

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:

**«Вебсистема керування сеансами та онлайн-
бронюваннями місць у кінотеатрі»**

Виконала студентка групи ІПЗ-22-1
ФРІЗ Олександра Іванівна

Керівник роботи:
ЧИЖМОТРИЯ Олена Геннадіївна

Рецензент:
САВІЦЬКИЙ Роман Святославович

Житомир – 2026

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

«__» лютого 2026 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **ФРІЗ Олександра Іванівна**

Керівник роботи: **ЧИЖМОТРЕ Олена Геннадіївна**

Тема роботи: «Вебсистема керування сеансами та онлайн-бронюваннями місць у кінотеатрі»,

затверджена Наказом закладу вищої освіти від «27» лютого 2026 р., №92/с

Вихідні дані для роботи: Вебсистема керування сеансами та онлайн-бронюваннями місць у кінотеатрі

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

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Чижмотря О.Г.	27.02.2026р.	27.02.2026р.
2	Чижмотря О.Г.	27.02.2026р.	27.02.2026р.
3	Чижмотря О.Г.	27.02.2026р.	27.02.2026р.

РЕФЕРАТ

Випускна кваліфікаційна робота бакалавра складається з вебсистеми керування сеансами та онлайн-бронюваннями місць у кінотеатрі та пояснювальної записки. Випускна кваліфікаційна робота складається з вступу, 3-х розділів, висновків, списку використаних джерел із 30 найменувань (2 стор.), додатків (22 стор.), 30 таблиць та 23 ілюстрації по тексту. Повний обсяг роботи становить 79 сторінок, з них 55 сторінок основного тексту.

Метою роботи є розробка вебсистеми адміністрування сеансами та онлайн-бронювання місць у кінотеатрі із застосуванням актуальних вебтехнологій, що забезпечить комфортний та результативний процес підбору сеансу й бронювання квитків.

Архітектура системи ґрунтується на клієнт-серверному підході, котрий гарантує стабільність, можливість подальшого розширення та високу продуктивність за умов паралельної активності великої кількості відвідувачів. Серверний компонент втілює бізнес-логіку опрацювання сеансів, конфігурації залів та резервувань, тоді як клієнтська складова пропонує динамічний інтерфейс для ознайомлення з афішею та обрання місць у глядацькій залі.

Вебсистема володіє інтуїтивно зрозумілим інтерфейсом, що уможливорює для користувачів швидкий перегляд актуального розкладу сеансів, зручне обрання місць на візуалізованій інтерактивній мапі зали та здійснення попереднього онлайн-бронювання.

КЛЮЧОВІ СЛОВА: ВЕБСЕРВІС, БРОНЮВАННЯ, КІНОТЕАТР, СЕАНС, РОЗКЛАД, ІНТЕРАКТИВНА СХЕМА ЗАЛУ, ОНЛАЙН-БРОНЮВАННЯ

					ІПЗ.КР.Б – 121 – 26 – ПЗ			
Змн.	Арк.	№ докум.	Підпис	Дата	Вебсистема керування сеансами та онлайн-бронюваннями місць у кінотеатрі	Літ.	Арк.	Аркушів
Розроб.	Фріз О.І.							
Керівник	Чижмотря О.Г.						3	79
Рецензент	Савіцький Р.С.					Житомирська політехніка, Група ІПЗ-22-1		
Зав. каф.	Вакалюк Т.А.							

ABSTRACT

The bachelor's final qualification work consists of a web system for managing sessions and online booking of seats in a cinema and an explanatory note. The final qualification work consists of an introduction, 3 chapters, conclusions, a list of sources used with 30 names (2 pages), appendices (22 pages), 30 tables and 23 illustrations in the text. The total volume of the work is 79 pages, of which 55 pages are the main text.

The purpose of the work is to develop a web system for administering sessions and online booking of seats in a cinema using current web technologies, which will ensure a comfortable and efficient process of selecting a session and booking tickets.

The system architecture is based on a client-server approach, which guarantees stability, the possibility of further expansion and high performance under conditions of parallel activity of a large number of visitors. The server component implements the business logic of processing sessions, hall configuration and reservations, while the client component offers a dynamic interface for viewing the poster and choosing the desired positions in the auditorium.

The web system has an intuitive interface that allows users to quickly view the current schedule of screenings, conveniently select seats on a visualized interactive hall map, and make advance online reservations.

KEYWORDS: WEB SERVICE, RESERVATION, CINEMA, SHOWING, SCHEDULE, INTERACTIVE HALL SCHEDULE, ONLINE BOOKING.

		Фріз О.І.			ІІЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
		Чижевотря О.Г.				4
Змн.	Арк.	№ докум.	Підпис	Дата		

ЗМІСТ

РЕФЕРАТ	3
ВСТУП	7
РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ РЕАЛІЗАЦІЇ СИСТЕМИ ОНЛАЙН БРОНЮВАННЯ	9
1.1. Постановка задачі	9
1.2. Аналіз аналогів програмного продукту	10
1.3. Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення	14
Висновки до першого розділу	16
РОЗДІЛ 2. ПРОЄКТУВАННЯ СИСТЕМИ БРОНЮВАННЯ КВИТКІВ.....	18
2.1. Визначення варіантів використання та об'єктно-орієнтованої структури системи	18
2.2. Розробка бази даних системи.....	26
2.3. Проєктування та реалізація алгоритмів роботи системи	30
2.4. Реалізація функціоналу системи бронювання квитків.....	34
Висновки до другого розділу	37
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З СИСТЕМОЮ БРОНЮВАННЯ КВИТКІВ	38
3.1. Порядок встановлення та налаштування параметрів додатку	38
3.2. Структура інтерфейсу.....	40
3.3. Тестування роботи програмного продукту	49
Висновки до третього розділу.....	52
ВИСНОВКИ.....	53
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	54
ДОДАТКИ.....	56

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ORM – Object-Relational Mapping

API – Application Programming Interface

HTTP/HTTPS – HyperText Transfer Protocol (Secure)

JWT – JSON Web Token

OAuth – Open Authorization

QR-код – Quick Response код

JSON – JavaScript Object Notation

SQL – Structured Query Language

		Фріз О.І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмотря О.Г.				6
Змн.	Арк.	№ докум.	Підпис	Дата		

ВСТУП

Актуальність теми зумовлена тим, що платформи онлайн-бронювання перетворюються на невід'ємний елемент сучасної індустрії дозвілля, забезпечуючи цифрову трансформацію сервісів та зростання зручності для відвідувачів. Вони відіграють роль об'єднання між кінотеатрами та аудиторією, вдосконалюючи процедури купівлі квитків та розподілення місць у залах.

Щодо вебплатформ для бронювання місць у кінотеатрі, то вони виступають ключовими засобами автоматизації робочих процесів, завдяки яким клієнти мають доступ до афіші сеансів, фільмів та інформації про вільні місця. Такі системи дають змогу глядачам заощаджувати час, оминаючи черги біля кас і наперед планувати власний відпочинок.

Окрім цього, зазначені платформи пропонують комфортний інструментарій для графічного відображення схеми залів, де користувачі можуть у динамічному режимі обирати місця та проводити онлайн оплату. Це позитивно впливає на продуктивність функціонування кінотеатрів та поліпшення User Experience. Вебсистема також становить робочий інструмент для адміністрації кінотеатру, відкриваючи можливості оновлення інформації, регулювання тарифної політики та моніторингу конкретних показів.

Метою випускної кваліфікаційної роботи постає проєктування алгоритмів діяльності та практичне втілення вебплатформи для бронювання місць у кінотеатрі.

Окреслена мета передбачає виконання таких завдань:

- опрацювання теоретичних та практичних вимог стосовно створення платформ онлайн-бронювання;
- встановлення структури та базової архітектури системи;
- аналіз наявних вебсервісів із продажу квитків і виявлення їхніх характерних рис та слабких сторін;
- аргументація та підбір технологічного стеку для реалізації клієнтської та серверної складових системи;
- проєктування структурних елементів та логіки роботи системи;

		Фріз О.І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмотря О.Г.				7
Змн.	Арк.	№ докум.	Підпис	Дата		

- програмна реалізація вебсистеми для бронювання.

Об'єктом дослідження виступають методики та інструментарій створення вебресурсів та інформаційних систем у галузі електронної торгівлі та надання послуг.

Предметом дослідження є застосування веборієнтованих технологій для побудови системи бронювання квитків.

У ході роботи було задіяно методи об'єктно-орієнтованого проектування, моделювання структур баз даних, вебпрограмування та проектування користувацьких інтерфейсів.

Розроблена вебсистема керування сеансами та онлайн-бронюваннями місць у кінотеатрі може бути впроваджена у практичну діяльність кінотеатрів як додатковий інструмент поліпшення процесу продажу квитків, управління сеансами та взаємодії з відвідувачами. Система дозволяє скоротити навантаження на працівників, забезпечити цілодобовий доступ до бронювання місць та підвищити ефективність роботи кінотеатру. Адміністративна панель надає кінотеатру зручні засоби для управління, що робить систему готовим рішенням для використання у кінотеатрі.

		Фріз О.І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмотря О.Г.				8
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ РЕАЛІЗАЦІЇ СИСТЕМИ ОНЛАЙН БРОНЮВАННЯ

1.1. Постановка задачі

Вебсистема для бронювання місць у кінотеатрі – це система, створена з метою автоматизації процесу продажу квитків та покращення якості обслуговування відвідувачів. Суть проєкту полягає в створенні зручного середовища для дистанційного вибору фільмів, перегляду розкладу та бронювання місць у залі.

Говорячи про такі платформи, зазвичай вони містять наступні складові:

- афіша для публікації актуальної інформації про фільми, що знаходяться в прокаті або плануються до показу;
- інтерактивна схема залу як інструмент, що дозволяє користувачам бачити реальне розташування місць у залі;
- розклад сеансів у вигляді графіку показів із можливістю фільтрації за датою, часом, форматом та залом;
- система оплати з інтеграцією платіжних сервісів для миттєвої оплати замовлення;
- особистий кабінет як простір, де користувачі мають можливість переглядати історію своїх замовлень, зберігати активні електронні квитки, керувати особистими даними.

Щодо технічних завдань для реалізації сайту, то вони сформульовані наступним чином, визначення стеку технологій для розробки вебплатформи та вибір фреймворку для забезпечення швидкодії, масштабованості та безпеки системи. Проєктування та розробка структури бази даних для зберігання інформації про фільми, сеанси, зали, місця, замовлення та користувачів, розробка інтуїтивно зрозумілого користувацького інтерфейсу з використанням відповідних бібліотек.

Створення алгоритмів перевірки доступності місць для уникнення подвійного бронювання одного й того ж місця та реалізація механізму генерації

		Фріз О.І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмотря О.Г.				9
Змн.	Арк.	№ докум.	Підпис	Дата		

електронного квитка. Розробка панелі керування для адміністраторів кінотеатру з метою додавання нових фільмів, формування розкладу сеансів, зміни цін та управління залами. Також завантаження мультимедійного контенту, такого як постери та банери, проведення тестування сценаріїв бронювання, оплати та скасування замовлень для виявлення та виправлення помилок.

Щодо безпеки слід застосувати хешування паролів та захищених протоколів передачі даних.

Підсумовуючи все вище наведене, переді мною постають такі задачі:

- пошук та аналіз існуючих аналогів систем бронювання;
- збір вимог до функціоналу платформи;
- проєктування ER-діаграми та створення таблиць бази даних;
- реалізація ролей користувачів (Користувач, Адміністратор);
- налаштування адміністративної панелі для керування контентом кінотеатру;
- обрання мови програмування та середовища розробки;
- визначення структури сайту;
- комплексне тестування розробленої системи.

1.2. Аналіз аналогів програмного продукту

Аналіз аналогів виступає суттєвим етапом у створенні будь-якого продукту чи сервісу, а також може принести користь при адмініструванні та вдосконаленні вже наявних проєктів. Якщо вести мову про аналіз загалом, то він є корисним для усвідомлення того, що вже представлено в обраній галузі досліджуваного продукту та які перспективи існують для поліпшення програмного рішення. Принциповим моментом є те, що це сприяє виявленню свіжих ідей, функціональних можливостей або методик, котрі можливо імплементувати в продукт чи сервіс задля підвищення його конкурентоздатності та відповідності запитам цільової аудиторії.

		Фріз О.І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмотря О.Г.				10
Змн.	Арк.	№ докум.	Підпис	Дата		

1. Multiplex

Офіційний вебресурс [13] наймасштабнішої мережі багатозальних кінотеатрів на території України. Він спрямований на максимальну автоматизацію процедури обслуговування клієнта без залучення касира.

Головна сторінка сайту вміщує динамічний слайдер із постерами фільмів та забезпечує швидкий доступ до афіші (рис. 1.1).

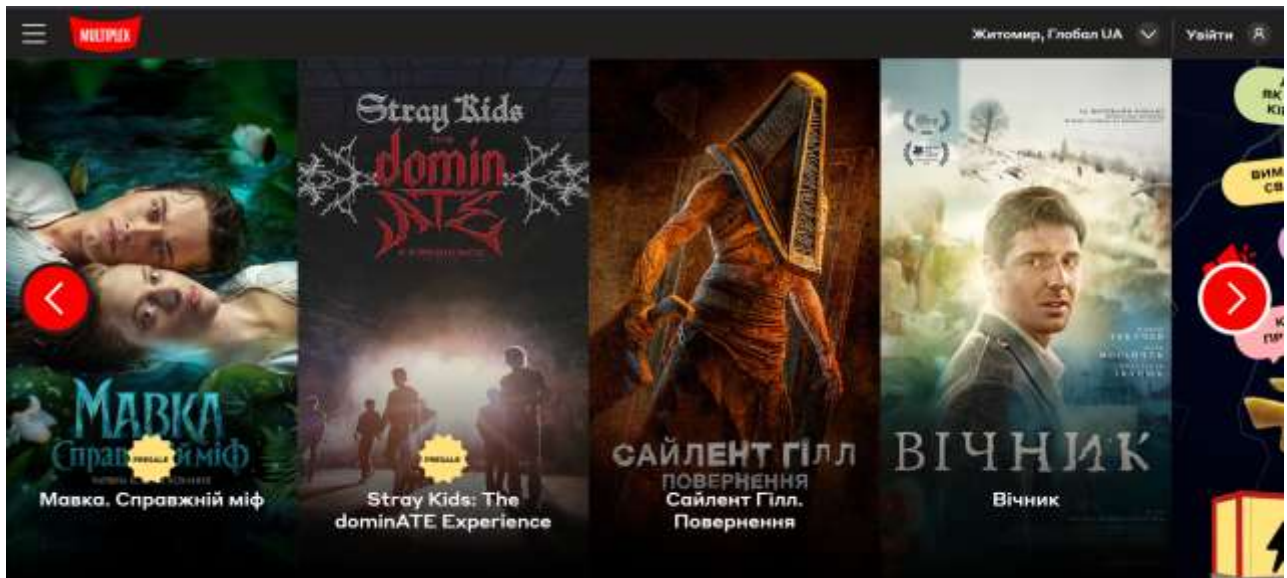


Рис. 1.1 – Головна сторінка сайту Multiplex

Розклад сеансів реалізовано у вигляді панелі з фільтрацією за датою, форматом та субтитрами, що дозволяє користувачу швидко знайти потрібний сеанс серед доступних (рис. 1.2).

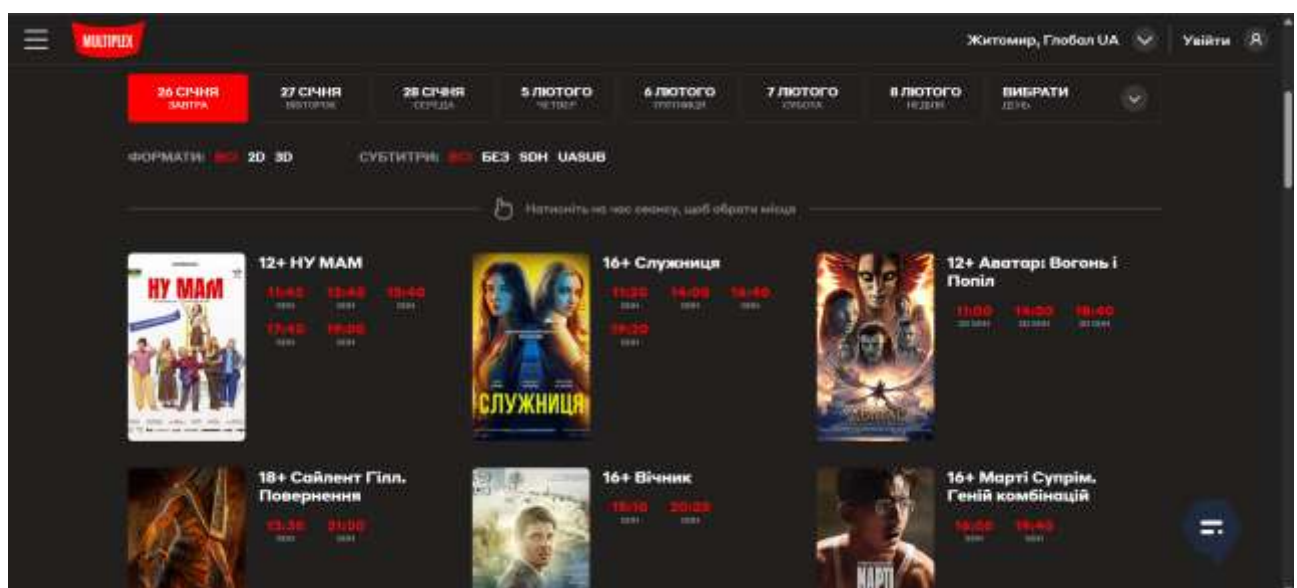


Рис. 1.2. Сторінка вибору сеансу сайту Multiplex

		Фріз О.І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижморя О.Г.				11
Змн.	Арк.	№ докум.	Підпис	Дата		

Серед ключових функцій наявні інтерактивна схема залу, де наочно продемонстровано категорії місць та їхню вартість. Особистий кабінет де фіксується хронологія переглядів, актуальні квитки та накопичені бонуси в межах програми заохочення постійних клієнтів. Також присутня фільтрація контенту та пошук кінострічок. Однією з цікавих функцій реалізовано Онлайн-бар, опція яка надає можливість дистанційного замовлення попкорну та напоїв, шляхом додавання до спільного кошика разом із квитком. Реалізовано відшкодування коштів за квитки, що не потребує особистого візиту до адміністратора кінотеатру.

Серед переваг сервісу варто відзначити інтуїтивно прозорий інтерфейс та високу оперативність роботи, наявність адаптованої мобільної версії та окремого застосунку, розгалужену систему лояльності з можливістю накопичення бонусів. Також наявна можливість придбання квитків з інтеграцією плажінних інструментів, зокрема Apple Pay та Google Pay.

Проте сервіс має й певні недоліки, у періоди надмірного навантаження на серверне обладнання виникає тимчасова недоступність інтерактивної схеми залу. Стартова сторінка містить значну кількість рекламних банерів, а процедура поновлення доступу до акаунту в разі зміни контактного номеру стає ускладненою.

