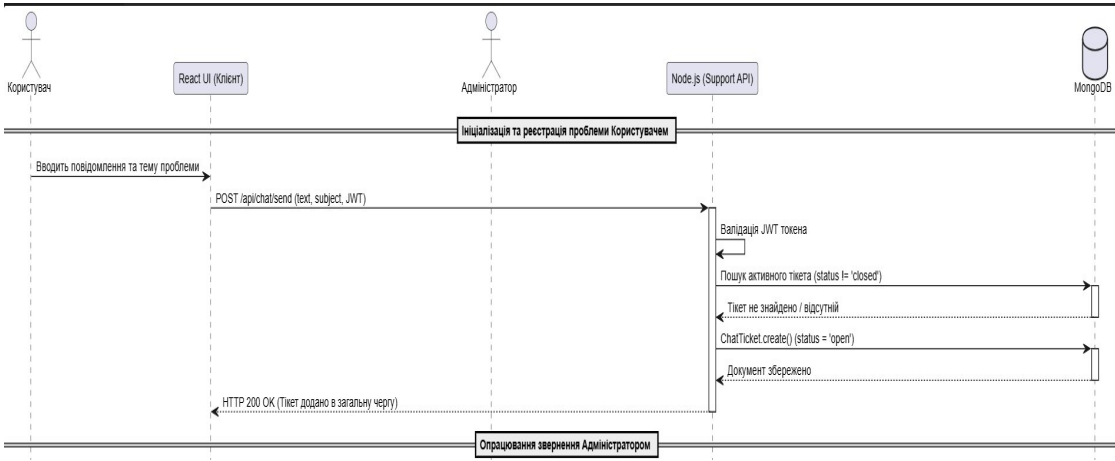


Рисунок 2.7 - Діаграма послідовності взаємодії із службою підтримки

Діаграма послідовності на рис. 2.7 відображає початковий етап взаємодії користувача із службою підтримки - ініціалізацію та реєстрацію звернення в системі. Процес розпочинається з того, що користувач через інтерфейс React UI вводить тему та текст проблеми, після чого клієнтський застосунок надсилає POST-запит на ендпоінт /api/chat/send з передачею тексту, теми звернення та JWT-токена в заголовок запиту.

На серверному рівні Node.js (Support API) виконує двоетапну перевірку: спочатку валідує JWT-токен для ідентифікації користувача, після чого здійснює пошук активного тікета з відповідним `userId` та статусом відмінним від `closed`. Якщо активне звернення відсутнє, Mongoose автоматично створює новий документ у колекції `ChatTicket` зі статусом `open`, фіксуючи тему звернення та первинне повідомлення користувача. Після успішного збереження документа в MongoDB сервер повертає клієнту статус HTTP 200 OK з підтвердженням того, що тікет додано до загальної черги очікування адміністратора.

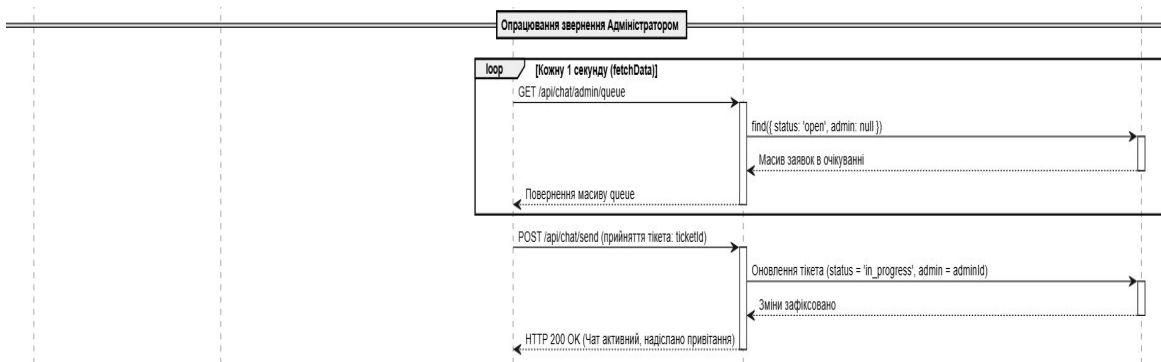


Рисунок 2.8 - Продовження діаграми послідовності взаємодії із службою підтримки

Рисунок 2.8 ілюструє етап опрацювання звернення адміністратором - від моніторингу черги тікетів до прийняття запиту в обробку. Адміністративна панель `AdminSupport.jsx` реалізує механізм періодичного оновлення черги через цикл `loop`, який кожну секунду надсилає GET-запит на ендпоінт `/api/chat/admin/queue`. Сервер виконує вибірку з колекції `ChatTicket` за умовою `{ status: 'open', admin: null }`, повертаючи масив усіх звернень що очікують на обробку, відсортованих за часом створення.

Після того як адміністратор обирає конкретний тікет із черги та натискає кнопку «ВЗЯТИ В РОБОТУ», клієнтський застосунок надсилає POST-запит на ендпоінт `/api/chat/send` з передачею ідентифікатора обраного тікета (`ticketId`). Сервер виконує атомарне оновлення документа в колекції `ChatTicket`: змінює значення поля `status` з `open` на `in_progress` та записує ідентифікатор адміністратора (`adminId`) у поле `admin`. Після фіксації змін у

MongoDB сервер повертає статус HTTP 200 OK з підтвердженням що чат активовано, та автоматично надсилає користувачу привітальне повідомлення від імені адміністратора, сигналізуючи про початок опрацювання його звернення.

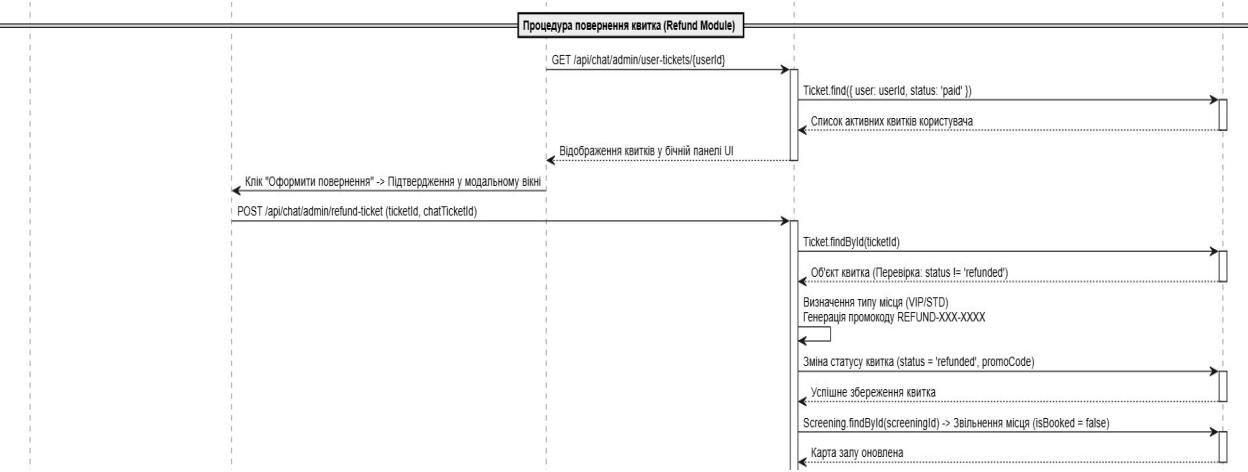


Рисунок 2.9 - Продовження діаграми послідовності взаємодії із службою підтримки

Рисунок 2.9 деталізує процедуру повернення квитка через модуль компенсації (Refund Module), інтегрований безпосередньо в інтерфейс адміністративного чату. Під час активного діалогу система автоматично виконує GET-запит на ендпоінт `/api/chat/admin/user-tickets/{userId}`, отримуючи з колекції `Ticket` перелік усіх активних квитків користувача за умовою `{ user: userId, status: 'paid' }`. Результат вибірки відображається у бічній панелі адміністратора, надаючи менеджеру повну картину поточних замовлень клієнта без необхідності переходу до інших розділів панелі керування.

Після того як адміністратор натискає кнопку «ОФОРМИТИ ПОВЕРНЕННЯ» та підтверджує дію у модальному вікні, клієнтський застосунок надсилає POST-запит на ендпоінт `/api/chat/admin/refund-ticket` з передачею ідентифікаторів квитка (`ticketId`) та поточного тікета підтримки (`chatTicketId`). Сервер виконує послідовність транзакційних операцій: спочатку знаходить документ квитка через `Ticket.findById(ticketId)` та перевіряє що його статус не є `refunded`, після чого визначає категорію місця (VIP або STD) для генерації відповідного компенсаційного промокоду у форматі `REFUND-XXX-XXXX`.

Далі сервер атомарно змінює статус квитка на refunded та записує згенерований промокод у поле promoCode. Паралельно виконується операція Screening.findById(screeningId), яка знаходить відповідний сеанс та змінює значення поля isBooked звільненого крісла на false, повертаючи місце до пулу доступних для бронювання.

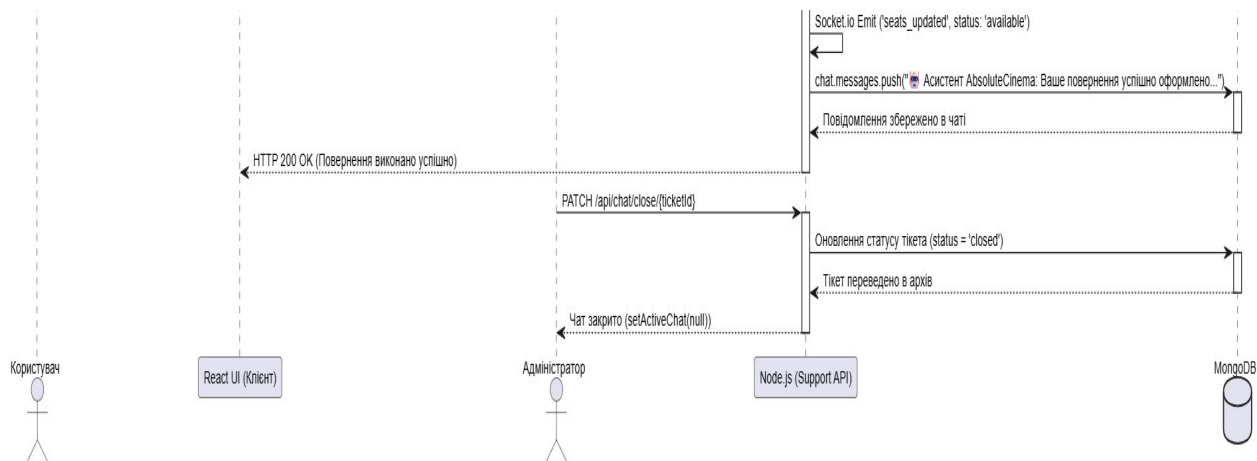


Рисунок 2.10 - Продовження діаграми послідовності взаємодії із службою підтримки

Рисунок 2.10 відображає фінальний етап циклу повернення квитка - синхронізацію стану залу в реальному часі, сповіщення користувача та закриття тикета підтримки. Після успішного збереження змін у колекціях Ticket та Screening сервер через Socket.io емітує подію seats_updated зі статусом available до відповідної кімнати сеансу (screeningId). Це миттєво оновлює інтерактивну схему залу для всіх користувачів, які переглядають сторінку бронювання цього сеансу, відображаючи щойно звільнене місце як доступне для повторного придбання.

Паралельно сервер формує автоматичне повідомлення від імені асистента AbsoluteCinema з підтвердженням успішного оформлення повернення та інформацією про згенерований промокод, яке додається до масиву messages тикета підтримки та зберігається в MongoDB. Клієнтський застосунок отримує статус HTTP 200 OK з підтвердженням що повернення виконано успішно.

Завершальним кроком алгоритму є закриття тикета підтримки. Адміністратор надсилає PATCH-запит на ендпоінт /api/chat/close/{ticketId}, після чого сервер

оновлює поле status документа на значення closed та переводить тикет до архіву. На стороні клієнтського інтерфейсу адміністратора викликається функція setActiveChat(null), що скидає активний діалог та повертає панель підтримки до відображення черги нових звернень. Таким чином завершується повний життєвий цикл звернення користувача - від початкової реєстрації проблеми до її вирішення з фінансовою компенсацією.

2.4. Реалізація функціоналу підсистеми інтерактивного бронювання та придбання квитків

Центральним модулем платформи AbsoluteCinema є підсистема інтерактивного вибору місць та бронювання, що інтегрує роботу з базою даних, платіжними шлюзами та механізмами синхронізації станів у реальному часі. Архітектура підсистеми побудована за принципом послідовної валідації даних на кожному етапі обробки запиту, що гарантує цілісність інформації та захист від типових вразливостей високонавантажених транзакційних систем.

Процес бронювання розпочинається з завантаження компонента Booking.jsx, який ініціює два паралельні асинхронні запити до серверного API. Перший запит GET /api/halls/:hallName отримує шаблонну конфігурацію залу з колекції Hall, що містить повну матрицю фізичних крісел з їхніми координатами, категоріями та прапорцями фізичної наявності (exists). Другий запит GET /api/screenings/:id отримує поточний стан сеансу з колекції Screening, включаючи динамічну матрицю місць з актуальними статусами бронювання.

На основі отриманих даних алгоритм фронтенду за допомогою Math.max() обчислює максимальні значення рядів та колонок, визначаючи габарити залу. Після цього через Array.from() генерується двовимірна матриця для відображення. Кожне крісло рендериться як окремий інтерактивний HTML-елемент, чий візуальний стан визначається на основі комбінації полів isBooked та lockedUntil зі стану сеансу. Місця розфарбовуються у відповідні кольори: вільні стандартні, вільні VIP, тимчасово заблоковані іншим користувачем та

		Сівченко М.Т.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		66

повністю заброньовані. Крісла з прапорцем `exists: false` повністю виключаються з рендерингу, зберігаючи коректну геометрію залу.

Стан вибраних користувачем місць керується через контекст `BookingContext`, який централізовано зберігає масив обраних крісел, розраховує динамічну вартість замовлення залежно від категорії місць та керує переходами між етапами оформлення. Використання `Context API` замість локального стану компонента забезпечує доступ до даних бронювання з будь-якого вкладеного компонента без необхідності передачі пропсів через проміжні рівні.

Однією з ключових інженерних задач підсистеми є захист від стану конкуренції (`Race Condition`), коли два або більше користувачі одночасно намагаються придбати одне й те саме місце. Для вирішення цієї проблеми реалізовано дворівневий механізм захисту.

Перший рівень - тимчасове блокування місць (`lock-seats`). У момент коли користувач обирає крісло на схемі залу та переходить до етапу оплати, клієнтський застосунок надсилає `POST`-запит на ендпоінт `/api/screenings/lock-seats` з масивом координат обраних місць та ідентифікатором сеансу. Сервер виконує атомарну операцію оновлення документа сеансу в колекції `Screening`: для кожного обраного крісла встановлює поле `lockedBy` рівним `userId` поточного користувача, а поле `lockedUntil` - часовій мітці що відповідає поточному моменту плюс п'ять хвилин. Якщо під час виконання цієї операції виявляється що будь-яке з обраних місць вже заблоковане іншим користувачем (тобто `lockedUntil` більше поточного часу та `lockedBy` не збігається з поточним `userId`), сервер повертає `HTTP 409 Conflict` з переліком конфліктних місць, а клієнтський застосунок оновлює схему залу та сповіщає користувача про необхідність обрати інші крісла.

Другий рівень захисту спрацьовує безпосередньо під час фіналізації покупки. При надходженні `POST`-запиту на ендпоінт `/api/tickets` сервер повторно перевіряє унікальність координат обраних місць у колекції `Ticket`, виконуючи пошук за комбінацією полів `screening`, `row` та `seatNumber`. Якщо запис вже існує, операція відхиляється з помилкою `HTTP 409 Conflict`. Така

		Сівченко М.Т.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		67

подвійна перевірка гарантує цілісність даних навіть у випадках коли термін тимчасового блокування вийшов, але паралельний запит від іншого користувача встиг завершити покупку раніше.

Для автоматичного звільнення місць від прострочених блокувань на сервері реалізовано фоновий процес очищення, який за допомогою cron-задачі періодично виконує оновлення колекції Screening, скидаючи значення полів lockedBy та lockedUntil для всіх крісел де термін блокування вже минув. Це запобігає накопиченню «мертвих» блокувань у базі даних та гарантує що місця повертаються до пулу доступних у разі якщо користувач покинув сторінку оплати без завершення транзакції.

Фінансова складова підсистеми бронювання реалізована через інтеграцію з міжнародним платіжним шлюзом Stripe. Процес оплати розділений на два незалежні етапи, що унеможливорює зберігання або обробку конфіденційних платіжних даних безпосередньо на серверах платформи.

На першому етапі при переході користувача до сторінки оплати компонент Payment.jsx надсилає POST-запит на ендпоінт /api/screenings/create-payment-intent, передаючи загальну вартість замовлення в гривнях. Бекенд-сервер взаємодіє з Stripe API, викликаючи метод stripe.paymentIntents.create() з параметрами суми та валюти, після чого отримує унікальний об'єкт платіжного наміру з полем clientSecret. Цей секретний ключ повертається на клієнтську частину та використовується виключно для ініціалізації захищеної форми введення платіжних реквізитів через @stripe/react-stripe-js.

На другому етапі форма введення даних картки будується на основі захищених iframe-компонентів CardElement від Stripe, які ізолюють платіжні реквізити від коду застосунку. Бібліотека Stripe.js автоматично валідує формат номера картки за алгоритмом Луна, перевіряє коректність терміну дії та CVV-коду, блокуючи відправку форми при виявленні синтаксичних помилок. Після натискання кнопки «СПЛАТИТИ» компонент PaymentUIComponent через State Pattern переводить локальний стан isProcessing у значення true, що активує компонент LoadingOverlay та блокує повторне надсилання форми,

		Сівченко М.Т.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремов Ю.М.				68
Змн.	Арк.	№ докум.	Підпис	Дата		

реалізуючи принцип відповідальності UI. Далі хук `stripe.confirmCardPayment()` передає токеновані дані на сервер Stripe для авторизації коштів та проходження протоколу 3D Secure.

Логіка завершення транзакції побудована за асинхронною моделлю обробки подій. Після підтвердження платежу на стороні Stripe сервер отримує підписану подію `payment_intent.succeeded` через Webhook-хук у вигляді POST-запиту на захищений ендпоінт `/api/payments/webhook`. Перед обробкою бекенд верифікує цифровий підпис події за допомогою `STRIPE_WEBHOOK_SECRET`, що унеможливлює підробку запитів. Після успішної верифікації асинхронно викликається `TransactionService.processPayment()`, який атомарно створює документ квитка у колекції `Ticket` зі статусом `paid` та звільняє тимчасове блокування місця в колекції `Screening`.

Підсистема бронювання інтегрує модуль обробки компенсаційних промокодів, які генеруються адміністратором при поверненні квитків через чат підтримки. Кожен промокод має формат `REFUND-XXX-XXXX` та прив'язаний до конкретної категорії місця (`VIP` або `STD`), що було повернуто. При введенні промокоду користувачем під час оформлення замовлення сервер виконує багаторівневу валідацію. На першому рівні перевіряється факт існування промокоду в базі даних та відсутність службового префіксу що позначає вже використаний код. На другому рівні сервер перевіряє термін дії промокоду - максимальний ліміт складає 90 днів з моменту генерації. На третьому рівні виконується перевірка відповідності категорії промокоду (`VIP` або `STD`) категорії обраних користувачем місць через порівняння поля `currentSeatType` зі структурою промокоду. Якщо категорії не збігаються, система повертає `HTTP 400 Bad Request` з відповідним повідомленням про помилку.

У момент фіналізації замовлення з успішно застосованим промокодом сервер виконує атомарну операцію його анулювання. Код промокоду модифікується службовим префіксом `USED-` через атомарну операцію

		Сівченко М.Т.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремов Ю.М.				69
Змн.	Арк.	№ докум.	Підпис	Дата		

findOneAndUpdate в MongoDB, що робить його невалідним для повторного використання та захищає систему від Replay Attacks. Поле promoCode у створеному документі квитка фіксує використаний код для збереження історії знижок у фінансовій звітності.

Для забезпечення актуального відображення стану місць у залі без необхідності перезавантаження сторінки реалізовано модуль реального часу на базі Socket.io. При підключенні до сторінки бронювання клієнтський застосунок приєднується до віртуальної WebSocket-кімнати з ідентифікатором поточного сеансу (screeningId), що дозволяє серверу адресно розсилати оновлення лише тим користувачам які переглядають схему конкретного залу. Подія seats_updated емітується сервером у трьох випадках: при тимчасовому блокуванні місць користувачем що перейшов до оплати, при успішному завершенні покупки та при поверненні квитка адміністратором через модуль компенсації. Корисне навантаження події містить оновлений масив місць сеансу, на основі якого клієнтський застосунок миттєво перерендерює схему залу без звернення до серверного API. Такий підхід замінює енергоємне HTTP-опитування (polling) постійним дуплексним з'єднанням, суттєво знижуючи навантаження на сервер при одночасному перегляді схеми залу великою кількістю користувачів під час пікового попиту на квитки популярних прем'єр. Модуль Socket.io також обслуговує комунікаційний канал чату підтримки, забезпечуючи миттєву доставку повідомлень між користувачем та адміністратором без затримок характерних для HTTP-запитів. Синхронізація клієнтських інтерфейсів чату здійснюється через механізм «оптимістичного оновлення» (Optimistic UI), коли повідомлення відображається в інтерфейсі відправника миттєво після введення, ще до отримання підтвердження від сервера, що забезпечує відчуття миттєвого відгуку системи [2].

Висновки до другого розділу

У другому розділі кваліфікаційної роботи виконано комплексне проєктування та технічне обґрунтування архітектури full-stack вебплатформи

		Сівченко М.Т.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремов Ю.М.				70
Змн.	Арк.	№ докум.	Підпис	Дата		

«AbsoluteCinema». На основі проведеного аналізу бізнес-вимог та специфіки предметної області було успішно реалізовано логічне моделювання системи, розроблено структуру збереження даних та деталізовано алгоритми взаємодії між клієнтськими й серверними компонентами додатка.

Основні практичні та теоретичні результати, отримані під час виконання розділу, полягають у наступному,

Обґрунтовано та впроваджено сучасний стек технологій. Для реалізації високої чуйності інтерфейсу обрано компонентну архітектуру React UI на фронтенді, а як безперервне, масштабоване серверне рішення середовище Node.js із фреймворком Express. Для персистентного збереження інформації інтегровано документ-орієнтовану СУБД MongoDB, взаємодія з якою оптимізована за допомогою інструментарію Mongoose ODM.

Розроблено поведінкові моделі за допомогою мови UML. Створено та деталізовано діаграми активності та послідовності, які покроково відображають життєвий цикл основних прецедентів системи, процес інтерактивного вибору місць, транзакційного бронювання та придбання квитків, а також привілейовані операції модерації та наповнення медіакаталогу адміністратором.

Реалізовано надійну підсистему безпеки та розмежування прав. На основі сесійних маркерів JWT впроваджено гнучке керування доступом на рівні захищеного вебAPI. Архітектурно закладено превентивну перевірку логічного прапорця блокування користувачів (isBlocked), що дозволяє миттєво ізолювати скомпрометовані або забанені акаунти й повністю блокувати їм доступ до контенту медіатеки ще до етапу виконання ресурсномістких запитів.

Оптимізовано структуру даних для зниження навантаження на СУБД. Застосовано ефективний підхід денормалізації NoSQL-документів у колекції movies. Завдяки винесенню логіки перерахунку та збереження інтегральних лічильників (ratingCount, ratingSum) безпосередньо у момент виставлення оцінки. Збереження агрегованих лічильників ratingSum та ratingCount

		Сівченко М.Т.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремюв Ю.М.				71
Змн.	Арк.	№ докум.	Підпис	Дата		

безпосередньо в документі фільму реалізує підхід денормалізації, що є ефективним методом оптимізації аналітичних запитів [8].

Вирішено проблему стану конкуренції (Race Condition). У підсистемі продажу квитків реалізовано ізольовану серверну валідацію унікальності обраних координат крісел (ряд, номер), яка повністю унеможлиблює ризик подвійного бронювання одного місця різними користувачами в один і той самий момент часу, повертаючи помилку HTTP 409 Conflict для запізнілих запитів.

Спроектовано модуль підтримки та фінансової компенсації (Support Hub), що автоматизує опрацювання експлуатаційних запитів. Логіка чат-системи побудована на базі тікетів із інтегрованим Refund-модулем, який дозволяє адміністраторам ініціювати повернення коштів безпосередньо з інтерфейсу адмін-панелі.

		Сівченко М.Т.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		72

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ВЕБПЛАТФОРМОЮ БРОНЮВАННЯ КВИТКІВ У КІНОТЕАТР

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

Вебплатформа «AbsoluteCinema» побудована на чіткому розділенні фронтенду та бекенду. Така сервісно-орієнтована архітектура дозволяє ізолювати логічні шари, розміщуючи їх на незалежних середовищах. Це забезпечує гнучкість деплою, легке масштабування під час пікових навантажень та високу відмовостійкість системи.

Топологія зв'язків та розміщення вузлів візуалізовані на діаграмі розгортання (рис. 3.1). Особлива увага приділена безпеці: клієнтський застосунок взаємодіє з сервером через шифрований HTTPS, що мінімізує ризики перехоплення.

Сервер Node.js функціонує в ізольованій підмережі, здійснюючи комунікацію з MongoDB через захищений драйвер Mongoose з обов'язковою автентифікацією. Для роботи модуля підтримки та динамічного оновлення карти залу між клієнтом та сервером розгорнуто постійний дуплексний канал WebSocket (Socket.io). Таке логічне розділення серверних потужностей дозволяє ефективно розподіляти ресурси та забезпечує захист конфіденційної інформації та платіжних операцій від зовнішніх загроз.

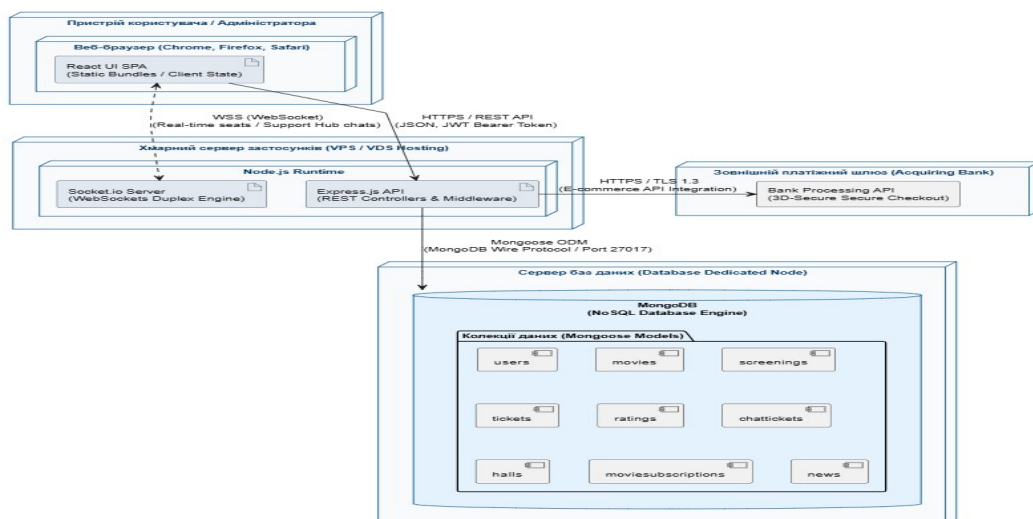


Рисунок 3.1 - Діаграма розгортання

		Сівченко М.Т.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		73

Вебплатформа «AbsoluteCinema» має чітко структуровану сервісно-орієнтовану архітектуру, розподілену на три основні компоненти: клієнтську частину (Frontend), логічне ядро (Backend) та рівень зберігання даних (Database). Схема їх взаємодії відображена на діаграмі розгортання (рис. 3.1).

Процес розгортання платформи спроектовано з урахуванням ізоляції фронтенду та бекенду, що дозволяє мінімізувати час простою системи (downtime) під час оновлень та забезпечує швидку конфігурацію середовища. Технічні характеристики вузлів розгортання, їхні програмні вимоги та конфігураційні параметри для кожного компонента детально описані у таблиці 3.1.

Таблиця 3.1

Етапи технологічного розгортання та збірки компонентів вебплатформи

Етапи збірки проєкту	Автоматизоване розгортання (через PM2 Runtime)	Ручна збірка та запуск (Node.js & Vite CLI)	Інфраструктурні компоненти (MongoDB / Mongoose)
Підготовчий етап	Встановлення на цільовому сервері середовища Node.js (LTS версії) та глобального менеджера процесів PM2. Клонування репозиторію проєкту з GitLab.	Інсталяція Node.js та пакетного менеджера npm на локальному робочому місці розробника або CI/CD-сервері.	Встановлення СУБД MongoDB Server та запуск системної служби mongod, або отримання Connection String до хмарного кластера.
Конфігурація середовища	Створення файлу .env у кореневому каталозі бекенду. Налаштування змінних оточення, PORT, MONGO_URI, секретного ключа JWT_SECRET.	Ініціалізація інсталяції залежностей на сервері або CI-платформі за допомогою команди npm ci або npm install.	Створення та налаштування користувача бази даних із правами читання та запису (readWrite) до цільової бази даних AbsoluteCinema.