2. Планета Кіно

Вебплатформа [14] однієї з мереж кінотеатрів в Україні, вона дозволяє не лише купувати квитки, а й керувати бонусами, дарувати сертифікати та купувати 3D-окуляри онлайн.

Головна сторінка сайту одразу відображає актуальний фільм із доступними сеансами на найближчі дні, що забезпечує швидкий перехід до вибору квитка (рис. 1.3).

		Фріз О.І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмотря О.Г.				12
Змн.	Арк.	№ докум.	Підпис	Дата		



Рис. 1.3 – Головна сторінка сайту Planet Kino

Сторінка вибору місць реалізована у вигляді інтерактивної схми залу з відображенням рядів і місць у реальному часі, що дозволяє зручно орієнтуватись у розташуванні вільних та зайнятих місць (рис. 1.4).

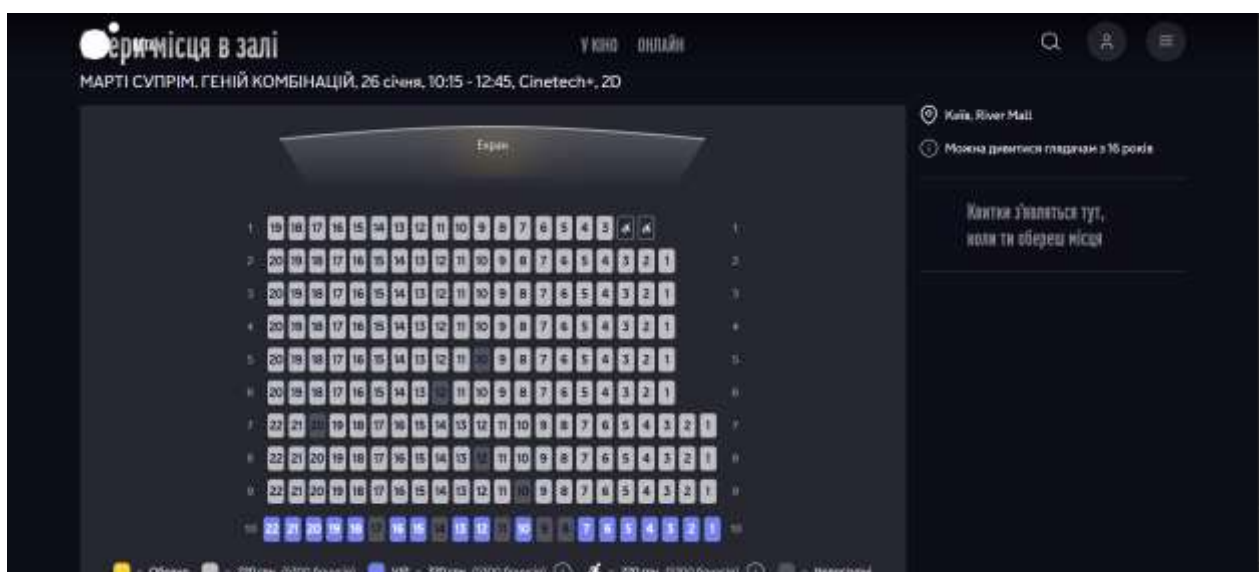


Рис. 1.4. – Інтерактивна мапа вибору місць сайту Planet Kino

Серед основних функцій програма лояльності, також кожен сеанс має чітке маркування технології показу з детальним описом та промо-матеріалами. Наявна функція, що дозволяє користувачеві самостійно скасувати замовлення та миттєво повернути бонуси або кошти. Кабінет користувача зберігає не лише історію переглядів, а й вподобані жанри, що дозволяє системі формувати персональні рекомендації щодо майбутніх прем'єр.

		Фріз О.І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмостря О.Г.				13
Змн.	Арк.	№ докум.	Підпис	Дата		

Щодо переваг сервісу, виділяється зручний користувацький інтерфейс, гнучка система повернення коштів, наявність окремого VIP-розділу з можливістю замовлення страв ресторанного типу, а також високий рівень захисту персональних даних та стабільність роботи під час пікових навантажень.

До недоліків належать менша географічна покритість, що обмежує цільову аудиторію, складна система реєстрації для нових користувачів, а також відсутність можливості бронювання квитків без попередньої оплати.

Таблиця 1.1

Порівняння аналогів

Назва	Multiplex	Planet Kino
Онлайн-бронювання	+	+
Купівля товарів бару	+	-
Інтерактивна схема	+	+
Система лояльності	+	+
Повернення онлайн	+	+
Мобільний додаток	+	+
Доступ на всіх пристроях	+	+
Доступ без реєстрації	+	+

1.3. Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення

Вибір архітектурної моделі та відповідного технологічного інструментарію має визначальний вплив на всі етапи життєвого циклу вебсистеми від початкового проєктування до подальшої експлуатації та масштабування. Коректно підібрана архітектура спроможна не лише спростити роботу, а й гарантувати відмовостійкість платформи в моменти пікових навантажень, а вдалий добір засобів реалізації забезпечує гнучкість системи та цілісність даних.

Архітектурні рішення безпосередньо позначаються на швидкодії серверної частини, адже навіть незначна затримка у відповіді здатна спричинити втрату

потенційного клієнта. Саме архітектурний каркас окреслює ступінь захищеності, прозоре розмежування прав доступу та ізоляція модулів.

У сучасній веброзробці найчастіше застосовуються монолітна та мікросервісна архітектурні моделі. Принциповою різницею між ними є те, що в моноліті всі елементи програми зосереджені в єдиній кодовій базі та послуговуються спільною базою даних, тоді як у мікросервісному підході функціонал дробиться на автономні, незалежні блоки.

Беручи до уваги специфіку завдань проєкту та критичну необхідність підтримання транзакцій під час продажу квитків, для реалізації серверної частини було обрано саме монолітну архітектуру. В умовах монолітної структури зі спільним сховищем даних організувати транзакційну обробку та блокування записів значно простіше та надійніше, ніж у розподіленому середовищі мікросервісів.

Загальна ж організація вебдодатку базується на класичній клієнт-серверній моделі, що передбачає чітке відокремлення клієнтської складової від серверної.

Визначившись із загальною архітектурною концепцією, було здійснено добір конкретних інструментальних засобів для практичної реалізації платформи. Для забезпечення технологічної єдності та високої продуктивності системи було обрано сучасний стек, що базується на мові JavaScript як на клієнтському, так і на серверному боці.

Для побудови користувацького інтерфейсу було обрано бібліотеку React [1]. Вона ґрунтується на компонентному підході, що дає змогу розбити інтерфейс кінотеатру на компоненти багаторазового застосування. Така архітектура значно полегшує управління станом застосунку та спрощує подальшу підтримку кодової бази. Завдяки використанню віртуального DOM, React забезпечує високу швидкодію оновлення інтерфейсу та необхідну реактивність. Крім того, широка екосистема бібліотеки та наявність великої кількості готових рішень дозволяють пришвидшити процес розробки.

Як серверне середовище було обрано Node.js. Це рішення є оптимальним з огляду на його асинхронну модель введення та виведення. Системи бронювання

		Фріз О.І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмотря О.Г.				15
Змн.	Арк.	№ докум.	Підпис	Дата		

часто зазнають пікових навантажень, коли значна кількість відвідувачів одночасно перевіряють наявність вільних посадкових позицій. Node.js здатен ефективно обслуговувати велику кількість одночасних з'єднань. Використання єдиної мови програмування JavaScript як на фронтенді, так і на бекенді спрощує комунікацію між частинами системи.

Вибір PostgreSQL як базу даних зумовлений тим, що ця об'єктно-реляційна СКБД є загальновизнаним стандартом надійності. Для системи реалізації квитків найвагомішим аспектом виступає цілісність даних. Крім того, PostgreSQL володіє потужним інструментарієм для роботи з даними у форматі JSON, що надає необхідної гнучкості для зберігання нестандартних конфігурацій.

Для спрощення процедури розгортання та гарантування ідентичності середовищ виконання застосунку у проєкті було задіяно технологію Docker. Вона дозволяє ізолювати клієнтську частину, серверний компонент та базу даних у вигляді окремих контейнерів разом із усіма необхідними залежностями. Такий підхід усуває проблему несумісності версій програмного забезпечення та забезпечує стабільне функціонування системи незалежно від операційної системи.

Підсумовуючи, застосування монолітної архітектури всередині клієнт-серверної моделі, підкріплене використанням зв'язки React, Node.js та PostgreSQL у поєднанні з контейнеризацією Docker, формує сучасний, гнучкий та надійний фундамент для побудови вебсистеми бронювання квитків у кінотеатрі.

Висновки до першого розділу

Проведено аналіз предметної області, за результатами якого сформульовано ключові вимоги до вебсистеми керування сеансами та онлайн-бронюваннями місць у кінотеатрі.

У процесі дослідження було розглянуто аналоги, виявлено їхні основні переваги та типов. На основі цього аналізу було сформовано перелік власних функціональних вимог до системи.

		Фріз О.І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмотря О.Г.				16
Змн.	Арк.	№ докум.	Підпис	Дата		

Спираючись на визначені вимоги, було описано вибір архітектурного рішення та визначено набір інструментальних засобів для реалізації проєкту, бібліотеку React для клієнтської частини, серверне середовище Node.js, СКБД PostgreSQL для зберігання даних та технологію контейнеризації Docker для спрощення розгортання.

Отримані результати створюють необхідне середовище для переходу до наступного етапу проєктування бази даних та програмної реалізації компонентів вебсистеми.

		Фріз О.І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмотря О.Г.				17
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 2. ПРОЄКТУВАННЯ СИСТЕМИ БРОНЮВАННЯ КВИТКІВ

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

Вебсистема керування сеансами та онлайн-бронюваннями місць у кінотеатрі пропонує відвідувачам зручний вебінтерфейс, за допомогою якого вони отримують змогу переглядати актуальну афішу, знайомитися з розкладом сеансів, обирати місця на інтерактивній мапі зали та оформлювати бронювання квитків.

Робота системи пов'язана з активністю користувачів на платформі. Усіх задіяних осіб можна логічно розподілити на дві ключові ролі – глядачі, зацікавлені в пошуку фільмів і адміністратори кінотеатру, які відповідають за наповнення контентом, складанням розкладу та моніторингом процесу. Кожен окремий сеанс охоплює відомості про кінострічку, час старту показу, поточний стан зайнятості місць. Задля візуалізації взаємозв'язків між головними складовими системи та її користувачами було побудовано діаграму варіантів використання (рис. 2.1).

Опис діаграми варіантів використання наведено в таблицях 2.1 – 2.7.

Таблиця 2.1

Опис прецеденту «Реєстрація особистого кабінету»

Назва прецеденту	Реєстрація особистого кабінету
Актори	Користувач
Короткий опис	Користувач створює власний обліковий запис у системі, вказуючи необхідні персональні.

Таблиця 2.2

Опис прецеденту «Перегляд афіші та розкладу»

Назва прецеденту	Перегляд афіші та розкладу
Актори	Користувач
Короткий опис	Користувач має змогу ознайомитися з переліком фільмів, що перебувають у прокаті або плануються до показу, також переглянути актуальний графік сеансів із зазначенням часу.

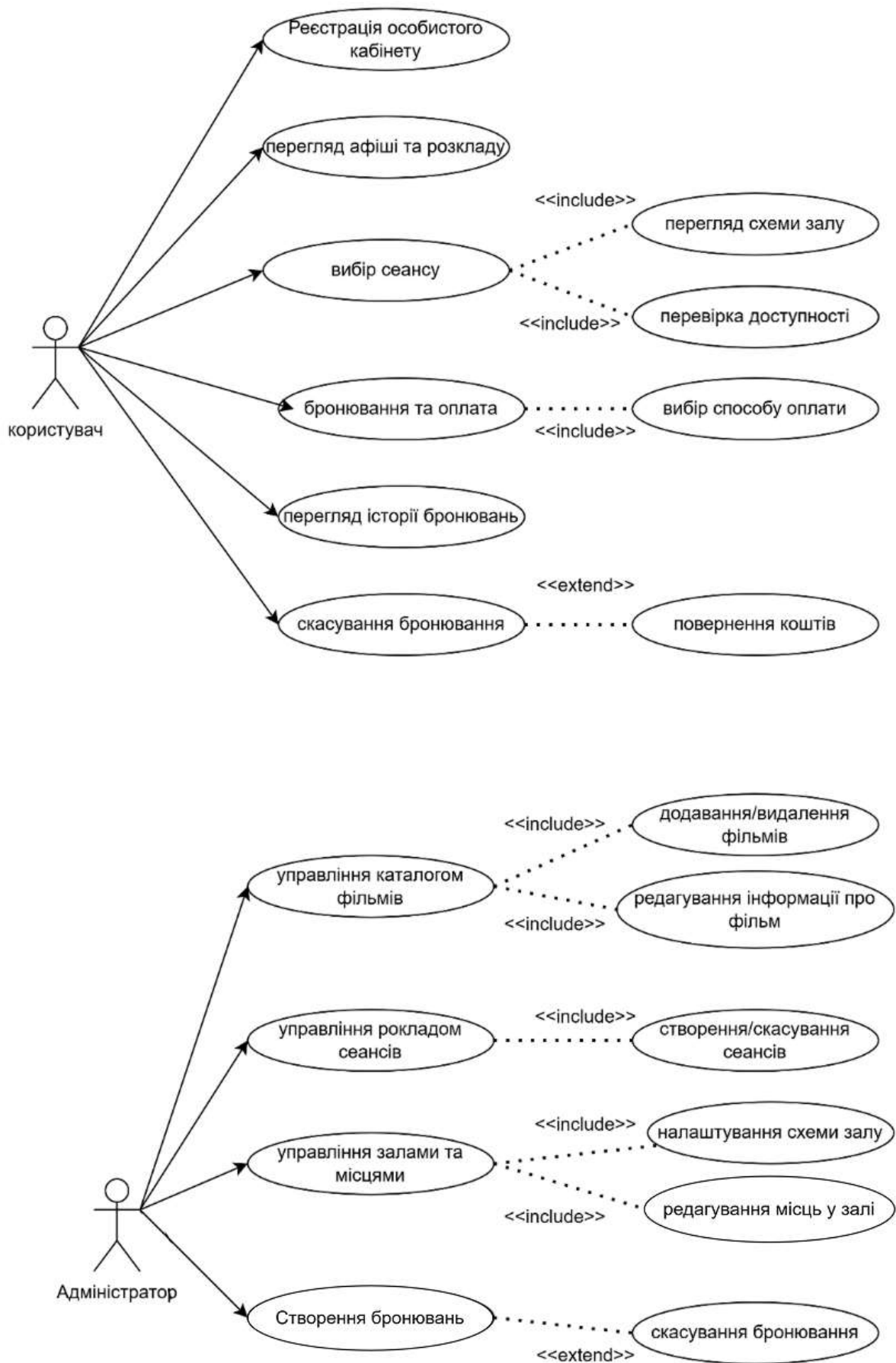


Рис. 2.1. Діаграма варіантів використання системи

Таблиця 2.3

Опис прецеденту «Вибір сеансу»

Назва прецеденту	Вибір сеансу
Актори	Користувач
Короткий опис	Користувач обирає конкретний сеанс із запропонованого розкладу для подальшого бронювання та купівлі квитків.

Таблиця 2.4

Опис прецеденту «Перегляд схеми залу»

Назва прецеденту	Перегляд схеми залу
Актори	Користувач
Короткий опис	Після вибору сеансу користувачеві відображається інтерактивна мапа зали, де візуально відображено розташування рядів, місць та їх статус.

Таблиця 2.5

Опис прецеденту «Перевірка доступності місць»

Назва прецеденту	Перевірка доступності місць
Актори	Користувач
Короткий опис	Система в реальному часі відображає актуальний стан заповненості залу та не дає змогу обрати викуплені місця.

Таблиця 2.6

Опис прецеденту «Бронювання та оплата»

Назва прецеденту	Бронювання та оплата
Актори	Користувач
Короткий опис	Користувач позначає місця та обирає спосіб оплати.

Таблиця 2.7

Опис прецеденту «Вибір способу оплати»

Назва прецеденту	Вибір способу оплати
Актори	Користувач
Короткий опис	У процесі оформлення бронювання користувач обирає зручний для нього метод розрахунку.

Таблиця 2.8

Опис прецеденту «Перегляд історії бронювань»

Назва прецеденту	Перегляд історії бронювань
Актори	Користувач
Короткий опис	Зареєстрований користувач має доступ до архіву бронювань, де є можливість переглянути історію придбаних та скасованих квитків.

Таблиця 2.9

Опис прецеденту «Скасування бронювання»

Назва прецеденту	Скасування бронювання
Актори	Користувач
Короткий опис	Користувач має можливість анулювати раніше оформлене бронювання.

Таблиця 2.10

Опис прецеденту «Управління каталогом фільмів»

Назва прецеденту	Управління каталогом фільмів
Актори	Адміністратор
Короткий опис	Адміністратор керує контентом афіші, додаючи нові фільми або видаляє застарілі.

Таблиця 2.11

Опис прецеденту «Додавання/видалення фільмів»

Назва прецеденту	Додавання/видалення фільмів
Актори	Адміністратор
Короткий опис	Внесення нових кінострічок до бази даних системи або повне вилучення тих, що завершили прокат.

Таблиця 2.12

Опис прецеденту «Редагування інформації про фільм»

Назва прецеденту	Редагування інформації про фільм
Актори	Адміністратор
Короткий опис	Оновлення даних фільму, зокрема опису, постера, тривалість, оцінка

Таблиця 2.13

Опис прецеденту «Управління розкладом сеансів»

Назва прецеденту	Управління розкладом сеансів
Актори	Адміністратор
Короткий опис	Формування розкладу шляхом призначення конкретним фільмам час, дату та зал для демонстрації.

Таблиця 2.14

Опис прецеденту «Створення/скасування сеансів»

Назва прецеденту	Створення/скасування сеансів
Актори	Адміністратор
Короткий опис	Додавання нових позицій до розкладу або видалення запланованих показів у разі зміни.

Таблиця 2.15

Опис прецеденту «Управління залами та місцями»

Назва прецеденту	Створення/скасування сеансів
Актори	Адміністратор
Короткий опис	Внесення параметрів глядацьких залів та визначення структури посадкових місць.

Таблиця 2.16

Опис прецеденту «Налаштування схеми залу»

Назва прецеденту	Налаштування схеми залу
Актори	Адміністратор
Короткий опис	Визначення наповнення зали: кількості рядів, кількості місць у ряду та наявності проходів.

Таблиця 2.17

Опис прецеденту «Підтвердження бронювань»

Назва прецеденту	Створення бронювань
Актори	Адміністратор
Короткий опис	Ручне створення окремих замовлень у разі виникнення непередбачуваних ситуацій або робота зі службою підтримки клієнтів.

Основною метою вебсисетми є створення єдиної цифрової платформи, що об'єднує глядачів та адміністраторів кінотеатру в рамках зручного середовища. Система забезпечує повний цикл взаємодії, від перегляду афіші та вибору місць до оплати й отримання електронного квитка.

Система орієнтована на два типи користувачів. Адміністратор отримує інструментарій керування, додавання та редагування фільмів із зазначенням строків прокату, налаштування розкладу сеансів, управління залами та схемами розміщення місць, моніторинг бронювань, а також керування користувачами. Натомість глядач отримує зручний інтерфейс для пошуку фільмів, перегляду детальної інформації про них, вибору кінотеатру та конкретного сеансу, інтерактивного вибору місць у залі, здійснення оплати та збереження квитків в особистому кабінеті.

Функціональні вимоги до системи передбачають підтримку двох способів аутентифікації – через створення профілю за допомогою електронної пошти та використовуючи обліковий запис Google. Інтерфейс має бути зрозумілим та забезпечувати швидкий доступ до потрібної інформації без зайвих перешкод. Процес бронювання супроводжється автоматичними email-сповіщеннями на кожному етапі, при створенні бронювання, підтвердженні оплати та скасуванні. Після успішної оплати система автоматично генерує квиток, який зберігається в особистому кабінеті користувача. Адміністратори мають доступ до окремої панелі керування з розширеним функціоналом, недоступним для звичайного користувача.

Нефункціональні вимоги охоплюють стабільну та безперервну роботу системи, захист персональних даних користувачів шляхом хешування паролів та використання HTTP-only cookie для зберігання токенів аутентифікації.

Серверна частина розгортається в Docker-контейнерах, що забезпечує ізольоване середовище виконання. Для обробки HTTP-запитів і реалізації бізнес-логіки застосовується Node.js [2] із фреймворком Express.js [3] та мовою TypeScript [10], що підвищує надійність коду завдяки статичній типізації. Усі дані зберігаються в реляційній базі даних PostgreSQL [4] з використанням ORM-

		Фріз О.І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмотря О.Г.				23
Змн.	Арк.	№ докум.	Підпис	Дата		

бібліотеки [5] для взаємодії з таблицями. Медіафайли, зокрема постери фільмів та аватари користувачів зберігаються у хмарному сховищі Amazon S3 [7].

Клієнтська частина реалізована на базі бібліотеки React із використанням Tailwind CSS [9] для стилізації компонентів. Застосунок працює в будь-якому сучасному браузері з підтримкою JavaScript і забезпечує асинхронну взаємодію із серверним API без перезавантаження сторінки.

У контексті вебсистеми онлайн бронювання особливе значення має забезпечення цілісності даних при одночасній роботі кількох користувачів. Механізм транзакцій PostgreSQL гарантує що не можливо виконати подвійне бронювання одного й того самого місця на сеанс. Реляційні зв'язки між таблицями не дозволять створити бронювання на неіснуючий сеанс або прив'язати його до видаленого користувача. Уся інформація про кінотеатри, зали, фільми, сеанси, користувачів та бронювань доступна як через клієнтський інтерфейс, так і через адміністративну панель.

Вимоги до апаратного забезпечення

Для забезпечення стабільної роботи серверної частини вебсистеми висуваються наступні мінімальні системні вимоги:

- 16 Гб пам'яті;
- 100 Гб вільного дискового простору;
- процесор з тактовою частотою не нижче 1GHz;
- доступ до Інтернету;
- операційна система Windows або Unix.

Аналіз функціональних вимог дозволив виділити наступні сутності, що забезпечать програмну реалізацію додатку. На рисунку 2.2. наведено діаграму класів, яка використовується для його роботи.

		Фріз О.І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмотря О.Г.				24
Змн.	Арк.	№ докум.	Підпис	Дата		

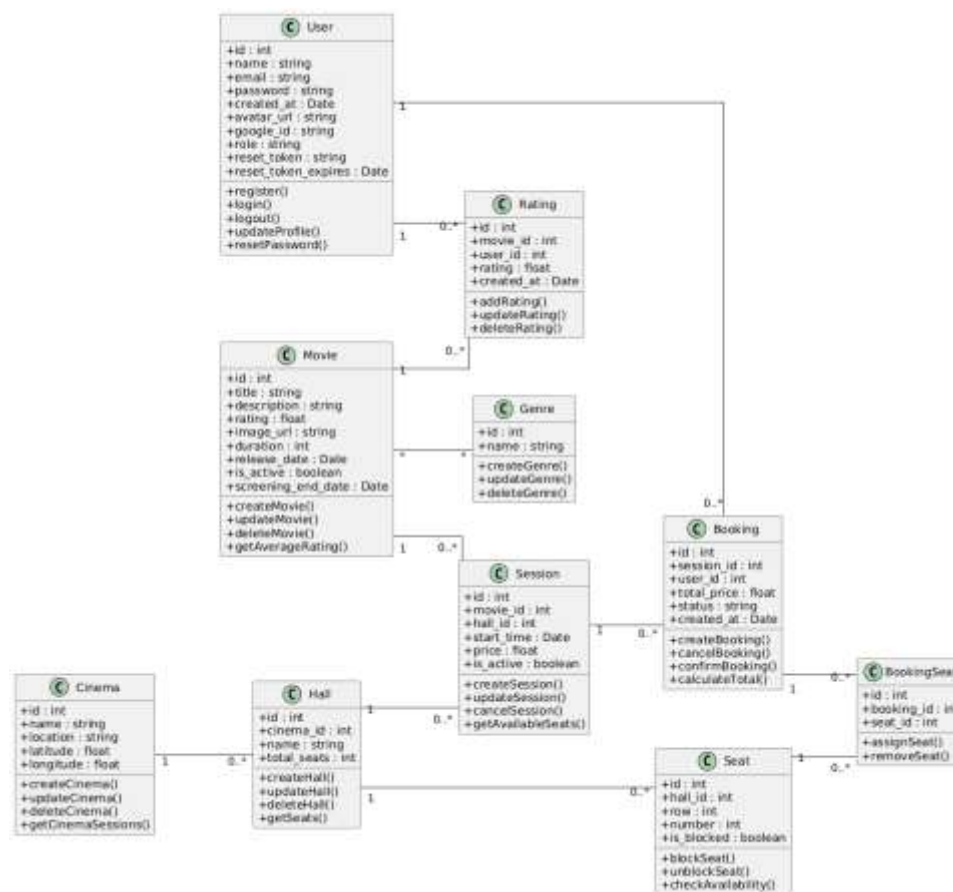


Рис.2.2. Діаграма класів

User – клас який відповідає за облікові записи користувачів. Реалізує методи register() та login() для реєстрації та аутентифікації, updateProfile() і resetPassword() для керування акаунтом, а саме редагування особистих даних та скидання паролю.

Rating – клас для оцінювання фільмів користувачами. Надає методи addRating(), updateRating() та deleteRating() які дозволяють виставляти оцінку, змінювати її та видаляти.

Movie – клас для зберігання інформації про фільми. Методи createMovie(), updateMovie(), deleteMovie() забезпечують керування каталогом фільмів з адмін панелі, а getAvarageRating() обчислює середню оцінку на основі відгуків.

Genre – допоміжний клас щоб розбивати на категорії фільми за жанром. Реалізує методи createGenre(), updateGenre() та deleteGenre() для керування довідником жанрів.

Session – клас, що описує параметри сеансу. Метод createSession() дозволяє адміністратору додавати новий сеанс до розкладу, updateSession() редагувати

його параметри, `cancelSession` скасовувати показ, а метод `getAvailableSeats()` дозволяє отримувати перелік вільних місць у залі на момент запиту.

`Booking` – клас який керує бронюванням місць. Метод `createBooking()` створює нове бронювання з перевіркою доступності місць, `cancelBooking()` скасовує його, `confirmBooking()` підтверджує після успішної оплати, `calculateTotal()` обчислює підсумкову суму.

`BookingSeat` – проміжний клас, що зв'язує конкретне бронювання з певними місцями у залі. Метод `assignSeat()` призначає місце до бронювання, `removeSeat()` звільняє місце.

`Seat` – клас, що описує окреме місце в залі. Методи `blockSeat()` та `unblockSeat()` дозволяють адміністратору керувати доступністю місць, а `checkAvailability()` перевіряє чи можна зарезервувати певне місце.

`Hall` – клас який описує зал кінотеатру. Надає CRUD-методи `createHall()`, `updateHall()`, `deleteHall()` які забезпечують керування залами з адмвн панелі, а `getSeats()` повертає перелік усіх місць конкретного залу.

`Cinema` – клас, що містить дані про кінотеатри, має методи `createCinema()`, `updateCinema()`, `deleteCinema()` які реалізують керування кінотеатрами, а `getCinemaSessions()` повертає розклад сеансів для конкретного кінотеатру.

2.2. Розробка бази даних системи

Для зберігання та опрацювання даних у вебсистемі керування сеансами й онлайн-резервування місць у кінотеатрі було задіяно систему управління базами даних PostgreSQL.

База даних системи складається з одинадцяти таблиць:

1. `users` – збереження кліристувачів
2. `cinemas` – збереження кінотеатрів
3. `halls` – збереження залів
4. `seats` – збереження місць
5. `movies` – збереження фільмів
6. `genres` – збереження жанрів

		Фріз О.І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмотря О.Г.				26
Змн.	Арк.	№ докум.	Підпис	Дата		

7. movie_genres – збереження зв'язку між фільмами та жанрами
8. sessions – збереження сеансів
9. bookings – збереження бронювань
10. booking_seats – збереження заброньованих місць
11. ratings – збереження рейтингів

Таблиця 2.18

Опис таблиці User

Поле	Тип	Призначення
id	int	Унікальний ідентифікатор користувача
name	varchar	Повне ім'я користувача
email	varchar	Електронна пошта
password	varchar	Хешований пароль
created_at	timestamp	Дата та час реєстрації облікового запису
avatar_url	varchar	Посилання на зображення аватара
google_id	varchar	Ідентифікатор google-користувача
role	varchar	Роль
reset_token	varchar	Токен для скидання паролю
reset_token_expires	timestamp	Дата та час закінчення дії токена

Таблиця 2.19

Опис таблиці Cinemas

Поле	Тип	Призначення
id	int	Унікальний ідентифікатор кінотеатру
name	varchar	Назва кінотеатру
location	varchar	Адреса кінотеатру
latitude	float	Географічна широта розташування кінотеатру
longitude	float	Географічна довгота розташування кінотеатру

Таблиця 2.20

Опис таблиці Halls

Поле	Тип	Призначення
id	int	Унікальний ідентифікатор залу
cinema_id	int	Зв'язок з таблицею кінотеатрів
name	varchar	Назва залу
total_seats	int	Загальна кількість місць

Таблиця 2.21

Опис таблиці Seats

Поле	Тип	Призначення
id	int	Унікальний ідентифікатор місця
hall_id	int	Зв'язок з таблицею залів
row	int	Номер ряду
number	int	Номер місця
is_blocked	boolean	Заблоковане місце

Таблиця 2.22

Опис таблиці Movies

Поле	Тип	Призначення
id	int	Унікальний ідентифікатор фільму
title	varchar	Назва фільму
description	text	Опис фільму
rating	int	Оцінка фільму
image_url	varchar	Посилання на постер фільму
duration	int	Тривалість в хвилинах
release_date	date	Дата виходу в прокат
created_at	timestamp	Дата додавання фільму до системи
is_active	boolean	Активний фільм
screening_end_date	date	Кінцева дата прокату

Таблиця 2.23

Опис таблиці Genres

Поле	Тип	Призначення
id	int	Унікальний ідентифікатор жанру
name	varchar	Назва жанру

Таблиця 2.24

Опис таблиці Movie_genres

Поле	Тип	Призначення
movie_id	int	Зв'язок з таблицею фільмів
genre_id	int	Зв'язок з таблицею жанрів

Таблиця 2.25

Опис таблиці Sessions

Поле	Тип	Призначення
id	int	Унікальний ідентифікатор сеансу
movie_id	int	Зв'язок з таблицею фільмів
hall_id	int	Зв'язок з таблицею залів
start_time	timestamp	Дата та час початку сеансу
price	decimal	Ціна за квиток
is_active	boolean	Активний сеанс

Таблиця 2.26

Опис таблиці Bookings

Поле	Тип	Призначення
id	int	Унікальний ідентифікатор бронювання
session_id	int	Зв'язок з таблицею сеансів
user_id	int	Зв'язок з таблицею користувачів
total_price	decimal	Загальна сума до сплати
created_at	timestamp	Дата та час оформлення бронювання
status	varchar	Статус бронювання

Таблиця 2.27

Опис таблиці Booking_seats

Поле	Тип	Призначення
id	int	Унікальний ідентифікатор запису
booking_id	int	Зв'язок з таблицею бронювань
seat_id	int	Зв'язок з таблицею місць

Таблиця 2.28

Опис таблиці Ratings

Поле	Тип	Призначення
id	int	Унікальний ідентифікатор оцінки
movie_id	int	Зв'язок з таблицею фільмів
user_id	int	Зв'язок з таблицею користувачів
rating	int	Значення оцінки
created_at	timestamp	Дата та час створення оцінки

База даних вебсистеми побудована на основі реляційної СУБД PostgreSQL, для взаємодії серверної частини з базою даних використовується ORM-бібліотека TypeORM, яка забезпечує типізований доступ до таблиць через entity-

		Фріз О.І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмотря О.Г.				29
Змн.	Арк.	№ докум.	Підпис	Дата		

класи та автоматично формує SQL-запити на основі визначених моделей і їхніх зв'язків.

Між таблицями встановлено зв'язки через зовнішні ключі, таким чином система не дозволить створити бронювання на неіснуючий сеанс або прив'язати місце до визначеного залу. При отриманні даних TypeORM автоматично формує JOIN-запити через механізм QueryBuilder, що дозволяє ефективно отримувати пов'язані записи з кількох таблиць одночасно.

Операція бронювання місць виконується в межах транзакції PostgreSQL, спочатку створюється запис у таблиці, після чого додаються відповідні записи. Якщо на будь-якому з етапів виникає помилка, транзакція не виконується, що робить неможливим виконати подвійний продаж одного місця. Прапорці `is_active` у таблицях фільмів та сеансів дозволяють виконувати м'яке видалення записів без порушення цілісності пов'язаних бронювань.

Автоматичне обслуговування бази даних реалізовано через планувальник завдань `node-cront`, який виконує запити для деактивації минулих сеансів та фільмів із завершеним терміном прокату із автоматичним скасуванням пов'язаних бронювань.

2.3. Проєктування та реалізація алгоритмів роботи системи

Вебсистема є зручною у використанні та містить основний функціонал, необхідний для інтуїтивної й комфортної взаємодії користувачів із системою. Алгоритм роботи вебдодатка представлено на рисунку 2.3, де детально описано та пояснено кожен етап його функціонування.

Точкою входу є відкриття системи користувачем, після чого система одразу виконує перевірку стану авторизації поточного сеансу. Якщо активна сесія існує то користувач одразу отримує доступ до повного функціоналу відповідно до своєї ролі без повторного введення даних. Після переходу на сайт користувач може скористатись обмеженими можливостями, такими як перегляд афіш та адрес кінотеатрів. При спробі зробити бронювання система виконає перевірку чи користувач авторизувався, у випадку якщо не авторизований то при

		Фріз О.І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмотря О.Г.				30
Змн.	Арк.	№ докум.	Підпис	Дата		

спробі забронювати місце користувача буде перенаправлено на сторінку з формою входу або реєстрації. Якщо ж авторизовано то відбувається перевірка на роль адміністратора.

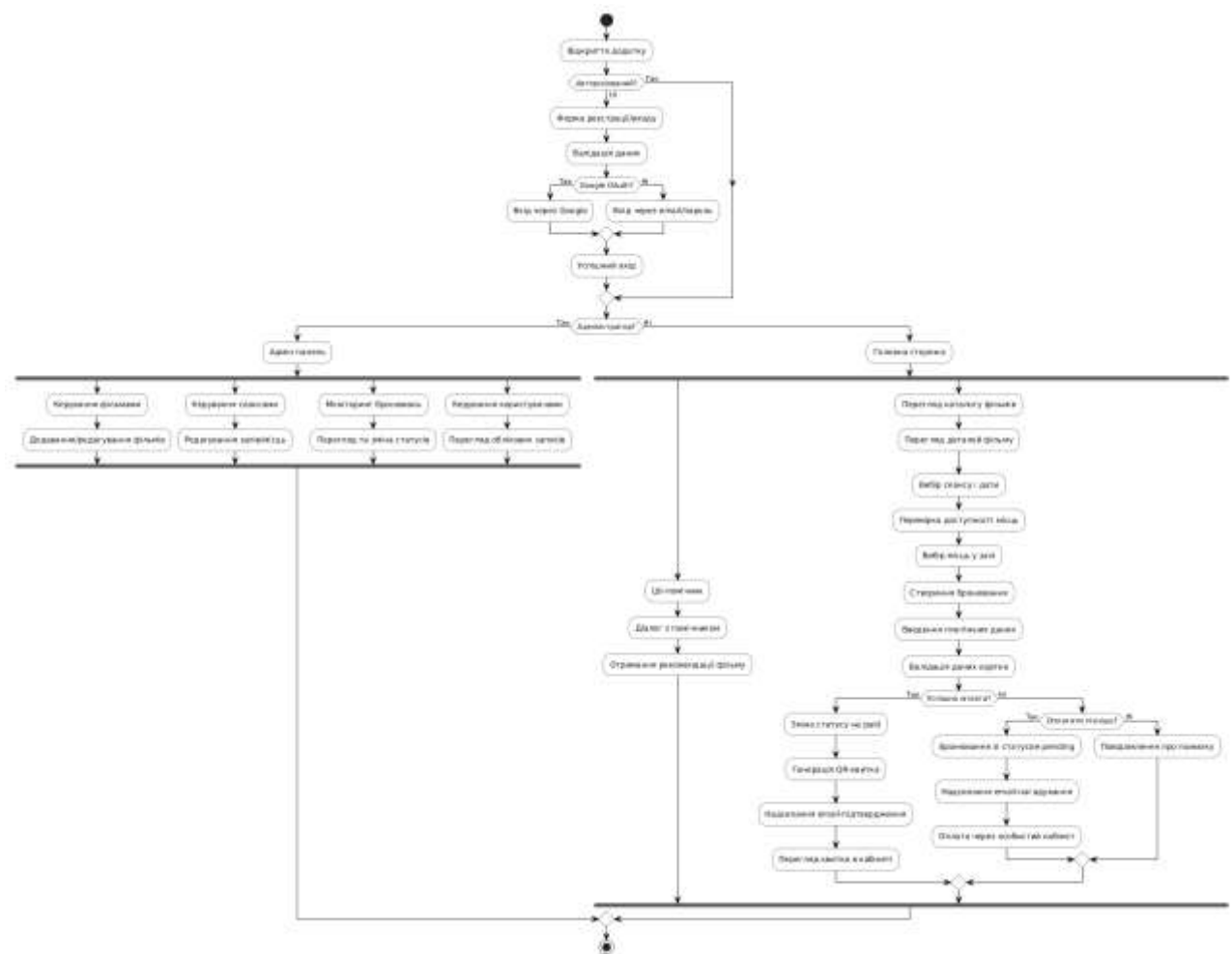


Рис. 2.3. Діаграма активності

На етапі аутентифікації користувач проходить валідацію введених даних, вебсистема підтримує два способи входу – через обліковий запис Google за протоколом OAuth та через електронну пошту і пароль. Алгоритм аутентифікації через OAuth працює наступним чином, система перенаправляє користувача на сторінку авторизації Google, після підтвердження доступу Google повертає код авторизації, а сервер обмінює його на токен та отримує дані користувача для виконання входу. При використанні Google OAuth система виконує перевірку наявності облікового запису з відповідною електронною поштою у базі даних. Якщо запис існує то виконується вхід, якщо ні – автоматично створюється новий

обліковий запис з поштою яку було обрано при авторизації. У разі виконання входу з використання електронної пошти та паролю, система звіряє введені дані з хешованим паролем у базі даних. У разі успішної авторизації формується JWT токен та зберігається для подальшої ідентифікації користувача.