продовж. табл.3.1

Компіляція та збірка релізу	Автоматична генерація оптимальних скриптів запуску. Для фронтенду виконується компіляція та мініфікація статики, npm run build у папці React-додатка.	Запуск локальної збірки. Процесор Vite компілює SPA-артефакти, оптимізує чанки й зберігає вихідні файли фронтенду в автономну директорію /dist.	Виконання первинної ініціалізації бази. Бекенд-додаток при першому запуску автоматично створює індекси (наприклад, унікальний складений індекс user_1_movie_1 для Rating).
Перенесення файлів (для ручного методу)	Не потрібно при прямому деплої, оскільки менеджер PM2 керує вихідним кодом безпосередньо з робочого каталогу на сервері.	Копіювання вмісту скомпільованої папки /dist фронтенду на веб-сервер (наприклад, Nginx) та перенесення коду Express-бекенду через SFTP/SSH.	Синхронізація схем Mongoose. Перевірка доступності NoSQL-колекцій і запуск первинних міграцій для наповнення каталогу фільмів (movies).
Запуск та фінальний моніторинг	Контейнеризація процесу у фоновому режимі з автоперезапуском при збоях.	Запуск Express-сервера вручну в CLI за допомогою утиліти production-запуску, NODE_ENV=production node server.js або через системні демони Linux.	Фінальна перевірка підключення через моніторинг консолі сервера. Перевірка статусу працездатності WebSocket-кімнат (Socket.io) та логів бази даних.

Під час перенесення «AbsoluteCinema» у продуктивне середовище ключовим завданням є налаштування змінних оточення та комплексного захисту даних. Оскільки архітектура розділяє клієнтську та серверну частини, першочерговою вимогою є коректна конфігурація шлюзу та безпека взаємодії компонентів.

		Сівченко М.Т.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		75

Захист з'єднання (HTTPS / Reverse Proxy), всі запити до API здійснюються через HTTPS. Забезпечення кібербезпеки вебзастосунків передбачає комплекс заходів захисту від несанкціонованого доступу, включаючи шифрування з'єднань та валідацію вхідних даних [29]. На VPS розгортається Nginx як Reverse Proxy з налаштованими SSL/TLS сертифікатами (Let's Encrypt), що шифрує JWT-токени та запобігає MITM-атакам. На рівні Node.js підключається бібліотека express-cors для обмеження переліку дозволених доменів, а пакет helmet формує захисні HTTP-заголовки відповідно до галузевих стандартів безпеки. Обидва модулі інтегруються як ланки Middleware-ланцюжка Express відповідно до принципу Dependency Inversion, що дозволяє замінювати або розширювати політики безпеки без втручання у бізнес-логіку маршрутів.

Ізоляція секретних даних, приватні ключі, паролі та рядки підключення до БД винесені у файл .env на сервері та додані до .gitignore. Для безпечного зберігання паролів у MongoDB використовується алгоритм хешування bcrypt.

Контейнеризація та ізоляція процесів, Node.js та MongoDB працюють як ізольовані процеси. Ізоляція серверних процесів та керування ресурсами операційної системи є необхідною умовою стабільного розгортання вебзастосунків [7]. Бекенд запускається менеджером PM2, який автоматично перезапускає додаток при помилках та ізолює робоче середовище від ОС. База даних обмежена локальним інтерфейсом localhost.

Автоматизація та моніторинг, за допомогою cron налаштовано щодобове створення знімків (mongodump) усіх колекцій із перенесенням у віддалене хмарне сховище. Оперативний моніторинг помилок, зокрема роботи Socket.io, здійснюється через логи PM2.

Кросбраузерність, під час збірки проєкту інструмент Vite спільно з Babel транспілює сучасний JavaScript та JSX у старіші стандарти. Це гарантує стабільний рендеринг інтерфейсу на різних пристроях, включаючи застарілі версії браузерів.

		Сівченко М.Т.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремов Ю.М.				76
Змн.	Арк.	№ докум.	Підпис	Дата		

3.2. Структура інтерфейсу вебплатформи

Клієнтська частина вебплатформи «AbsoluteCinema» функціонує безпосередньо у веб-браузері кінцевого користувача, забезпечуючи взаємодію з бекенд-АРІ як у локальному середовищі розробника (через localhost), так і у продуктивному середовищі при розміщенні на хмарному сервері у відкритому доступі. Проєктування користувацького інтерфейсу вебсистеми базується на принципах компонентно-орієнтованого дизайну та забезпечення інтуїтивної навігації [18]. Інтерфейс застосунку спроектовано за принципом компонентно-орієнтованого дизайну (Component-Driven Design), що дозволяє динамічно оновлювати вміст сторінок без їх повного перезавантаження.

Після успішного завантаження головної сторінки додатка користувачеві відкривається адаптивна навігаційна панель та слайдер із актуальними кінопрем'єрами. Головне меню системи містить такі функціональні розділи, що повністю відповідають внутрішній структурі NoSQL-колекцій бази даних, що забезпечує високий рівень синхронізації між візуальним відображенням та структурою збереження даних. Зокрема, розмежування компонентів на логічні блоки дозволяє реалізувати гнучку систему доступу до сервісів, таких як інтерактивний план зали, профіль користувача з історією бронювань, розділ фінансових операцій та модуль відстеження актуальних сеансів.

Головна сторінка вебплатформи AbsoluteCinema, зображена на рис. 3.2, є першою точкою взаємодії відвідувача з системою. Інтерфейс виконано у темній кольоровій схемі з акцентними червоними елементами, що формує кінематографічну атмосферу та відповідає загальній стилістиці розважального сервісу.

У верхній частині сторінки розміщено навігаційну панель з логотипом платформи AbsoluteCinema та іконками швидкого доступу до новин, чату підтримки та особистого кабінету користувача. Центральний блок містить головний слоган «Твій квиток у світ ABSOLUTE кіно» з підзаголовком «Бронювання місць у найкращих залах міста», що одразу інформує відвідувача про основне призначення платформи.

		Сівченко М.Т.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		77

Нижче розташовано інтерактивний горизонтальний слайдер секції «Зараз у кіно», який відображає картки поточних кінострічок з постерами та рейтингами глядачів. Навігація між картками реалізована через кнопки прокрутки, що дозволяє переглядати весь актуальний репертуар без переходу на окрему сторінку каталогу. Дані для слайдера завантажуються асинхронним GET-запитом до ендпоінта /api/movies з фільтром isShowingInCinema: true, що гарантує відображення лише актуальних кінострічок поточного прокату.

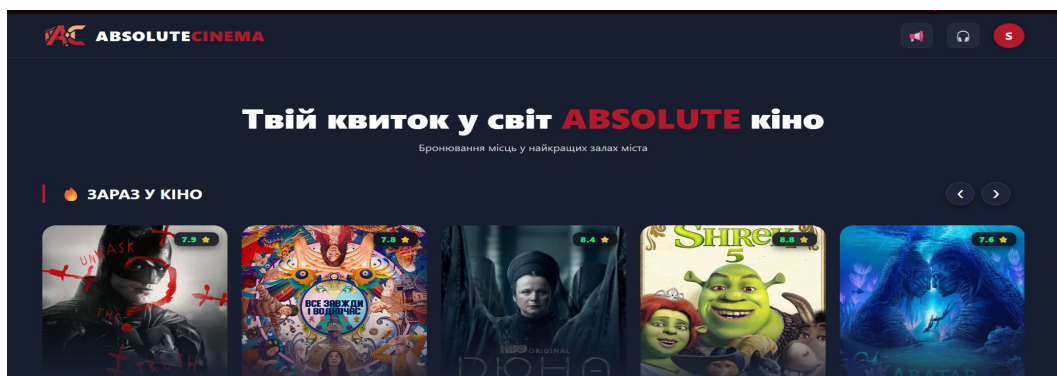


Рисунок 3.2 - Головна сторінка вебплатформи AbsoluteCinema

Рисунок 3.3 демонструє інтерфейс фільтрації та навігації по розкладу сеансів платформи. У верхній частині екрана розміщено горизонтальний календарний слайдер з датами, що дозволяє користувачу швидко переключатися між днями та переглядати доступні сеанси на конкретну дату. Активна дата виділяється кольоровим індикатором, а перелік дат автоматично генерується на основі наявних сеансів у колекції Screening.

Нижче розташовано два інструменти фільтрації контенту. Перший - сортування за рейтингом медіатеки, що дозволяє впорядкувати список фільмів за середнім балом глядачів у порядку зростання або спадання. Другий - фільтр за жанрами, реалізований у вигляді горизонтального набору інтерактивних тегів з повним переліком жанрових категорій: «Всі жанри», «Екшн», «Фентезі», «Пригоди», «Бойові мистецтва», «Мультфільм», «Комедія», «Сімейний», «Фантастика», «Бойовик», «Аніме», «Трилер», «Драма», «Жахи», «Тригер», «Супергерої», «Містика», «Історія», «Кримінал». Активний

		Сівченко М.Т.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремов Ю.М.				78
Змн.	Арк.	№ докум.	Підпис	Дата		

жанровий фільтр підсвічується червоним кольором. При виборі жанру клієнтський застосунок надсилає GET-запит до /api/movies з відповідним параметром фільтрації, оновлюючи список фільмів без перезавантаження сторінки.

У нижній частині екрана відображається секція «Розклад сеансів» з групуванням показів за датою - наприклад «Четвер, 25 червня» - та картками фільмів з їхніми рейтингами, що забезпечує зручну навігацію по розкладу кінотеатру.

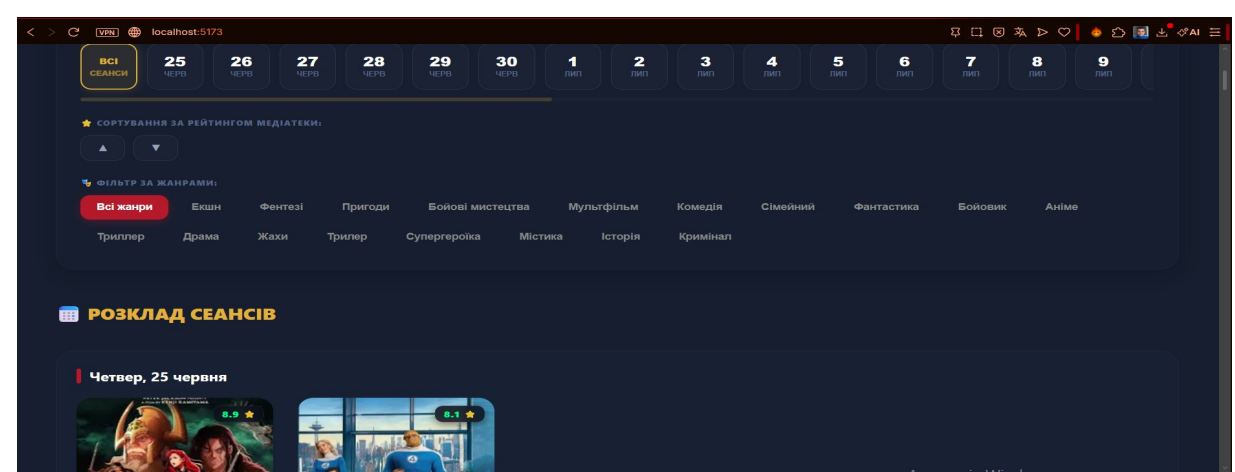


Рисунок 3.3 - Фільтрація фільмів

Рисунок 3.4 відображає сторінку детальної інформації про кінострічку на прикладі фільму «Бійцівський клуб». Сторінка завантажується після вибору фільму з каталогу або розкладу сеансів і формується на основі даних отриманих асинхронним GET-запитом до ендпоінта /api/movies/:id, де параметр ідентифікатора зчитується з URL через хук useParams.

Візуальна компоновка сторінки розділена на два блоки. Лівий блок містить вертикальний постер фільму високої роздільної здатності, завантажений за адресою збереженою в полі posterUrl документа колекції Movie. Правий інформаційний блок відображає повний набір метаданих кінострічки: глядацький рейтинг 8.8 з іконкою зірки, рік випуску 1999, повну назву фільму, інтерактивну шкалу оцінювання «Оцініть прем'єру» для авторизованих користувачів, жанрові теги («Драма», «Трилер», «Кримінал») у вигляді стилізованих бейджів, тривалість стрічки 139 хв. та ім'я режисера

		Сівченко М.Т.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремов Ю.М.				79
Змн.	Арк.	№ докум.	Підпис	Дата		

Девід Фінчер. Нижче розміщено текстовий блок з детальним описом сюжету фільму.

Інтерактивна шкала оцінювання відображається лише для авторизованих користувачів - для неавторизованих відвідувачів цей елемент приховується на рівні умовного рендерингу React. При виставленні оцінки клієнтський застосунок надсилає POST-запит до /api/ratings з передачею JWT-токена, після чого сервер атомарно оновлює лічильники ratingSum та ratingCount у документі фільму та повертає оновлений середній бал для відображення в інтерфейсі [1].

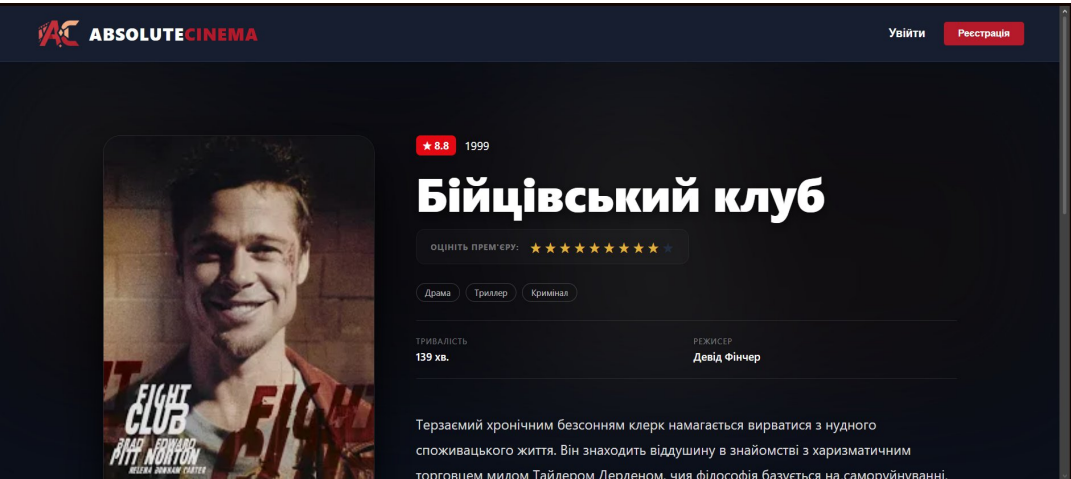


Рисунок 3.4 - Перегляд сторінки фільму

Рисунок 3.5 демонструє інтерфейс інтерактивної схеми залу для сеансу фільму «Людина-павук: Повземо крізь Всесвіт» у залі OverHall 25 червня о 17:40. Сторінка бронювання є найбільш технологічно складним компонентом платформи, оскільки поєднує динамічний рендеринг матриці крісел, синхронізацію стану в реальному часі та захист від одночасного бронювання одного місця кількома користувачами.

Верхня частина інтерфейсу містить інформаційний блок з назвою фільму, статусом «Зараз у кіно», назвою залу та точним часом початку сеансу. Умовне позначення «ЕКРАН» візуально орієнтує глядача відносно розташування переднього краю зали. Основну частину екрана займає двовимірна матриця крісел, згенерована компонентом Booking.jsx на основі масиву seats отриманого з колекції Screening. Зал відображає шість рядів по

чотирнадцять місць у кожному, з нумерацією рядів по обидва боки схеми для зручності орієнтації.

На момент знімку всі місця відображаються у темно-синьому кольорі що відповідає статусу вільних стандартних крісел. При наведенні курсору на крісло воно змінює колір, сигналізуючи про можливість вибору. Після кліку обране місце виділяється активним кольором та додається до масиву вибраних крісел у BookingContext, одночасно оновлюючи розрахунок загальної вартості замовлення в нижній панелі. Актуальність стану місць підтримується через постійне WebSocket-з'єднання Socket.io, яке миттєво відображає зміни внесені іншими користувачами без необхідності оновлення сторінки [2].

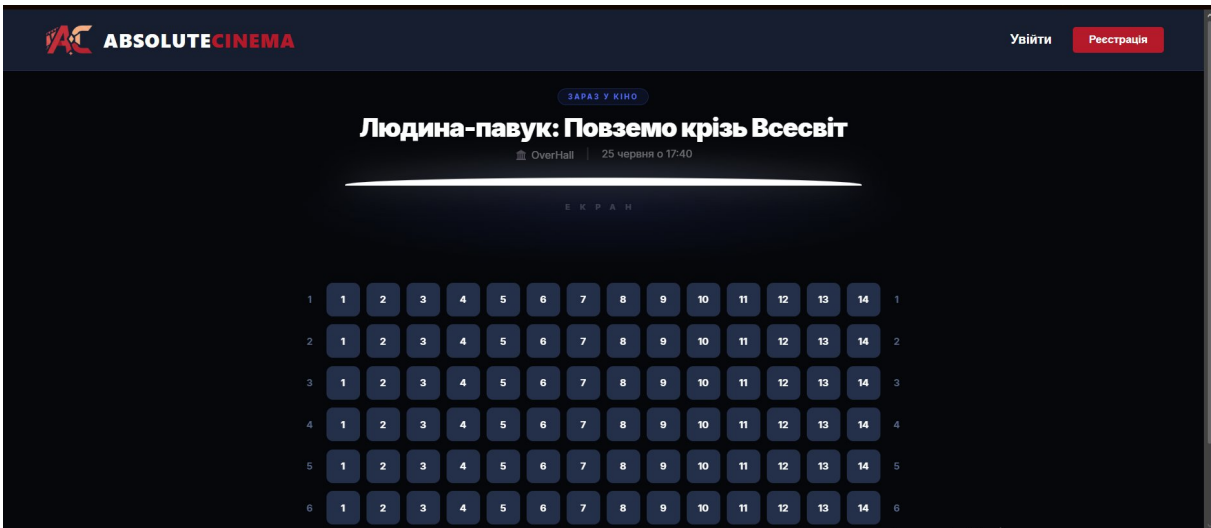


Рисунок 3.5 - Вигляд бронювання місць на кінопрем'єру

Рисунок 3.6 ілюструє поведінку системи при спробі неавторизованого відвідувача взаємодіяти зі схемою залу. У момент кліку на будь-яке крісло компонент Booking.jsx перевіряє наявність валідного JWT-токена у localStorage. За його відсутності замість відправки запиту на сервер клієнтський застосунок активує стан showAuthModal, що відкриває модальне вікно авторизації поверх затемненої схеми залу.

Модальне вікно містить іконку замка, заголовок «ПОТРІБНА АВТОРИЗАЦІЯ» та пояснювальний текст «Щоб обирати місця та купувати квитки в кінотеатрі AbsoluteCinema, будь ласка, увійдіть у свій акаунт або зареєструйтеся». Користувачу пропонується три варіанти дії: перехід до

форми входу через кнопку «УВІЙТИ», перехід до форми реєстрації через кнопку «РЕЄСТРАЦІЯ» або закриття модального вікна через посилання «СКАСУВАТИ» для повернення до перегляду схеми залу без авторизації.

Така реалізація захисту є дворівневою: перший рівень спрацьовує на стороні клієнта через перевірку токена в localStorage ще до відправки будь-якого запиту, другий рівень - на стороні сервера через middleware protect, який відхиляє запити до захищених ендпоінтів /api/screenings/lock-seats та /api/screenings/book-multiple-seats з поверненням статусу HTTP 401 Unauthorized за відсутності валідного токена в заголовку запиту [5].

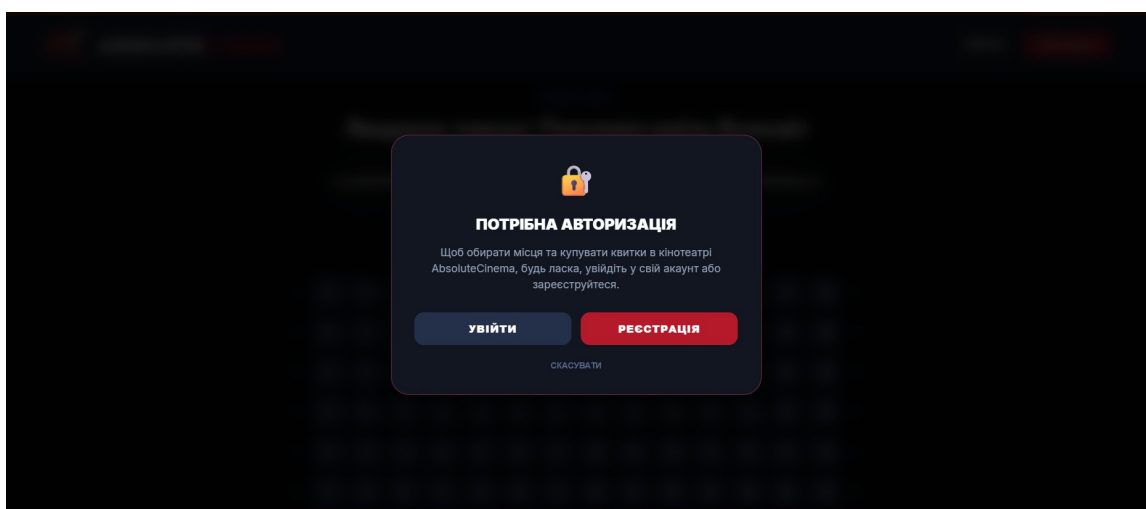


Рисунок 3.6 - Спроба бронювання незареєстрованим користувачем

Неавторизований користувач має можливість перегляду розкладу сеансів кінотеатру, також перегляд місць бронювань, бронювання недоступне допоки користувач не авторизується в системі (рис.3.2 - 3.6). При спробі взаємодії з ендпоінтами, що відповідають за блокування місць (/api/screenings/lock-seats) або оформлення замовлення (/api/screenings/book-multiple-seats), клієнтський застосунок надсилає запит, який сервер перевіряє за допомогою прошарку авторизації protect. У разі відсутності валідного JWT-токена або неавторизованого статусу сесії, система повертає відповідний HTTP-статус, ініціюючи автоматичний перехід гостя на сторінку входу або реєстрації.

		Сівченко М.Т.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		82

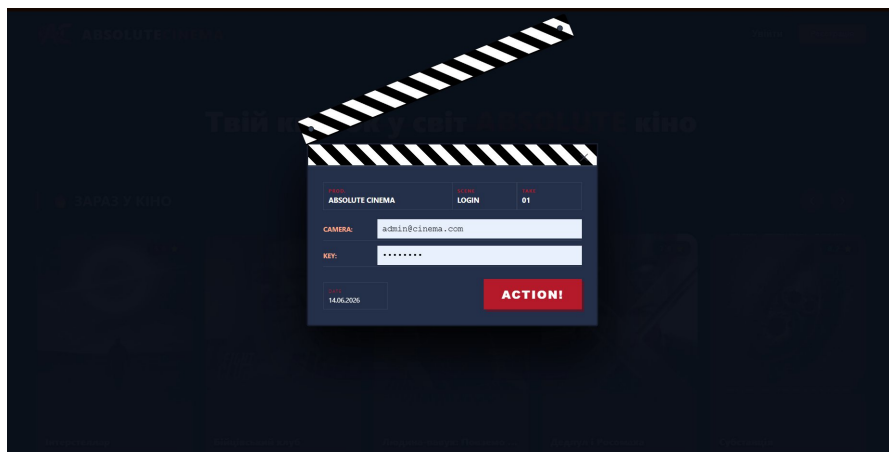


Рисунок 3.7 - Вікно авторизації користувача

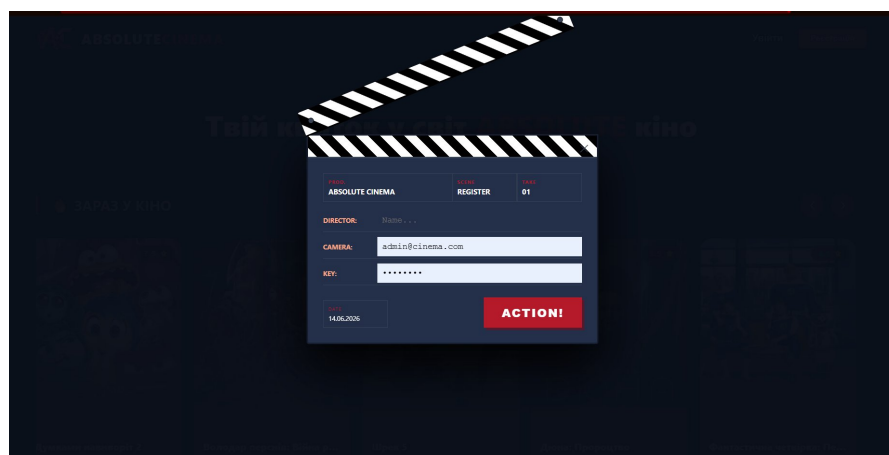


Рисунок 3.8 - Вікно реєстрації користувача

У разі спроби неавторизованого відвідувача взаємодіяти з інтерактивною схемою залу (клікнути на крісло) або перейти до етапу оплати, система блокує подальші дії та активує стан `showAuthModal`, відкриваючи інтерфейс авторизації. Форма вимагає введення електронної пошти (email) та пароля (password). Під час відправлення дані асинхронно відправляються методом POST на ендпоінт бекенду `/api/auth/login`. Сервер виконує пошук користувача в базі даних MongoDB, порівнює хешовані паролі за допомогою бібліотеки `bcryptjs` і, у разі успіху, генерує сесійний токен JWT (JSON Web Token) терміном дії на 1 добу. На фронтенді токен та базовий об'єкт користувача зберігаються у `localStorage`, після чого відбувається м'яке перезавантаження сторінки (`window.location.reload()`) для синхронного оновлення стану Хедера та відкриття доступу до вибору квитків.

		Сівченко М.Т.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		83

Рисунок 3.9 демонструє інтерфейс персоналізованої новинної стрічки для авторизованого користувача платформи. Після успішної автентифікації у навігаційній панелі з'являється іконка дзвіночка, натискання на яку відкриває випадające вікно «НОВИНИ» з актуальними сповіщеннями системи.

На відміну від загальнодоступного контенту, новинна стрічка авторизованого користувача формується динамічно на основі його підписок та придбаних квитків. У прикладі на рисунку відображаються три персоналізовані сповіщення про відкриття продажу квитків: на фільм «Бійцівський клуб» у залі OverHall, на фільм «Людина-павук: Повземо крізь Всесвіт» у залі OverHall та на фільм «Дедпул і Росомаха». Кожне сповіщення містить мініатюру постера фільму, заголовок повідомлення та текст з деталями сеансу і закликом до бронювання.

Технічно формування персоналізованої стрічки реалізовано через запит до ендпоінта /api/news з передачею JWT-токена. Сервер виконує вибірку документів колекції News за двома умовами: загальні публікації з isAlert: false та цільові сповіщення де поле targetUsers містить userId поточного користувача. Кнопка «ОЧИСТИТИ ВСЕ» у правому верхньому куті вікна дозволяє користувачу одночасно позначити всі сповіщення як прочитані, надсилаючи відповідний запит до серверного API.

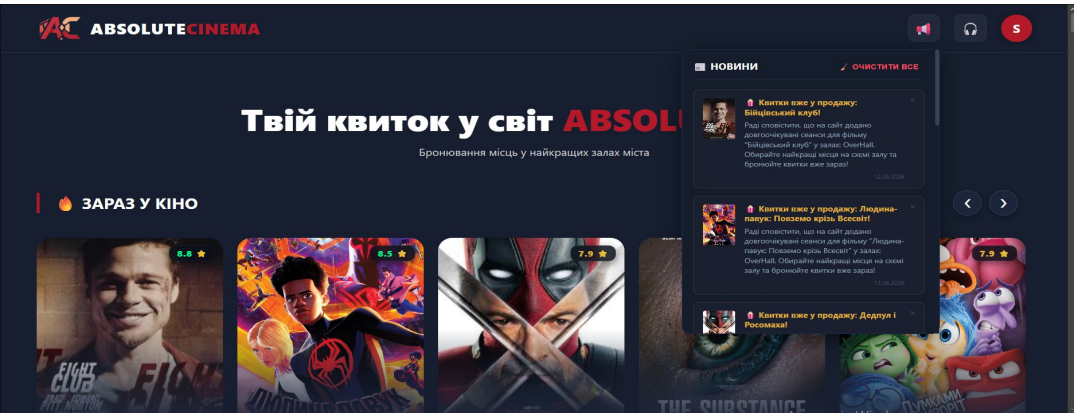


Рисунок 3.9 - Перелік новин для зареєстрованого користувача

Рисунок 3.10 ілюструє функціонал підписки на анонсовані фільми на прикладі кінострічки «Interstellar» режисера Крістофера Нолана тривалістю 169 хвилин. Сторінка фільму містить короткий синопсис та блок «СТАТУС

ПРЕМ'ЄРИ», який відображається замість стандартного розкладу сеансів у випадку коли для даної кінострічки ще не створено жодного активного показу в колекції Screening.

Блок статусу прем'єри інформує користувача повідомленням «На жаль, активних сеансів для цього фільму наразі немає, але ви можете підписатися на сповіщення» та пропонує жовту кнопку «ПІДПИСАТИСЯ НА СПОВІЩЕННЯ». При натисканні на кнопку авторизований користувач ініціює створення документа в колекції MovieSubscription через POST-запит до відповідного ендпоінта з передачею userId та movieId. Сервер перевіряє відсутність дублюючого запису для цієї пари ідентифікаторів та зберігає новий документ підписки зі значенням notified: false.

Після успішного оформлення підписки кнопка змінює свій стан на «Ви підписані», що реалізується через оновлення локального стану компонента React без перезавантаження сторінки. У момент коли адміністратор додає новий сеанс для цього фільму в систему, бекенд автоматично виконує вибірку всіх активних підписок з notified: false для відповідного movieId та генерує персоналізовані записи в колекції News з масивом targetUsers, що забезпечує доставку сповіщення кожному підписнику у їхню особисту новинну стрічку [1].

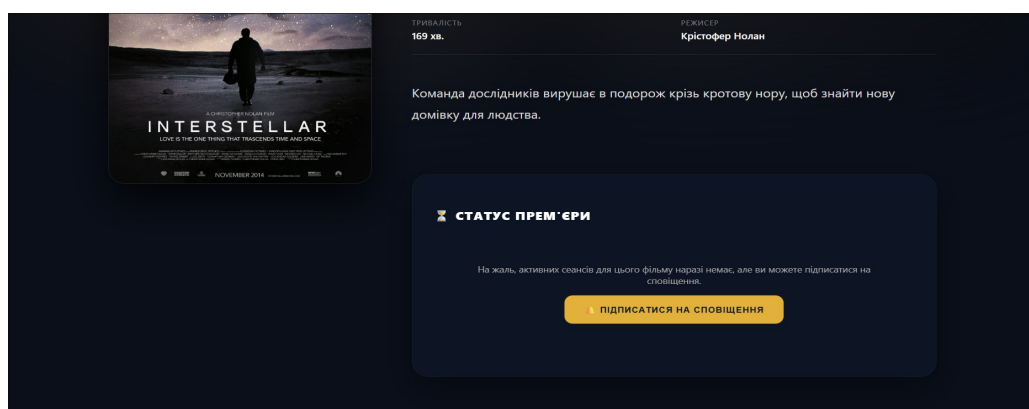


Рисунок 3.10 - Можливість підписатися на анонсовані фільми користувачем

Рисунок 3.11 демонструє роботу системи автоматичних сповіщень при скасуванні кіносеансу на прикладі фільму «Дюнами навпоріг 2». У новинній стрічці авторизованого користувача відображається критичне сповіщення з

		Сівченко М.Т.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		85

жовтою іконкою попередження та заголовком «СЕАНС СКАСОВАНО: "Дюнами навпоріг 2"».

Текст сповіщення інформує користувача про деталі скасування: «З технічних причин сеанс на фільм "Дюнами навпоріг 2" (3 липня о 21:00, Зал MainHall) скасовано. Ваша квитки анульовано, а компенсаційний промокод чекає у розділі "Мої квитки"». Таке детальне повідомлення формується автоматично на рівні серверної логіки при виконанні адміністратором операції масового скасування сеансу через алгоритм processBulkRefundForScreening. Технічно цей процес включає кілька послідовних операцій: сервер виконує вибірку всіх квитків зі статусом paid для скасованого сеансу, атомарно змінює їх статус на refunded, генерує індивідуальний компенсаційний промокод для кожного постраждалого користувача та створює персоналізовані записи в колекції News з масивом targetUsers що містить ідентифікатори власників анульованих квитків. Паралельно через Socket.io емітується подія оновлення схеми залу для звільнення всіх заброньованих місць скасованого сеансу. Поряд із критичним сповіщенням у стрічці відображаються також стандартні новини про відкриття продажу квитків на інші фільми, що підтверджує коректну роботу системи розмежування типів публікацій за прапорцем isAlert [1].

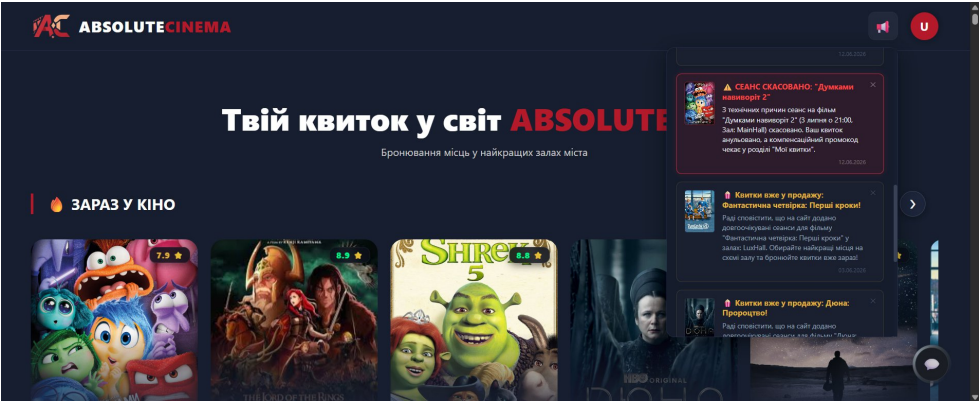


Рисунок 3.11 - Повідомлення про скасований сеанс для користувача

Рисунок 3.12 демонструє новинну стрічку користувача після відкриття продажу квитків на фільми, на які він раніше оформив підписку. У випадваючому вікні «НОВИНИ» відображаються три персоналізовані

сповіщення про старт продажу: на фільм «Інтерстеллар», «Бійцівський клуб» у залі OverHall та «Людина-павук: Повземо крізь Всесвіт».

Кожне сповіщення містить мініатюру постера фільму, заголовок «Квитки вже у продажу» з назвою кінострічки та текст з деталями додавання сеансу і закликом до бронювання. Поява цих сповіщень є результатом роботи автоматичного алгоритму системи підписок: у момент коли адміністратор створює новий сеанс для фільму, бекенд виконує вибірку всіх документів колекції MovieSubscription де movie збігається з ідентифікатором доданого фільму та notified має значення false. Для кожного знайденого підписника генерується окремий запис у колекції News з isAlert: false та відповідним userId у масиві targetUsers, після чого поле notified підписки оновлюється до значення true для запобігання повторній розсилці.

Кнопка «ОЧИСТИТИ ВСЕ» у правому верхньому куті дозволяє користувачу одночасно очистити всі прочитані сповіщення зі стрічки через відповідний DELETE-запит до серверного API, після чого список оновлюється без перезавантаження сторінки завдяки оновленню локального стану компонента React [1].

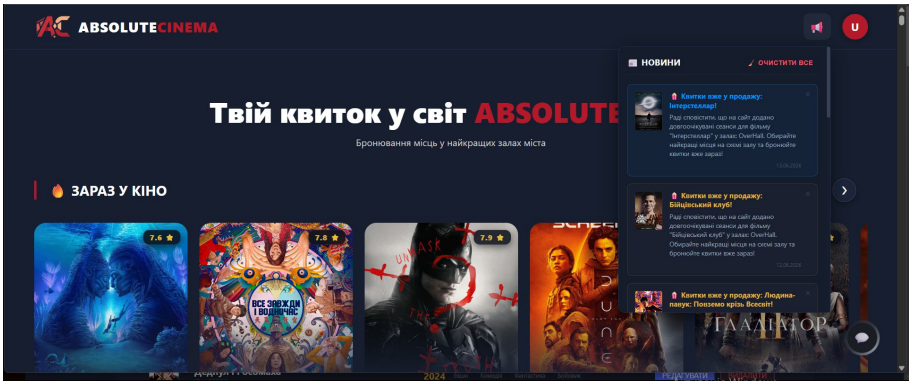


Рисунок 3.12 - Повідомлення про кінопрем'єру на яку підписався користувач

Після успішної автентифікації користувачу відкривається персоналізований розділ «Стрічка новин». На відміну від загальних публікацій, цей інтерфейс динамічно адаптується під відвідувача: крім глобальних маркетингових матеріалів, сюди виводяться таргетовані системні попередження, що стосуються поточного профілю.

Для фільмів зі статусом «Скоро у прокаті» доступна функція «Підписатися на оновлення». При її активації фронтенд фіксує зв'язок `userId` та `movieId` у колекції `MovieSubscription`, забезпечуючи користувачеві пріоритетне інформування про відкриття продажу квитків.

У разі скасування сеансу система запускає алгоритм `processBulkRefundForScreening`. Він автоматично скасовує квитки всіх постраждалих користувачів, виконуючи масове повернення коштів та звільняючи ресурси бази даних, що мінімізує негативний вплив на клієнтський досвід.

Рисунки 3.13 та 3.14 відображають інтерфейс особистого кабінету авторизованого користувача платформи. Сторінка профілю завантажується після автентифікації та формується на основі даних отриманих через GET-запит до ендпоінта `/api/users/profile` з передачею JWT-токена в заголовок запиту.

Верхній блок сторінки, зображений на рис. 3.13, містить аватар користувача з ініціалами, його нікнейм (`User5`), електронну адресу (`user5@cinema.com`) та рольовий індикатор CINEMAN. Нижче розміщено інтерактивний лічильник «ВСЬОГО КВИТКІВ» що відображає загальну кількість придбаних квитків - у даному прикладі 35. Натискання на лічильник ініціює перехід до детальної сторінки історії замовлень користувача.

Нижній блок профілю, зображений на рис. 3.14, містить додаткові відомості про активність користувача на платформі. Поле «ДАТА РЕЄСТРАЦІЇ» відображає місяць та рік створення облікового запису - «січень 2026 р.», що зчитується з поля `createdAt` документа колекції `User`. Секція «ВАШІ ДОСЯГНЕННЯ» відображає набір іконок-нагород що відображають активність користувача на платформі. Завершує сторінку кнопка «ВИЙТИ З АКАУНТУ», натискання на яку очищає JWT-токен з `localStorage` та скидає стан авторизації в клієнтському застосунку, повертаючи користувача до гостьового режиму без перезавантаження сторінки.

		Сівченко М.Т.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремов Ю.М.				88
Змн.	Арк.	№ докум.	Підпис	Дата		

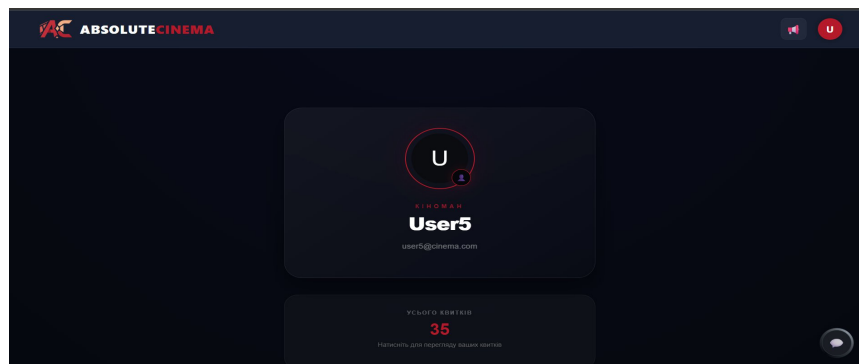


Рисунок 3.13 - Особистий акаунт користувача

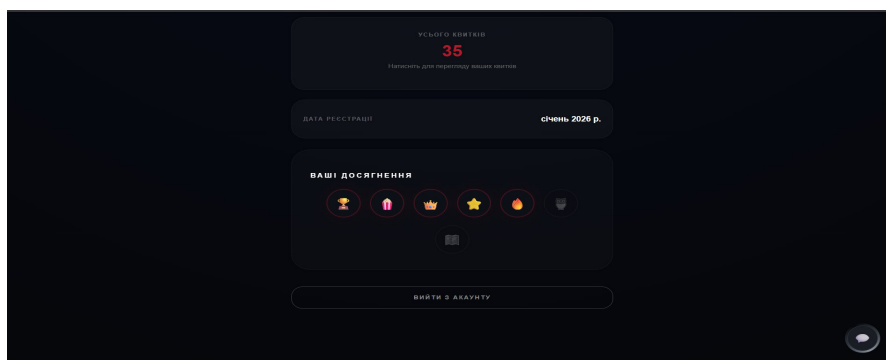


Рисунок 3.14 - Особистий акаунт користувача

Рисунок 3.15 демонструє інтерфейс сторінки «МОЇ КВИТКИ», яка надає користувачу повний доступ до історії його замовлень та активних квитків. Дані завантажуються через GET-запит до ендпоінта `/api/tickets/my` з передачею JWT-токена, після чого сервер виконує вибірку всіх документів колекції `Ticket` де поле `user` збігається з ідентифікатором поточного користувача, використовуючи метод `populate('screening')` для отримання деталей сеансу за один запит до бази даних.

Інтерфейс сторінки містить два рівні фільтрації квитків. Перший рівень - фільтр за статусом з вкладками «Всі», «Активні», «Використані» та «Повернуті», що дозволяє швидко знайти потрібний квиток залежно від стану поля `status` в документі колекції `Ticket`. Другий рівень - фільтр за категорією крісла з варіантами «Всі типи», «Standard» та «VIP», що фільтрує квитки за значенням поля `type`.

		Сівченко М.Т.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремов Ю.М.				89
Змн.	Арк.	№ докум.	Підпис	Дата		

Кожна картка квитка містить постер фільму, його назву («Мортал Комбат 2»), назву залу (OverHall), дату та час сеансу (12 липня о 18:00), номер ряду та місця (4/7 та 3/7), кнопку «ЗБЕРЕГТИ КВИТОК» для завантаження електронного квитка та унікальний QR-код для сканування на вході до кінозалу. QR-код генерується на основі поля orderId документа квитка та слугує засобом верифікації при проході глядача на сеанс, після чого поле isUsed квитка оновлюється до значення true.

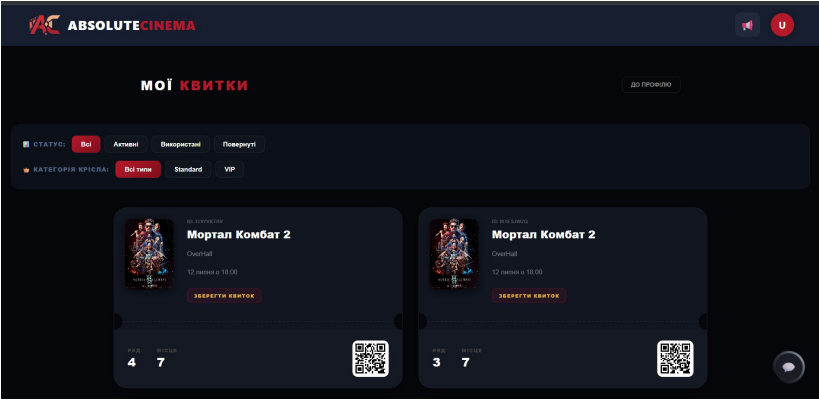


Рисунок 3.15 - Сторінка з усіма квитками користувача

Особистий кабінет користувача складається з декількох логічних блоків. Інформаційна панель відображає базові дані: ID, ім'я користувача та електронну адресу. Окрему увагу приділено модулю статистики profile-stats, який за допомогою асинхронних запитів виводить агреговані дані про активність користувача, обчислені в реальному часі на основі історії покупок.

Сторінка з історією замовлень забезпечує повний вивід придбаних квитків. Для оптимізації запитів до MongoDB застосовується агрегація через метод .populate('screening'). Це дозволяє за один етап отримати масив квитків із вкладеними деталями відповідних сеансів: назвою фільму, постером, часом показу та назвою кінозалу. Такий підхід мінімізує кількість запитів до бази даних, забезпечуючи високу швидкодію інтерфейсу при великих обсягах історії замовлень.

Рисунок 3.16 демонструє інтерфейс віджета SupportChat.jsx вбудованого модуля зв'язку з адміністрацією кінотеатру, який доступний авторизованому користувачу з будь-якої сторінки платформи через іконку

		Сівченко М.Т.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремов Ю.М.				90
Змн.	Арк.	№ докум.	Підпис	Дата		

чату у навігаційній панелі. Віджет відкривається як випадаюча панель у правій частині екрана поверх основного контенту без перезавантаження сторінки.

Панель містить дві вкладки: «ЧАТ» для ініціювання нового звернення або продовження активного діалогу та «ІСТОРІЯ» для перегляду архіву закритих тікетів підтримки. На початковому екрані відображається заголовок «З чим допомогти?» та пропозиція обрати категорію звернення. Система пропонує п'ять тематичних категорій: «Питання щодо оплати», «Технічна підтримка», «Квитки та сеанси», «Акаунт та підписка» та «Інше». Активна категорія «Квитки та сеанси» виділена червоним кольором, що сигналізує про її поточний вибір.

Технічно перед відображенням інтерфейсу вибору категорії компонент `SupportChat.jsx` виконує перевірку статусу `isBannedSupport` поточного користувача через запит до серверного API. У разі якщо адміністратор заблокував доступ користувача до чату підтримки, замість інтерфейсу вибору категорії відображається відповідне повідомлення про обмеження. При виборі категорії та надсиланні першого повідомлення клієнтський застосунок ініціює POST-запит до ендпоінта `/api/chat/send` з передачею теми звернення та тексту повідомлення, що запускає процес створення нового тікета в колекції `ChatTicket` [2].

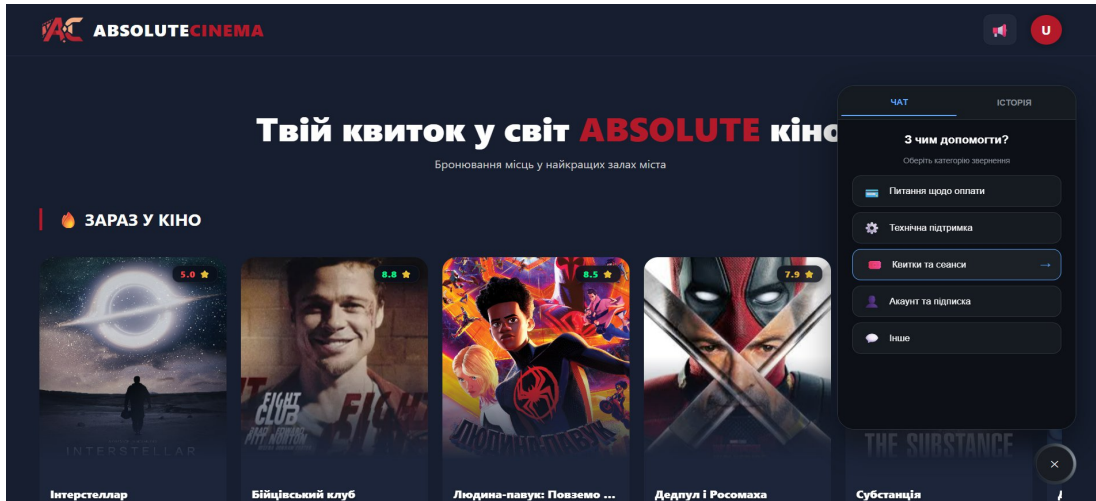


Рисунок 3.16 - Вікно зв'язку з адміністрацією сайту

		Сівченко М.Т.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремов Ю.М.				91
Змн.	Арк.	№ докум.	Підпис	Дата		

Рисунок 3.17 ілюструє початковий етап створення звернення користувачем після вибору категорії «Квитки та сеанси» у віджеті підтримки. Інтерфейс чату переходить до активного режиму діалогу, відображаючи обрану категорію у вигляді тегу у верхній частині панелі та перше повідомлення користувача «Я хочу повернути квиток» у вигляді бульбашки повідомлення з правого боку чату.

Нижня частина панелі містить поле введення тексту з підказкою «Напишіть нам...» та кнопку відправки повідомлення. Інтерфейс реалізує механізм «оптимістичного оновлення» (Optimistic UI) - повідомлення відображається в чаті миттєво після натискання кнопки відправки, ще до отримання підтвердження від сервера, що створює відчуття миттєвого відгуку системи.

Технічно при відправці першого повідомлення клієнтський застосунок надсилає POST-запит до ендпоінта `/api/chat/send` з параметрами `text` (текст повідомлення), `subject` (обрана категорія «Квитки та сеанси») та JWT-токеном в заголовок. Сервер декодує токен для отримання `userId`, перевіряє відсутність активного тікета для цього користувача та створює новий документ у колекції `ChatTicket` зі статусом `open` та першим повідомленням у масиві `messages`. Після збереження документа в `MongoDB` тікет автоматично потрапляє до черги звернень в адміністративній панелі `AdminSupport.jsx`, очікуючи на призначення відповідального менеджера [2].

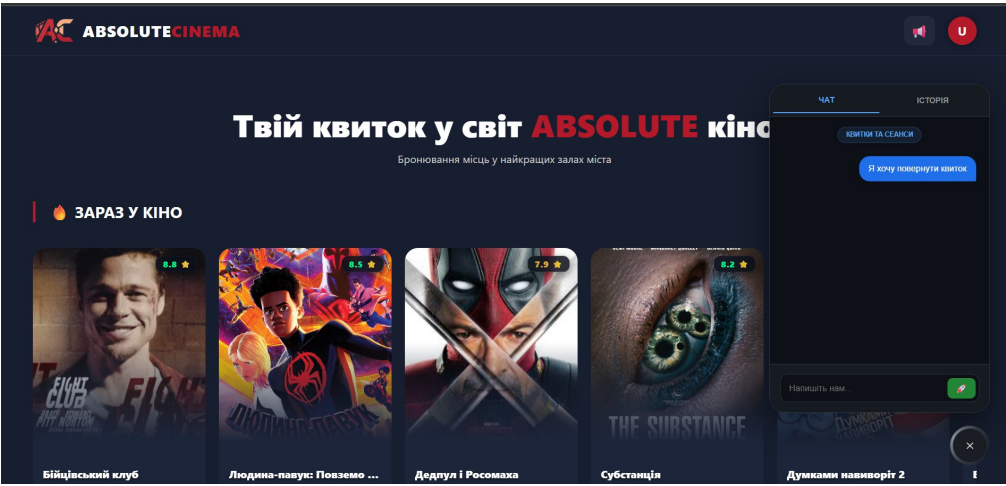


Рисунок 3.17 - Звернення користувача до адміністрації

		Сівченко М.Т.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремов Ю.М.				92
Змн.	Арк.	№ докум.	Підпис	Дата		

Рисунок 3.18 демонструє інтерфейс адміністративного модуля Support Hub - центру обробки звернень користувачів, доступного виключно для облікових записів з роллю admin. Панель розділена на три функціональні зони що забезпечують повний контроль над процесом опрацювання запитів.

Ліва колонка містить два розділи: «В ОЧІКУВАННІ (0)» для нових тікетів що ще не взяті в обробку та «МОЇ ЧАТИ (1)» для активних діалогів поточного адміністратора. У розділі активних чатів відображається тікет користувача User5 з категорією «Квитки та сеанси» та превью останнього повідомлення «Вітаю! Я прийняв ваше звернення. Чи...», виділений зеленою індикаторною точкою що сигналізує про активний статус діалогу.

Центральна зона відображає повний лог переписки з користувачем. Повідомлення «Я хочу повернути квиток» від користувача відображається зліва, тоді як автоматичне привітання «Вітаю! Я прийняв ваше звернення. Чим можу допомогти?» від адміністратора - справа у жовтій бульбашці. У верхній частині центральної зони розміщено кнопку «ЗАКРИТИ ТІКЕТ» для завершення діалогу після вирішення питання.

Права колонка «АКТИВНІ КВИТКИ КЛІЄНТА» є ключовим інструментом модуля компенсації. Система автоматично виконує GET-запит до /api/chat/admin/user-tickets/{userId} та відображає всі активні квитки користувача: «Супермен» (Зал IMAX, Ряд 5, Місце 4, 200 грн), «Думками навиворіт 2» (Зал Main Hall, Ряд 6, Місце 9, 150 грн) та «Думками навиворіт 2» (Зал Main Hall, Ряд 6, Місце 8, 150 грн). Під кожним квитком розміщено червону кнопку «ОФОРМИТИ ПОВЕРНЕННЯ» для ініціювання процедури компенсації безпосередньо з інтерфейсу чату [2].

Така триколонкова компоновка інтерфейсу реалізує принцип єдиної точки керування інцидентом: адміністратор одночасно веде діалог із клієнтом у центральній зоні та має миттєвий доступ до всіх його активних квитків у правій панелі, не залишаючи сторінки підтримки. Це унеможливорює необхідність паралельного пошуку інформації в окремих розділах адмін-панелі та скорочує час опрацювання кожного звернення. Архітектурно права

		Сівченко М.Т.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		93

колонка функціонує як вбудований мікромодуль із власним циклом запитів до бекенду, що автоматично оновлює перелік квитків при зміні статусу діалогу, забезпечуючи актуальність даних протягом усього сеансу підтримки.

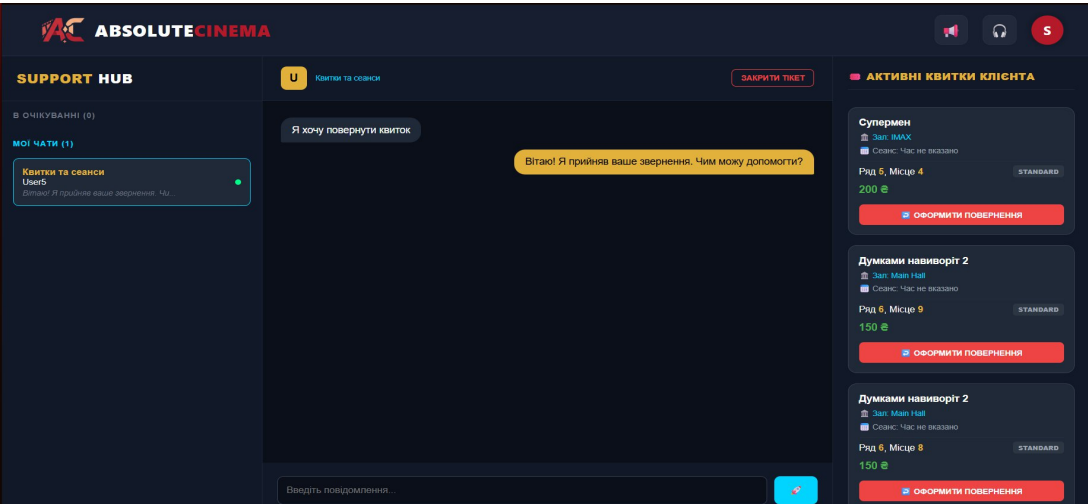


Рисунок 3.18 - Вигляд зі сторони адміністратора на звернення

Рисунок 3.19 ілюструє інтерфейс чату підтримки з точки зору користувача після того як адміністратор прийняв звернення в обробку. На відміну від адміністративної панелі Support Hub, користувач бачить спрощений інтерфейс діалогу у вигляді випадаючої панелі з правого боку екрана з вкладками «ЧАТ» та «ІСТОРИЯ».

У верхній частині панелі відображається тег обраної категорії «КВИТКИ ТА СЕАНСИ». Хронологія переписки містить три повідомлення: початкове звернення користувача «Я хочу повернути квиток» у синій бульбашці справа, автоматичне привітання від адміністратора «Вітаю! Я прийняв ваше звернення. Чим можу допомогти?» у темній бульбашці зліва та уточнювальне запитання «Який саме квиток і на який сеанс ви хочете повернути?» також від адміністратора.

Синхронізація повідомлень між інтерфейсом користувача та адміністративною панеллю реалізована через механізм періодичного оновлення з інтервалом в одну секунду за допомогою функції fetchData. Нові повідомлення від адміністратора з'являються в чаті користувача автоматично без необхідності перезавантаження сторінки або виконання додаткових дій.

Завдяки механізму «оптимістичного оновлення» власні повідомлення користувача відображаються миттєво після відправки, створюючи відчуття плавного та безперервного діалогу в реальному часі [2].

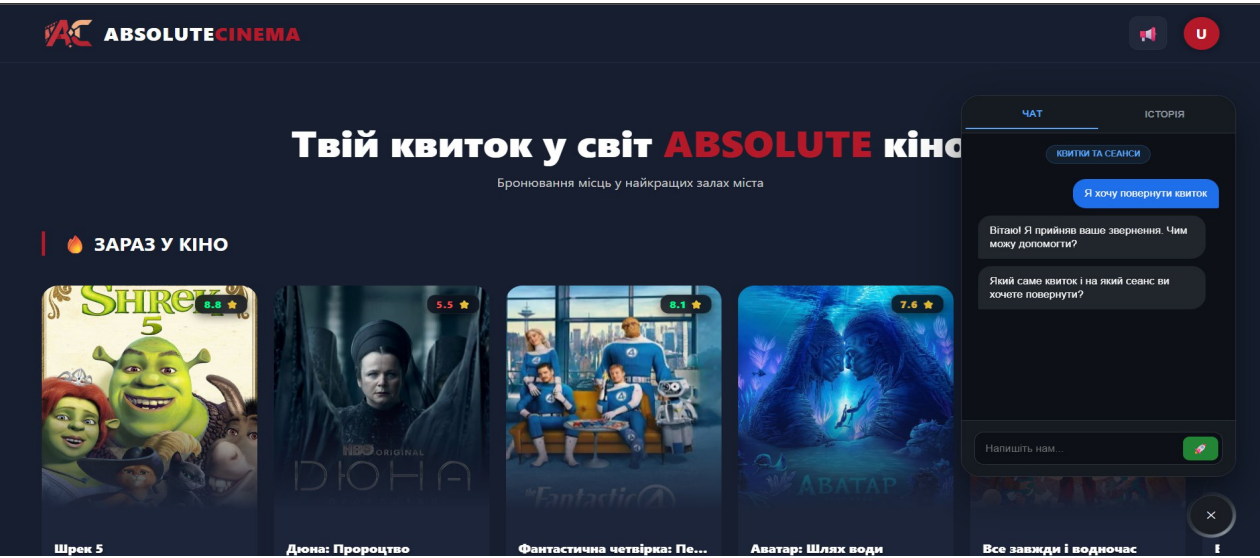


Рисунок 3.19 - Вигляд зі сторони користувача на відповідь від адміністрації

Рисунок 3.20 демонструє продовження діалогу між користувачем та адміністратором у чаті підтримки на етапі уточнення деталей звернення. Після отримання уточнювального запитання від адміністратора «Який саме квиток і на який сеанс ви хочете повернути?» користувач надає конкретну інформацію про квиток що підлягає поверненню - два послідовні повідомлення у синіх бульбашках справа: «Мортал Комбат 2, 18 липня о 18:00» та «Ряд 4 Місце 7». Розділення інформації на два окремі повідомлення є природною поведінкою користувача в чат-інтерфейсі. Кожне повідомлення обробляється сервером як окремий запит до ендпоінта `/api/chat/send` та додається як новий субдокумент до масиву `messages` відповідного тікета в колекції `ChatTicket`. Поле `updatedAt` документа тікета автоматично оновлюється при кожному новому повідомленні завдяки опції `{ timestamps: true }` схеми `Mongoose`, що дозволяє адміністратору сортувати чергу активних звернень за часом останньої активності.

На стороні адміністративної панелі `Support Hub` нові повідомлення користувача з'являються в реальному часі завдяки механізму періодичного

оновлення. Отримавши достатньо інформації для ідентифікації квитка, адміністратор може перейти до правої колонки «АКТИВНІ КВИТКИ КЛІЄНТА» де відображаються всі придбані квитки користувача, знайти відповідний квиток на «Мортал Комбат 2» та ініціювати процедуру повернення через кнопку «ОФОРМИТИ ПОВЕРНЕННЯ» [2].