Паролі зберігаються у хешованому вигляді що унеможливорює отримання оригінального пароля навіть у разі витоку даних із бази. Якщо авторизація виконана невдало, неправильний пароль або неіснуючий обліковий запис, користувач отримує відповідне повідомлення про помилку та може повторити спробу або скористатись функцією скидання паролю через електронну пошту.

Після виконання успішного входу система визначає роль користувача, передбачено дві ролі – адміністратор та звичайний користувач. У разі виконання входу від адміністратора відбувається перехід на адмін панель, а користувач на головну сторінку вебсистеми.

У разі входу адміністратора відбувається перехід на адміністративну панель. Вона включає шість основних модулів керування. Перший, керування фільмами, який дозволяє додавати нові фільми із завантаженням постерів та редагувати наявні. При завантаженні постера реалізовано алгоритм обробки зображень, файл приймається сервером через бібліотеку `multer` після чого автоматично конвертується у формат `WebP`, що зменшує розмір файлу та забезпечує швидше завантаження сторінки. Сконвертований файл завантажується на хмарне сховище `Amazon S3`, а отримане посилання на зображення зберігається у базі даних і прив'язується до відповідного фільму. Другий, керування сеансами, дозволяє формувати розклад показів та призначати зали. Алгоритм створення сеансу передбачає вибір фільму, залу, дати часу початку та ціна квитка. Після підтвердження система автоматично копіює схему місць для даного сеансу. Третій, моніторинг бронювань, де адміністратор може переглядати усі бронювання системи, змінювати їх статуси та створювати бронювання від імені користувача. Четвертий, керування користувачами, що надає можливість переглядати облікові записи та керувати ними. П'ятий керування залами, де є можливість відредагувати схему залу, видалити або

		Фріз О.І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмоторя О.Г.				32
Змн.	Арк.	№ докум.	Підпис	Дата		

додавши як окремо місця так і ряди. Також виконати блокування місця, натиснувши на відповідне місце чорним кольором тим самим заблоковується місце і воно стає неможливе до бронювання. Останній шостий модуль включає керування кінотеатрами, де відбувається перегляд адрес кінотеатрів а також зміна їх координат на мапі. На головній сторінці розміщено дашборд із базовою статистикою.

Якщо ж вхід виконав звичайний користувач, він переходить на головну сторінку сайту, де має можливість переглянути фільми які зараз в прокаті та обрати сеанс до перегляду, забронювати місця обравши на інтерактивній мапі, виконати оплату та отримати квиток в особистому кабінеті, в додаток користувач буде отримувати сповіщення на пошту при різних умовах, підтвердження бронювання, підтвердження оплати, скасування бронювання. Головна сторінка відображає п'ять фільмів з найвищим рейтингом який виствляють авторизовані в системі користувачі як рекомендовані. На сторінці каталогу фільмів доступна фільтрація за жанром, рейтингом, тривалістю, датою виходу. Також має можливість перейти на сторінку перегляду адрес на яких знаходяться кінотеатри та переглянути на карті міста відповідні позначки локацій. При натисканні на позначку локації кінотеатру на карті відображається його назва та адреса а також можливість відкрити локацію в Google Maps за допомогою координат, які пркдставлені у системі.

Алгоритм роботи вебсистеми також передбачає окремий суенарій для ІІІ-помічника. Користувач може у будь-який момент звернутись до помічника, який веде діалог та формує персональну рекомендацію фільму. Роботу ІІІ-помічника реалізовано наступним чином, при кожному зверненні до моделі через Groq API динамічно передається актуальний список фільмів що наразі перебувають у прокаті. Помічним послідовно задає питання про настрій користувача, юажаний жанр та компанію для перегляду, аналізує надані відповіді та формує персоналізовану рекомендацію виключно з доступних фільмів у системі. Після отримання рекомендацій користувач може одразу перейти до вибору сеансу на рекомендований фільм та придбати на нього квиток.

		Фріз О.І.			ІІЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
		Чижемотря О.Г.				33
Змн.	Арк.	№ докум.	Підпис	Дата		

Невід’ємною частиною алгоритму роботи системи є автоматичний планувальник завдань реалізований на базі бібліотеки node-cron. Планувальник запускається кожні п’ять хвилин та виконує перевірку стану усіх сеансів і фільмів у базі даних. Він деактивує сеанси з минулою датою початку та фільми із завершеним терміном прокату, після чого автоматично скасовує всі пов’язані активні бронювання та надсилає сповіщення на електронну пошту відповідним користувачам. Це забезпечує актуальність даних у системі в автоматичному режимі.

2.4. Реалізація функціоналу системи бронювання квитків

Розглянемо основні етапи реалізації функціоналу вебсистеми керування сеансами та онлайн-бронюваннями місць у кінотеатрі.

Для повноцінного користування системою користувач повинен спочатку зареєструвати обліковий запис або увійти через наявний акаунт Google [15]. Реєстрація передбачає введення імені, електронної пошти та пароля, який зберігається у базі даних у хешованому вигляді. Після аутентифікації користувач отримує доступ до особистого кабінету, де може редагувати персональні дані, змінювати фото профілю та пароль. Також реалізовано функцію скидання паролю через електронну пошту, система надсилає лист для відновлення доступу до акаунту. При виконанні входу користувач має й інші переваги, такі як встановлювати рейтинг фільмам та виконувати оплату квитків, доступні функції як перегляд детальної інформації про фільм, адреси кінотеатрів та запитати пораду у ШІ агента.

Перейдемо до розгляду вибору фільму та сеансу, де на сторінці фільму користувач може переглянути детальну інформацію про нього, виставити оцінку та обрати сеанс. Нижче відображається секція вибору сеансу з фільтрацією по назві кінотеатру. Розклад досупний на сім днів вперед з робивкою за датою і часом. Система автоматично приховує сеанси, що вже розпочалися або до початку яких залишилось менше двох хвилин, що унеможливорює бронювання на сеанс що вже почався. Якщо усі місця на сеанс розпродано, він відображається з відповідною позначкою і недоступний до вибору.

		Фріз О.І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмотря О.Г.				34
Змн.	Арк.	№ докум.	Підпис	Дата		

Наступне це III-помічник для вибору фільму, для допомоги користувачам у виборі фільму реалізовано інтелектуального чат-помічника на базі мовної моделі через API Groq [8]. Помічник веде діалог з користувачем, задаючи питання про настрій, з ким він буде перегляти фільм, бажаний жанр, після чого формує персональну рекомендацію виключно з тих фільмів, що наразі є в прокаті. Список фільмів передається до моделі динамічно при кожному запиті, що гарантує актуальність рекомендацій.

Після вибору сеансу користувач переходить на інтерактивну схему залу, де відображаються всі місця з їх статусами – вільне, зайняте чи заблоковане адміністратором. Заблоковані та зайняті місця неможливо забронювати. При натисканні на вільне місце воно позначається червоним кольором і стає недоступним для інших користувачів, оновлення відбувається в реальному часі розсилаючи оновлення схеми залу всім користувачам які знаходяться на сторінці. Після позначення бажаних місць відображається загальна вартість та кнопка переходу до оформлення бронювання та оплати квитків.

При підтвердженні вибору місць є два варіанти оплати. При виборі оплати зараз, у відображену форму потрібно ввести номер банківської картки, ім'я власника картки, термін дії та CVV-код. Після валідації даних система звертається до платіжного шлюзу. У разі успішної оплати статус бронювання змінюється на оплачено і користувач отримує підтвердження на електронну пошту. У разі невдалої оплати буде надіслано повідомлення про помилку і може спробувати ще раз. При виборі оплати пізніше, система створює запис бронювання зі статусом pending та надсилає користувачу email-сповіщення з деталями. Таке бронювання зберігається в особистому кабінеті де є можливість виконати оплату.

Після підтвердження бронювання в особистому кабінеті користувачу стає доступним електронний квиток. Він містить інформацію про фільм, кінотеатр, зал, дату та час сеансу, обрані місця та суму оплати. До квитка також автоматично генерується QR-код [17], який містить усі дані бронювання у форматі JSON і призначений для пред'явлення на касі кінотеатру.

		Фріз О.І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмотря О.Г.				35
Змн.	Арк.	№ докум.	Підпис	Дата		

В особистому кабінеті користувач може переглядати всю історію своїх бронювань з відображенням їх статусів. Активні бронювання можна скасувати, після чого буде надіслано відповідне сповіщення на електронну пошту. Бронювання на сеанси, що вже відбулись, автоматично позначаються як недійсні. Скасовані та просрочені бронювання можна видалити зі списку квитків.

У системі реалізовано планувальник завдань на базі бібліотеки `node-cron`, який кожні п'ять хвилин перевіряє стан сеансів та фільмів. Сеанси з минулою датою початку автоматично стають неактивні, а всі пов'язані бронювання скасовуються та надсилається відповідне повідомлення на електронну пошту користувача. Фільми у яких минув термін прокату також будуть скасовані разом з усіма майбутніми сеансами. Це забезпечує актуальність даних у системі без ручного втручання адміністратора.

Говорячи про панель адміністратора це окрема захищена панель керування, недоступна для звичайного користувача. Вона включає в собі управління фільмами із завантаженням постерів на хмарне сховище Amazon S3 де зображення автоматично генеруються у формат WebP для зменшення розміру файлу та швидшого завантаження на сторінці.

Модуль налаштування розкладу сеансів дозволяє формувати графік показів та призначати фільмам конкретні зали, дату, час та ціну квитка. При створенні сеансу система автоматично копіює схему місць з обраного залу. Адміністратор може редагувати параметри сеансу або скасовувати його, що автоматично тягне за собою скасування усіх пов'язаних бронювань.

Модуль керування залами та конітеатрами надає можливість управління схемою місць у залах з можливістю видалення, додавання та блокування крісел. Для кінотеатрів можна зазначати їх назви, адресу в системі координат та геолокації для відображення на інтерактивній мапі міста.

Модуль бронювань забезпечує перегляд та керування усіма замовленнями системи. Окремою функцією є можливість створення бронювання адміністратором від імені клієнта, якщо користувач зареєстрований у системі –

		Фріз О.І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмотря О.Г.				36
Змн.	Арк.	№ докум.	Підпис	Дата		

квиток з'явиться в його особистому кабінеті, якщо ні – квиток буде надіслано на вказану пошту при створенні бронювання. Це зручно для обслуговування клієнтів які звертаються до кінотеатру через касу або телефонним дзвінком на гарячу лінію.

На головній сторінці панелі розміщено дашборд із базовою ствтичтикою системи, де відображена загальна кількість бронювань, активних сеансів, зареєстрованих користувачів та фільмів у прокаті, що дає адміністратору швидкий загальний огляд поточного стану системи.

Висновки до другого розділу

У другому розділі було виконано проєктування вебсистеми керування сеансами та онлайн-бронюваннями місць у кінотеатрі. На етапі визначення варіантів використання було визначено два основних актори системи, це користувач та адміністратор і описано їх взаємодії із системою.

На основі функціональних вимог побудовано діаграму класів з визначеними атрибутами, методами та зв'язками між ними, що охоплюють повний цикл роботи системи від авторизації до генерації електронного квитка.

Спроектовано реляційну базу даних PostgreSQL, встановлені зовнішні ключі та механізм транзакцій, що гарантують цілісність даних і не дають можливості виконати подвійне бронювання одного місця. Для взаємодії з базою даних обрано ORM-бібліотеку TypeORM. Додатково реалізовано планувальник завдань, який автоматично деактивує неактивні сеанси та скасовує пов'язані бронювання із надсиланням сповіщень на електронну пошту користувача.

Отримані результати проєктування формують основу для подальшої програмної реалізації.

		Фріз О.І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмотря О.Г.				37
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З СИСТЕМОЮ
БРОНЮВАННЯ КВИТКІВ

3.1. Порядок встановлення та налаштування параметрів додатку

Для встановлення і налаштування додатку необхідно виконати розгортання основних компонентів системи і забезпечення взаємодії між ними. Для візуального відображення архітектури було розроблено діаграму розгортання (рис. 3.1).

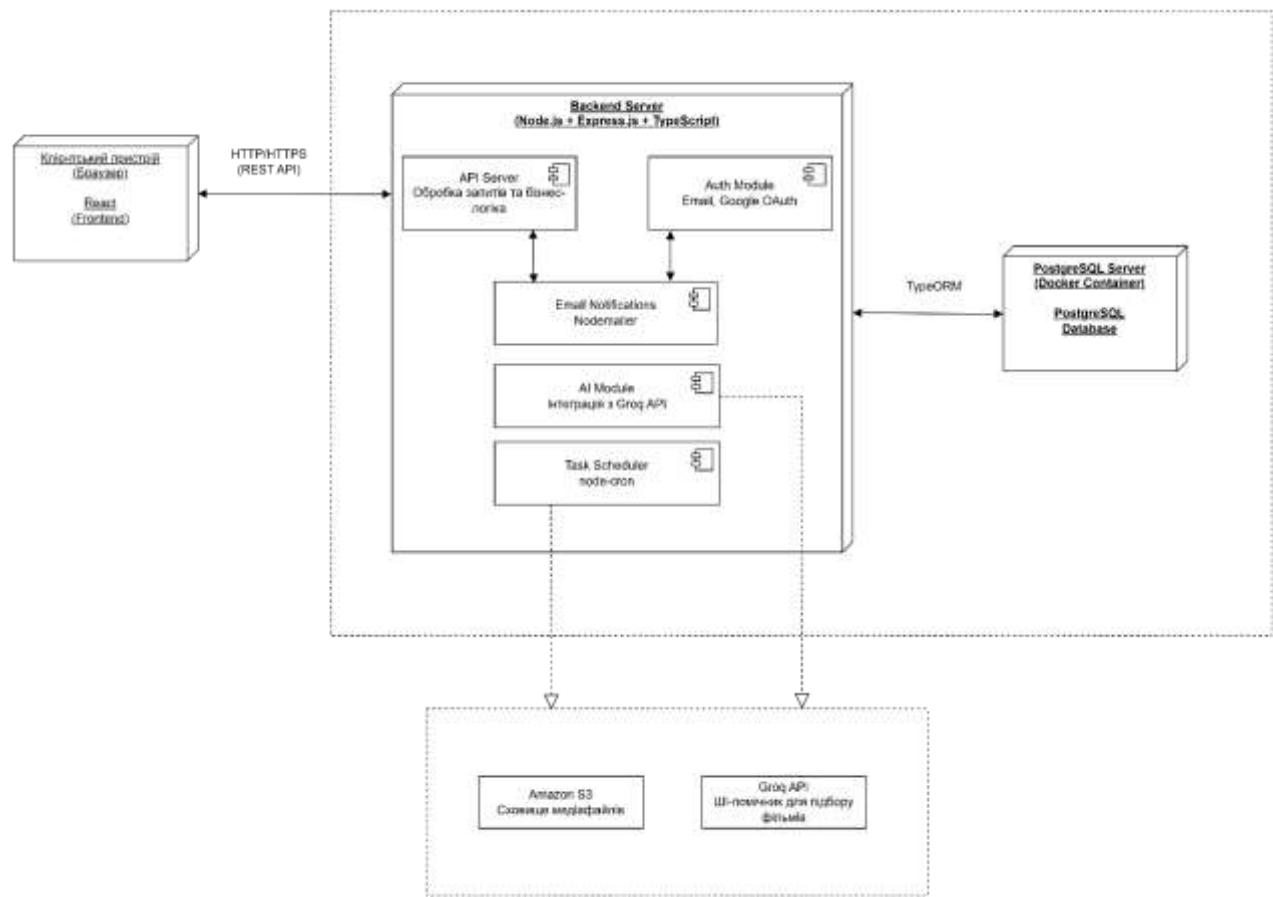


Рис. 3.1. Діаграма розгортання додатку для бронювання

Клієнтська частина являє собою React-застосунок, що виконується у браузері користувача. Вона відповідає за відображення інтерфейсу, взаємодію з користувачем та обмін даними із серверною частиною через HTTP-запити.

Перед початком встановлення необхідно переконатися, що на машині розробника встановлено Node.js версію не нижче 18.x та менеджер пакетів npm. Після клонування репозиторію виконується команда встановлення залежностей,

для локальної розробки застосунк запускається у режимі гарячого перезавантаження, а для продакшн-збірки виконується `npm run build`.

Серверна частина побудована на основі Node.js та Express.js з використанням TypeScript і розгортається у Docker-контейнері. Вона включає API-сервер, який відповідає за обробку запитів від клієнтської частини та реалізацію бізнес-логіки. Також модуль аутентифікації через Email та Google OAuth, модуль сповіщень на електронну пошту на базі Nodemailer [11], ШІ-модуль для взаємодії з Groq API та планувальник завдань на базі node-cron для автоматичної деактивації минулих сеансів та фільмів. Для встановлення серверної частини у режимі розробки необхідно у директорії backend виконати `npm install` для завантаження залежностей, після чого запустити сервер командою `npm run dev`. Конфігурація серверної частини здійснюється через файл `.env`, де визначаються параметри. API-сервер відповідає за маршрутизацію запитів від клієнтської частини та реалізацію бізнес-логіки. Express.js виступає основним фреймворком, що обробляє вхідні HTTP-запити. Структура маршрутів організована за принципом RESTful API, де кожен ресурс має відповідний набір ендпоінтів.

Планувальник завдань на базі node-cron виконує регулярні фонові завдання за розкладом, визначеним у форматі cron-виразів. Основні завдання це автоматична деактивація минулих сеансів та архівування застарілих фільмів.

PostgreSQL Server – окремий контейнер бази даних, взаємодія з яким здійснюється через ORM-бібліотеку TypeORM з серверного боку. База даних зберігає всю потрібну інформацію яка використовується у вебсистемі.

Сповіщення надсилаються на базі Nodemailer який дозволяє надсилати листи про підтвердження реєстрації, нагадування про бронювання, сповіщення про скасування. Шаблони листів зберігаються у вигляді HTML-файлів з підстановними змінними, які заповнюються динамічно перед відправкою.

До зовнішніх сервісів належать Amazon S3 для зберігання медіафайлів, таких як постери, аватари користувачів. Для підключення необхідно у консолі AWS створити S3-bucket, налаштувати публічний доступ на читання

		Фріз О.І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмотря О.Г.				39
Змн.	Арк.	№ докум.	Підпис	Дата		

медіафайлів, а також створити ІАМ-користувача з правами на запис до відповідного бакета. А також Groq API для роботи ШІ-помічника з підбору фільмів. Для підключення потрібно зареєструватись на платформі Groq, тримати API-ключ та додати його до змінної середовища. Усі запити до Groq API виконуються серверною частиною, що не дає можливості аитоку ключа на клієнтську сторону.

Оскільки усі компоненти розгортаються у Docker-контейнерах за допомогою Docker Compose [6], це забезпечує ізольоване середовище виконання, яке спрощує налаштування та гарантує однакову поведінку системи у різних середовищах. Файл `docker-compose.yml` знаходиться у кореневій директорії проєкту і описує три сервіси, бекенд частину, базу даних та за необхідності фронтенд. Для розгортання використовується єдина команда `docker-compose up --build`.

3.2. Структура інтерфейсу

Інтерфейс вебсистеми спроектовано з пріорітетом на зручність користувача, забезпечуючи виконання усіх необхідних операцій, від перегляду афіші до проведення оплати. Сторінки системи логічно пов'язані між собою та дозволяють користувачеві послідовно переходити від одного етапу до наступного.

На головній сторінці наявний перелік рекомандованих фільмів, він визначається за найвищою оцінкою користувачів та відображає 5 фільмів з найбільшим рейтингом (рис. 3.2).

		Фріз О.І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмотря О.Г.				40
Змн.	Арк.	№ докум.	Підпис	Дата		

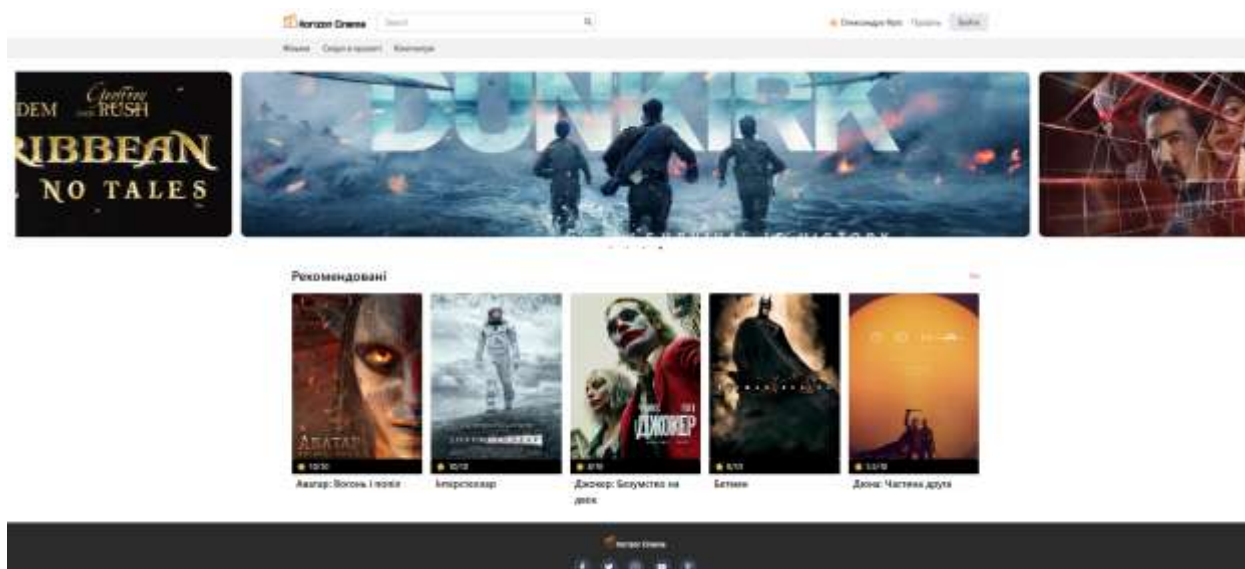


Рис. 3.2. Головна сторінка

Користувач має можливість перейти до сторінки «Фільми» де потрібно буде обрати фільм до перегляду, також наявна фільтрація яку можна використовувати в комбінації по жанрах, рейтингу, тривалістю, датою (рис. 3.3).

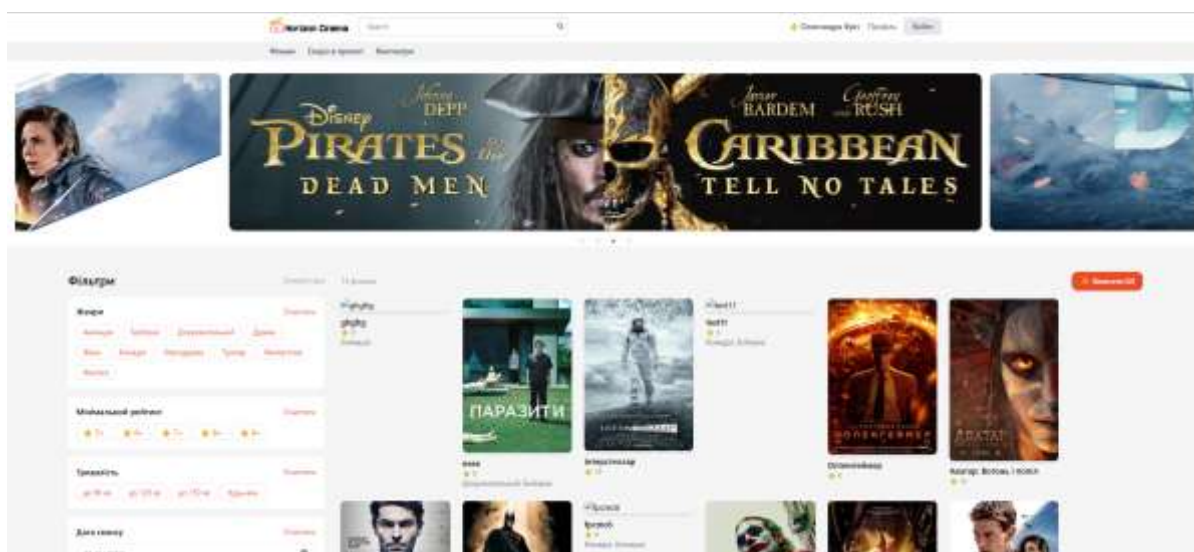


Рис. 3.3. Сторінка фільмів

Також коли глядач не може визначитись який фільм переглянути він може запитати поради в ШІ агента та зробити свій вибір натиснувши на конпку «Запитати ШІ» (рис. 3.4).

		Фріз О.І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмотря О.Г.				41
Змн.	Арк.	№ докум.	Підпис	Дата		

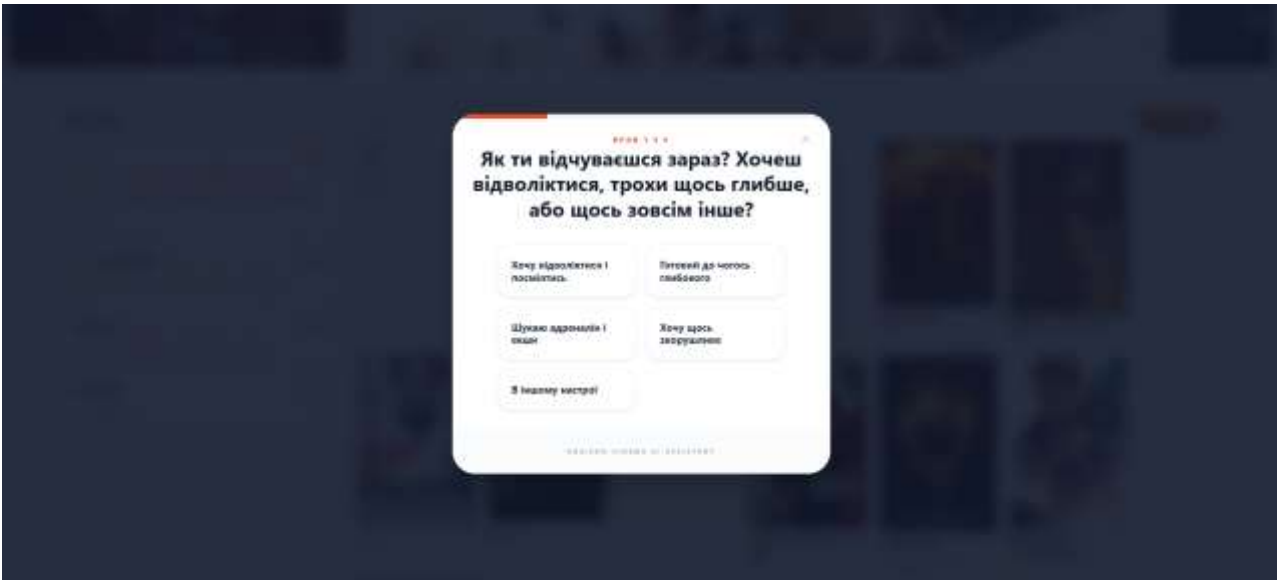


Рис. 3.4. Вікно ІІІ

Після того як користувач обере фільм він потрапляє на сторінку вибору сеансів та може переглянути детальний опис до фільму, жанр, дату виходу, до якого числа буде в прокаті. Якщо користувач увійшов у систему він може поставити оцінку фільму та сформувати загальний рейтинг. Також нижче під деталями є можливість обрати сеанс, доступно до перегляду на 7 днів вперед. Користувачу слід визначитись із датою та часом, а також підібрати цінову категорію (рис. 3.5).



Рис. 3.5. Сторінка вибору сеансу та опису фільму

Коли користувач обрав сеанс він переходить на сторінку де відображена інтерактивна мапа розподілу місць у залі. На ній відображено екран, ряди та місця. При натисканні на певне місце воно стає червоним і тоді інший користувач не зможе оплатити це місце поки поточний користувач не вийде з цієї сторінки. Також місця можуть мати інші кольори, такі як сіре – місце зайняте, чорний – місце заблоковане адміністрацією. Після вибору місця з'являється кнопка для переходу до оплати квитка (рис. 3.6).

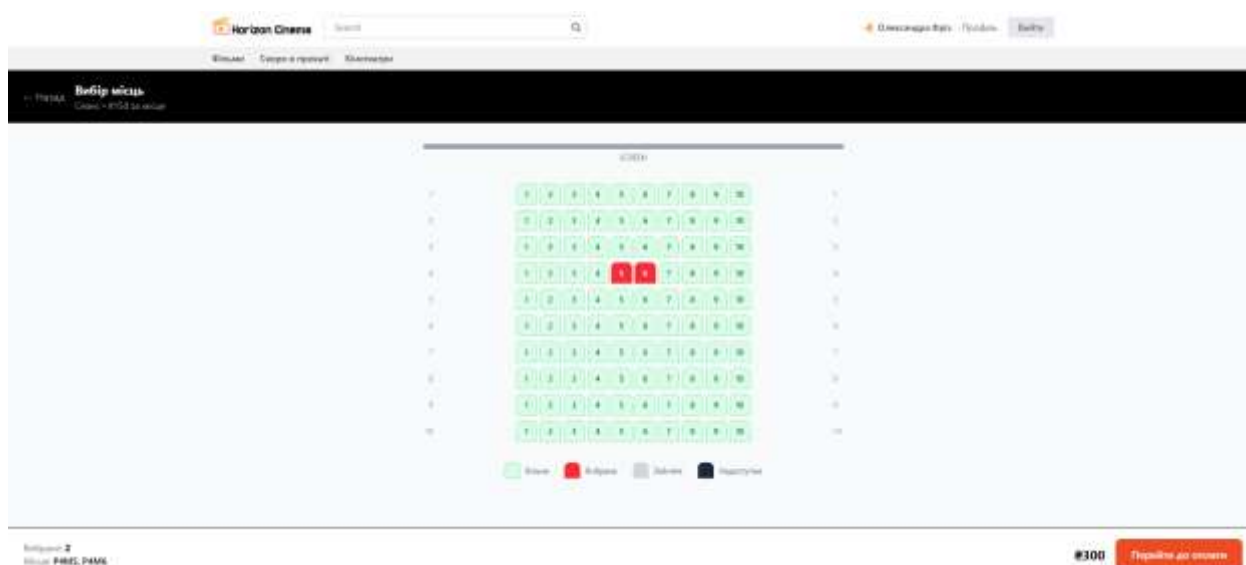


Рис. 3.6. Сторінка вибору місць та перехід до оплати

Коли користувач обрав місця та перейде до оплати він може обрати або оплатити зараз ввівши дані картки або оплатити пізніше, тоді в особистому кабінеті з'явиться квиток і на ньому буде підпис в залежності від методу оплати, зі статусом оплачено або очікує оплати. Після проведення платежу користувачу прийде сповіщення про бронювання на електронну пошту (рис. 3.7).

		Фріз О.І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмотря О.Г.				43
Змн.	Арк.	№ докум.	Підпис	Дата		

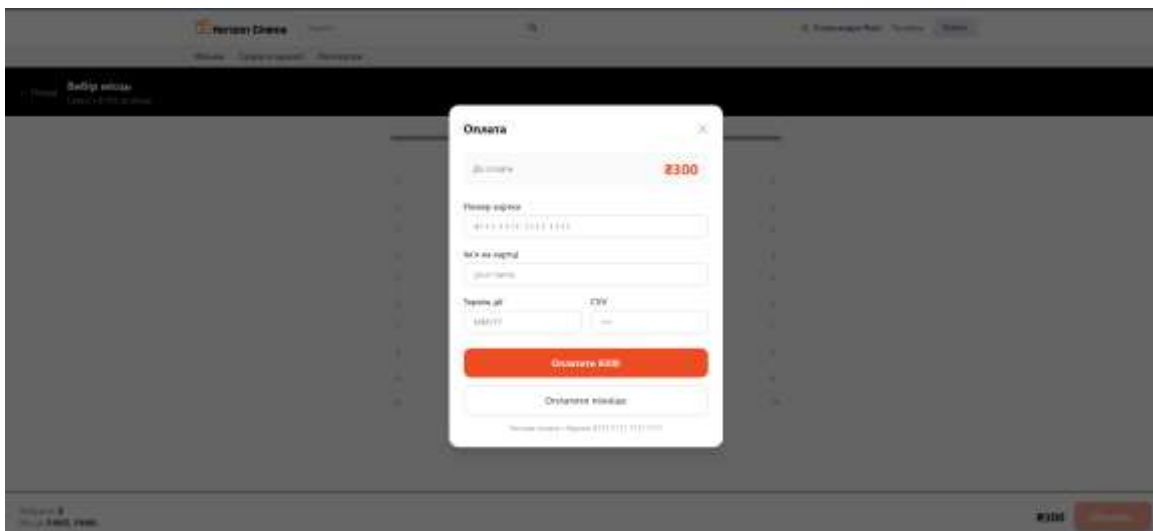


Рис. 3.7. Вікно проведення платежу

У випадку коли користувач обрав оплату одразу, він перейде на сторінку успішної оплати і далі є можливість переглянути бронювання або перейти на сторінку фільмів та обрати ще інші сеанси до перегляду (рис. 3.8).

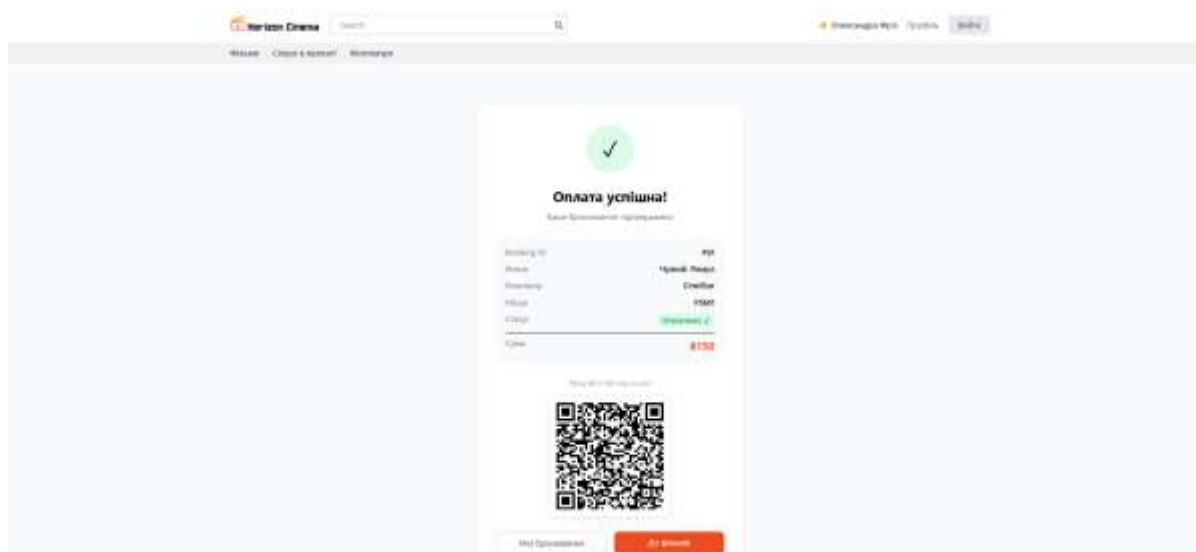


Рис. 3.8. Сторінка успішної оплати

У випадку коли було обрано оплату пізніше то користувача перенаправляє на сторінку особистого кабінету де квиток буде у статусі очікує оплати (рис. 3.9).