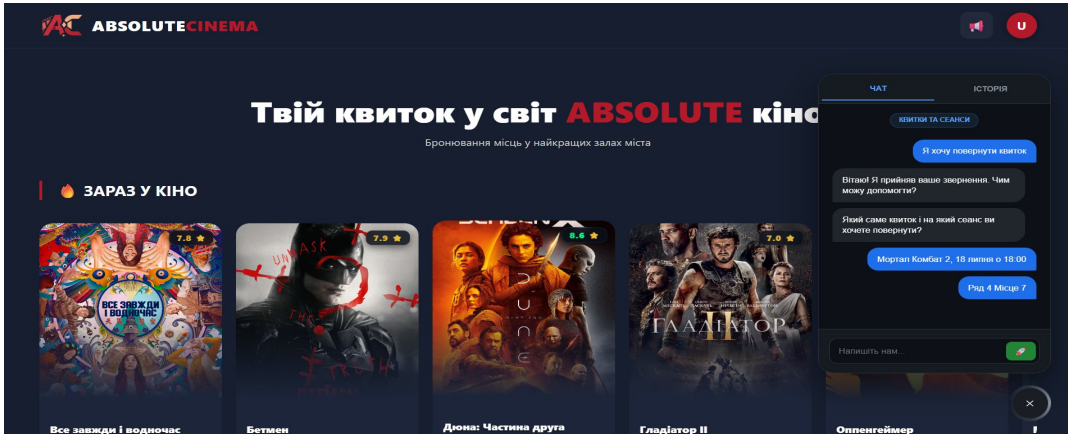


Рисунок 3.20 - Опис проблеми

Рисунок 3.21 ілюструє модальне вікно підтвердження операції повернення квитка, яке відкривається в адміністративній панелі Support Hub після натискання кнопки «ОФОРМИТИ ПОВЕРНЕННЯ» в картці відповідного квитка. Вікно містить заголовок «Підтвердження повернення», інформаційний блок з назвою фільму («Мортал Комбат 2»), номером ряду та місця («Ряд 4, Місце 7 (STANDARD)»), а також попереджувальний текст: «Ви впевнені, що хочете анулювати цей квиток? Клієнту в чат буде автоматично надіслано сервісне повідомлення, а компенсаційний промокод з'явиться в його кабінеті». Адміністратору пропонується два варіанти дії: скасування операції через кнопку «Скасувати» або підтвердження повернення через червону кнопку «Оформити повернення». Така реалізація запобіжного діалогу (confirmation dialog) унеможливорює випадкове анулювання квитка та інформує адміністратора про подальші автоматичні наслідки операції - сповіщення клієнта та генерацію промокода - ще до фактичного виконання транзакції на сервері.

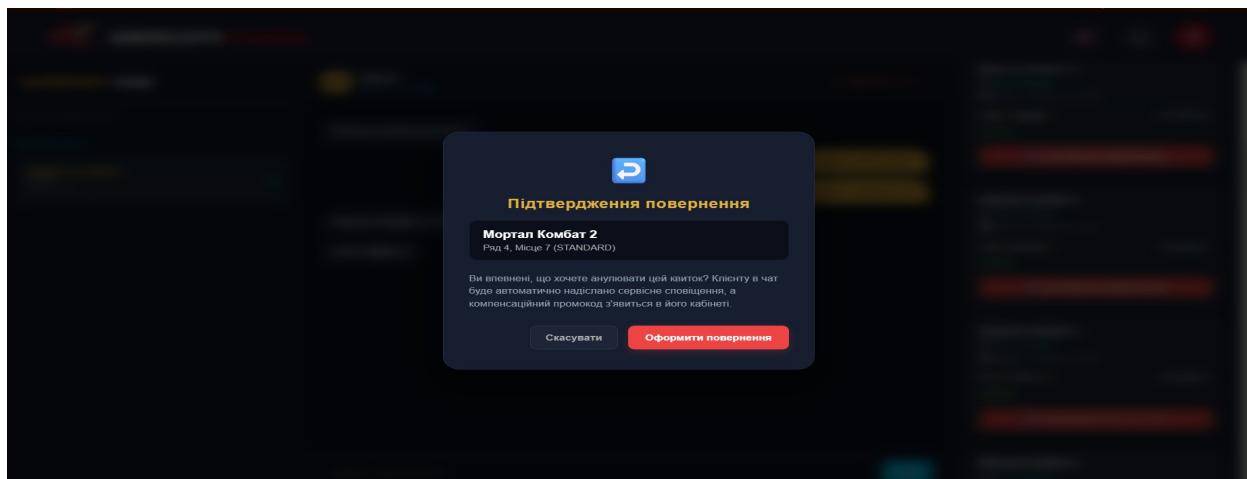


Рисунок 3.21 - Процес повернення квитка для користувача

Рисунок 3.22 ілюструє результат успішного виконання операції повернення квитка з точки зору користувача. Після підтвердження адміністратором операції в адміністративній панелі (рис. 3.21) у чаті користувача автоматично з'являється системне повідомлення від «Асистент AbsoluteCinema» наступного змісту: «Ваше повернення успішно оформлено! Квиток на фільм "Мортал Комбат 2" (Ряд 4, Місце 7, Категорія: STANDARD) анульовано. Ми вже згенерували для вас компенсаційний промокод на STD-місце. Його надіслано у ваш особистий кабінет (розділ "Мої квитки"). Чекаємо на вас на наступних прем'єрах!».

Технічно дане повідомлення формується на стороні сервера автоматично, без участі адміністратора, у момент успішного завершення транзакційної операції анулювання квитка. Бекенд додає системний субдокумент до масиву messages відповідного тікета в колекції ChatTicket із позначкою службового відправника (наприклад, sender: 'system' або 'admin-auto'), що відрізняє його від звичайних повідомлень, написаних адміністратором вручну. Завдяки механізму періодичного оновлення інтерфейсу чату (fetchData з інтервалом в одну секунду) це повідомлення з'являється в чаті користувача в реальному часі без необхідності оновлення сторінки.

		Сівченко М.Т.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		97

Окрім текстового підтвердження, дана подія супроводжується паралельними серверними операціями: зміною статусу квитка на refunded у колекції tickets, звільненням заброньованого місця у відповідній колекції screenings та генерацією унікального компенсаційного промокода, який стає доступним користувачеві в розділі «Мої квитки» особистого кабінету.

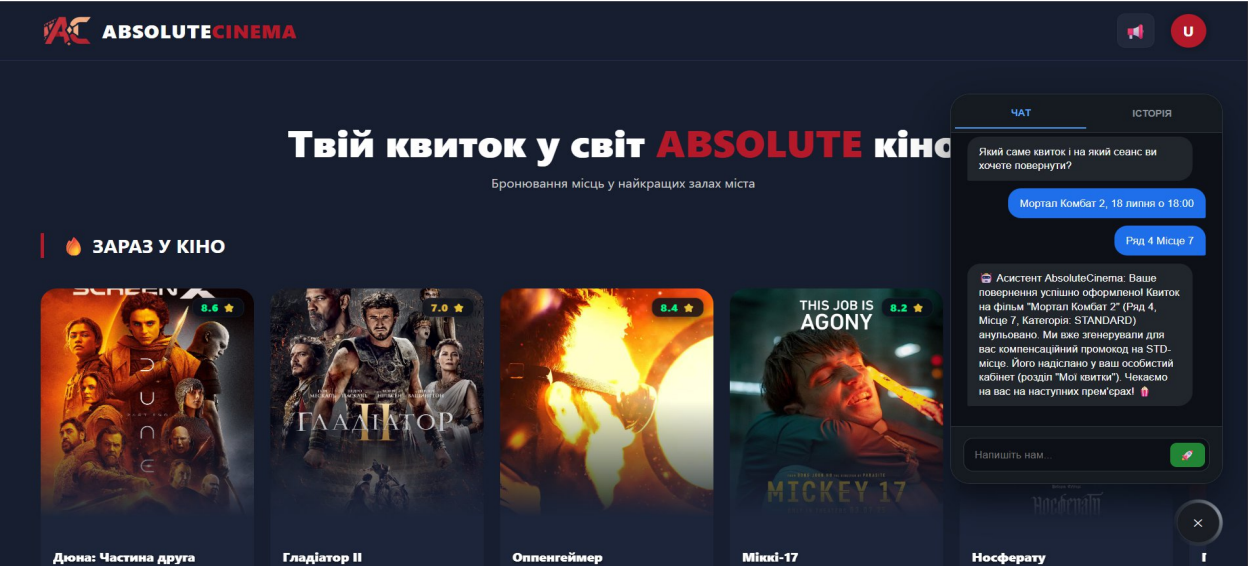


Рисунок 3.22 - Повернення квитка користувачу і вивід повідомлення

Рисунок 3.23 демонструє відображення анульованого квитка в особистому кабінеті користувача після успішного оформлення повернення адміністратором. Картка квитка на фільм «Мортал Комбат 2» (зал OverHall, 18 липня о 18:00, ряд 4, місце 7) отримує діагональний червоний штамп «ПОВЕРНУТО (STANDARD)», що візуально сигналізує про зміну статусу документа з paid на refunded у колекції tickets. Нижче картки відображається блок компенсаційного промокоду з унікальним кодом у форматі REFUND-STD-TSRCMV-9131, терміном дії до 14 вересня 2026 року та позначкою «АНУЛЬОВАНО» у правому нижньому куті. Аналогічну структуру мають і попередні картки з промокодами REFUND-STD-ONLYNB-5002 та REFUND-STD-5G8T8M-9597, що підтверджує коректну роботу алгоритму масової генерації компенсацій. Промокод зберігається у відповідному полі документа колекції tickets та стає доступним користувачеві одразу після завершення транзакційної операції на сервері без необхідності оновлення сторінки.

		Сівченко М.Т.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Сфремов Ю.М.				98
Змн.	Арк.	№ докум.	Підпис	Дата		

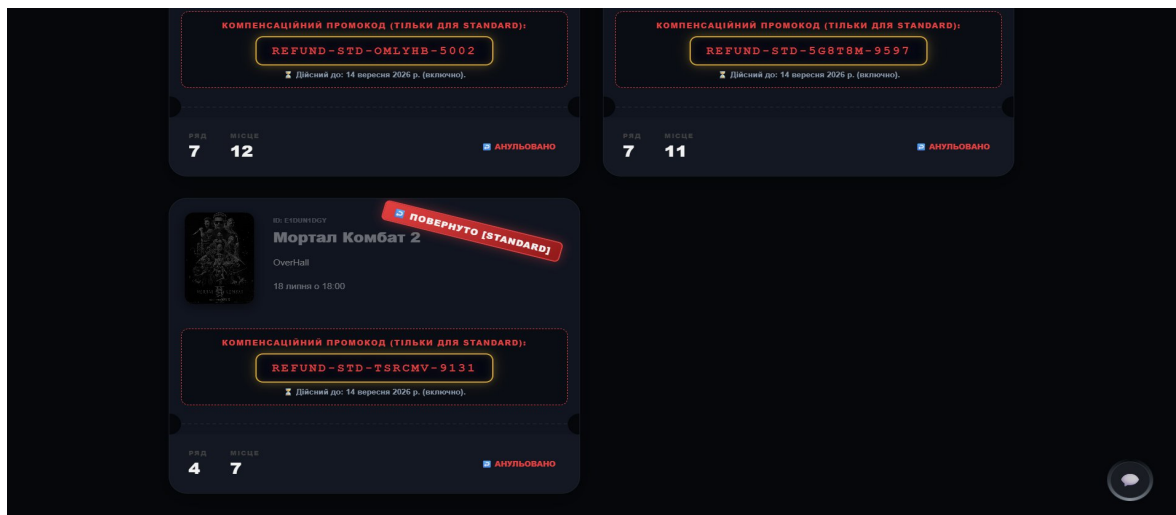


Рисунок 3.23 - Повернутий квиток з промокодом

Процес ініціалізації звернення описано на рисунках 3.16 – 3.17. Користувач відкриває віджет SupportChat.jsx, де система перевіряє статус isBannedSupport. У штатному режимі клієнт обирає тему звернення (оплата, квитки, технічні збої), що ініціює створення документа в колекції ChatTicket зі статусом waiting.

Обробка запиту на стороні адміністратора відображена на рисунку 3.18. В адмін-панелі AdminSupport.jsx запити оновлюються в реальному часі. Після натискання «ВЗЯТИ В РОБОТУ» надсилається POST-запит, який закріплює тикет за adminId, змінює статус на active та автоматично надсилає привітання.

Логіка діалогу та розв'язання проблеми представлена на рисунках 3.19 – 3.20. Синхронізація клієнтських інтерфейсів здійснюється з інтервалом в 1 секунду. Завдяки механізму «оптимістичного оновлення» (Optimistic UI) повідомлення відображаються миттєво. Завершення діалогу адміністратором переводить тикет у статус closed і переносить його до архіву.

Ініціалізація процедури повернення коштів ілюструється на рис. 3.21. Під час діалогу система автоматично виконує GET-запит до /api/chat/admin/user-tickets/\${targetUserId}, вилучаючи всі активні квитки користувача на майбутні сеанси, які відображаються у бічній панелі адміністратора.

Процес анулювання квитка та генерації компенсації (див. рис. 3.21 – 3.23) активується кнопкою «ОФОРМИТИ ПОВЕРНЕННЯ». Після підтвердження бекенд виконує транзакційну операцію: змінює статус квитка на refunded, генерує унікальний компенсаційний промокод та звільняє заброньоване місце в системі. Атомарність операції повернення гарантує цілісність даних - у разі збою на будь-якому етапі транзакція відкочується, запобігаючи виникненню неузгодженого стану між колекціями tickets та screenings. Коректність реалізації цього ланцюжка була верифікована в ході функціонального тестування за допомогою інструменту Postman [28].

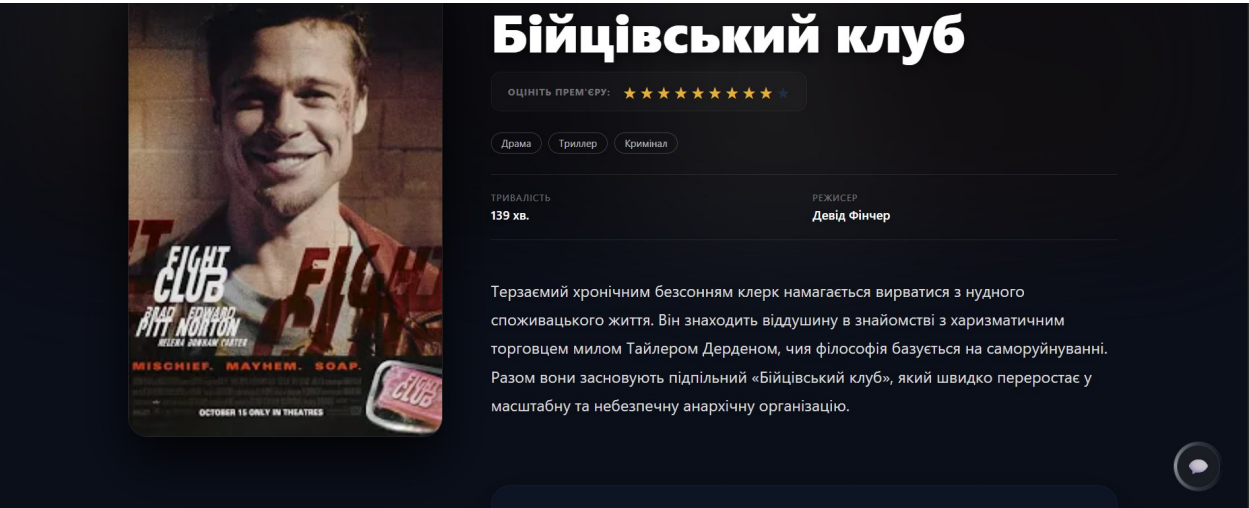


Рисунок 3.24 - Сторінка фільму

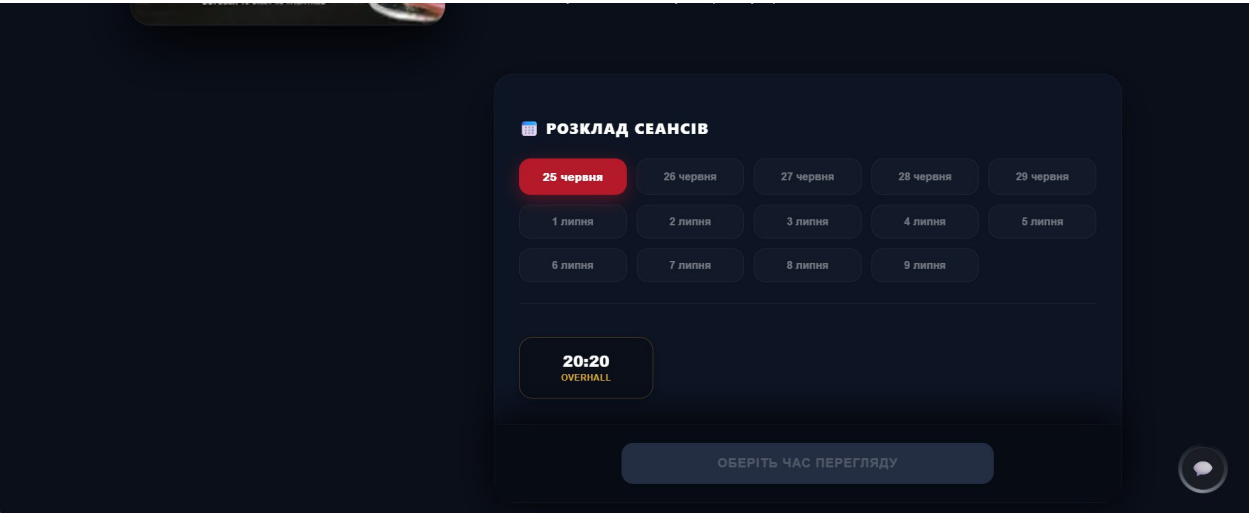


Рисунок 3.25 - Продовження сторінки фільму

		Сівченко М.Т.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремов Ю.М.				100
Змн.	Арк.	№ докум.	Підпис	Дата		

Динамічна сторінка фільму є ключовим інтерфейсом платформи «AbsoluteCinema», призначеним для надання вичерпної інформації про стрічку, візуалізації медіаконтенту та переходу до вибору сеансів.

Візуальне оформлення та інформаційна панель, представлені на рис. 3.24, реалізовані через динамічне зчитування параметрів URL (movieSlug або movieId) за допомогою хука useParams. Після монтування компонента ініціюється асинхронний GET-запит до /api/movies/{id}, результати якого рендеряться у відповідні блоки інтерфейсу.

Медіа-панель, відображає широкоформатний фоновий банер (backdrop) із розмиттям та вертикальний постер фільму (posterUrl) високої роздільної здатності. Текстовий блок метаданих, містить назву фільму, рік випуску (year), режисера (director), тривалість стрічки у хвилинах та перелік жанрів, оформлених у вигляді стилізованих тегів (badges). Сюжетна лінія (Synopsis), детальний опис кінострічки для ознайомлення відвідувача з фабулою фільму.

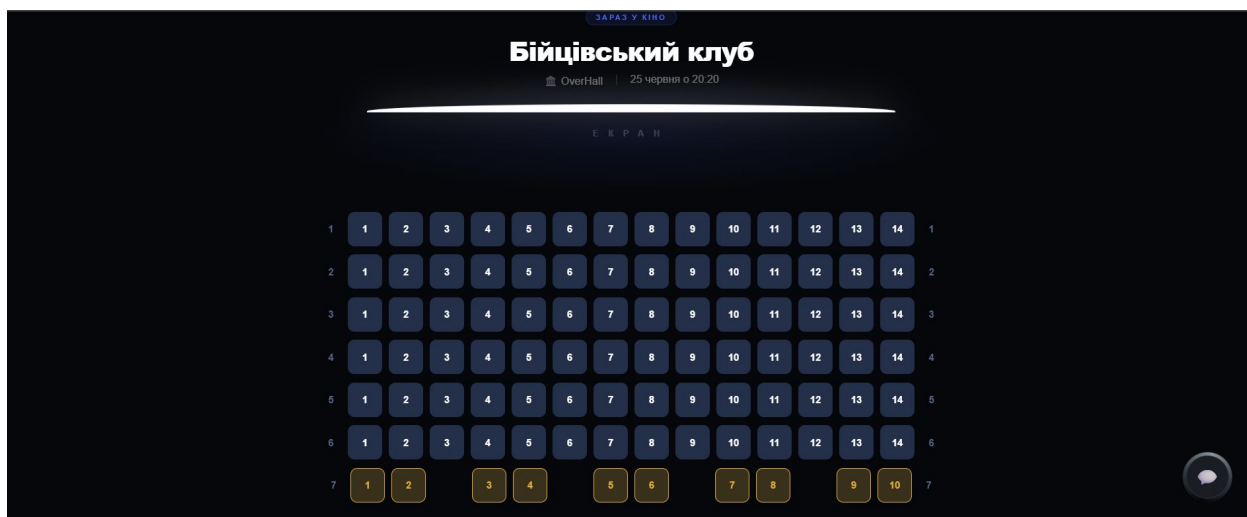


Рисунок 3.26 - Сторінка вибору місць на сеанс кіно

Модуль вибору місць є найбільш технологічно складною частиною інтерфейсу «AbsoluteCinema». Він забезпечує візуалізацію унікальної конфігурації залу, динамічний розрахунок вартості квитків та захист від конфліктів одночасного бронювання (Race Condition).

Після відкриття компонента Booking.jsx виконується запит до API для отримання поточної структури сеансу. На основі отриманого масиву крісел

		Сівченко М.Т.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремов Ю.М.				101
Змн.	Арк.	№ докум.	Підпис	Дата		

алгоритм фронтенду за допомогою `Math.max()` вираховує габарити залу (кількість рядів та місць), після чого через `Array.from()` генерує двовимірну матрицю для відображення. Кожне крісло рендериться як окремий інтерактивний HTML-елемент, чий візуальний стан, колір та доступність для взаємодії, визначається на основі статусу, зчитаного з бази даних. Цей механізм гарантує користувачеві актуальну картину зайнятості місць та зручний вибір позицій відповідно до їх категорії (VIP або STD). Компонентна архітектура React дозволяє ізолювати логіку відображення кожного крісла в окремому елементі, що спрощує керування станом інтерфейсу та забезпечує високу швидкість відгуку при одночасному оновленні великої кількості елементів схеми залу [19].

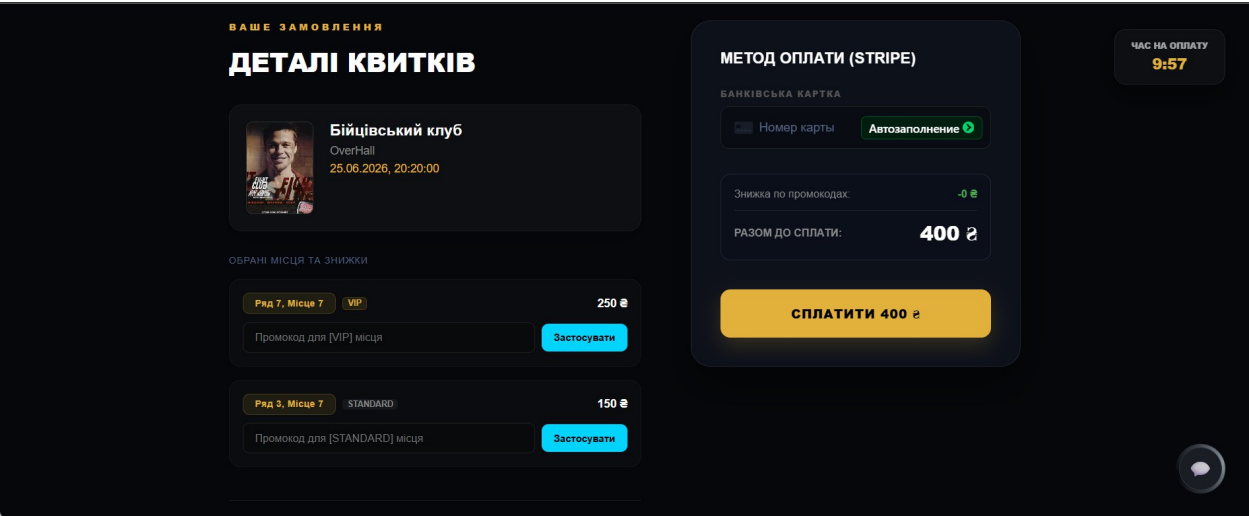


Рисунок 3.27 - Сторінка оплати квитків

Безпека фінансових транзакцій у системі «AbsoluteCinema» реалізована через інтеграцію з міжнародним платіжним шлюзом Stripe. Процес оплати (див. рис. 3.27) розділений на два етапи, що унеможливорює обробку або зберігання конфіденційних даних картки на серверах кінотеатру.

Ініціалізація на бекенді: під час переходу до оплати фронтенд надсилає POST-запит на ендпоінт `/api/screenings/create-payment-intent`, передаючи загальну вартість замовлення. Бекенд-сервер взаємодіє з API Stripe, генерує унікальний `clientSecret` і повертає його на клієнтську частину.

Безпечне введення даних: форма введення реквізитів картки будується за допомогою захищених `iframe`-компонентів `@stripe/react-stripe-js`.

		Сівченко М.Т.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремов Ю.М.				102
Змн.	Арк.	№ докум.	Підпис	Дата		

Рисунок 3.28 - Заповнені дані для оплати

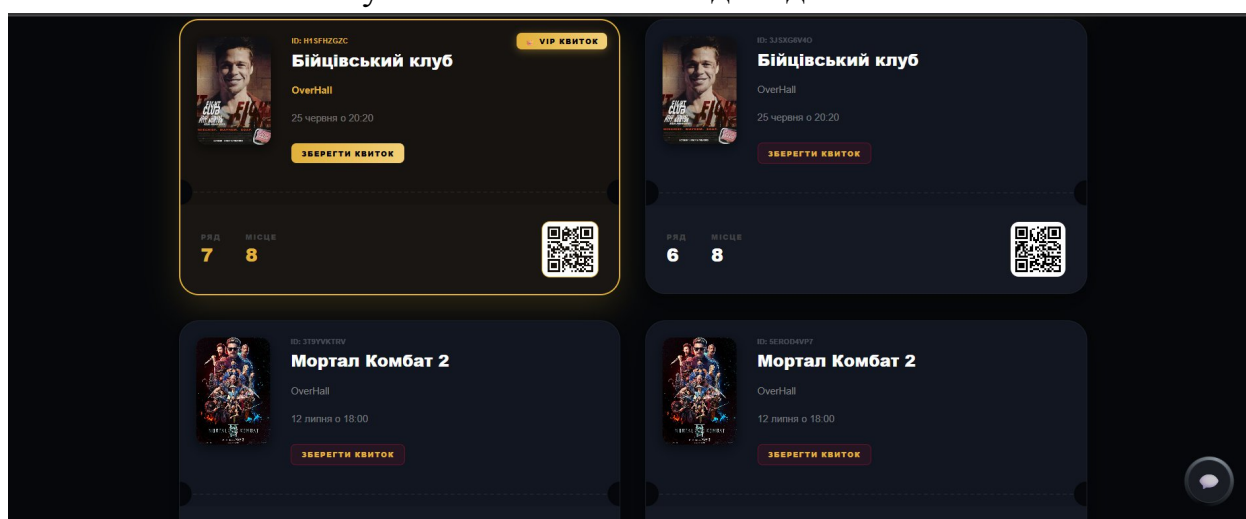


Рисунок 3.29 - Поява квитків в профілі після успішної покупки

Процес введення та валідації даних (рис. 3.28) реалізований через CardElement від Stripe, який забезпечує моніторинг дій користувача в реальному часі. Бібліотека Stripe.js автоматично перевіряє формат даних за допомогою алгоритму Луна та інших валідаторів. У разі синтаксичних помилок інтерфейс змінює стан на критичний, виводячи повідомлення про помилку та блокуючи відправку запиту.

Після натискання кнопки «СПЛАТИТИ» компонент PaymentUIComponent через State Pattern переводить локальний стан isProcessing у значення true, що активує компонент LoadingOverlay та блокує повторне надсилання форми. Такий підхід реалізує принцип відповідальності UI - візуальний стан компонента повністю відокремлений від бізнес-логіки транзакції та керується виключно результатом виклику

		Сівченко М.Т.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремов Ю.М.				103
Змн.	Арк.	№ докум.	Підпис	Дата		

`stripe.confirmCardPayment()`. У разі синтаксичних помилок у реквізитах картки бібліотека `Stripe.js` активує вбудований валідатор на основі алгоритму Луна, повертаючи об'єкт помилки без відправки запиту на сервер. Далі хук `stripe.confirmCardPayment()` передає токеновані дані на сервер Stripe для авторизації коштів та проходження протоколу 3D Secure.

Логіка завершення транзакції побудована за асинхронною моделлю обробки подій (рис. 3.29). Після підтвердження платежу на стороні Stripe сервер отримує підписану подію `payment_intent.succeeded` через Webhook-хук у вигляді POST-запиту на захищений ендпоінт `/api/payments/webhook`. Перед обробкою бекенд верифікує цифровий підпис події за допомогою `STRIPE_WEBHOOK_SECRET`, що унеможливлює підробку запитів. Після успішної верифікації асинхронно викликається `TransactionService.processPayment()`, який атомарно створює документ квитка у колекції `tickets` зі статусом `paid` та звільняє тимчасове блокування місця в колекції `screenings`.

Рисунок 3.30 відображає головний екран адміністративної панелі керування каталогом фільмів (`AdminDashboard.jsx`). Інтерфейс побудовано за модульним принципом із використанням системи динамічних табів. У центральній частині розміщено інтерактивну таблицю з переліком усіх кінострічок, що містить мініатюру постера, унікальний ідентифікатор документа MongoDB, назву фільму, рік випуску, жанри та кнопки керування - редагування і видалення запису. У верхній частині інтерфейсу розміщено пошуковий рядок для миттєвої фільтрації таблиці за назвою, роком, режисером або жанром без перезавантаження сторінки.