		Фріз О.І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмотря О.Г.				44
Змн.	Арк.	№ докум.	Підпис	Дата		

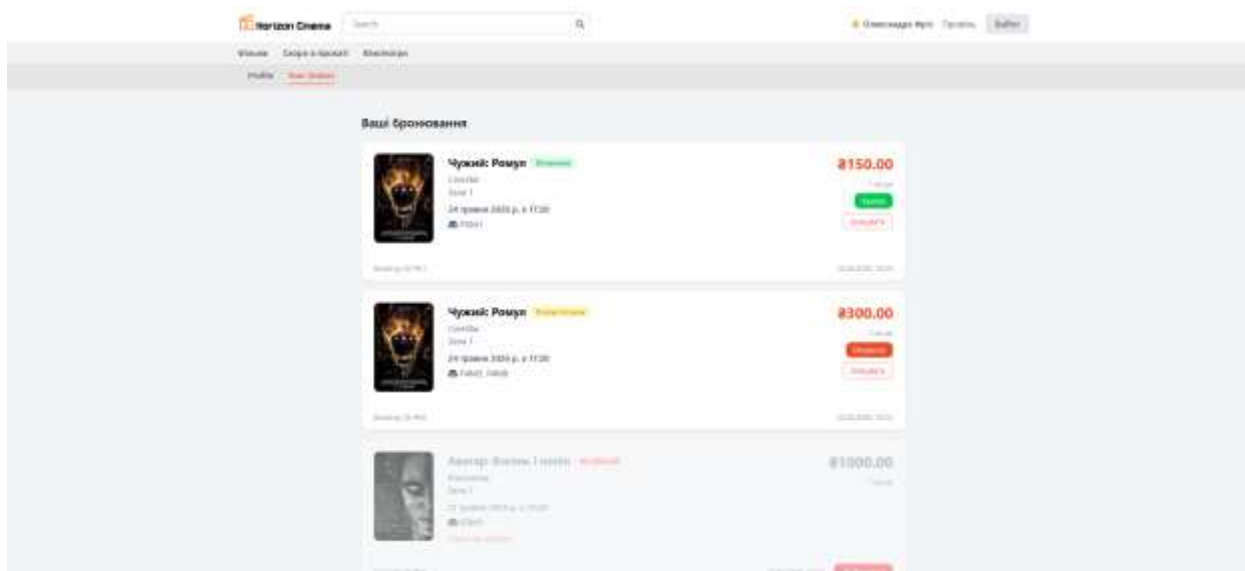


Рис. 3.9. Квитки в особистому кабінеті

Також користувач може перейти на сторінку «Скоро в прокаті», на ній можна переглянути фільми, які будуть в майбутньому показані в кінотеатрі, до кожного фільму зазначена дата виходу, коли настає ця дата то афіша фільму переходить з «Скоро в прокаті» на «Фільми» і користувачі зможуть обрати на нього сеанс (рис. 3.10).



Рис. 3.10. Сторінка фільмів які скоро будуть у прокаті

З можливостей ще можна переглянути адреси кінотеатрів на сторінці «Кінотеатри», там представлені адреси та назви кінотеатрів і також карта міста з позначками геолокацій кожного з них. Це дозволяє користувачу швидко

орієнтуватись у місті, визначити найближчий до нього кінотеатр та спланувати маршрут.(рис. 3.11).

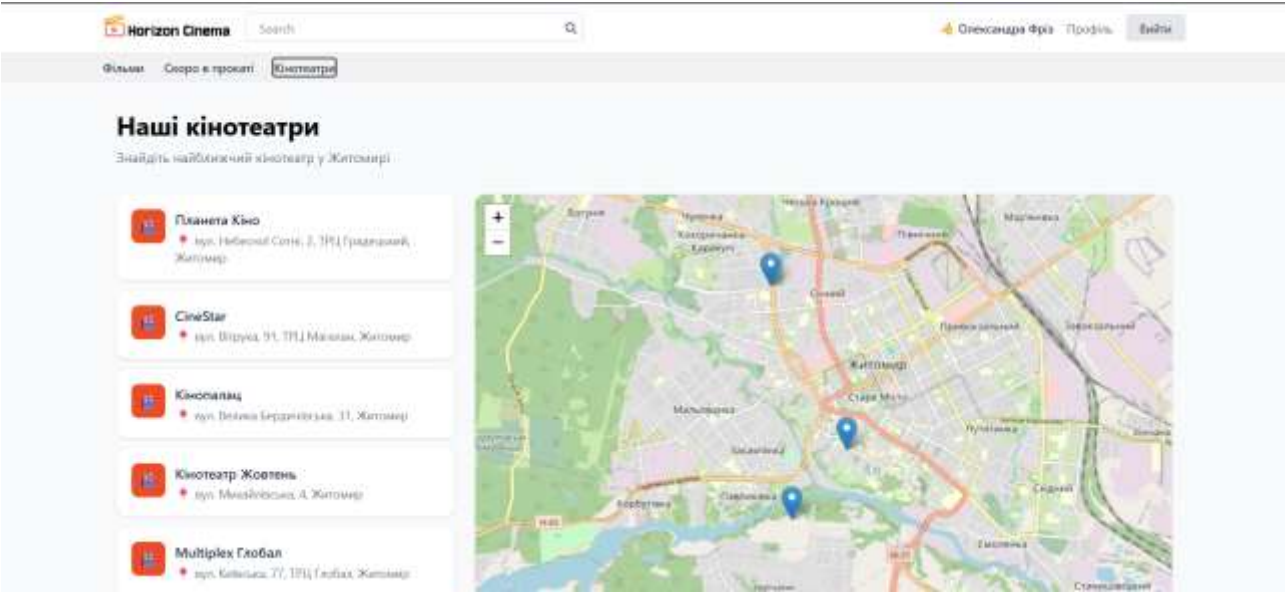


Рис. 3.11. Кінотеатри міста

Якщо користувач зареєстрований та виконав вхід в систему він може переглянути особистий кабінет в якому може на вкладці «Профіль» відредагувати свої особисті дані такі як фото профілю, електронну пошту, ім'я та пароль(рис. 3.12).

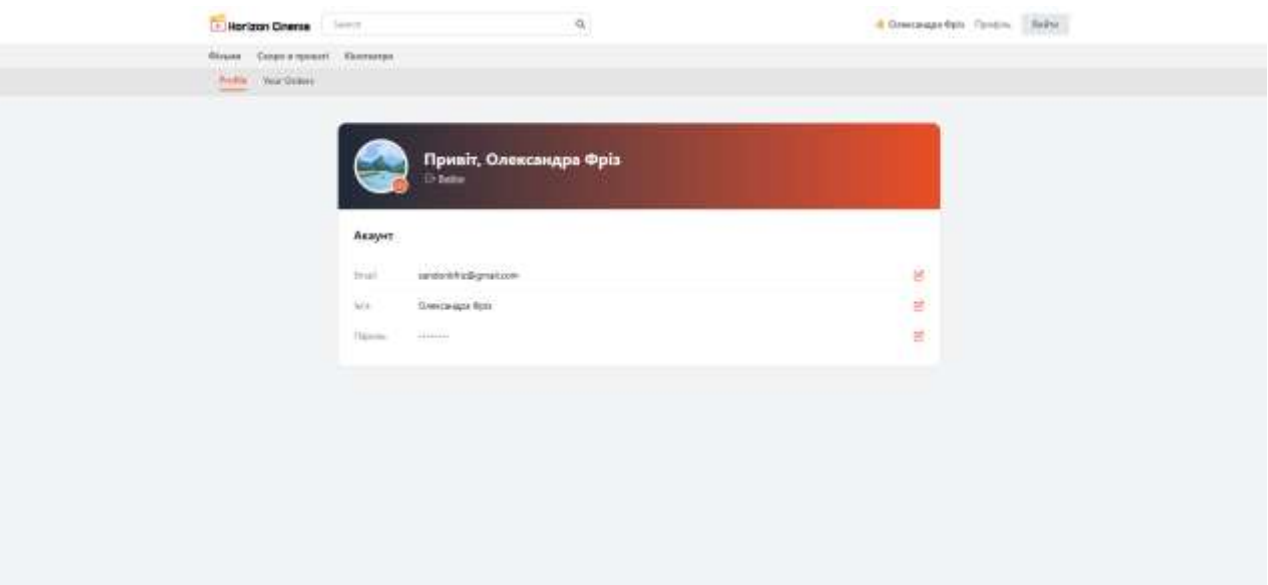


Рис. 3.12. Профіль користувача

Також якщо перейти на вкладку «Мої бронювання» можна переглянути бронювання а саме, ті що в очікуванні оплати, оплачені, скасовані самостійно, скасовані системою (рис. 3.13).

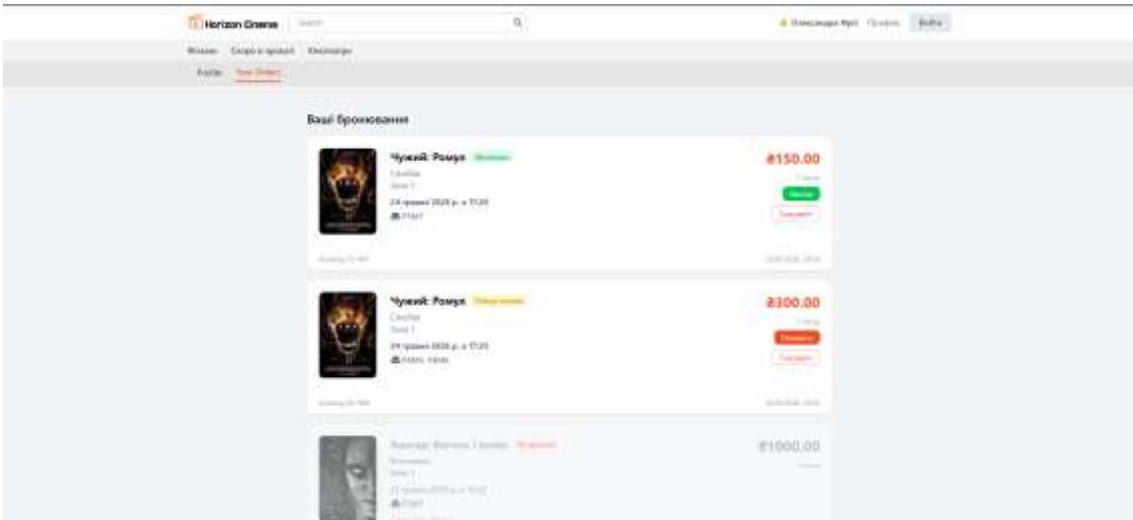


Рис. 3.13. Сторінка бронювань користувача

Щоб переглянути квиток, слід натиснути на кнопку «Квиток» та побачити QR-код на екрані (рис. 3.14).

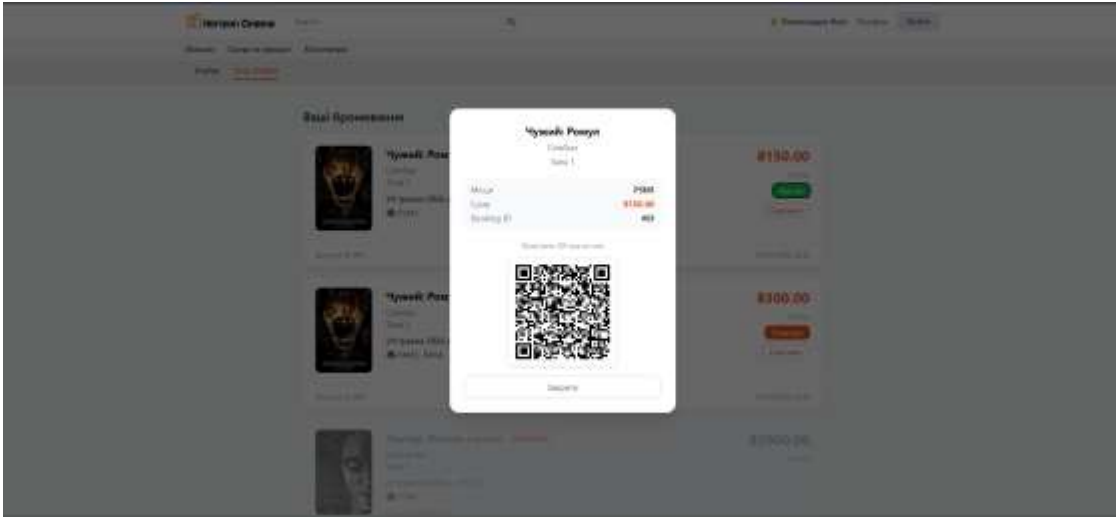


Рис. 3.14. Квиток

Якщо користувач не увійшов у систему він не зможе встановити оцінку на фільм, створити бронювання та переглянути особистий кабінет. Щоб це зробити йому слід натиснути кнопку «Увійти», його перенаправить на сторінку з формою входу куди потрібно внести особисті дані або скористатись входом за допомогою Google сервісу (рис 3.15).

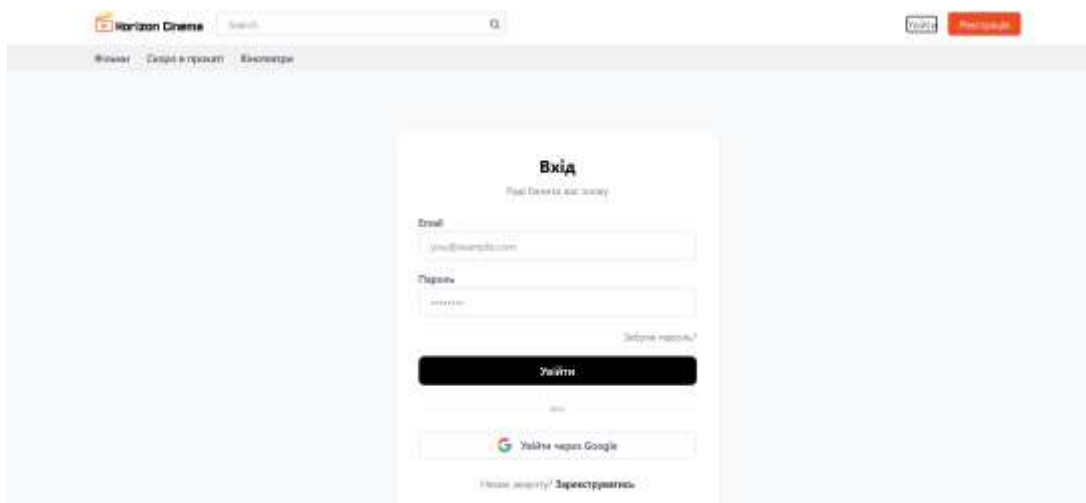


Рис. 3.15. Сторінка входу

У випадку якщо користувач не має створеного акаунту він може виконати реєстрацію натиснувши на кнопку «Зареєструватись» на формі входу або в хедері сторінки на кнопку «Реєстрація». Перейшовши на сторінку реєстрації потрібно ввести свої персональні дані та реєстрація буде виконана (рис 3.16).

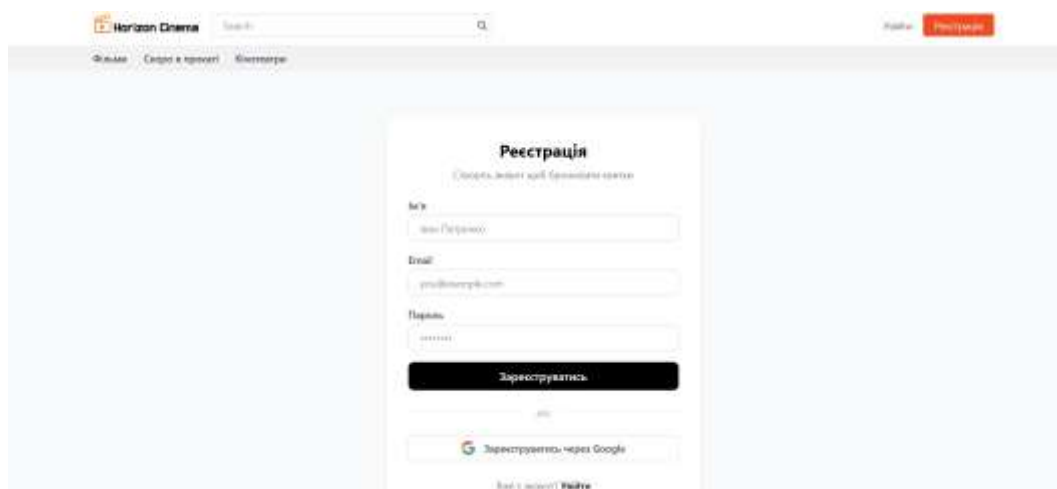


Рис. 3.16. Сторінка реєстрації

Система також має адміністративну панель на якій є можливість управляти системою, а саме керувати бронюваннями, фільмами, сеансами, місцями, користувачами. Щоб увійти до цієї панелі слід ввести дані адміністратора та він буде мати певні права, відмінні від звичайного користувача (рис 3.17).

		Фріз О.І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмотря О.Г.				48
Змн.	Арк.	№ докум.	Підпис	Дата		

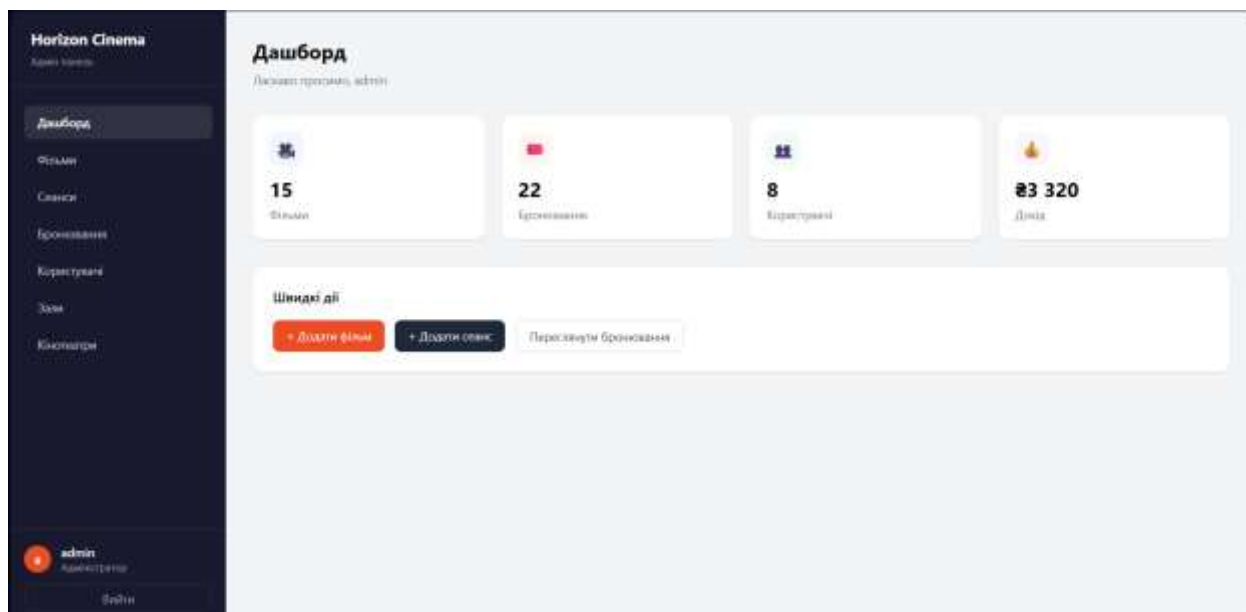


Рис. 3.17. Головна сторінка адмін панелі

3.3. Тестування роботи програмного продукту

Метою тестування є перевірка відповідності розробленої вебсистеми керування сеансами та онлайн-бронюваннями місць у кінотеатрі встановленим функціональним вимогам, а також виявлення можливих дефектів до введення системи в експлуатацію. Тестування проводилось методом функціонального ручного тестування заздалегідь складеними тест-кейсами, що охоплюють усі ключові сценарії використання системи. Для кожного тест-кейсу визначено передумову, послідовність кроків виконання, очікуваний результат та зафіксовано фактичний результат перевірки.

Таблиця 3.1

Перелік тест-кейсів перевірки функціональності системи

№	Назва кейсу	Передумова	Кроки	Очікуваний результат	Статус
ТС-01	Реєстрація нового користувача	Користувач не зареєстрований	1.Відкрити сторінку реєстрації. 2.Заповнити поля форми. 3.Натиснути «Зареєструватись».	Обліковий запис створено	Схвалено

Продовження табл. 3.1

ТС-02	Вхід через Google OAuth	Наявний акаунт в Google	1.Відкрити сторінку входу. 2.Натиснути «Увійти через Google» 3.Обрати акаунт. 4.Підтвердити доступ.	Авторизація успішна	Схвалено
ТС-03	Перегляд інформації про фільм	Користувач авторизований	1.Перейти на сторінку обраного фільму. 2.Переглянути детальний опис.	Інформація відображається коректно	Схвалено
ТС-04	Фільтрація сеансів за кінотеатром	Сторінка фільму відкрита	1.Перейти на список фільмів. 2.Обрати фільм. 3.Під описом до фільму обрати кінотеатр для фільтрації.	Відображаються сеанси лише обраного кінотеатру	Схвалено
ТС-05	Вибір місця в залі	Обраний активний сеанс	1.Обрати фільм та сеанс. 2.Обрати вільне місце на схемі залу. 3.Перейти до оплати.	Місце позначається червоним кольором як обране та доступна кнопка оплати.	Схвалено
ТС-06	Бронювання з оплатою пізніше	Обрано сеанс та місця	1.Обрати сеанс та місця. 2.Натиснути на кнопку «Оплатити». 3.У формі оплати обрати спосіб «Оплатити пізніше»	Створено бронювання зі статусом «Очікує оплати».	Схвалено
ТС-07	Неможливість вибору зайнятого місця	Обраний сеанс існує, є зайняті місця	1.Обрати сеанс. 2.Перейти на схему залу. 3.Обрати зайняте місце.	Місце не обирається, відображається відповідна позначка про заборону.	Схвалено
ТС-08	Виконати оплату картою	Обрані місця в залі	1.Обрати сеанс та місця. 2.Натиснути на кнопку «Оплатити». 3.У формі оплати заповнити всі поля. 4.Натиснути на кнопку «Оплатити».	Створено бронювання зі статусом «Оплачено». Надіслано сповіщення на електронну пошту.	Схвалено

Закінчення табл. 3.1

ТС-09	Перегляд електронного квитка	Оплачене бронювання	1.Перейти до особистого кабінету. 2.Перейти на вкладку «Мої бронювання». 3.На сплаченому квитку натиснути на кнопку «Квиток».	Відображається квиток з інформацією про сеанс та QR-код.	Схвалено
ТС-10	Скасування бронювання	Бронювання активне	1.Перейти до особистого кабінету. 2.Перейти на вкладку «Мої бронювання». 3.На квитку натиснути на кнопку «Скасувати».	Статус змінюється на «Скасовано». Надіслано сповіщення про скасування на електронну пошту.	Схвалено
ТС-11	AI-помічник: рекомендація фільму	-	1.Відкрити чат помічника на сторінці фільмів. 2.Дати відповіді на питання.	Отримано рекомендацію фільму та відображено кнопки для переходу на купівлю квитку.	Схвалено
ТС-12	Автоматичне скасування сеансів	Створити сеанс з минулою датою	1.Дочекатися запуску планувальника (5 хв). 2.Перевірити статус сеансу.	Сеанс отримує статус «Неактивного», бронювання скасовується та надсилається сповіщення.	Схвалено
ТС-13	Блокування місця в залі на адмін панелі	Виконати вхід від адміністратора	1.Відкрити схему залу на вкладці «Зали». 2.Обрати варіант «Блокування» 3.Обрати місце	Місце недоступне до бронювання користувачами.	Схвалено
ТС-14	Створення бронювання з адмін панелі	Виконати вхід від адміністратора	1.Перейти на вкладку «Бронювання». 2.Натиснути на кнопку «Створити бронювання». 3.Ввести дані клієнта. 4.Підтвердити бронювання.	Бронювання створено, сповіщення надіслано на електронну пошту користувача та/або відображено квиток в кабінеті.	Схвалено

За результатами проведеного функціонального тестування було виконано 14 тест-кейсів, що охоплюють основні сценарії роботи системи. Усі тести було успішно продено, критичних та нукритичних дефектів виявлено не було.

Загальний результат тестування свідчить про готовність системи до експлуатації, усі перевірені функції працюють відповідно до визначених вимог.

Висновки до третього розділу

У даному розділі було розглянуто інтерфейс та поррядок роботи вебсистеми, а також описано процес її налаштування та тестування. Було визначено основні компоненти архітектури системи, принцип їх взаємодії та особливості розгортання за допомогою Docker, що забезпечує стабільність роботи та зручність підтримки програмного продукту.

У процесі аналізу інтерфейсу було описано основні сторінки системи та сценарії їх взаємодії з користувачем. Також було приділено увагу адміністративній панелі, яка забезпечує можливість керувати наповненням системи даними.

Після завершення роботи над розробкою вебсистеми було проведене функціональне ручне тестування, яке підтвердило коректність роботи усіх основних модулів системи. Тест-кейси були успішно виконані, а результати перевірки показали відповідність реалізованого продукту поставленим функціональним вимогам.

		Фріз О.І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмотря О.Г.				52
Змн.	Арк.	№ докум.	Підпис	Дата		

ВИСНОВКИ

Під час виконання випускної кваліфікаційної роботи бакалавра було спроектовано та реалізовано вебсистему керування сеансами та онлайн-бронюваннями місць у кінотеатрі .

На початковому етапі було вивчено предметну область та проведено аналіз існуючих аналогів. На основі виявлених переваг та недоліків існуючих рішень сформовано перелік функціональних та нефункціональних вимог до розроблюваної системи та зроблено вибір до інструментальних засобів реалізації.

На етапі проектування визначено двох основних акторів та описано сценарії їх взаємодії. Побудовано діаграму класів, спроектовану базу даних та розроблено алгоритм роботи системи. Також реалізовано автоматичний планувальник завдань для обробки службових процесів.

Розроблений інтерфейс забезпечує повний цикл взаємодії користувача із системою від перегляду афіші та вибору фільму через інтерактивну схему залу до створення бронювання та отримання квитка в особистому кабінеті. Для адміністратора реалізовано окрему панель керування з повним набором інструментів управління сеансами, залами, фільмами, користувачами та бронюваннями.

Результатом кваліфікаційної роботи є готова до впровадження та подальшого розвитку вебсистема керування сеансами та онлайн-бронюваннями місць у кінотеатрі, яка відповідає поставленим вимогам та успішно пройшла тестування без виявлених критичних дефектів. Система забезпечує зручний процес пошуку сеансу для глядачів, а також надає ефективний інструментарій для адміністрування та керування основними процесами кінотеатру.

Розроблена система може бути використана як основа для автоматизації роботи сучасних кінотеатрів та подальшого розширення функціональних можливостей.

		Фріз О.І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмотря О.Г.				53
Змн.	Арк.	№ докум.	Підпис	Дата		

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Документація React [Електронний ресурс]: <https://react.dev/learn>
2. Документація Node.js [Електронний ресурс]: <https://nodejs.org/en/docs>
3. Документація Express.js. [Електронний ресурс]: <https://expressjs.com>
4. Документація PostgreSQL [Електронний ресурс]: <https://www.postgresql.org/docs>
5. Документація TypeORM [Електронний ресурс]: <https://typeorm.io>
6. Документація Docker [Електронний ресурс]: <https://docs.docker.com>
7. Документація Amazon S3 [Електронний ресурс]: <https://docs.aws.amazon.com/s3>
8. Документація Groq API [Електронний ресурс]: <https://console.groq.com/docs>
9. Документація Tailwind CSS [Електронний ресурс]: <https://tailwindcss.com/docs>
10. Документація TypeScript [Електронний ресурс]: <https://www.typescriptlang.org/docs>
11. Документація Nodemailer [Електронний ресурс]: <https://nodemailer.com/about>
12. Документація node-cron [Електронний ресурс]: <https://github.com/node-cron/node-cron>
13. Офіційний сайт кінотеатру Multiplex [Електронний ресурс]: <https://multiplex.ua>
14. Офіційний сайт кінотеатру Planet Kino [Електронний ресурс]: <https://planetkino.ua>
15. Документація Google OAuth 2.0 [Електронний ресурс]: <https://developers.google.com/identity/protocols/oauth2>
16. Документація JWT (JSON Web Token) [Електронний ресурс]: <https://jwt.io/introduction>
17. Документація бібліотеки qrcode для Node.js [Електронний ресурс]: <https://www.npmjs.com/package/qrcode>
18. Документація Docker Compose [Електронний ресурс]: <https://docs.docker.com/compose>

19. Документація Axios [Електронний ресурс]: <https://axios-http.com/docs/intro>
20. Wieringa R. Design Methods for Reactive Systems. – San Francisco : Morgan Kaufmann, 2003.
21. Документація React Router [Електронний ресурс]: <https://reactrouter.com/en/main>
22. Фаулер М. UML. Основи : короткий довідник / М. Фаулер. – К. : Діалектика, 2019. – 192 с.
23. Соммервілл І. Інженерія програмного забезпечення. – 10-те вид. – К. : Вільямс, 2021. – 816 с.
24. Документація Vite [Електронний ресурс]: <https://vitejs.dev/guide>
25. Документація passport.js (аутентифікація) [Електронний ресурс]: <https://www.passportjs.org/docs>
26. Документація Google Maps JavaScript API [Електронний ресурс]: <https://developers.google.com/maps/documentation/javascript>
27. Документація Groq – моделі LLaMA [Електронний ресурс]: <https://console.groq.com/docs/models>
28. UML Class Diagram Tutorial [Електронний ресурс]: <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-class-diagram/>
29. PlantUML Documentation [Електронний ресурс]: <https://plantuml.com/>
30. JWT Introduction [Електронний ресурс]: <https://jwt.io/introduction>

ДОДАТКИ

		Фріз О.І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмотря О.Г.				56
Змн.	Арк.	№ докум.	Підпис	Дата		

Фрагменти вихідного коду вебсистеми

```
import { Router, Response } from "express";
import Groq from "groq-sdk";
import { authMiddleware, AuthRequest } from "../middleware/auth.middleware";

const router = Router();

router.post("/recommend", authMiddleware, async (req: AuthRequest, res: Response): Promise<void> => {
  try {
    const { messages, movies } = req.body;
    const client = new Groq({ apiKey: process.env.GROQ_API_KEY });

    const systemPrompt = `Ти – кінокритик і персональний гід кінотеатру
    "Horizon Cinema".
    Твоя мета: за 3 кроки зрозуміти людину і порекомендувати їй ОДИН ідеальний
    фільм зі списку.

    СПИСОК ДОСТУПНИХ ФІЛЬМІВ:
    ${movies}

    СТРУКТУРА КВІЗУ (суворо 3 кроки + фінал):

    Крок 1 – Емоційний стан
    Запитай як людина себе почуває зараз або чого хоче від вечора.
    Варіанти мають відображати різні емоційні стани, наприклад:
    ["Хочу відволіктись і посміятись", "Готовий до чогось глибокого", "Шукаю
    адреналін і екшн", "Хочу щось зворушливе"]

    Крок 2 – Контекст перегляду
    Запитай з ким і в якій атмосфері людина дивитиметься.
    Варіанти, наприклад:
    ["Сам/сама, хочу занурення", "З другом/подругою", "З родиною або дітьми",
    "На побаченні"]

    Крок 3 – Один уточнюючий параметр
    Обери ОДНЕ з: тривалість, темп фільму, або епоха/сеттинг – залежно від
    відповідей кроків 1-2.
    Наприклад якщо людина хоче адреналін і йде з другом – запитай про темп:
    ["Швидкий монтаж, без зупинок", "Є час розкрити персонажів", "Не важливо"]

    Крок 4 (isFinal: true) – Рекомендація
    Проаналізуй всі відповіді і знайди найкращий збіг у списку фільмів.
    У "message" напиши 2-3 речення: чому саме цей фільм підходить цій людині в
    цей момент.
    Будь конкретним – посилайся на те що людина відповіла.
    У "recommendedTitle" вкажи ТОЧНУ назву фільму зі списку (копіюй дослівно).
```

		Фріз О.І.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Чижмотря О.Г.				57
Змн.	Арк.	№ докум.	Підпис	Дата		

ФОРМАТ ВІДПОВІДІ – ТІЛЬКИ ВАЛІДНИЙ JSON:

```
{
  "step": число від 1 до 4,
  "message": "питання або фінальний коментар",
  "options": ["варіант1", "варіант2", "варіант3", "варіант4"],
  "isFinal": false,
  "recommendedTitle": null
}
```

На фінальному кроці:

```
{
  "step": 4,
  "message": "персональний коментар чому цей фільм",
  "options": [],
  "isFinal": true,
  "recommendedTitle": "Точна назва фільму зі списку"
}
```

ВАЖЛИВО:

- Відповідай ТІЛЬКИ українською мовою
- Варіанти – короткі (3-7 слів), живі, не канцелярські
- НЕ рекомендуй фільми яких немає у списку
- На кроках 1-3 "isFinal" завжди false, "recommendedTitle" завжди null
- Питання мають звучати як від живої людини, не як анкета`;

```
const groqMessages = [
  { role: "system" as const, content: systemPrompt },
  ...messages.map((m: any) => ({
    role: m.role === "assistant" ? ("assistant" as const) : ("user" as const),
    content: m.text,
  })),
];
```

```
const completion = await client.chat.completions.create({
  model: "llama-3.1-8b-instant",
  max_tokens: 500,
  temperature: 0.7,
  response_format: { type: "json_object" },
  messages: groqMessages,
});
```

```
const raw = completion.choices[0]?.message?.content || "";
```

```
try {
  const parsed = JSON.parse(raw);
```

```
  // Підстраховка якщо модель не дотримала формату
  if (!parsed.step) parsed.step = Math.ceil(messages.length / 2) + 1;
  if (!parsed.options) parsed.options = [];
```

		Фріз О.І.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Чижмотря О.Г.				58
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    if (parsed.isFinal === undefined) parsed.isFinal = false;
    if (!parsed.recommendedTitle) parsed.recommendedTitle = null;

    res.json({ success: true, ...parsed });
  } catch (parseError) {
    console.error("JSON Parse Error:", raw);
    res.json({
      success: true,
      step: 1,
      message: "Як ти себе почуваєш сьогодні ввечері?",
      options: ["Хочу відволіктись і посміятись", "Готовий до чогось глибокого", "Шукаю адреналін", "Хочу щось зворушливе"],
      isFinal: false,
      recommendedTitle: null,
    });
  }
} catch (err) {
  console.error("Groq API Error:", err);
  res.status(500).json({ success: false, message: "AI Service Error" });
}
});

export default router;

-----

import React, { createContext, useContext, useEffect, useState } from
"react";

const AuthContext = createContext(null);

export const AuthProvider = ({ children }) => {
  const [user, setUser] = useState(null);
  const [loading, setLoading] = useState(true);

  useEffect(() => {
    fetch("/api/auth/me", { credentials: "include" })
      .then(res => res.ok ? res.json() : null)
      .then(data => {
        setUser(data);
        setLoading(false);
      })
      .catch(() => setLoading(false));
  }, []);

  const login = async (email, password) => {
    const res = await fetch("/api/auth/login", {
      method: "POST",
      headers: { "Content-Type": "application/json" },
      credentials: "include",
      body: JSON.stringify({ email, password }),
    });
  };

```

```

    const data = await res.json();
    if (!res.ok) throw new Error(data.message);
    setUser(data.user);
    return data.user;
  };

const register = async (name, email, password) => {
  const res = await fetch("/api/auth/register", {
    method: "POST",
    headers: { "Content-Type": "application/json" },
    credentials: "include",
    body: JSON.stringify({ name, email, password }),
  });
  const data = await res.json();
  if (!res.ok) throw new Error(data.message);
  setUser(data.user);
  return data.user;
};

const logout = async () => {
  await fetch("/api/auth/logout", {
    method: "POST",
    credentials: "include",
  });
  setUser(null);
};

return (
  <AuthContext.Provider value={{ user, loading, login, register, logout,
setUser }}>
    {children}
  </AuthContext.Provider>
);
};

export const useAuth = () => useContext(AuthContext);

-----

import React, { useState } from "react";
import { useNavigate, Link, useLocation } from "react-router-dom";
import { useAuth } from "../context/AuthContext";
import { validateEmail, validatePassword } from "../utils/validators";

const Login = () => {
  const navigate = useNavigate();
  const { login } = useAuth();

  const location = useLocation();
  const from = location.state?.from || "/";

```

```

const [form, setForm] = useState({ email: "", password: "" });
const [errors, setErrors] = useState({});
const [serverError, setServerError] = useState("");
const [loading, setLoading] = useState(false);

const handleChange = (e) => {
  const { name, value } = e.target;
  setForm(prev => ({ ...prev, [name]: value }));
  if (errors[name]) {
    setErrors(prev => ({ ...prev, [name]: null }));
  }
};

const validate = () => {
  const emailErr = validateEmail(form.email);
  const newErrors = {};
  if (emailErr) newErrors.email = emailErr;
  if (!form.password) newErrors.password = "Пароль обов'язковий";

  setErrors(newErrors);
  return Object.keys(newErrors).length === 0;
};

const handleSubmit = async (e) => {
  e.preventDefault();
  setServerError("");
  if (!validate()) return;

  setLoading(true);
  try {
    const loggedUser = await login(form.email.trim(), form.password);
    if (loggedUser.role === "admin") navigate("/admin");
    else navigate(from);
  } catch (err) {
    setServerError(err.message);
  } finally {
    setLoading(false);
  }
};

const inputClass = (field) =>
`w-full border rounded-lg px-4 py-2 text-sm focus:outline-none
focus:ring-2 transition ${
  errors[field]
  ? "border-red-400 focus:ring-red-300"
  : "border-gray-300 focus:ring-black"
}`;

return (
  <div className="min-h-screen bg-gray-50 flex items-center justify-center
  px-4">

```

		Фріз О.І.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Чижмотря О.Г.				61
Змн.	Арк.	№ докум.	Підпис	Дата		

```

<div className="bg-white rounded-2xl shadow-md w-full max-w-md p-8">

  <h1 className="text-2xl font-bold text-center mb-2">Вхід</h1>
  <p className="text-center text-gray-500 text-sm mb-6">Раді бачити
вас знову</p>

  {serverError && (
    <div className="bg-red-50 text-red-600 text-sm px-4 py-3 rounded-
lg mb-4">
      {serverError}
    </div>
  )}

  <form onSubmit={handleSubmit} className="space-y-4" noValidate>
    <div>
      <label className="block text-sm font-medium text-gray-700 mb-
1">Email</label>
      <input
        type="email"
        name="email"
        value={form.email}
        onChange={handleChange}
        placeholder="you@example.com"
        className={inputClass("email")}
      />
      {errors.email && <p className="text-red-500 text-xs mt-
1">{errors.email}</p>}}
    </div>