Впровадження Webhook-механізму як єдиної точки підтвердження транзакції є принциповим архітектурним рішенням з точки зору фінансової безпеки системи. На відміну від підходу, де фронтенд самостійно сигналізує серверу про успішний платіж, Webhook-модель гарантує, що квиток створюється виключно на підставі верифікованої події від Stripe, підписаної секретним ключем `STRIPE_WEBHOOK_SECRET`. Це унеможливлює

		Сівченко М.Т.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		104

маніпуляції з боку клієнта, наприклад підробку запиту на створення квитка без реального списання коштів. Таким чином, навіть у разі переривання з'єднання між браузером користувача та сервером після підтвердження платежу на стороні Stripe, механізм Webhook автоматично завершить транзакцію та створить квиток у базі даних, забезпечуючи цілісність фінансових операцій незалежно від стану клієнтського з'єднання.

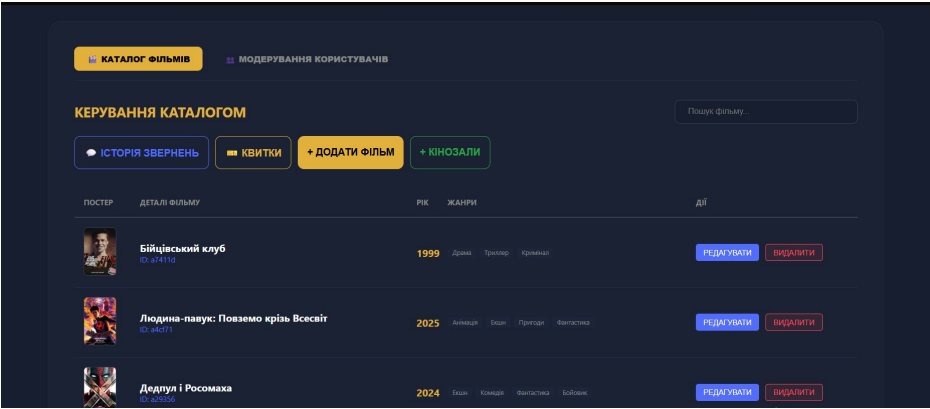


Рисунок 3.30 - Вигляд адмін-панелі

Рисунок 3.31 ілюструє інтерфейс центру обробки та архіву звернень служби підтримки в адміністративній панелі. Сторінка відображає зведену таблицю всіх тікетів системи із зазначенням ідентифікатора звернення, нікнейму та електронної адреси клієнта, обраної категорії запиту, поточного статусу тікета (відкритий, активний або закритий) та дати останнього оновлення. Адміністратор має можливість перейти до повного логу переписки з будь-яким користувачем для аналізу якості обслуговування або вирішення спірних ситуацій. Дані оновлюються в реальному часі за допомогою механізму WebSocket-з'єднання.

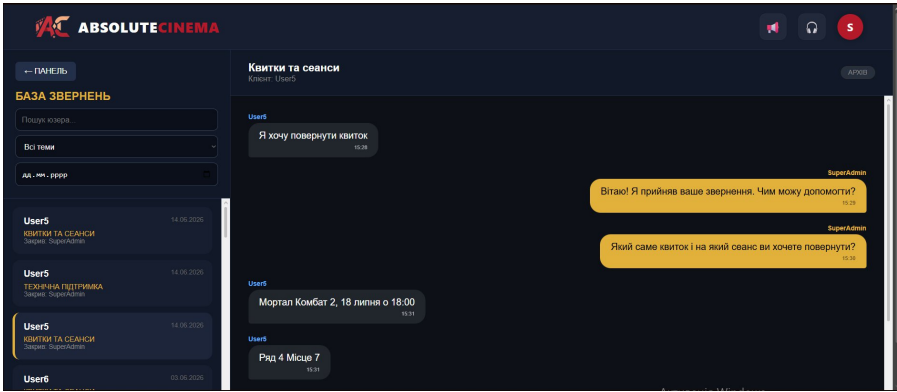


Рисунок 3.31 - Сторінка з історією звернень користувачів

		Сівченко М.Т.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремов Ю.М.				105
Змн.	Арк.	№ докум.	Підпис	Дата		

Рисунок 3.32 демонструє сторінку глобального реєстру проданих квитків із вбудованим модулем їх анулювання. Таблиця містить зведену інформацію про кожен придбаний квиток: ідентифікатор транзакції, нікнейм покупця, назву фільму та залу, дату й час сеансу, номер ряду та місця, категорію крісла (Standard або VIP), вартість квитка та його поточний статус. Для кожного запису доступна кнопка ручного скасування транзакції, після підтвердження якої автоматично запускається ланцюжок звільнення місця у схемі залу та генерується сервісне сповіщення для відповідного користувача.

КОРИСТУВАЧ	ФІЛЬМ / СЕАНС	КАТЕГОРІЯ	МІСЦЕ	СТАТУС	ДІЇ
User5 user5@cinema.com	Людина-павук: Повземо крізь Всесвіт 25.06.2026, 17:40:00	STANDARD OverHall	Ряд 6, Місце 8	АКТИВНИЙ	ДЕТАЛІ АНУЛЮВАТИ
User5 user5@cinema.com	Людина-павук: Повземо крізь Всесвіт 25.06.2026, 17:40:00	VIP OverHall	Ряд 7, Місце 8	АКТИВНИЙ	ДЕТАЛІ АНУЛЮВАТИ
User5 user5@cinema.com	Бійцівський клуб 25.06.2026, 20:20:00	VIP OverHall	Ряд 7, Місце 8	АКТИВНИЙ	ДЕТАЛІ АНУЛЮВАТИ
User5 user5@cinema.com	Бійцівський клуб 25.06.2026, 20:20:00	STANDARD OverHall	Ряд 6, Місце 8	АКТИВНИЙ	ДЕТАЛІ АНУЛЮВАТИ
User5 user5@cinema.com	Думками наживоріт 2 03.07.2026, 21:00:00	STANDARD MainHall	Ряд 3, Місце 7	АКТИВНИЙ	ДЕТАЛІ АНУЛЮВАТИ
User5 user5@cinema.com	Думками наживоріт 2 03.07.2026, 21:00:00	STANDARD MainHall	Ряд 3, Місце 6	АКТИВНИЙ	ДЕТАЛІ АНУЛЮВАТИ
User5 user5@cinema.com	Володар пернів: Війна роґірімів 05.07.2026, 19:00:00	STANDARD OverHall	Ряд 6, Місце 7	АКТИВНИЙ	ДЕТАЛІ АНУЛЮВАТИ

Рисунок 3.32 - Сторінка усіх квитків користувачів та можливістю їх анулювання

Рисунки 3.33 та 3.34 відображають екрани додавання та редагування кінострічок у адміністративній панелі. Форма містить поля для введення назви фільму, URL-адреси постера та фонового банера, текстового опису, масиву жанрових тегів, числових параметрів тривалості (у хвиликах) та року випуску, імені режисера, а також перемикача статусу показу isShowingInCinema. При редагуванні наявного запису форма автоматично виконує GET-запит до відповідного ендпоінта та попередньо заповнює всі поля актуальними значеннями з бази даних. Збереження змін ініціює PUT-запит, який оновлює документ у колекції Movie без порушення цілісності пов'язаних сеансів.

НОВИЙ ФІЛЬМ

ПОСТЕР ФІЛЬМУ файл не вибрано

НАЗВА

ОПИС

РІК

РЕЙТИНГ (БАЗОВИЙ)

ОПУБЛІКУВАТИ ФІЛЬМ

Рисунок 3.33 - Сторінка додавання/редагування фільмів

ПОСТЕР ФІЛЬМУ файл не вибрано

НАЗВА

Бійцівський клуб

ОПИС

Терасмий хронічним безсонням клерк намагається вирватися з нудного споживацького життя. Він знаходить віддушину в знайомстві з харизматичним торговцем милом Тайлером Дерденом, чия філософія базується на саморуйнуванні. Разом вони засновують підпільний «Бійцівський клуб», який швидко переростає у масштабну та небезпечну анархічну організацію.

РІК

1999

РЕЙТИНГ (БАЗОВИЙ)

8,8

ТРИВАЛІСТЬ (ХВ)

139

РЕЖИСЕР

Девід Фінчер

ЗБЕРЕГТИ ЗМІНИ

Рисунок 3.34 - Сторінка додавання/редагування фільмів

Рисунок 3.35 демонструє інтерфейс конструктора та редактора кінозалів. Адміністратор вводить унікальну назву залу та задає геометричні параметри сітки місць: кількість рядів і кількість крісел у кожному ряду. На основі введених даних бекенд автоматично генерує повну матрицю місць із координатами кожного крісла та зберігає шаблон залу в колекцію Hall. Під час створення нових сеансів цей шаблон копіюється, формуючи динамічну схему місць із індивідуальними полями статусу для кожного покупця.

		Сівченко М.Т.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремов Ю.М.				107
Змн.	Арк.	№ докум.	Підпис	Дата		

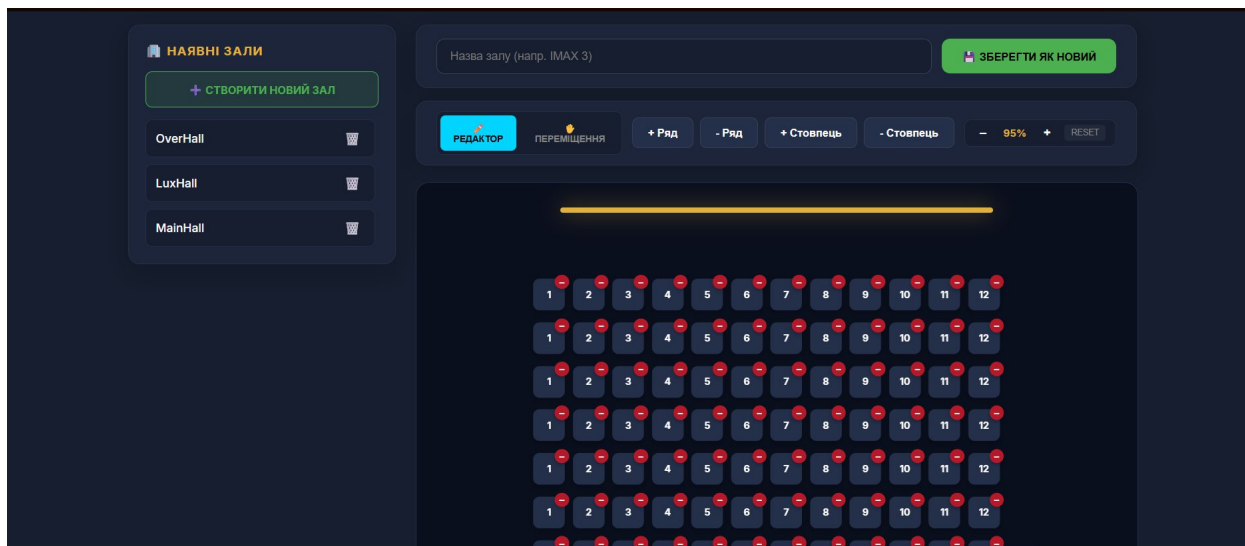


Рисунок 3.35 - Сторінка додавання/редагування кінозалів

Рисунок 3.36 відображає панель модерування користувачів із вбудованою антифлуд-системою. Таблиця містить перелік усіх зареєстрованих облікових записів із зазначенням нікнейму, електронної адреси, ролі користувача та індикаторів підозрілої активності: лічильника покинутих сторінок оплати (abandonedPaymentsCount) та частоти повторних кліків по кнопці оплати на основі масиву таймштампів (paymentClickTimestamps). Адміністратор має можливість заблокувати доступ конкретного користувача до чату підтримки (isBannedSupport) або до платформи загалом безпосередньо з цього інтерфейсу.

ДВОЕТАПНЕ БЛОКУВАННЯ ТА РОЛІ					
КОРИСТУВАЧ	ЕЛЕКТРОННА ПОШТА	РОЛЬ	ПОКИНУТА ОПЛАТА	ПЕРИОДИЧНІСТЬ	КЕРУВАННЯ ДОСТУПОМ
Universal Pictures ID: 6a1820f4ae01d1c0fa1944d1	universal@absolute.cinema	Distributor	0 разів	Немає флуду	БЛОКУВАТИ БРОНЬ БЛОКУВАТИ ЧАТ
Walt Disney Studios ID: 6a1820f3ae01d1c0fa1944ce	disney@absolute.cinema	Distributor	0 разів	Немає флуду	БЛОКУВАТИ БРОНЬ БЛОКУВАТИ ЧАТ
Warner Bros. Pictures ID: 6a1820f3ae01d1c0fa1944cb	warner@absolute.cinema	Distributor	0 разів	Немає флуду	БЛОКУВАТИ БРОНЬ БЛОКУВАТИ ЧАТ
SuperAdmin ID: 696ec595c686041dabf3d683	admin@cinema.com	Admin	4 разів	Менше хвилини	БЛОКУВАТИ БРОНЬ БЛОКУВАТИ ЧАТ
User7 ID: 696556ab6d368b158ff8d70c	user7@cinema.com	User (Клієнт)	0 разів	Немає флуду	БЛОКУВАТИ БРОНЬ БЛОКУВАТИ ЧАТ
User6 ID: 696556185012440249ca0b6c	user6@cinema.com	User (Клієнт)	0 разів	Немає флуду	БЛОКУВАТИ БРОНЬ БЛОКУВАТИ ЧАТ
User5 ID: 696556185012440249ca0b6c	user5@cinema.com	User (Клієнт)	0 разів	Немає флуду	БЛОКУВАТИ БРОНЬ БЛОКУВАТИ ЧАТ

Рисунок 3.36 - Сторінка модерування користувачами

Головний екран керування каталогом (Рис. 3.30), Центральна панель адміністратора (AdminDashboard.jsx) побудована за модульним принципом із використанням системи динамічних табів. Перший таб забезпечує повний контроль над каталогом фільмів. Дані виводяться у вигляді інтерактивної таблиці з можливістю миттєвої фільтрації через пошуковий рядок (searchTerm). Система розбиває запит на окремі символи, виконуючи наскрізний пошук за назвою, роком випуску, режисером чи жанрами. Кожен рядок таблиці містить прев'ю постера, унікальний шестизначний ідентифікатор документа з бази даних MongoDB та кнопки швидкого переходу до редагування чи безповоротного видалення об'єкта.

Центр обробки черги та історії звернень (Рис. 3.31), Цей інтерфейс слугує архівом та координаційним центром для служби підтримки. За допомогою періодичних асинхронних запитів та технології WebSocket система агрегує закриті й чинні тікети, сортуючи їх за часом останнього оновлення. Адміністратор має можливість переглянути повний лог листування з будь-яким клієнтом для аналізу якості обслуговування або вирішення спірних ситуацій.

Глобальний реєстр та модуль анулювання квитків (Рис. 3.32), Спеціалізована сторінка обліку продажів відображає зведену інформацію про всі викуплені квитки в системі. Інтерфейс інтегрований із фінансовим роутом бекенду, що дозволяє адміністратору у ручному режимі скасувати будь-яку транзакцію (наприклад, за усним зверненням клієнта на касі). При цьому автоматично запускається ланцюжок звільнення крісел у схемі залу та створюється сервісне сповіщення про повернення коштів.

Управління сутностями фільмів (Рис. 3.33 - Рис. 3.34), Екрани додавання та редагування медіафайлів реалізують складні форми валідації даних. Бекенд вимагає суворого дотримання типів, назва, лінк на постер, опис, масив жанрів та числові поля для тривалості й року релізу. При редагуванні форма автоматично робить GET-запит, предзаповнює поля поточними значеннями з

		Сівченко М.Т.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремов Ю.М.				109
Змн.	Арк.	№ докум.	Підпис	Дата		

бази, а кнопка збереження відправляє PUT/PATCH-запит, актуалізуючи дані в MongoDB без порушення цілісності зв'язків із сеансами.

Конструктор та редактор кінозалів (Рис. 3.35), цей модуль відповідає за оцифрування фізичних залів кінотеатру. Адміністратор задає унікальну назву залу та конфігурує геометрію сітки (кількість рядів та крісел у ряду). На основі цих даних бекенд автоматично розраховує координати кожного місця (матрицю крісел) та записує шаблон залу в колекцію Hall. Під час створення нових сеансів цей шаблон копіюється, створюючи динамічну схему місць із індивідуальними мітками lockedUntil для майбутніх покупців.

Панель модерування користувачів (Антифлуд-система), Другий таб адмін-панелі є ключовим інструментом забезпечення безпеки кінотеатру, який реалізує розширений функціонал моніторингу. Система в реальному часі виводить список усіх зареєстрованих клієнтів і містить вбудовану інтелектуальну систему моніторингу залишку неоплачених замовлень. Панель відображає лічильник покинутих сторінок оплати (abandonedPaymentsCount) та вираховує середню періодичність флуду на основі масиву таймштампів кліків (paymentClickTimestamps).

Вся архітектура адмін-панелі проєктувалася з фокусом на масштабованість та мінімізацію людського фактора під час управління ресурсами. Завдяки використанню асинхронних архітектурних патернів та централізованої обробки подій, адміністратори отримують актуальні дані в режимі реального часу.

Рисунок 3.37 демонструє головний екран аналітичної панелі дистриб'ютора (DistributorDashboard.jsx). У верхній частині інтерфейсу розміщено три зведені метричні віджети, що розраховуються на сервері в реальному часі: загальний виторг - 2 550 грн, кількість проданих квитків - 13 штук та загальна кількість фільмів дистриб'ютора - 11 стрічок з посиланням «ДЕТАЛЬНІШЕ» для переходу до повного переліку. Нижня частина екрана поділена на два аналітичні блоки. Лівий блок містить інтерактивний графік фінансових надходжень із перемикачами часового діапазону: «7 Днів», «20

		Сівченко М.Т.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремов Ю.М.				110
Змн.	Арк.	№ докум.	Підпис	Дата		

Днів», «Рік» та «Загально». На момент знімку активовано режим «20 Днів», що відображає динаміку касових зборів у вигляді лінійного графіка. Правий блок - кругова діаграма «Розподіл виручки по фільмах» - візуалізує частку кожної кінострічки у загальному виторзі дистриб'ютора. При наведенні курсору на сегмент діаграми відображається підказка з назвою фільму та сумою зборів - наприклад, «28 років потому: 1 150 грн».

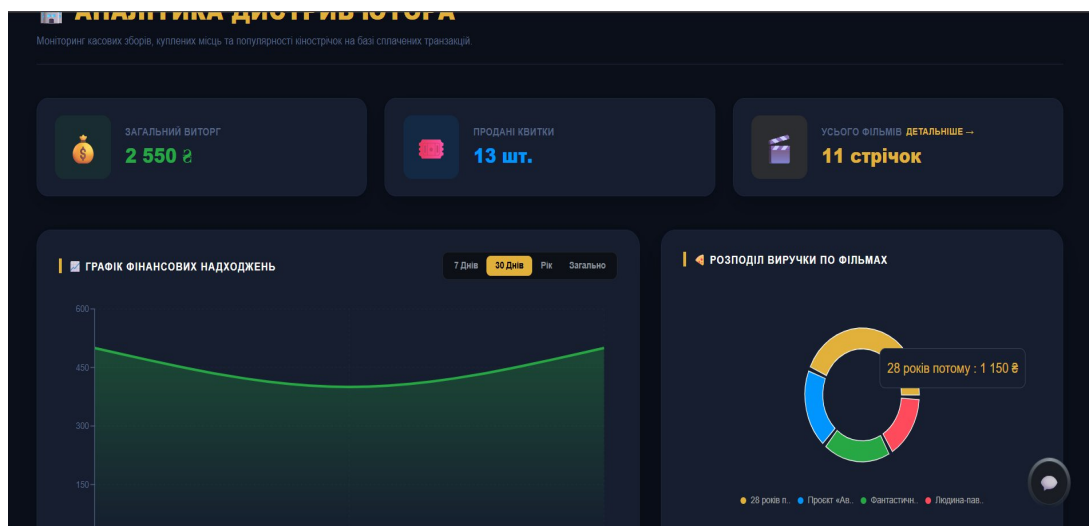


Рисунок 3.37 - Сторінка дистриб'ютора

Рисунок 3.38 відображає продовження аналітичної сторінки дистриб'ютора з двома додатковими блоками звітності. Верхній блок містить стовпчасту діаграму порівняння касових зборів між фільмами каталогу дистриб'ютора: «28 років потому», «Проєкт "Аве Марія"», «Фантастична четвірка» та «Людина-павук». Висота стовпця відповідає кількості проданих квитків на кожну стрічку, що дозволяє наочно порівняти популярність фільмів. Нижній блок - зведена комерційна відомість - являє собою табличне представлення тих самих даних у структурованому вигляді. Таблиця містить стовпці: постер фільму з кольоровим індикатором, назва кінострічки, кількість проданих квитків та загальні збори у гривнях. Зокрема, «28 років потому» - 7 квитків на суму 1 150 грн, «Проєкт "Аве Марія"» - 2 квитки на 500 грн, «Фантастична четвірка: Перші кроки» - 2 квитки на 500 грн, «Людина-павук: Повземо крізь Всесвіт» - 2 квитки на 400 грн.

		Сівченко М.Т.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремов Ю.М.				111
Змн.	Арк.	№ докум.	Підпис	Дата		

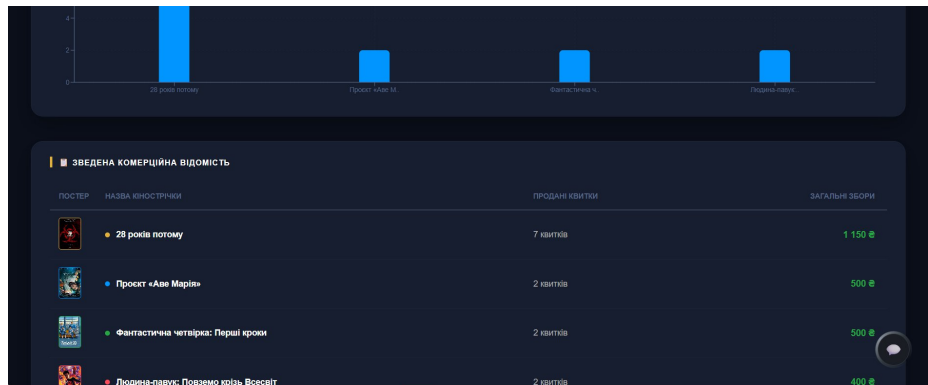


Рисунок 3.38 - Сторінка дистриб'ютора

Рисунок 3.39 ілюструє режим детального перегляду фільмів дистриб'ютора, який активується після натискання на віджет «Усього фільмів 11 стрічок» та переключення інтерфейсу дашборда у стан відображення 'movies'. У верхній частині зберігаються три зведені метричні віджети (загальний виторг, продані квитки, кількість фільмів), при цьому активний віджет «Усього фільмів» виділяється жовтою рамкою, сигналізуючи про поточний режим перегляду. Основну частину екрана займає перелік кінострічок у вигляді карток, кожна з яких містить постер фільму, його назву, унікальний ідентифікатор документа MongoDB та відсоткову частку у загальних касових зборах (наприклад, «28 років потому» - частка у касі 45,1%, «Проект "Аве Марія"» - 19,6%). У правій частині кожної картки розміщено два показники: кількість проданих квитків та загальні касові збори у гривнях. Цей режим підтягує всі фільми, де поле distributor у схемі Movie збігається з ідентифікатором поточної сесії дистриб'ютора, включаючи як активні стрічки в прокаті, так і архівні позиції.

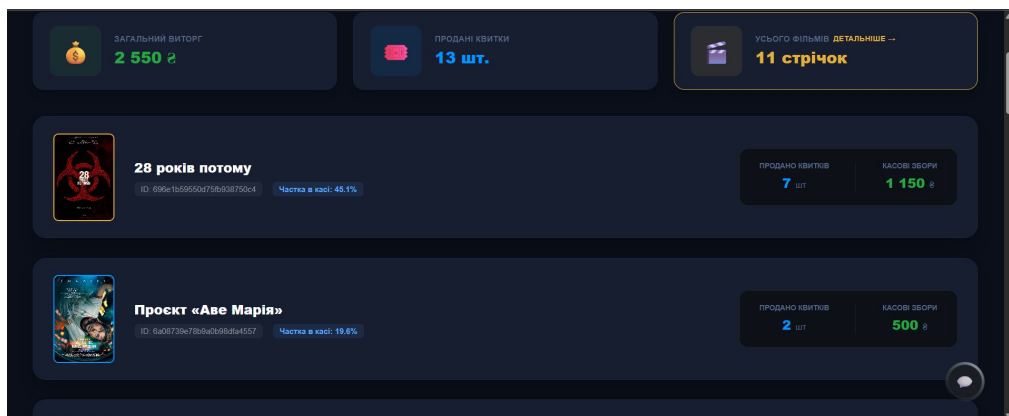


Рисунок 3.39 - Фільми закріплені за дистриб'ютором

		Сівченко М.Т.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремов Ю.М.				112
Змн.	Арк.	№ докум.	Підпис	Дата		

Екран загальної аналітики та віджети швидкого доступу (Рис. 3.37), Головна сторінка кабінету (DistributorDashboard.jsx) містить три базові метричні віджети, які розраховуються на сервері за допомогою високопродуктивних методів підрахунку документів MongoDB (.countDocuments()) та математичної агрегації суми (\$sum),

Загальний виторг (totalRevenue) - відображає накопичену суму у гривнях від продажу квитків на фільми, закріплені за цим дистриб'ютором.

Продані квитки (totalTickets) - сумарна кількість успішно викуплених посадкових місць.

Усього фільмів (totalMovies) - кількість активних та архівних кінострічок дистриб'ютора, зареєстрованих у системі.

Екран детальної статистики фільмів (Left-Join на рівні БД), При кліку на інтерактивний віджет «Усього фільмів» інтерфейс дашборда перемикає стан відображення у режим 'movies'. Головною архітектурною особливістю цього екрана є ізоляція та повне виведення контенту правовласника. На відміну від стандартних фінансових звітів, які відображають лише успішні продажі, цей модуль підтягує абсолютно всі фільми, де поле distributor у схемі Movie збігається з ID поточної сесії дистриб'ютора (distributorId), переведеним у тип mongoose.Types.ObjectId.

3.3. Тестування вебплатформи

Етап оцінювання працездатності створеної системи є ключовою фазою інженерного циклу, тестування програмного забезпечення є обов'язковим етапом інженерного циклу, що дозволяє виявити логічні та функціональні дефекти до розгортання системи в продуктивному середовищі [28], оскільки для високонавантажених сервісів стабільність архітектурних зв'язків визначає загальну життєздатність продукту. Перед безпосереднім розгортанням платформи «AbsoluteCinema» було здійснено суцільний покроковий аудит кожного структурного компонента й кодової бази додатка. Базовий пріоритет під час виконання цих процедур полягав у превентивному виявленні логічних

		Сівченко М.Т.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремов Ю.М.				113
Змн.	Арк.	№ докум.	Підпис	Дата		

чи функціональних дефектів, локалізації системних збоїв та підтвердженні повної відповідності продукту критеріям, зафіксованим у технічному завданні.

Для ізольованого дослідження поведінки серверних інтерфейсів (API) застосовувався спеціалізований програмний комплекс Postman. Цей інструментарій функціонує як віртуальний клієнт, що дозволяє проектувати, конструювати та відправляти мережеві запити різного рівня складності без залучення графічної оболонки сайту.

Завдяки інтегрованому інструментарію Postman, розробник отримує повний контроль над параметрами передачі даних, включаючи тонке налаштування HTTP-заголовків (Headers), токенів доступу, механізмів обробки файлів cookies, а також розробку скриптів для автоматичного тестування та динамічної генерації вхідних масивів. Використання цього середовища дозволило змоделювати екстремальні сценарії навантаження та протестувати логіку захисту від зловживань (антифлуд-контроль) ще до інтеграції коду в реальну інфраструктуру.

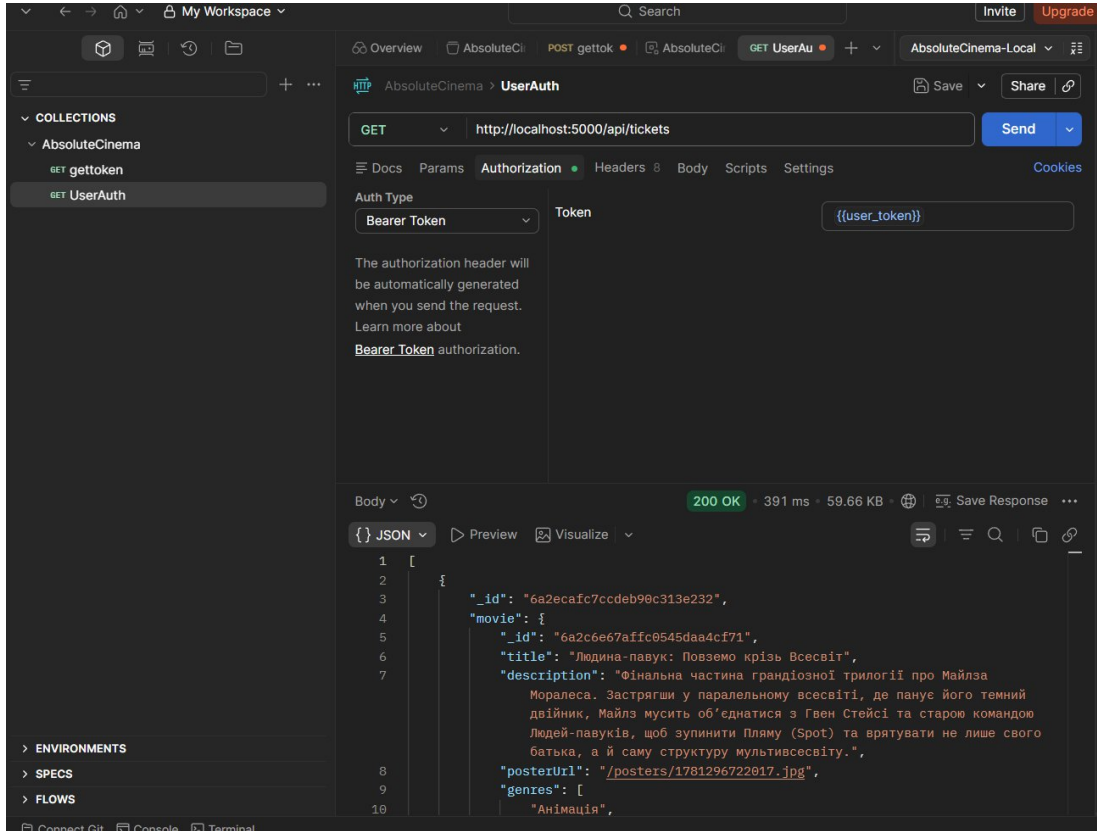


Рисунок 3.40 - Приклад запиту для тесту API

		Сівченко М.Т.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремюв Ю.М.				114
Змн.	Арк.	№ докум.	Підпис	Дата		

Для підтвердження коректності роботи шару проміжного програмного забезпечення (Middleware) захисту транзакційних маршрутів, було проведено симуляцію запиту на отримання історії куплених місць поточного користувача. Цей процес наочно демонструє взаємодію між клієнтським токеном та внутрішньою системою контролю доступу сервера.

Архітектура та конфігурація запиту (Рис. 3.40), тестування виконувалося через безпечний HTTP-метод GET за адресою кінцевої точки `http://localhost:5000/api/tickets`. Оскільки цей маршрут є закритим і вимагає обов'язкової ідентифікації клієнта, у вкладці Authorization було активовано тип авторизації Bearer Token. Замість ручного копіювання тимчасових ключів було застосовано динамічну конструкцію виклику глобальної змінної середовища `{{user_token}}`. Цей параметр автоматично підставляє актуальний рядок JWT-токена, який був згенерований та записаний у середовище AbsoluteCinema-Local під час попереднього успішного виконання процедури логіну (`/api/auth/login`).

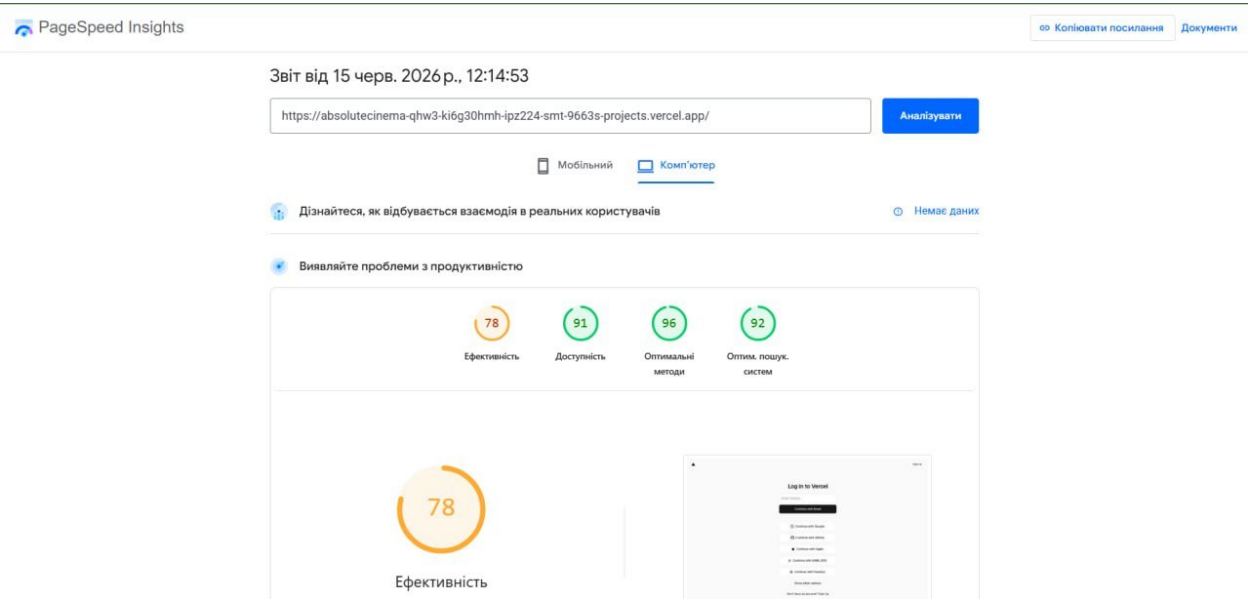


Рисунок 3.41 - Результати тестування на PageSpeed

Для оцінки швидкодії та ефективності оптимізації клієнтської частини розробленої вебплатформи було проведено тестування за допомогою інструменту Google PageSpeed Insights. Результати аналізу завантаження сторінок відображено на (Рис.3.41.) Отримані показники підтверджують

ефективність впроваджених методів оптимізації, таких як ледаче завантаження зображень (lazy loading), мініфікація статичних файлів та використання кешування, що забезпечує високий рівень продуктивності для кінцевого користувача.

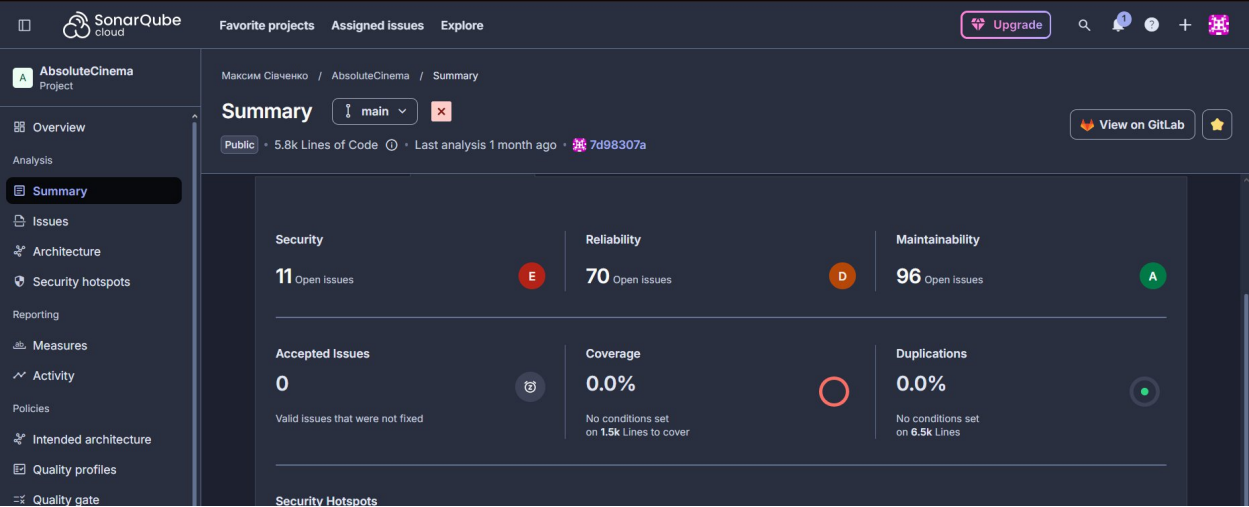


Рисунок 3.42 - SonarCloud тест

Для забезпечення високої якості програмного коду, перевірки дотримання стандартів написання та уникнення дублювання логіки було застосовано платформу SonarCloud. Цей інструмент дозволив провести комплексне сканування репозиторію в автоматичному режимі. Результати аналізу якості коду наведено на (Рис. 3.42.)

Після завершення етапу розробки було проведено комплексне тестування всіх функціональних компонентів вебплатформи. Процес тестування дозволив виявити та своєчасно усунути потенційні програмні помилки, що забезпечило повну відповідність системи технічним вимогам, сформульованим на початкових стадіях проєктування.

Окрім функціональної перевірки, особливу увагу було приділено забезпеченню кібербезпеки системи. Аналіз вразливостей проводився відповідно до рекомендацій OWASP щодо критичних загроз безпеці вебплатформ, зокрема захисту від ін'єкцій на рівні запитів до бази даних та міжсайтового скриптингу [30]. Зокрема, проведено аналіз вхідних даних та налаштування HTTP-заголовків безпеки для підвищення стійкості до поширених векторів атак (зокрема NoSQL-ін'єкцій та XSS-вразливостей).

Реалізовані заходи безпеки дозволили мінімізувати ризики несанкціонованого доступу до бази даних та забезпечити конфіденційність інформації користувачів.

Комплексний підхід до контролю якості, що охоплює оптимізацію програмного коду, тестування REST API та перевірку продуктивності в умовах високого навантаження, гарантує стабільність роботи системи під час її реальної експлуатації. Комбінація автоматизованих методів перевірки та ручного тестування дозволила усунути виявлені недоліки та створила надійну архітектурну основу для майбутнього масштабування проєкту, що значно полегшує подальше обслуговування та модернізацію програмного продукту.

Висновки до третього розділу

У цьому розділі дипломної роботи було проведено повний цикл розгортання та тестування вебплатформи «AbsoluteCinema». Практична реалізація охопила проєктування діаграми розгортання, що відображає надійну архітектурну взаємодію між клієнтською частиною, сервером додатків та базою даних, а також налаштування продуктивного хостингу.

Реалізований інтерфейс користувача вирізняється адаптивністю та логічною структурованістю. Виконано розробку ключових функціональних модулів, від інтерактивного каталогу фільмів та динамічного конструктора залів до системи лояльності та чату підтримки. Впровадження методології «оптимістичного оновлення» (Optimistic UI) у модулях діалогу дозволило досягти миттєвого відгуку системи, що значно підвищує зручність користувацького досвіду.

Окремо слід відзначити архітектурну готовність системи до вертикального та горизонтального масштабування. Завдяки використанню менеджерів процесів (PM2) та ізоляції служб у контейнеризованій середовищі, платформа «AbsoluteCinema» здатна ефективно перерозподіляти обчислювальні ресурси залежно від поточного рівня трафіку. Впроваджена модель автоматизованого резервного копіювання даних та моніторингу логів

		Сівченко М.Т.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремов Ю.М.				117
Змн.	Арк.	№ докум.	Підпис	Дата		

у реальному часі мінімізує ризики втрати інформації, забезпечуючи безперервність бізнес-процесів кінотеатру навіть за умов виникнення критичних відмов окремих вузлів інфраструктури.

		Сівченко М.Т.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремов Ю.М.				118
Змн.	Арк.	№ докум.	Підпис	Дата		

ВИСНОВКИ

У процесі виконання випускної кваліфікаційної роботи бакалавра було здійснено комплексне дослідження та аналіз існуючих рішень у сфері розробки систем бронювання квитків та взаємодії з користувачами в онлайн-кінотеатрах. Головною метою першого розділу роботи стало проведення детального конкурентного аналізу, що дозволило виокремити ключові переваги та функціональні можливості, які стали фундаментом для розробки власної вебплатформи «AbsoluteCinema».

У межах практичної частини було вивчено актуальні архітектурні підходи та сформульовано технічні вимоги. У другому розділі здійснено архітектурне проектування: розроблено модель предметної області, спроектовано структуру бази даних MongoDB та візуалізовано ключові сценарії використання системи через UML-діаграми. Запропоновані рішення гарантують масштабованість, відмовостійкість та відповідність моделі всім вимогам до сучасного сервісу.

У третьому розділі було реалізовано програмний продукт та описано процес його розгортання. Окрему увагу приділено розробці клієнтської частини, що забезпечує адаптивність та швидкість інтерфейсу. Після реалізації проведено повний цикл тестування: від перевірки API через Postman до комплексного аудиту продуктивності (Google PageSpeed Insights) та якості коду (SonarCloud), що дозволило усунути критичні помилки.

У роботі продемонстровано застосування передових концепцій, зокрема принципу розмежування обов'язків (Separation of Concerns) між клієнтським та серверним рівнями, впровадження бізнес-логіки та використання механізмів WebSocket для комунікації в реальному часі. Отримані результати підтверджують обґрунтованість обраних технологічних рішень та готовність платформи до реальної експлуатації.

Практичне завдання - розроблено вебплатформу бронювання квитків у кінотеатр.

		Сівченко М.Т.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремов Ю.М.				119
Змн.	Арк.	№ докум.	Підпис	Дата		

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Сівченко М.Т., Єфремов Ю.М. Архітектура системи персоналізації та UX у вебплатформах бронювання квитків // Тези доповідей XVI Міжнародної наукової конференції студентів та молодих вчених, м. Одеса, 14-15 травня 2026 р. Одеса : Одеська політехніка, 2026. С. 1–3.
2. Сівченко М.Т., Єфремов Ю.М. Вебплатформа бронювання квитків із використанням технологій реального часу // Тези доповідей XVI Міжнародної наукової конференції студентів та молодих вчених, м. Одеса, 14-15 травня 2026 р. Одеса : Одеська політехніка, 2026. С. 1–3.
3. Буров Є. В. Комп'ютерні мережі та вебтехнології : навч. посіб. Львів : Бакушевич, 2019. 312 с.
4. Глибовець М. М. Архітектура програмних систем : навч. посіб. Київ : НаУКМА, 2018. 245 с.
5. Зацуха О. В., Кудрявцева С. П. Проєктування інформаційних систем : навч. посіб. Київ : Центр учбової літератури, 2021. 288 с.
6. Трофименко О. Г., Прокоп Ю. В. Організація баз даних та знань : навч. посіб. Одеса : Фенікс, 2017. 264 с.
7. Шеховцов В. А. Операційні системи та архітектура комп'ютерних мереж. Київ : Видавнича група ВНУ, 2020. 432 с.
8. Литвин В. В., Пасічник В. В. Проєктування інформаційних систем інтелектуального аналізу даних. Львів : Новий Світ-2000, 2022. 340 с.
9. Ситник В. Ф. Проєктування баз даних : навч.-метод. посіб. Київ : КНЕУ, 2016. 220 с.
10. Кравець П. О. Вебтехнології та проєктування вебсистем : навч. посіб. Львів : Видавництво Львівської політехніки, 2021. 196 с.
11. Поморова О. В., Говорущенко Т. О. Технології створення програмних продуктів. Хмельницький : ХНУ, 2018. 182 с.
12. Мельник А. О. Розподілені інформаційні системи та хмарні технології : монографія. Київ : Інтерсервіс, 2023. 210 с.

		Сівченко М.Т.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Єфремов Ю.М.				120
Змн.	Арк.	№ докум.	Підпис	Дата		

13. Федорчук Є. Н. Розробка серверних застосунків на базі асинхронних платформ : метод. вказівки. Львів : ЛНТУ, 2022. 88 с.
14. Пасічник В. В. Глобальні інформаційні системи та технології : електрон. конспект. Львів : ПП Сорока, 2019. 416 с.
15. Node.js. Асинхронне середовище виконання JavaScript [Електронний ресурс]. – Режим доступу до ресурсу: <https://nodejs.org/uk/docs/> (дата звернення: 10.01.2026).
16. Берко А. Ю., Алексеева В. О. Технології створення та підтримки вебсайтів : навч. посіб. Львів : Магнолія, 2020. 270 с.
17. Коробейнікова Т. І. Розробка інтерфейсів користувача за допомогою сучасних JavaScript-фреймворків. Київ : КНУТД, 2021. 144 с.
18. Дунець Р. Б., Сидор А. Р. Проєктування користувацьких інтерфейсів вебсистем. Львів : Видавництво Старого Лева, 2019. 180 с.
19. Гризун Л. Е. Створення односторінкових вебзастосунків (SPA) : навч.-метод. посіб. Харків : ХНПУ, 2022. 112 с.
20. React. Офіційна документація для розробників [Електронний ресурс]. – Режим доступу до ресурсу: <https://uk.react.dev/> (дата звернення: 12.01.2026).
21. Клименко О. В. Мережеві протоколи та технології обміну даними в режимі реального часу. Дніпро : Середняк Т. К., 2020. 156 с.
22. Кулаков Ю. О., Луцький Г. М. Комп'ютерні мережі : підручник. Київ : Юніор, 2016. 400 с.
23. WebSocket Protocol : специфікація стандарту RFC 6455 [Електронний ресурс]. – Режим доступу до ресурсу: <https://tools.ietf.org/html/rfc6455> (дата звернення: 15.01.2026).
24. Нікольський Ю. В., Щербина Ю. М. Системи баз даних : практикум. Львів : Магнолія-2006, 2018. 284 с.
25. Коваленко О. М. Неструктуровані та нереляційні бази даних (NoSQL) : навч. посіб. Полтава : ПУЕТ, 2021. 198 с.
26. Плєскач В. Л., Захарова Р. Ф. Інформаційні системи та технології : підручник. Київ : КНИГА, 2019. 350 с.

		Сівченко М.Т.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремов Ю.М.				121
Змн.	Арк.	№ докум.	Підпис	Дата		

27. MongoDB. Офіційна документація. Керівництво з моделювання даних та роботи з документами [Електронний ресурс]. – Режим доступу до ресурсу: <https://www.mongodb.com/docs/> (дата звернення: 18.01.2026).
28. Креней С. О. Тестування програмного забезпечення : практичний підхід. Київ: ВХВ, 2017. 210 с.
29. Бондаренко М. Ф. Безпека вебзастосунків та захист від мережевих загроз. Харків : ХНУРЕ, 2020. 165 с.
30. OWASP. Рекомендації щодо безпеки вебплатформ. Звіт про критичні вразливості [Електронний ресурс]. – Режим доступу до ресурсу: <https://owasp.org> (дата звернення: 22.01.2026).

		Сівченко М.Т.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремов Ю.М.				122
Змн.	Арк.	№ докум.	Підпис	Дата		

ДОДАТКИ

		Сівченко М.Т.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремов Ю.М.				123
Змн.	Арк.	№ докум.	Підпис	Дата		

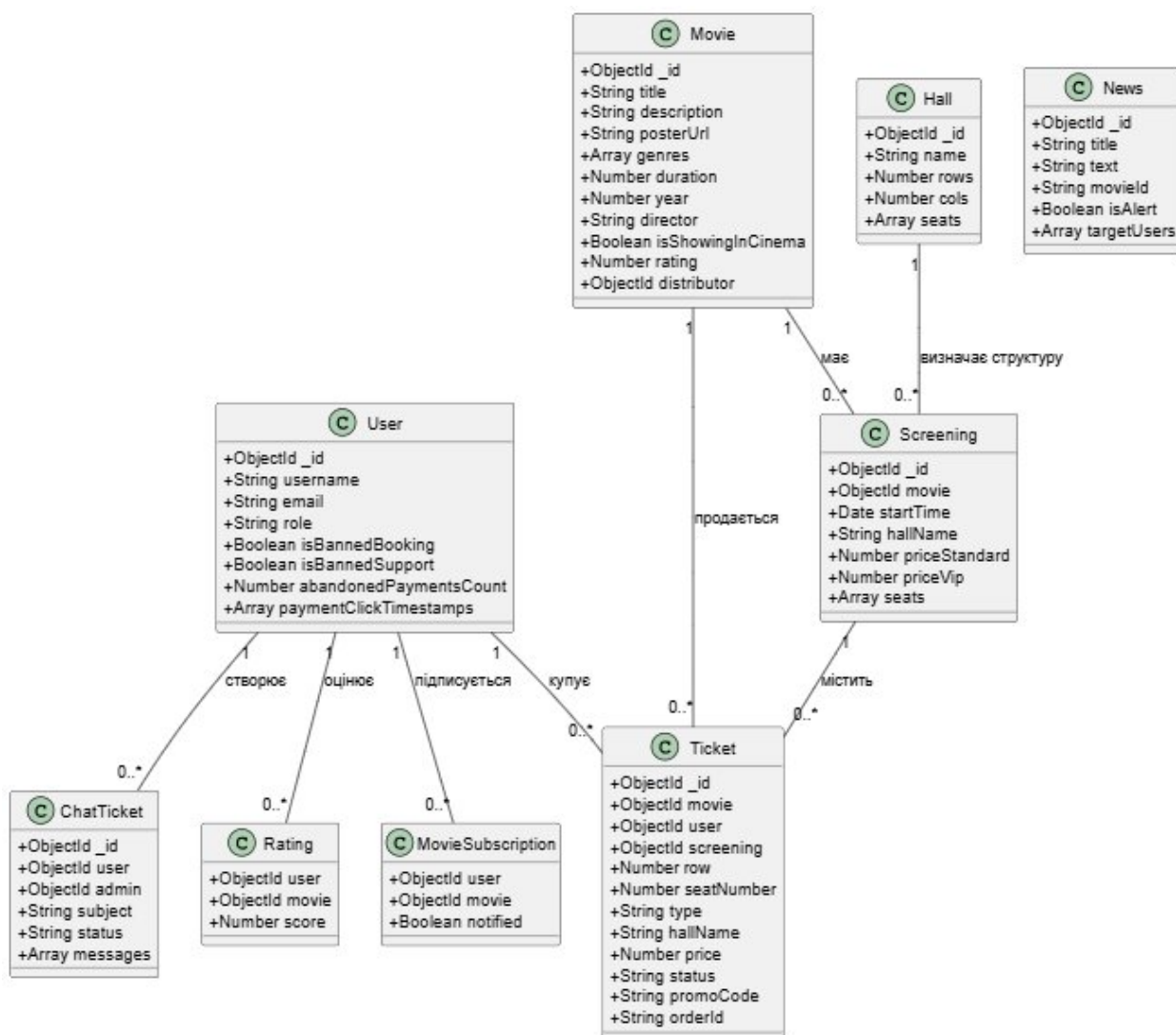


Рис.Скорочена діаграма класів доменних моделей

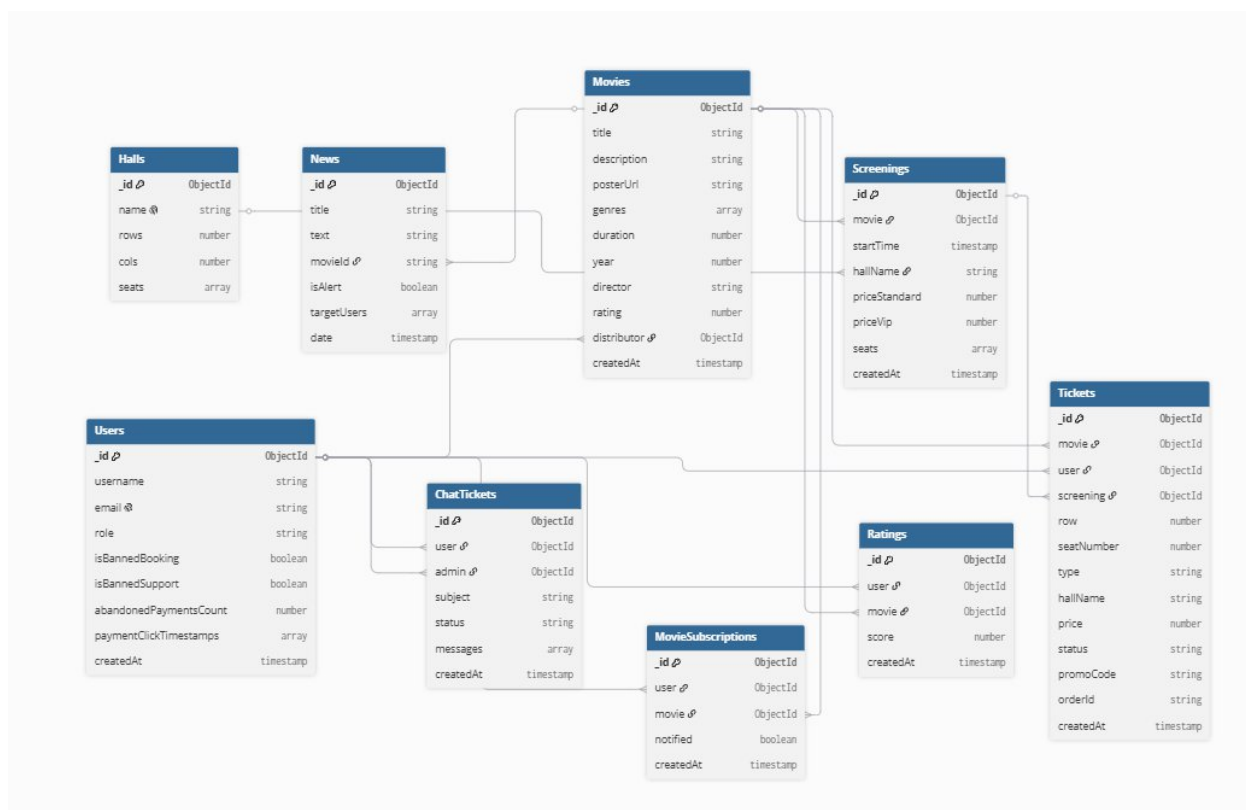


Рис.Детальна таблиця бази даних

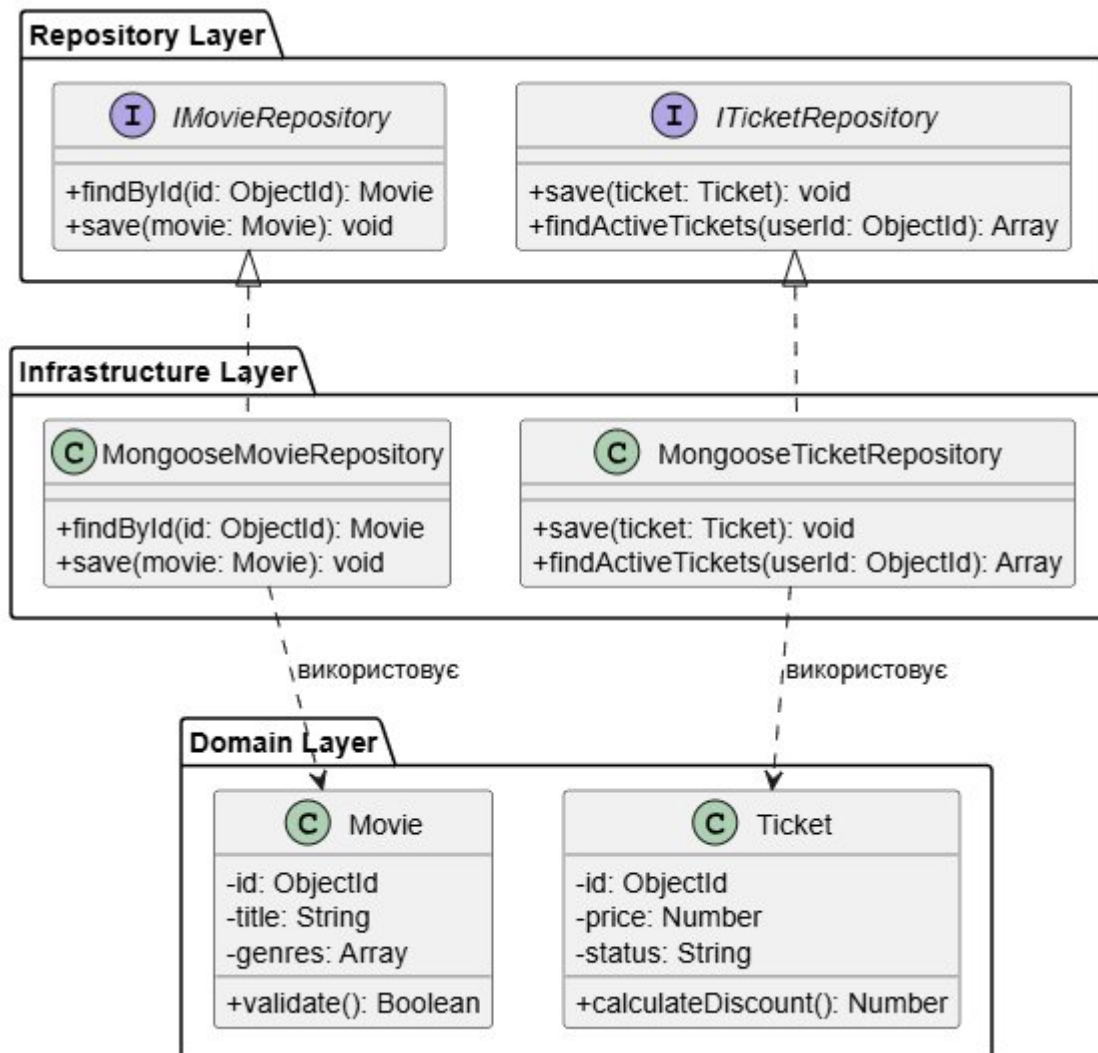


Рис. Діаграма класів при використанні Domain Model + Repositories

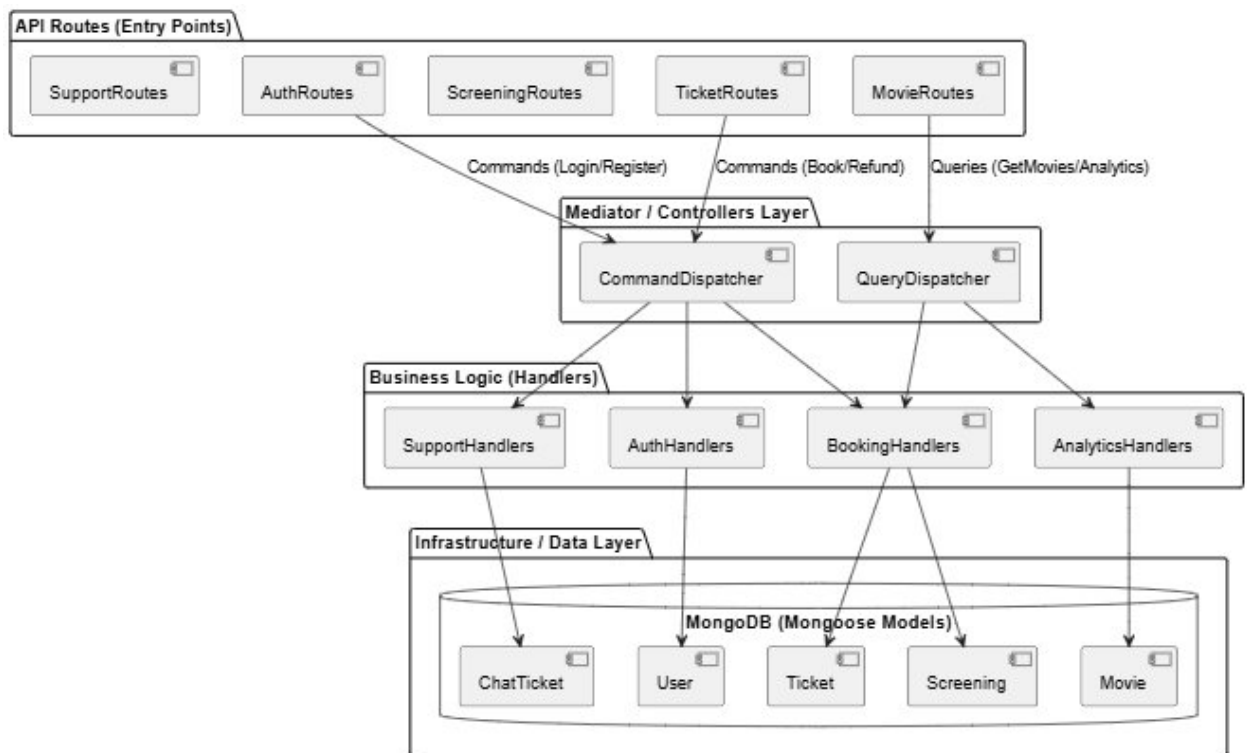


Рис.Діаграма класів при використанні CQRS та Mediator

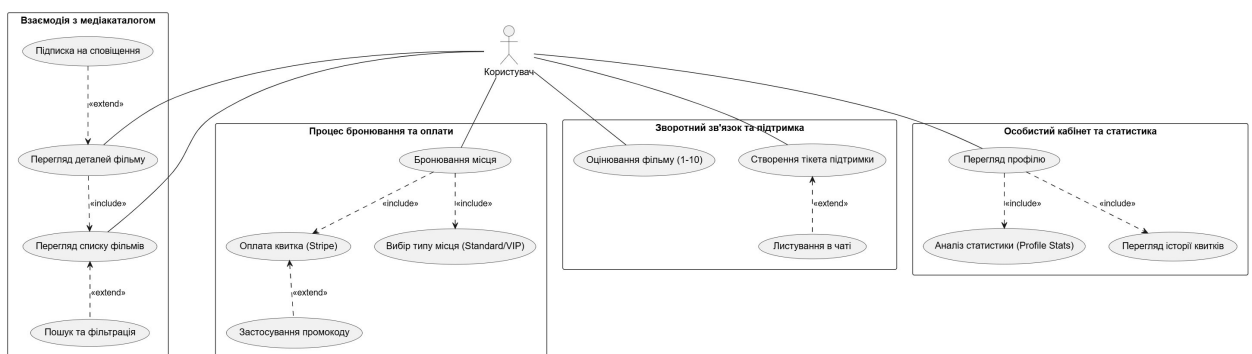


Рис.Діаграма use case для користувачів



Рис.Діаграма use case для адміністрації



Рис.Коротка архітектура вебплатформи

Лістинг моделі ChatTicket.js

```

const mongoose = require('mongoose');

const messageSchema = new mongoose.Schema({
  sender: {
    type: mongoose.Schema.Types.ObjectId,
    ref: 'User',
    required: true
  },
  text: {
    type: String,
    required: true
  },
  createdAt: {
    type: Date,
    default: Date.now
  }
});

const chatTicketSchema = new mongoose.Schema({
  user: {
    type: mongoose.Schema.Types.ObjectId,
    ref: 'User',
    required: true
  },
  admin: {
    type: mongoose.Schema.Types.ObjectId,
    ref: 'User',
    default: null
  },
  subject: {
    type: String,
    default: "Загальне питання про роботу сайту"
  },
  status: {
    type: String,
    enum: ['open', 'in_progress', 'closed'],
    default: 'open'
  },
  messages: [messageSchema],
  timestamps: true
});

chatTicketSchema.index({user: 1, status: 1});

module.exports = mongoose.model('ChatTicket', chatTicketSchema);

```

		Сівченко М.Т.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Сфремов Ю.М.				130
Змн.	Арк.	№ докум.	Підпис	Дата		

Лістинг моделі Hall.js

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

const hallSchema = new mongoose.Schema({
  name: {type: String, required: true},
  rows: Number,
  cols: Number,
  seats: [{
    row: Number,
    col: Number,
    type: {type: String, enum: ['standard', 'vip'], default: 'standard'},
    exists: {type: Boolean, default: true},
    x: Number,
    y: Number
  }]
});

module.exports = mongoose.model('Hall', hallSchema);
```

Лістинг моделі Movie.js

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

const screeningSchema = new mongoose.Schema({
  dateTime: {type: Date, required: true},
  hall: {type: String, required: true},
  priceStandard: {type: Number, required: true},
  priceVip: {type: Number, required: true}
});

const movieSchema = new mongoose.Schema({
  title: {type: String, required: true},
  description: String,
  posterUrl: String,
  genres: [String],
  duration: {type: Number},
  year: Number,
  director: String,
  actors: [String],
  isShowingInCinema: {type: Boolean, default: true},
  rating: {type: Number, default: 0},
  ratingSum: {type: Number, default: 0},
  ratingCount: {type: Number, default: 0},
  trailerUrl: {type: String, default: ""},

  distributor: {
    type: mongoose.Schema.Types.ObjectId,
```

		Сівченко М.Т.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Сфремов Ю.М.				131
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    ref, 'User',
    required, true
  }
}, {timestamps, true});

module.exports = mongoose.model('Movie', movieSchema);

```

Лістинг моделі MovieSubscription.js

```

const mongoose = require('mongoose');

const movieSubscriptionSchema = new mongoose.Schema({
  user: {type: mongoose.Schema.Types.ObjectId, ref: 'User', required: true},
  movie: {type: mongoose.Schema.Types.ObjectId, ref: 'Movie', required: true},
  notified: {type: Boolean, default: false},
  createdAt: {type: Date, default: Date.now}
});

movieSubscriptionSchema.index({user: 1, movie: 1}, {unique: true});

module.exports = mongoose.model('MovieSubscription', movieSubscriptionSchema);

```

Лістинг моделі News.js

```

const mongoose = require('mongoose');

const newsSchema = new mongoose.Schema({
  title: {type: String, required: true},
  text: {type: String, required: true},
  movieId: {type: String, default: null},
  posterUrl: {type: String, default: null},
  date: {type: Date, default: Date.now},
  isAlert: {type: Boolean, default: false},
  targetUsers: [String]
});

module.exports = mongoose.model('News', newsSchema);

```

Лістинг моделі Rating.js

```

const mongoose = require('mongoose');

const ratingSchema = new mongoose.Schema({
  user: {
    type: mongoose.Schema.Types.ObjectId,
    ref: 'User',

```

		Сівченко М.Т.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Сфремов Ю.М.				132
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    required,true
  },
  movie,{
    type,mongoose.Schema.Types.ObjectId,
    ref,'Movie',
    required,true
  },
  score,{
    type,Number,
    required,true,
    min,1,
    max,10
  }
},{timestamps,true});

ratingSchema.index({user,1,movie,1},{unique,true});

module.exports=mongoose.model('Rating',ratingSchema);

```

Лістинг моделі Screening.js

```

const mongoose=require('mongoose');

const screeningSchema=new mongoose.Schema({
  movie,{type,mongoose.Schema.Types.ObjectId,ref,'Movie',required,true},
  startTime,{type,Date,required,true},
  hallName,{type,String,required,true},

  priceStandard,{type,Number,required,true},
  priceVip,{type,Number,required,false,default,0},

  seats,[
    {
      row,{type,Number,required,true},
      number,{type,Number,required,true},
      type,{type,String,enum,['standard','vip'],default,'standard'},

      isBooked,{type,Boolean,default,false},
      bookedBy,{type,mongoose.Schema.Types.ObjectId,ref,'User',default,null},
      lockedUntil,{type,Date,default,null},
      lockedBy,{type,mongoose.Schema.Types.ObjectId,ref,'User',default,null}
    }
  ],
},{timestamps,true});

module.exports=mongoose.model('Screening',screeningSchema);

```

Лістинг моделі Ticket.js

```

const mongoose = require('mongoose');

const ticketSchema = new mongoose.Schema({
  movie: {type: mongoose.Schema.Types.ObjectId, ref: 'Movie', required: true},
  user: {type: mongoose.Schema.Types.ObjectId, ref: 'User', required: true},
  screening: {type: mongoose.Schema.Types.ObjectId, ref: 'Screening', required: true},
  row: {type: Number, required: true},
  seatNumber: {type: Number, required: true},
  type: {
    type: String,
    enum: ['standard', 'vip'],
    default: 'standard',
    required: true
  },
  hallName: {type: String, required: true},
  startTime: {type: Date, required: true},
  price: {type: Number, required: true},
  status: {
    type: String,
    enum: ['booked', 'paid', 'refunded'],
    default: 'paid'
  },
  promoCode: {
    type: String,
    default: null
  },
  isUsed: {type: Boolean, default: false},
  orderId: {type: String, default: () => Math.random().toString(36).substr(2, 9).toUpperCase()},
  createdAt: {type: Date, default: Date.now}
});

module.exports = mongoose.model('Ticket', ticketSchema);

```

Лістинг моделі Ticket.js

```

const mongoose = require('mongoose');

const userSchema = new mongoose.Schema({
  username: {type: String, required: true, trim: true},
  email: {type: String, required: true, unique: true, lowercase: true, trim: true},
  password: {type: String, required: true},
  role: {type: String, enum: ['user', 'admin', 'distributor'], default: 'user'},
  isBannedBooking: {type: Boolean, default: false},
  isBannedSupport: {type: Boolean, default: false},

  abandonedPaymentsCount: {type: Number, default: 0},
  paymentClickTimestamps: [{type: Date}],
  }, {timestamps: true});

```

		Сівченко М.Т.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Сфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		134

```
module.exports=mongoose.model('User',userSchema);
```

Лістинг authRoutes.js

```
const express=require('express');
const router=express.Router();
const bcrypt=require('bcryptjs');
const jwt=require('jsonwebtoken');
const User=require('../models/User');
const Ticket=require('../models/Ticket');
const Rating=require('../models/Rating');
const {protect}=require('../middleware/authMiddleware');

router.get('/profile-stats',protect,async(req,res)=>{
  try{
    const userId=req.user._id;

    const userTickets=await Ticket.find({
      $or:[{user:userId},{bookedBy:userId},{userId:userId}]
    }).populate('screening');

    const totalTickets=userTickets.length;

    const hasVip=userTickets.some(t=>t.seatType==='vip' || t.type==='vip');

    const hasNightSession=userTickets.some(t=>{
      if(!t.screening || !t.screening.startTime)return false;
      const hour=new Date(t.screening.startTime).getHours();
      return hour>=22 || hour<6;
    });

    const uniqueHalls=new Set(userTickets.map(t=>t.screening?.hallName ||
      t.screening?.hall).filter(Boolean));
    const uniqueHallsCount=uniqueHalls.size;

    let totalRatings=0;
    try{
      totalRatings=await Rating.countDocuments({user:userId});
    }catch(e){
      totalRatings=0;
    }

    res.json({
      totalTickets,
      hasVip,
      totalRatings,
      hasNightSession,
      uniqueHallsCount
    });
  }catch(err){
```

		Сівченко М.Т.			ІПЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
		Сфремов Ю.М.				135
Змн.	Арк.	№ докум.	Підпис	Дата		

```

res.status(500).json({message,"Помилкасервера,"+err.message});
}
});

router.get('/profile',protect,async(req,res)=>{
try{
res.json(req.user);
}catch(err){
res.status(500).json({message,err.message});
}
});

router.patch('/update-profile',protect,async(req,res)=>{
try{
const{subscribeToNews}=req.body;

constupdatedUser=awaitUser.findByIdAndUpdate(
req.user._id,
{$set,{subscribeToNews,subscribeToNews}},
{new,true,runValidators,true}
);

if(!updatedUser){
returnres.status(404).json({message,"Користувачанезнайдено"}});
}

res.json({
user,{
id,updatedUser._id,
username,updatedUser.username,
email,updatedUser.email,
role,updatedUser.role,
subscriptionType,updatedUser.subscriptionType,
hasSubscription,updatedUser.hasSubscription,
subscriptionExpires,updatedUser.subscriptionExpires,
subscribeToNews,updatedUser.subscribeToNews,
createdAt,updatedUser.createdAt
}
});
}catch(err){
res.status(500).json({message,"Помилкасервераприоновненніпрофілю,"+err.message});
}
});

router.post('/register',async(req,res)=>{
try{
const{username,email,password}=req.body;

constuserExists=awaitUser.findOne({email});
if(userExists)returnres.status(400).json({message,"Такийemailвжеzareєстровано"}});

```

		Сівченко М.Т.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Сфремов Ю.М.				136
Змн.	Арк.	№ докум.	Підпис	Дата		

```

const salt = await bcrypt.genSalt(10);
const hashedPassword = await bcrypt.hash(password, salt);

const newUser = new User({
  username,
  email,
  password, hashedPassword,
  role, 'user',
  subscribeToNews, false
});

await newUser.save();

const token = jwt.sign(
  {id, newUser._id, role, newUser.role},
  process.env.JWT_SECRET,
  {expiresIn, '1d'}
);

res.status(201).json({
  token,
  user, {
    id, newUser._id,
    username, newUser.username,
    email, newUser.email,
    role, newUser.role,
    subscriptionType, newUser.subscriptionType,
    hasSubscription, newUser.hasSubscription,
    subscriptionExpires, newUser.subscriptionExpires,
    subscribeToNews, newUser.subscribeToNews,
    createdAt, newUser.createdAt
  }
});
} catch (err) {
  res.status(500).json({message, err.message});
}
});

router.post('/login', async (req, res) => {
  try {
    const {email, password} = req.body;

    const user = await User.findOne({email});
    if (!user) return res.status(400).json({message, "Користувача не знайдено"});

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

    const token = jwt.sign(

```

		Сівченко М.Т.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Сфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		137

```

{id,user._id,role,user.role},
process.env.JWT_SECRET,
{expiresIn,'1d'}
);

res.json({
  token,
  user,{
    id,user._id,
    username,user.username,
    email,user.email,
    role,user.role,
    subscriptionType,user.subscriptionType,
    hasSubscription,user.hasSubscription,
    subscriptionExpires,user.subscriptionExpires,
    subscribeToNews,user.subscribeToNews??false,
    createdAt,user.createdAt
  }
});
}catch(err){
  res.status(500).json({message,err.message});
}
});

router.post('/track-payment-intent',protect,async(req,res)=>{
  try{
    constuser=awaitUser.findById(req.user._id);
    if(!user)returnres.status(404).json({message,"Користувачанезнайденно"});

    user.abandonedPaymentsCount=(user.abandonedPaymentsCount || 0)+1;

    if(!user.paymentClickTimestamps)user.paymentClickTimestamps=[];
    user.paymentClickTimestamps.push(newDate());

    if(user.paymentClickTimestamps.length>10){
      user.paymentClickTimestamps.shift();
    }

    awaituser.save();
    res.json({success,true,abandonedCount,user.abandonedPaymentsCount});
  }catch(err){
    res.status(500).json({message,err.message});
  }
});

router.get('/admin/users',protect,async(req,res)=>{
  try{
    if(req.user.role!=='admin'){
      returnres.status(403).json({message,"Доступзаборонено.Потрібніправаадміністратора."});
    }
  }

```

		Сівченко М.Т.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Сфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		138

```

const users = await User.find().select('-password').sort({createdAt: -1});
res.json(users);
} catch (err) {
res.status(500).json({message, err.message});
}
});

router.put('/admin/users/:id', protect, async (req, res) => {
try {
if (req.user.role !== 'admin') {
return res.status(403).json({message, "Доступ заборонено. Потрібні права адміністратора."});
}
const {role, isBannedBooking, isBannedSupport} = req.body;

const user = await User.findById(req.params.id);
if (!user) return res.status(404).json({message, "Користувача не знайдено"});

if (role) user.role = role;
if (isBannedBooking !== undefined) user.isBannedBooking = isBannedBooking;
if (isBannedSupport !== undefined) user.isBannedSupport = isBannedSupport;

await user.save();
res.json({message, "Дані користувача успішно оновлено!", user});
} catch (err) {
res.status(500).json({message, err.message});
}
});

router.post('/reset-my-flood-counter', protect, async (req, res) => {
try {
const user = await User.findById(req.user._id);
if (!user) return res.status(404).json({message, "Користувача не знайдено"});

user.abandonedPaymentsCount = 0;
await user.save();

res.json({success, true, message, "Лічильник успішно скинуто"});
} catch (err) {
res.status(500).json({message, "Помилка сервера, " + err.message});
}
});

module.exports = router;

Лістинг chatRoutes.js
const express = require('express');
const router = express.Router();
const ChatTicket = require('../models/ChatTicket');
const Ticket = require('../models/Ticket');
const Screening = require('../models/Screening');

```

		Сівченко М.Т.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Сфремов Ю.М.				139
Змн.	Арк.	№ докум.	Підпис	Дата		

```

const{protect}=require('../middleware/authMiddleware');

router.get('/admin/user-tickets/:userId',protect,async(req,res)=>{
if(req.user.role!=='admin')returnres.status(403).send("Немаєдоступу");
try{
consttickets=awaitTicket.find({
user:req.params.userId,
status,{ $in:['paid',null,undefined]}
})
.populate('movie')
.populate({
path,'screening',
select,'startTimehallName'
});

res.json(tickets);
}catch(err){
res.status(500).json({message,"Помилказавантаженняквитків",err.message});
}
});

router.post('/admin/refund-ticket',protect,async(req,res)=>{
if(req.user.role!=='admin')returnres.status(403).send("Немаєдоступу");

try{
const{ticketId,chatTicketId}=req.body;

constticket=awaitTicket.findById(ticketId).populate('movie');
if(!ticket)returnres.status(404).json({message,"Квитокнезнайдено"});
if(ticket.status==='refunded')returnres.status(400).json({message,"Квитоквжеповернуто"});

constseatType=ticket.type||'standard';
constisVip=seatType.toLowerCase()===vip';
constprefix=isVip?'VIP','STD';

constgeneratedPromo=`REFUND-${prefix}-${Math.random().toString(36).substring(2,
8).toUpperCase()}-${Math.floor(1000+Math.random()*9000)}`;

ticket.status='refunded';
ticket.promoCode=generatedPromo;
ticket.type=seatType;
awaitticket.save();

constscreening=awaitScreening.findById(ticket.screening);
if(screening){
constseat=screening.seats.find(s=>s.row===ticket.row&&s.number===ticket.seatNumber);
if(seat){
seat.isBooked=false;
seat.bookedBy=null;
}
}
}

```

		Сівченко М.Т.			ІІЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
		Сфремов Ю.М.				140
Змн.	Арк.	№ докум.	Підпис	Дата		

```

await screening.save();

if(global.io){
global.io.to(ticket.screening.toString()).emit('seats_updated',{
seatIds,seat?[seat._id],[],
status,'available'
});
}
}

const chat=await ChatTicket.findById(chatTicketId);
if(chat){
chat.messages.push({
sender:req.user._id,
text,` Асистент Absolute Cinema, Ваше повернення успішно оформлено! Квиток на фільм "${ticket.movie?.title || 'Кіно'}" (Ряд ${ticket.row}, Місце ${ticket.seatNumber}, Категорія, ${seatType.toUpperCase()}) анульовано. Ми вже сгенерували для вас компенсаційний промокод на ${prefix}-місце. Його надіслано у ваш особистий кабінет (розділ "Мої квитки"). Чекаємо на вас на наступних прем'єрах! 🎬`,
createdAt,new Date()
});
await chat.save();
}

res.json({message,"Повернення успішне",promoCode,generatedPromo});
}catch(err){
res.status(500).json({message,"Помилка при поверненні",+err.message});
}
});

router.post('/validate-promo',protect,async(req,res)=>{
try{
const{code,currentSeatType}=req.body;
const ticketWithPromo=await Ticket.findOne({promoCode,code,status,'refunded'});

if(!ticketWithPromo){
return res.status(404).json({message,"Промокод не дійсний або вже використаний"});
}

const refundDate=ticketWithPromo.updatedAt || ticketWithPromo.createdAt;
const now=new Date();
const diffTime=Math.abs(now-refundDate);
const diffDays=Math.ceil(diffTime/(1000*60*60*24));

if(diffDays>90){
return res.status(400).json({
message,"Термін дії цього промокоду вичерпано. Промокод компенсації дійсний лише протягом 3 місяців з моменту скасування квитка."
});
}
}
}

```

		Сівченко М.Т.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Сфремов Ю.М.				141
Змн.	Арк.	№ докум.	Підпис	Дата		

```

const promoSeatType = ticketWithPromo.type || 'standard';

if (currentSeatType && promoSeatType.toLowerCase() !== currentSeatType.toLowerCase()) {
  return res.status(400).json({
    message: `Цей промокод створено для крісел типу[${promoSeatType.toUpperCase()}]. Вине
    можете застосувати його на місці типу[${currentSeatType.toUpperCase()}]!`
  });
}

res.json({
  valid: true,
  discount: ticketWithPromo.price
});
} catch (err) {
  res.status(500).json({ message: "Помилка валідації," + err.message });
}
});

router.post('/consume-promo', protect, async (req, res) => {
  try {
    const { code } = req.body;
    const ticket = await Ticket.findOne({ promoCode: code });
    if (ticket) {
      ticket.promoCode = `USED-${ticket.promoCode}`;
      await ticket.save();
    }
    res.json({ success: true });
  } catch (err) {
    res.status(500).json({ message: err.message });
  }
});

router.get('/my-active-ticket', protect, async (req, res) => {
  try {
    const ticket = await ChatTicket.findOne({ user: req.user._id, status: { $ne: 'closed' } })
      .populate('messages.sender', 'username role');
    res.json(ticket);
  } catch (err) {
    res.status(500).json({ message: err.message });
  }
});

router.post('/send', protect, async (req, res) => {
  try {
    const { text, ticketId, subject } = req.body;
    let ticket;

    if (ticketId) {
      ticket = await ChatTicket.findById(ticketId);
    }
  }
});

```

		Сівченко М.Т.			ІІЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
		Сфремов Ю.М.				142
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    }else{
    ticket=await ChatTicket.findOne({user:req.user._id,status,{ $ne:'closed'}});
    }

    if(!ticket){
    ticket=await ChatTicket.create({
    user:req.user._id,
    subject,subject || "Загальне питання",
    messages,[{sender:req.user._id,text}]
    });
    }else{
    ticket.messages.push({sender:req.user._id,text});
    if(req.user.role==='admin'&&!ticket.admin){
    ticket.admin=req.user._id;
    ticket.status='in_progress';
    }
    }

    await ticket.save();
    const updated=await ChatTicket.findById(ticket._id)
    .populate('messages.sender','username role')
    .populate('admin','username');
    res.json(updated);
    }catch(err){
    res.status(500).json({message,err.message});
    }
    });

    router.get('/my-history',protect,async(req,res)=>{
    try{
    const history=await ChatTicket.find({user:req.user._id,status,'closed'})
    .populate('admin','username')
    .sort({createdAt:-1});
    res.json(history);
    }catch(err){
    res.status(500).json({message,err.message});
    }
    });

    router.get('/ticket/:ticketId',protect,async(req,res)=>{
    try{
    const ticket=await ChatTicket.findById(req.params.ticketId)
    .populate('messages.sender','username role')
    .populate('admin','username');
    if(!ticket) return res.status(404).json({message,"Не знайдено"});
    res.json(ticket);
    }catch(err){
    res.status(500).json({message,err.message});
    }
    });

```

		Сівченко М.Т.			ІПЗ.КР.Б – 121 – 26 – ІПЗ	Арк.
		Сфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		143

```

router.get('/admin/queue',protect,async(req,res)=>{
if(req.user.role!=='admin')returnres.status(403).send("Noaccess");
try{
consttickets=awaitChatTicket.find({status,'open',admin,null})
.populate('user','usernameemail')
.sort({createdAt,-1});
res.json(tickets);
}catch(err){
res.status(500).json({message,err.message});
}
});

router.get('/admin/my-chats',protect,async(req,res)=>{
if(req.user.role!=='admin')returnres.status(403).send("Noaccess");
try{
consttickets=awaitChatTicket.find({admin:req.user._id,status,'in_progress'})
.populate('user','usernameemail')
.sort({updatedAt,-1});
res.json(tickets);
}catch(err){
res.status(500).json({message,err.message});
}
});

router.patch('/close/ticketId',protect,async(req,res)=>{
try{
constticket=awaitChatTicket.findById(req.params.ticketId);
if(!ticket)returnres.status(404).json({message,"Чатнезнайдено"});
ticket.status='closed';
awaitticket.save();
res.json({message,"Чатуспішнозакритий",ticket});
}catch(err){
res.status(500).json({message,err.message});
}
});

router.get('/admin/history',protect,async(req,res)=>{
if(req.user.role!=='admin')returnres.status(403).send("Немаєдоступу");
try{
consthistory=awaitChatTicket.find({status,'closed'})
.populate('user','usernameemail')
.populate('admin','username')
.populate({
path,'messages.sender',
select,'username'role'
})
.sort({updatedAt,-1});
res.json(history);
}catch(err){

```

		Сівченко М.Т.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Сфремов Ю.М.				144
Змн.	Арк.	№ докум.	Підпис	Дата		

```
res.status(500).json({message,err.message});
}
});
```

```
module.exports=router;
```

Лістинг distributorRoutes.js

```
const express=require('express');
const router=express.Router();
const mongoose=require('mongoose');
const Movie=require('../models/Movie');
const Ticket=require('../models/Ticket');
const {protect}=require('../middleware/authMiddleware');

const verifyDistributorOrAdmin=(req,res,next)=>{
  if(req.user.role!=='distributor'&&req.user.role!=='admin'){
    return res.status(403).json({message,"Доступзаборонено.Кабінетпризначенийдля
    дистриб'юторівтаадміністраторів."});
  }
  next();
};

router.get('/analytics',protect,verifyDistributorOrAdmin,async(req,res)=>{
  try{
    const distributorId=new mongoose.Types.ObjectId(req.user.id || req.user._id);
    const {period='7days'}=req.query;

    let myMovies=[];
    if(req.user.role==='admin'){
      myMovies=await Movie.find();
    }else{
      myMovies=await Movie.find({distributor,distributorId});
    }

    const myMovieIds=myMovies.map(m=>m._id);

    if(myMovieIds.length===0){
      return res.json({
        summary:{totalRevenue,0,totalTickets,0,totalMovies,0},
        movieSales,[],
        dailySales,[]
      });
    }

    const totalTickets=await Ticket.countDocuments({
      status,'paid',
      movie,{ $in,myMovieIds}
    });

    const salesAggregate=await Ticket.aggregate([
```

		Сівченко М.Т.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Сфремов Ю.М.				145
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    {$match,{status,'paid',movie,{ $in,myMovieIds}}},
    {$group,{_id,null,totalRevenue,{ $sum,"$price"}}}
  ]);
  consttotalRevenue=salesAggregate[0]?.totalRevenue || 0;
  consttotalMovies=myMovieIds.length;

  constactiveSalesData=awaitTicket.aggregate([
    {$match,{status,'paid',movie,{ $in,myMovieIds}}},
    {
      $group,{
        _id,"$movie",
        ticketsSold,{ $sum,1},
        revenue,{ $sum,"$price"}
      }
    }
  ]);

  constmovieSalesData=myMovies.map(movie=>{
    constsalesInfo=activeSalesData.find(s=>s._id.toString()===movie._id.toString());
    return{
      _id,movie._id,
      name,movie.title || movie.name,
      posterUrl,movie.posterUrl || movie.poster || "",
      ticketsSold,salesInfo?salesInfo.ticketsSold,0,
      value,salesInfo?salesInfo.revenue,0
    };
  });
  movieSalesData.sort((a,b)=>b.value-a.value);

  letstartDate=null;
  letdateFormat="%d.%m";

  if(period==='7days'){
    startDate=newDate();
    startDate.setDate(startDate.getDate()-7);
  }elseif(period==='30days' || period==='month'){
    startDate=newDate();
    startDate.setDate(startDate.getDate()-30);
  }elseif(period==='year'){
    startDate=newDate();
    startDate.setFullYear(startDate.getFullYear()-1);
    dateFormat="%m.%Y";
  }
  if(period==='all'){
    dateFormat="%m.%Y";
  }

  constmatchStage={status,'paid',movie,{ $in,myMovieIds}};
  if(startDate){
    matchStage.createdAt={ $gte,startDate};
  }

```

		Сівченко М.Т.			ІІЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
		Сфремов Ю.М.				146
Змн.	Арк.	№ докум.	Підпис	Дата		

```

}

const dailySalesData = await Ticket.aggregate([
  {$match, matchStage},
  {
    $group, {
      _id, {$dateToString, {format, dateFormat, date, "$createdAt"}},
      revenue, {$sum, "$price"},
      tickets, {$sum, 1}
    }
  },
  {$sort, {"_id", 1}},
  {
    $project, {
      date, "$_id",
      "Збори(грн)", "$revenue",
      "Квитки(шт)", "$tickets",
      _id, 0
    }
  }
]);

res.json({
  summary, {totalRevenue, totalTickets, totalMovies},
  movieSales, movieSalesData,
  dailySales, dailySalesData
});

} catch (err) {
  console.error("DISTRIBUTORANALYTICSEERROR", err);
  res.status(500).json({message, "Помилка сервера аналітики"});
}
});

module.exports = router;

```

Лістинг hallRoutes.js

```

const express = require('express');
const router = express.Router();
const Hall = require('../models/Hall');
const {protect} = require('../middleware/authMiddleware');

const verifyAdmin = (req, res, next) => {
  if (req.user && req.user.role === 'admin') {
    return next();
  }
  return res.status(403).json({message, "Доступ заборонено. Потрібні права адміністратора."});
};

router.get('/', async (req, res) => {

```

		Сівченко М.Т.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Сфремов Ю.М.				147
Змн.	Арк.	№ докум.	Підпис	Дата		

```

try{
consthalls=awaitHall.find();
res.json(halls);
}catch(err){
res.status(500).json({message,err.message});
}
});

router.post('/create',protect,verifyAdmin,async(req,res)=>{
try{
consthall=newHall(req.body);
awaithall.save();
res.status(201).json(hall);
}catch(err){
res.status(500).json({message,err.message});
}
});

router.put('/update/,id',protect,verifyAdmin,async(req,res)=>{
try{
const{name,seats}=req.body;

constupdatedHall=awaitHall.findByIdAndUpdate(
req.params.id,
{name,seats},
{new,true,runValidators,true}
);

if(!updatedHall){
returnres.status(404).json({message,"Кінозалнезнайдено."});
}

res.json(updatedHall);
}catch(err){
res.status(500).json({message,"Помилкаоновленнязалу,"+err.message});
}
});

router.delete('/',id',protect,verifyAdmin,async(req,res)=>{
try{
constdeletedHall=awaitHall.findByIdAndDelete(req.params.id);

if(!deletedHall){
returnres.status(404).json({message,"Кінозалнезнайдено."});
}

res.json({message,"Кінозалуспішновидалено."});
}catch(err){
res.status(500).json({message,"Помилкавидаленнязалу,"+err.message});
}
}

```

		Сівченко М.Т.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Сфремов Ю.М.				148
Змн.	Арк.	№ докум.	Підпис	Дата		

```
});
```

```
module.exports=router;
```

Лістинг movieRoutes.js

```
const express=require('express');
const multer=require('multer');
const path=require('path');
const fs=require('fs');
const mongoose=require('mongoose');
const router=express.Router();
const Movie=require('../models/Movie');
const Hall=require('../models/Hall');
const Screening=require('../models/Screening');
const News=require('../models/News');
const Rating=require('../models/Rating');
const User=require('../models/User');
const Ticket=require('../models/Ticket');
const {protect}=require('../middleware/authMiddleware');
const MovieSubscription=require('../models/MovieSubscription');

const storage=multer.diskStorage({
  destination,(req,file,cb)=>{
    const frontendPath=path.join(__dirname,'../../frontend/public/posters');
    cb(null,frontendPath);
  },
  filename,(req,file,cb)=>{
    cb(null,Date.now()+path.extname(file.originalname));
  }
});
const upload=multer({storage});

const processBulkRefundForScreening=async(screeningId)=>{
  try{
    const activeTickets=await Ticket.find({
      screening:screeningId,
      status:'paid'
    }).populate('movie');

    if(!activeTickets || activeTickets.length===0) return;

    const firstTicket=activeTickets[0];
    const movieTitle=firstTicket.movie?.title || 'Кіно';
    const posterUrl=firstTicket.movie?.posterUrl || null;
    const sessionDate=new Date(firstTicket.startTime).toLocaleString('uk-UA',{
      day:'numeric',
      month:'long',
      hour:'2-digit',
      minute:'2-digit'
    });
  }
```

		Сівченко М.Т.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Сфремов Ю.М.				149
Змн.	Арк.	№ докум.	Підпис	Дата		

```

const userIdsWithTickets = activeTickets.map(t => t.user.toString());

for (const ticket of activeTickets) {
  const seatType = ticket.type || 'standard';
  const prefix = seatType.toLowerCase() === 'vip' ? 'VIP' : 'STD';
  const generatedPromo = `REFUND-${prefix}-${Math.random().toString(36).substring(2, 8).toUpperCase()}-${Math.floor(1000 + Math.random() * 9000)}`;

  ticket.status = 'refunded';
  ticket.promoCode = generatedPromo;
  await ticket.save();
}

const movieSlug = movieTitle
  .toLowerCase()
  .trim()
  .replace(/\s+/g, '-')
  .replace(/[\^a-z0-9a-яіієє-]/gi, '');

const refundNotification = new News({
  title: `⚠️ СЕАНССКАСОВАНО, "${movieTitle}"`,
  text: `З технічних причин сеанс на фільм "${movieTitle}" (${sessionDate}), Зал, ${firstTicket.hallName} скасовано. Ваш квиток анульовано, а компенсаційний промокод чекає у розділі "Мої квитки".`,
  movieId: firstTicket.movie?._id || movieSlug,
  posterUrl: posterUrl,
  date: new Date(),
  isAlert: true,
  targetUsers: userIdsWithTickets
});

await refundNotification.save();

if (global.io) {
  global.io.emit('screening_cancelled_alert', {
    screeningId: screeningId.toString(),
    userIds: userIdsWithTickets
  });
}
} catch (error) {
  console.error("Помилка масового скасування квитків", error.message);
}
};

const generateSeatsForScreening = async (hallName) => {
  try {
    const hall = await Hall.findOne({ name: hallName });
    if (!hall) return [];
    return hall.seats
  }

```

		Сівченко М.Т.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремов Ю.М.				150
Змн.	Арк.	№ докум.	Підпис	Дата		

```

.filter(s=>s.exists===true)
.map((s,index)=>({
row,s.row || 1,
number,s.col || s.number || (index+1),
type,s.type || 'standard',
isBooked,false,
bookedBy,null
}));
}catch(error){
return[];
}
};

router.get('/admin/distributors-list',protect,async(req,res)=>{
try{
if(req.user.role!=='admin')returnres.status(403).json({message,"Noadminaccess"});
constdistributors=awaitUser.find({role,'distributor'}).select('usernameemail_id');
res.json(distributors);
}catch(err){
res.status(500).json({message,err.message});
}
});

router.get('/',async(req,res)=>{
try{
constmovies=awaitMovie.find().sort({createdAt,-1});
constmoviesWithScreenings=awaitPromise.all(movies.map(async(movie)=>{
constscreenings=awaitScreening.find({movie,movie._id});
return{
...movie._doc,
screenings,screenings.map(sc=>({
_id,sc._id,
dateTime,sc.startTime,
hall,sc.hallName,
priceStandard,sc.priceStandard,
priceVip,sc.priceVip
})))
});
res.json(moviesWithScreenings);
}catch(err){
res.status(500).json({message,err.message});
}
});

router.get('/news-feed',async(req,res)=>{
try{
constnewsFeed=awaitNews.find().sort({date,-1}).limit(20);
letcurrentUserId=null;

```

```

const authHeader = req.headers.authorization;
if (authHeader && authHeader.startsWith('Bearer ')) {
  try {
    const jwt = require('jsonwebtoken');
    const token = authHeader.split(' ')[1];
    const decoded = jwt.verify(token, process.env.JWT_SECRET);
    const currentUserId = decoded.id ? decoded.id.toString() : decoded._id.toString();
  } catch (e) {}
}

const filteredNews = newsFeed.filter(item => {
  const isCancellationAlert = item.title?.includes('СЕАНССКАСОВАНО');
  const isSubscriptionAlert = item.title?.includes('Квитки вже у продажу');

  if (isCancellationAlert) {
    if (!currentUserId) return false;
    if (item.targetUsers && item.targetUsers.length > 0) {
      const stringUsers = item.targetUsers.map(u => u.toString().trim());
      return stringUsers.includes(currentUserId.trim());
    }
    return false;
  }

  if (isSubscriptionAlert) {
    if (!currentUserId) return false;
    if (!item.targetUsers || item.targetUsers.length === 0) return true;
    const stringUsers = item.targetUsers.map(u => u.toString().trim());
    return stringUsers.includes(currentUserId.trim());
  }

  return true;
});

res.json(filteredNews.slice(0, 10));
} catch (err) {
  res.status(500).json({ message, err.message });
}
});

router.get('/admin/edit/:id', protect, async (req, res) => {
  try {
    if (req.user.role !== 'admin') return res.status(403).json({ message, "No admin access" });
    const movie = await Movie.findById(req.params.id);
    if (!movie) return res.status(404).json({ message, "Фільм не знайдено" });

    const globalScreenings = await Screening.find({ movie, req.params.id });
    const formattedScreenings = globalScreenings.map(sc => ({
      ...sc._doc,
      _id, sc._id,
      dateTime, sc.startTime,

```

```

hall,sc.hallName,
priceStandard,sc.priceStandard!==undefined&&sc.priceStandard!==null?sc.priceStandard,150,
priceVip,sc.priceVip!==undefined&&sc.priceVip!==null?sc.priceVip,0
});

res.json({
...movie._doc,
screenings,formattedScreenings
});
}catch(err){
res.status(500).json({message,err.message});
}
});

router.post('/add',protect,upload.single('poster'),async(req,res)=>{
try{
if(req.user.role!=='admin')returnres.status(403).json({message,"Noadminaccess"});
constmovieData=JSON.parse(req.body.data);

constduplicateMovie=awaitMovie.findOne({
title,{regex,newRegExp(`^${movieData.title.trim()}$`,`i`)},
year,Number(movieData.year),
director,{regex,newRegExp(`^${movieData.director.trim()}$`,`i`)}
});

if(duplicateMovie){
if(req.file){
constfilePath=path.join(__dirname,'../frontend/public/posters',req.file.filename);
if(fs.existsSync(filePath))fs.unlinkSync(filePath);
}
returnres.status(400).json({
message,`Фільм"${movieData.title}"(${movieData.year}р.,режисер,${movieData.director})вже існуєвбазіданихплатформи`
});
}

if(req.file)movieData.posterUrl=`/posters/${req.file.filename}`;

const{screenings,generatedNews,...onlyMovieData}=movieData;

constnewMovie=newMovie({
...onlyMovieData,
screenings,screenings.map(sc=>({
dateTime,sc.dateTime,
hall,sc.hall,
priceStandard,Number(sc.priceStandard)||150,
priceVip,Number(sc.priceVip)||0
})))
});
awaitnewMovie.save();

```

		Сівченко М.Т.			ІПЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
		Сфремов Ю.М.				153
Змн.	Арк.	№ докум.	Підпис	Дата		

```

if(movieData.isShowingInCinema&&screenings?.length>0){
constscreeningsToCreate=screenings.map(sc=>({
movie,newMovie._id,
startTime,sc.dateTime,
hallName,sc.hall,
priceStandard,Number(sc.priceStandard)||150,
priceVip,Number(sc.priceVip)||0,
seats,[]
}));

for(constscToCreateofscreeningsToCreate){
scToCreate.seats=awaitgenerateSeatsForScreening(scToCreate.hallName);
}
awaitScreening.insertMany(screeningsToCreate);

if(generatedNews){
constuserIds=awaitgetSubscriptionUserIds(newMovie._id.toString());

constnewNews=newNews({
title,generatedNews.title,
text,generatedNews.text,
movieId,newMovie._id.toString(),
posterUrl,newMovie.posterUrl||null,
date,newDate(),
isAlert,userIds.length>0,
targetUsers,userIds
});
awaitnewNews.save();

if(global.io&&userIds.length>0){
global.io.emit('movie_subscription_alert',{
movieId,newMovie._id.toString(),
movieTitle,newMovie.title,
title,generatedNews.title,
text,generatedNews.text
});
}
if(global.io)global.io.emit('global_news_alert');
}

res.status(201).json(newMovie);
}catch(err){
res.status(500).json({message,err.message});
}
});

constgetSubscriptionUserIds=async(movieId)=>{
try{
constallSubs=awaitMovieSubscription.find({});
constfiltered=allSubs.filter(s=>s.movie.toString()===movieId.toString().trim());

```

		Сівченко М.Т.			ІІЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
		Сфремов Ю.М.				154
Змн.	Арк.	№ докум.	Підпис	Дата		

```

console.log(`[SUBCHECK]movieId=${movieId},всьогопідписок,${allSubs.length},знайдено,
${filtered.length}`);
returnfiltered.map(s=>s.user.toString());
}catch(err){
console.error("ПомилкаgetSubscriptionUserIds",err);
return[];
}
};
router.put('/update/:id',protect,upload.single('poster'),async(req,res)=>{
try{
if(req.user.role!=='admin')returnres.status(403).json({message,"Noadminaccess"});
constmovieData=JSON.parse(req.body.data);
constmovieId=req.params.id;

constduplicateMovie=awaitMovie.findOne({
_id,{$_id:movieId},
title,{$_id:newRegExp(`^${movieData.title.trim()}$`,`i`)},
year,Number(movieData.year),
director,{$_id:newRegExp(`^${movieData.director.trim()}$`,`i`)}
});

if(duplicateMovie){
if(req.file){
constfilePath=path.join(__dirname,'../frontend/public/posters',req.file.filename);
if(fs.existsSync(filePath))fs.unlinkSync(filePath);
}
returnres.status(400).json({
message,`Неможливозберегтизміни.Фільмзнавою"${movieData.title}"(${movieData.year}р.,
режисер,${movieData.director})вжеzareєстрованийусистемі!`
});
}

constmovieToUpdate=awaitMovie.findById(movieId);
if(!movieToUpdate)returnres.status(404).json({message,"Фільмнезнайдено"});

if(req.file){
if(movieToUpdate.posterUrl&&movieToUpdate.posterUrl.startsWith('/posters/')){
constoldFilename=movieToUpdate.posterUrl.replace('/posters/','');
constoldFilePath=path.join(__dirname,'../frontend/public/posters',oldFilename);
if(fs.existsSync(oldFilePath))fs.unlinkSync(oldFilePath);
}
movieData.posterUrl=`/posters/${req.file.filename}`;
}

const{screenings,generatedNews,...onlyMovieData}=movieData;

deleteonlyMovieData._id;
deleteonlyMovieData.__v;
deleteonlyMovieData.createdAt;
deleteonlyMovieData.updatedAt;

```

		Сівченко М.Т.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Сфремов Ю.М.				155
Змн.	Арк.	№ докум.	Підпис	Дата		

```

deleteonlyMovieData.screenings;

constcurrentScreeningsInDb=awaitScreening.find({movie,movied});
for(constoldScofcurrentScreeningsInDb){
conststillExists=screenings.some(newSc=>
newSc._id===oldSc._id.toString()||
(newSc.dateTime===oldSc.startTime.toISOString().slice(0,16)&&newSc.hall===oldSc.hallName)
);

if(!stillExists){
awaitprocessBulkRefundForScreening(oldSc._id);
awaitScreening.findByIdAndDelete(oldSc._id);
}
}

constcleanMovieScreenings=screenings.map(sc=>({
dateTime,sc.dateTime,
hall,sc.hall,
priceStandard,sc.priceStandard!==undefined&&sc.priceStandard!=="?
Number(sc.priceStandard),150,
priceVip,sc.priceVip!==undefined&&sc.priceVip!=="?Number(sc.priceVip),0
}));

constupdatedMovie=awaitMovie.findByIdAndUpdate(
movied,
{
$set,{
...onlyMovieData,
screenings,cleanMovieScreenings
}
},
{new,true,runValidators,true}
);

if(movieData.isShowingInCinema&&screenings&&screenings.length>0){
for(constscofscreenings){
constexists=awaitScreening.findOne({movie,movied,startTime,sc.dateTime,hallName,sc.hall});
if(!exists){
constnewSc=newScreening({
movie,movied,
startTime,sc.dateTime,
hallName,sc.hall,
priceStandard,sc.priceStandard!==undefined&&sc.priceStandard!=="?
Number(sc.priceStandard),150,
priceVip,sc.priceVip!==undefined&&sc.priceVip!=="?Number(sc.priceVip),0,
seats,awaitgenerateSeatsForScreening(sc.hall)
});
awaitnewSc.save();
}
}
}

```

		Сівченко М.Т.			ІІЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
		Сфремов Ю.М.				156
Змн.	Арк.	№ докум.	Підпис	Дата		

```

}

if(generatedNews&&screenings?.length>0){
  const allSubs=await MovieSubscription.find({});
  console.log('BCIΠΔΠICKI','JSON.stringify(allSubs));
  console.log('ШУКАЄМОmovieId','movieId,typeofmovieId);

  const newNews=new News({
    title,generatedNews.title,
    text,generatedNews.text,
    movieId,movieId,
    posterUrl,updatedMovie.posterUrl || null,
    date,new Date(),
    isAlert,userIds.length>0,
    targetUsers,userIds
  });
  await newNews.save();

  if(global.io&&userIds.length>0){
    global.io.emit('movie_subscription_alert',{
      movieId,movieId,
      movieTitle,updatedMovie.title,
      title,generatedNews.title,
      text,generatedNews.text
    });
  }
  if(global.io)global.io.emit('global_news_alert');
}
res.json(updatedMovie);
} catch (err){
  res.status(500).json({message,err.message});
}
});

router.get('/:id/subscription-status',protect,async(req,res)=>{
  try{
    const movieId=req.params.id;
    const cleanMovieIdStr=movieId.toString().trim();
    const movieObjId=new mongoose.Types.ObjectId(cleanMovieIdStr);

    const subscription=await MovieSubscription.findOne({
      user:req.user._id,
      $or:[
        {movie,cleanMovieIdStr},
        {movie,movieObjId}
      ]
    });
    res.json({isSubscribed,!subscription});
  } catch (err){
    res.status(500).json({message,err.message});
  }
}

```

		Сівченко М.Т.			ІІЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
		Сфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		157

```

}
});

router.post('/',id/subscribe',protect,async(req,res)=>{
try{
constmovieId=req.params.id;
constuserId=req.user._id;
constcleanMovieIdStr=movieId.toString().trim();
constmovieObjId=newmongoose.Types.ObjectId(cleanMovieIdStr);

constexistingSub=awaitMovieSubscription.findOne({
user,userId,
$or,[
{movie,cleanMovieIdStr},
{movie,movieObjId}
]
});

if(existingSub)returnres.status(400).json({message,"Вивжепідписанінасповіщенняцього
фільму."});

constsubscription=newMovieSubscription({user,userId,movie,movieObjId});
awaitsubscription.save();

res.status(201).json({message,"Виуспішнопідписалисянасповіщення!",isSubscribed,true});
}catch(err){
res.status(500).json({message,err.message});
}
});

router.delete('/',id/unsubscribe',protect,async(req,res)=>{
try{
constmovieId=req.params.id;
constuserId=req.user._id;
constcleanMovieIdStr=movieId.toString().trim();
constmovieObjId=newmongoose.Types.ObjectId(cleanMovieIdStr);

constdeletedSub=awaitMovieSubscription.findOneAndDelete({
user,userId,
$or,[
{movie,cleanMovieIdStr},
{movie,movieObjId}
]
});

if(!deletedSub)returnres.status(404).json({message,"Підпискунезнайдено."});
res.json({message,"Підпискускасовано.",isSubscribed,false});
}catch(err){
res.status(500).json({message,err.message});
}
}

```

		Сівченко М.Т.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Сфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		158

```

});

router.get('/', identifier, async (req, res) => {
  try {
    const { identifier } = req.params;
    let movie;
    if (/^[0-9a-fA-F]{24}$/.test(identifier)) {
      movie = await Movie.findById(identifier);
    }
    if (!movie) {
      const decodedSlug = decodeURIComponent(identifier).replace(/-/g, '');
      movie = await Movie.findOne({ title: { $regex: new RegExp(decodedSlug, 'i') } });
    }
    if (!movie) return res.status(404).json({ message: "Фільм не знайдено" });

    const averageRating = movie.rating ? movie.rating.toFixed(1) : '0.0';
    let hasRatedBefore = false;
    let userVoteScore = 0;

    const authHeader = req.headers.authorization;
    if (authHeader && authHeader.startsWith('Bearer ')) {
      try {
        const jwt = require('jsonwebtoken');
        const token = authHeader.split(' ')[1];
        const decoded = jwt.verify(token, process.env.JWT_SECRET);
        const existingRating = await Rating.findOne({ user: decoded.id, movie: movie._id });
        if (existingRating) {
          hasRatedBefore = true;
          userVoteScore = existingRating.score;
        }
      } catch (e) {}
    }

    res.json({
      ...movie._doc,
      displayRating: Number(averageRating),
      hasRated, hasRatedBefore,
      userRating, userVoteScore
    });
  } catch (err) {
    res.status(500).json({ message: err.message });
  }
});

router.post('/', id/rate, protect, async (req, res) => {
  try {
    const { userRating = 0 } = req.body;
    const movieId = req.params.id;
    const userId = req.user._id;

```

		Сівченко М.Т.			ІПЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
		Сфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		159

```

if(!userRating || userRating<1 || userRating>10){
return res.status(400).json({message,"Оцінка повинна бути від 1 до 10"});
}

const movie = await Movie.findById(movieId);
if(!movie) return res.status(404).json({message,"Фільм не знайдено"});

const alreadyRated = await Rating.findOne({user:userId,movie:movieId});
if(alreadyRated){
return res.status(400).json({message,"Вивже оцінювали цей фільм!"});
}

const newRatingRecord = new Rating({user:userId,movie:movieId,score:Number(userRating)});
await newRatingRecord.save();

movie.ratingSum += Number(userRating);
movie.ratingCount += 1;

const calculatedAverage = Number((movie.ratingSum/movie.ratingCount).toFixed(1));
movie.rating = calculatedAverage;
await movie.save();

res.json({
message,"Дякуємо!",
displayRating,calculatedAverage,
userRating,Number(userRating)
});
} catch (err){
res.status(500).json({message,err.message});
}
});

router.delete('/:id',protect,async(req,res)=>{
try{
if(req.user.role !== 'admin') return res.status(403).json({message,"No admin access"});
const movieId = req.params.id;

const movie = await Movie.findById(movieId);
if(!movie) return res.status(404).json({message,"Фільм не знайдено"});

const screeningsToDelete = await Screening.find({movie:movieId});
for(const sc of screeningsToDelete){
await processBulkRefundForScreening(sc._id);
}

await News.deleteMany({movieId:movieId});
if(movie.posterUrl && movie.posterUrl.startsWith('/posters/')){
const filename = movie.posterUrl.replace('/posters/','');
const filePath = path.join(__dirname,'../frontend/public/posters',filename);
if(fs.existsSync(filePath)) fs.unlinkSync(filePath);
}
}
}

```

		Сівченко М.Т.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Сфремов Ю.М.				160
Змн.	Арк.	№ докум.	Підпис	Дата		

```

}

await Movie.findByIdAndDelete(movieId);
await Screening.deleteMany({movie,movieId});

if(global.io)global.io.emit('global_news_alert');

res.json({message,"Фільм,сеанситановиниуспішновидалено!"});
}catch(err){
res.status(500).json({message,err.message});
}
});

module.exports=router;

```

Лістинг screeningRoutes.js

```

const express=require('express');
const router=express.Router();
const Screening=require('../models/Screening');
const Ticket=require('../models/Ticket');
const Hall=require('../models/Hall');
const {protect}=require('../middleware/authMiddleware');

const stripe=require('stripe')(process.env.STRIPE_SECRET_KEY);

router.post('/create',async(req,res)=>{
try{
const {movie,startTime,hallId,priceStandard,priceVip}=req.body;

const hall=await Hall.findById(hallId);
if(!hall)return res.status(404).json({message,"Зал не знайдено"});

const generatedSeats=hall.seats.map(s=>({
row,s.row,
number,s.col,
type,s.type || 'standard',
isBooked,false,
bookedBy,null,
lockedUntil,null,
lockedBy,null
})));

const newScreening=new Screening({
movie,
startTime,
hallName,hall.name,
priceStandard,
priceVip,
seats,generatedSeats
});

```

		Сівченко М.Т.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Сфремов Ю.М.				161
Змн.	Арк.	№ докум.	Підпис	Дата		

```

await new Screening.save();
res.status(201).json(new Screening);
} catch (err) {
res.status(500).json({ message, err.message });
}
});

router.post('/lock-seats', protect, async (req, res) => {
const { screeningId, seatIds } = req.body;
const userId = req.user.id || req.user._id;
try {
const screening = await Screening.findById(screeningId);
if (!screening) return res.status(404).json({ message, "Сеанс не знайдено" });

const now = new Date();
const newLockTime = new Date(now.getTime() + 10 * 60000);

let finalResponseTime = newLockTime;

for (let seatId of seatIds) {
const seat = screening.seats.id(seatId);
if (seat && !seat.isBooked) {

const isAlreadyLockedByMe = seat.lockedUntil &&
new Date(seat.lockedUntil) > now &&
String(seat.lockedBy) === String(userId);

if (isAlreadyLockedByMe) {
finalResponseTime = seat.lockedUntil;
} else {
seat.lockedUntil = newLockTime;
seat.lockedBy = userId;
}
}
}

await screening.save();

if (global.io) {
global.io.to(screeningId).emit('seats_updated', {
seatIds, seatIds,
status, 'locked',
lockedUntil, finalResponseTime,
lockedBy, userId
});
}

res.json({ message, "Заблоковано", expiresAt, finalResponseTime });
} catch (err) {

```

		Сівченко М.Т.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Сфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		162

```

console.error("Помилка локумість", err);
res.status(500).json({message, "Помилка блокування"});
}
});

router.post('/create-payment-intent', protect, async (req, res) => {
  try {
    const {totalPrice} = req.body;

    if (!totalPrice || totalPrice <= 0) {
      return res.status(400).json({message, "Некоректна сума оплати"});
    }

    const paymentIntent = await stripe.paymentIntents.create({
      amount, Math.round(totalPrice * 100),
      currency, 'uah',
      payment_method_types, ['card'],
    });

    res.json({clientSecret, paymentIntent.client_secret});
  } catch (err) {
    console.error("Помилка Stripe", err);
    res.status(500).json({message, "Помилка сервісу оплати Stripe", err.message});
  }
});

router.post('/book-multiple-seats', protect, async (req, res) => {
  const {screeningId, seats} = req.body;
  const userId = req.user.id || req.user._id;
  try {
    const screening = await Screening.findById(screeningId);
    if (!screening) return res.status(404).json({message, "Сеанс не знайдено"});

    const ticketsData = [];
    const seatIds = seats.map(s => s.id);

    for (let selectedSeat of seats) {
      const seatInDb = screening.seats.id(selectedSeat.id);

      if (seatInDb && !seatInDb.isBooked) {
        const finalPrice = seatInDb.type === 'vip'
          ? screening.priceVip
          : screening.priceStandard;

        seatInDb.isBooked = true;
        seatInDb.bookedBy = userId;
        seatInDb.lockedUntil = null;
        seatInDb.lockedBy = null;

        ticketsData.push({

```

		Сівченко М.Т.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Сфремов Ю.М.				163
Змн.	Арк.	№ докум.	Підпис	Дата		

```

movie,screening.movie,
user,userId,
screening,screeningId,
row,seatInDb.row,
seatNumber,seatInDb.number,
type,seatInDb.type || 'standard',
hallName,screening.hallName,
startTime,screening.startTime,
price,finalPrice,
status,'paid'
});
}
}

if(ticketsData.length>0){
awaitTicket.insertMany(ticketsData);
awaitscreening.save();

if(global.io){
global.io.to(screeningId).emit('seats_updated',{
seatIds,seatIds,
status,'booked'
});
}
res.json({message,"Успішнокуплено",ticketsCount,ticketsData.length});
}else{
res.status(400).json({message,"Місцявжезайняті"});
}

}catch(err){
console.error(err);
res.status(500).json({message,"Помилкаоплати,"+err.message});
}
});

router.get('/movie/,movieId',async(req,res)=>{
try{
constscreenings=awaitScreening.find({
movie:req.params.movieId,
startTime,{ $gte,newDate()})
}).sort({startTime,1});

res.json(screenings);
}catch(err){
res.status(500).json({message,"Помилказавантаженнясеансів,"+err.message});
}
});

router.get('/,id',async(req,res)=>{
try{

```

```
const screening = await Screening.findById(req.params.id).populate('movie');
res.json(screening);
} catch (err) {
res.status(500).json({ message, err.message });
}
});
```

```
module.exports = router;
```

Лістинг ticketRoutes.js

```
const express = require('express');
const router = express.Router();
const Ticket = require('../models/Ticket');
const { protect } = require('../middleware/authMiddleware');

router.get('/', protect, async (req, res) => {
  try {
    const tickets = await Ticket.find({ user: req.user._id })
      .populate('movie')
      .sort({ createdAt: -1 });

    res.json(tickets);
  } catch (err) {
    console.error("Помилка на бекенді квитків", err);
    res.status(500).json({ message, err.message });
  }
});

router.patch('/use/:id', protect, async (req, res) => {
  try {
    const ticket = await Ticket.findById(req.params.id);

    if (!ticket) {
      return res.status(404).json({ message, 'Квиток не знайдено' });
    }

    if (ticket.user.toString() !== req.user._id.toString() && req.user.role !== 'admin') {
      return res.status(401).json({ message, 'Немає доступу до цього квитка' });
    }

    if (ticket.isUsed) {
      return res.status(400).json({ message, 'Цей квиток вже був використано' });
    }

    ticket.isUsed = true;
    await ticket.save();

    res.json({ message, 'Квиток успішно використано!', ticket });
  } catch (err) {
    res.status(500).json({ message, err.message });
  }
});
```

		Сівченко М.Т.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Сфремов Ю.М.				165
Змн.	Арк.	№ докум.	Підпис	Дата		

```

}
});

router.get('/all',protect,async(req,res)=>{
try{
if(req.user.role!=='admin')returnres.status(403).json({message,"Доступзаборонено"});

consttickets=awaitTicket.find()
.populate('user','usernameemail')
.populate('movie','title')
.sort({createdAt:-1});

res.json(tickets);
}catch(err){
res.status(500).json({message,err.message});
}
});

router.delete('/:id',protect,async(req,res)=>{
try{
if(req.user.role!=='admin')returnres.status(403).json({message,"Доступзаборонено"});

constticket=awaitTicket.findByIdAndDelete(req.params.id);
if(!ticket)returnres.status(404).json({message,"Квитокнезнайдено"});

res.json({message,"Квитокуспішноповернуто/анульовано"});
}catch(err){
res.status(500).json({message,err.message});
}
});

module.exports=router;

```

		Сівченко М.Т.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремов Ю.М.				166
Змн.	Арк.	№ докум.	Підпис	Дата		