    <div>
      <label className="block text-sm font-medium text-gray-700 mb-
1">Пароль</label>
      <input
        type="password"
        name="password"
        value={form.password}
        onChange={handleChange}
        placeholder="....."
        className={inputClass("password")}
      />
      {errors.password && <p className="text-red-500 text-xs mt-
1">{errors.password}</p>}}
    </div>

    <div className="text-right">
      <Link to="/forgot-password" className="text-sm text-gray-500
hover:text-black">
        Забули пароль?
      </Link>
    </div>
  </form>

```

```

        <button
          type="submit"
          disabled={loading}
          className="w-full bg-black text-white py-2 rounded-lg font-
semibold hover:bg-gray-800 transition disabled:opacity-50"
        >
          {loading ? "Вхід..." : "Увійти"}
        </button>
      </form>

      <div className="relative my-6">
        <div className="absolute inset-0 flex items-center">
          <div className="w-full border-t border-gray-200" />
        </div>
        <div className="relative flex justify-center text-xs text-gray-
400">
          <span className="bg-white px-3">або</span>
        </div>
      </div>

      <a href="/api/auth/google"
        className="w-full flex items-center justify-center gap-3 border
border-gray-300 hover:border-gray-400 text-gray-700 py-2 rounded-lg text-sm
font-medium transition"
      >
        
        Увійти через Google
      </a>

      <p className="text-center text-sm text-gray-500 mt-6">
        Немає акаунту?{" "}
        <Link to="/register" className="text-black font-semibold
hover:underline">Зареєструватись</Link>
      </p>
    </div>
  </div>
);
};

export default Login;

-----

import React, { useState } from "react";
import { useNavigate, Link } from "react-router-dom";
import { useAuth } from "../context/AuthContext";
import { validateName, validateEmail, validatePassword } from
"../utils/validators";

const Register = () => {

```

		Фріз О.І.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Чижмотря О.Г.				63
Змн.	Арк.	№ докум.	Підпис	Дата		

```

const navigate = useNavigate();
const { register } = useAuth();

const [form, setForm] = useState({ name: "", email: "", password: "" });
const [errors, setErrors] = useState({});
const [serverError, setServerError] = useState("");
const [loading, setLoading] = useState(false);

const handleChange = (e) => {
  const { name, value } = e.target;
  setForm(prev => ({ ...prev, [name]: value }));
  if (errors[name]) {
    setErrors(prev => ({ ...prev, [name]: null }));
  }
};

const validate = () => {
  const nameErr = validateName(form.name);
  const emailErr = validateEmail(form.email);
  const passwordErr = validatePassword(form.password);

  const newErrors = {};
  if (nameErr) newErrors.name = nameErr;
  if (emailErr) newErrors.email = emailErr;
  if (passwordErr) newErrors.password = passwordErr;

  setErrors(newErrors);
  return Object.keys(newErrors).length === 0;
};

const handleSubmit = async (e) => {
  e.preventDefault();
  setServerError("");
  if (!validate()) return;

  setLoading(true);
  try {
    await register(form.name.trim(), form.email.trim(), form.password);
    navigate("/");
  } catch (err) {
    setServerError(err.message);
  } finally {
    setLoading(false);
  }
};

const inputClass = (field) =>
  `w-full border rounded-lg px-4 py-2 text-sm focus:outline-none
  focus:ring-2 transition ${
    errors[field]
      ? "border-red-400 focus:ring-red-300"
  }

```

		Фріз О.І.			ІІЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
		Чижмотря О.Г.				64
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        : "border-gray-300 focus:ring-black"
    }`;

    return (
      <div className="min-h-screen bg-gray-50 flex items-center justify-center px-4">
        <div className="bg-white rounded-2xl shadow-md w-full max-w-md p-8">

          <h1 className="text-2xl font-bold text-center mb-2">Реєстрація</h1>
          <p className="text-center text-gray-500 text-sm mb-6">
            Створіть акаунт щоб бронювати квитки
          </p>

          {serverError && (
            <div className="bg-red-50 text-red-600 text-sm px-4 py-3 rounded-
lg mb-4">
              {serverError}
            </div>
          )}

          <form onSubmit={handleSubmit} className="space-y-4" noValidate>
            <div>
              <label className="block text-sm font-medium text-gray-700 mb-
1">Ім'я</label>
              <input
                type="text"
                name="name"
                value={form.name}
                onChange={handleChange}
                placeholder="Іван Петренко"
                className={inputClass("name")}
              />
              {errors.name && <p className="text-red-500 text-xs mt-
1">{errors.name}</p>}
            </div>

            <div>
              <label className="block text-sm font-medium text-gray-700 mb-
1">Email</label>
              <input
                type="email"
                name="email"
                value={form.email}
                onChange={handleChange}
                placeholder="you@example.com"
                className={inputClass("email")}
              />
              {errors.email && <p className="text-red-500 text-xs mt-
1">{errors.email}</p>}
            </div>
          </form>
        </div>
      </div>
    );
  }
}

```

```

        <div>
            <label className="block text-sm font-medium text-gray-700 mb-
1">Пароль</label>
            <input
                type="password"
                name="password"
                value={form.password}
                onChange={handleChange}
                placeholder="....."
                className={inputClass("password")}
            />
            {errors.password && <p className="text-red-500 text-xs mt-
1">{errors.password}</p>}
        </div>

        <button
            type="submit"
            disabled={loading}
            className="w-full bg-black text-white py-2 rounded-lg font-
semibold hover:bg-gray-800 transition disabled:opacity-50"
        >
            {loading ? "Реєстрація..." : "Зареєструватись"}
        </button>
    </form>

    <div className="relative my-6">
        <div className="absolute inset-0 flex items-center">
            <div className="w-full border-t border-gray-200" />
        </div>
        <div className="relative flex justify-center text-xs text-gray-
400">
            <span className="bg-white px-3">або</span>
        </div>
    </div>

    <a href="/api/auth/google"
        className="w-full flex items-center justify-center gap-3 border
border-gray-300 hover:border-gray-400 text-gray-700 py-2 rounded-lg text-sm
font-medium transition"
    >
        
        Зареєструватись через Google
    </a>

    <p className="text-center text-sm text-gray-500 mt-6">
        Вже є акаунт?{" "}
        <Link to="/login" className="text-black font-semibold
hover:underline">Увійти</Link>
    </p>

```

```

        </div>
    </div>
    );
};

export default Register;

-----

import React, { useEffect, useState } from "react";
import { useParams, useNavigate } from "react-router-dom";
import PaymentModal from "../components/shared/PaymentModal";

const SeatSelection = () => {
    const { sessionId } = useParams();
    const navigate = useNavigate();

    const [seats, setSeats] = useState([]);
    const [selectedSeats, setSelectedSeats] = useState([]);
    const [sessionInfo, setSessionInfo] = useState(null);
    const [loading, setLoading] = useState(true);
    const [showPayment, setShowPayment] = useState(false);
    const [bookingId, setBookingId] = useState(null);
    const [booking, setBooking] = useState(false);

    useEffect(() => {
        fetch(`/api/sessions/${sessionId}`)
            .then(res => res.json())
            .then(data => setSessionInfo(data))
            .catch(err => console.error(err));

        fetch(`/api/seats?session_id=${sessionId}`)
            .then(res => res.json())
            .then(data => { setSeats(data); setLoading(false); })
            .catch(err => { console.error(err); setLoading(false); });
    }, [sessionId]);

    const toggleSeat = (seat) => {
        if (seat.isBooked || seat.isBlocked) return;
        setSelectedSeats(prev => {
            const exists = prev.find(s => s.id === seat.id);
            if (exists) return prev.filter(s => s.id !== seat.id);
            return [...prev, seat];
        });
    };

    const getSeatStatus = (seat) => {
        if (seat.isBlocked) return "blocked";
        if (seat.isBooked) return "booked";
        if (selectedSeats.find(s => s.id === seat.id)) return "selected";
        return "available";
    };

```

		Фріз О.І.			ІІЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
		Чижемотря О.Г.				67
Змн.	Арк.	№ докум.	Підпис	Дата		

```

};

const getSeatClass = (status) => {
  const base = "w-8 h-8 rounded-t-lg text-xs flex items-center justify-
center font-medium transition-colors";
  if (status === "blocked") return `${base} bg-gray-800 text-gray-600
cursor-not-allowed`;
  if (status === "booked") return `${base} bg-gray-300 text-gray-400
cursor-not-allowed`;
  if (status === "selected") return `${base} bg-red-500 text-white cursor-
pointer`;
  return `${base} bg-green-100 text-green-800 hover:bg-green-300 border
border-green-300 cursor-pointer`;
};

const rows = seats.reduce((acc, seat) => {
  if (!acc[seat.row]) acc[seat.row] = [];
  acc[seat.row].push(seat);
  return acc;
}, {});

const totalPrice = sessionInfo ? selectedSeats.length * sessionInfo.price
: 0;

const handleCheckout = async () => {
  if (selectedSeats.length === 0 || booking) return;
  setBooking(true);

  try {
    const res = await fetch("/api/seats/book", {
      method: "POST",
      headers: { "Content-Type": "application/json" },
      credentials: "include",
      body: JSON.stringify({
        session_id: parseInt(sessionId),
        seat_ids: selectedSeats.map(s => s.id),
        total_price: totalPrice,
      }),
    });
  };

  const data = await res.json();

  if (data.success) {
    setBookingId(data.booking_id);
    setShowPayment(true);
  } else if (res.status === 409) {
    alert(data.message || "Місця вже зайняті. Оберіть інші.");
    setSelectedSeats([]);
    setBooking(false);
    const updated = await fetch(`/api/seats?session_id=${sessionId}`);
    setSeats(await updated.json());
  }
};

```

		Фріз О.І.			ПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмотря О.Г.				68
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    } else {
      alert(data.message || "Помилка бронювання");
      setBooking(false);
    }
  } catch (err) {
    console.error(err);
    alert("Помилка з'єднання");
    setBooking(false);
  }
};

// Скасовує бронювання якщо користувач закрив модал без оплати
const handleClosePayment = async () => {
  if (bookingId) {
    try {
      await fetch("/api/seats/cancel", {
        method: "POST",
        headers: { "Content-Type": "application/json" },
        credentials: "include",
        body: JSON.stringify({ booking_id: bookingId }),
      });
    } catch (err) {
      console.error(err);
    }
  }
  setShowPayment(false);
  setBookingId(null);
  setBooking(false);
  setSelectedSeats([]);
};

const handlePaymentSuccess = () => {
  navigate(`/booking-success/${bookingId}`);
};

const handlePayLater = async () => {
  try {
    await fetch("/api/seats/notify-pending", {
      method: "POST",
      headers: { "Content-Type": "application/json" },
      credentials: "include",
      body: JSON.stringify({ booking_id: bookingId }),
    });
  } catch (err) {
    console.error(err);
  }
  setShowPayment(false);
  navigate("/profile?tab=Your Orders");
};

```

```

    if (loading) return <div className="text-white p-10">Завантаження...</div>;

    return (
      <div className="min-h-screen bg-gray-50">

        {showPayment && bookingId && (
          <PaymentModal
            totalPrice={totalPrice}
            bookingId={bookingId}
            onSuccess={handlePaymentSuccess}
            onClose={handleClosePayment}
            onPayLater={handlePayLater}
          />
        )}

        <div className="bg-black text-white px-6 py-4 flex items-center gap-4">
          <button onClick={() => navigate(-1)} className="text-gray-400 hover:text-white">
            ← Назад
          </button>
          <div>
            <h1 className="font-bold text-lg">Вибip мicць</h1>
            {sessionInfo && (
              <p className="text-sm text-gray-400">Сeанс •
                {sessionInfo.price} за мicцe</p>
            )}
          </div>
        </div>

        <div className="max-w-2xl mx-auto px-4 mt-8">
          <div className="w-full h-2 bg-gray-400 rounded mb-1" />
          <p className="text-center text-xs text-gray-500 mb-8">SCREEN</p>

          <div className="space-y-2">
            {Object.keys(rows).sort((a, b) => Number(a) - Number(b)).map(row
=> (
              <div key={row} className="flex items-center gap-2">
                <span className="text-xs text-gray-400 w-4 text-right">{row}</span>
                <div className="flex gap-1 flex-1 justify-center">
                  {rows[row].sort((a, b) => a.number - b.number).map(seat => {
                    const status = getSeatStatus(seat);
                    return (
                      <button
                        key={seat.id}
                        onClick={() => toggleSeat(seat)}
                        className={getSeatClass(status)}
                        title={`Ряд ${seat.row}, Мicцe ${seat.number}`}
                      >

```

		Фріз О.І.			ІІЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
		Чижмoтpя О.Г.				70
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        {seat.number}
      </button>
    );
  })}
</div>
  <span className="text-xs text-gray-400 w-4">{row}</span>
</div>
  )})}
</div>

<div className="flex justify-center gap-6 mt-8 text-xs text-gray-
600">
  <div className="flex items-center gap-2">
    <div className="w-6 h-6 rounded-t-lg bg-green-100 border border-
green-300" />
    <span>Вільне</span>
  </div>
  <div className="flex items-center gap-2">
    <div className="w-6 h-6 rounded-t-lg bg-red-500" />
    <span>Вибране</span>
  </div>
  <div className="flex items-center gap-2">
    <div className="w-6 h-6 rounded-t-lg bg-gray-300" />
    <span>Зайняте</span>
  </div>
  <div className="flex items-center gap-2">
    <div className="w-6 h-6 rounded-t-lg bg-gray-800" />
    <span>Недоступне</span>
  </div>
</div>
</div>

{selectedSeats.length > 0 && (
  <div className="fixed bottom-0 left-0 right-0 bg-white border-t
shadow-lg px-6 py-4 flex items-center justify-between">
    <div>
      <p className="text-sm text-gray-500">
        Вибрано: <span className="font-bold text-
black">{selectedSeats.length}</span>
      </p>
      <p className="text-sm text-gray-500">
        Місяць: <span className="font-medium text-black">
          {selectedSeats.map(s => `P${s.row}M${s.number}`)}.join(", ")}
        </span>
      </p>
    </div>
    <div className="flex items-center gap-4">
      <p className="font-bold text-xl">€{totalPrice}</p>
      <button
        onClick={handleCheckout}
        disabled={booking}

```



```

const [cvv, setCvv] = useState("");
const [name, setName] = useState("");
const [processing, setProcessing] = useState(false);
const [error, setError] = useState("");

const formatCard = (val) =>
  val.replace(/\D/g, "").slice(0, 16).replace(/(.{4})/g, "$1 ").trim();

const formatExpiry = (val) =>
  val.replace(/\D/g, "").slice(0, 4).replace(/^(\d{2})/, "$1/");

const handlePay = async () => {
  setError("");

  const rawCard = card.replace(/\s/g, "");
  if (rawCard.length !== 16) { setError("Невірний номер картки"); return; }

  if (expiry.length < 5) { setError("Невірна дата"); return; }
  if (cvv.length < 3) { setError("Невірний CVV"); return; }
  if (!name.trim()) { setError("Введіть ім'я власника картки"); return; }

  setProcessing(true);
  await new Promise(resolve => setTimeout(resolve, 2000));

  if (rawCard.startsWith("0000")) {
    setError("Платіж відхилено. Спробуйте іншу картку.");
    setProcessing(false);
    return;
  }

  try {
    const res = await fetch("/api/seats/pay", {
      method: "POST",
      headers: { "Content-Type": "application/json" },
      credentials: "include",
      body: JSON.stringify({ booking_id: bookingId }),
    });

    const data = await res.json();
    if (data.success) {
      onSuccess();
    } else {
      setError("Помилка підтвердження платежу");
    }
  } catch (err) {
    setError("Помилка сервера");
  } finally {
    setProcessing(false);
  }
};

```

```

return (
  <div className="fixed inset-0 bg-black/60 z-50 flex items-center
justify-center px-4">
    <div className="bg-white rounded-2xl shadow-xl w-full max-w-md p-6">

      <div className="flex justify-between items-center mb-6">
        <h2 className="text-xl font-bold">Оплата</h2>
        <button onClick={onClose} className="text-gray-400 hover:text-
black text-xl">x</button>
      </div>

      <div className="bg-gray-50 rounded-xl px-4 py-3 mb-6 flex justify-
between items-center">
        <span className="text-gray-500 text-sm">До сплати</span>
        <span className="text-2xl font-bold text-
[#EC4D25]">€{totalPrice}</span>
      </div>

      {error && (
        <div className="bg-red-50 text-red-600 text-sm px-4 py-2 rounded-
lg mb-4">
          {error}
        </div>
      )}

      <div className="space-y-4">
        <div>
          <label className="text-sm font-medium text-gray-700 block mb-
1">Номер картки</label>
          <input
            type="text"
            value={card}
            onChange={e => setCard(formatCard(e.target.value))}
            placeholder="4111 1111 1111 1111"
            className="w-full border border-gray-300 rounded-lg px-4 py-2
text-sm focus:outline-none focus:ring-2 focus:ring-black tracking-widest"
            />
        </div>

        <div>
          <label className="text-sm font-medium text-gray-700 block mb-
1">Ім'я на картці</label>
          <input
            type="text"
            value={name}
            onChange={e => setName(e.target.value.toUpperCase())}
            placeholder="your name"
            className="w-full border border-gray-300 rounded-lg px-4 py-2
text-sm focus:outline-none focus:ring-2 focus:ring-black"
            />
        </div>
      </div>
    </div>
  </div>
)

```

```

<div className="flex gap-4">
  <div className="flex-1">
    <label className="text-sm font-medium text-gray-700 block mb-1">Термін дії</label>
    <input
      type="text"
      value={expiry}
      onChange={e => setExpiry(formatExpiry(e.target.value))}
      placeholder="MM/YY"
      className="w-full border border-gray-300 rounded-lg px-4 py-2 text-sm focus:outline-none focus:ring-2 focus:ring-black"
    />
  </div>
  <div className="flex-1">
    <label className="text-sm font-medium text-gray-700 block mb-1">CVV</label>
    <input
      type="password"
      value={cvv}
      onChange={e => setCvv(e.target.value.replace(/\D/g, ""))}
      placeholder="..."
      className="w-full border border-gray-300 rounded-lg px-4 py-2 text-sm focus:outline-none focus:ring-2 focus:ring-black"
    />
  </div>
</div>

<button
  onClick={handlePay}
  disabled={processing}
  className="w-full mt-6 bg-[#EC4D25] hover:bg-red-600 text-white py-3 rounded-xl font-semibold transition disabled:opacity-50 flex items-center justify-center gap-2"
  >
  {processing ? (
    <>
      <svg className="animate-spin h-5 w-5 text-white" viewBox="0 0 24 24" fill="none">
        <circle className="opacity-25" cx="12" cy="12" r="10" stroke="currentColor" strokeWidth="4"/>
        <path className="opacity-75" fill="currentColor" d="M4 12a8 8 0 0 1 8 0 0 18-8v8z"/>
      </svg>
      Обробка платежу...
    </>
  ) : (
    `Оплатити ₾${totalPrice}`
  )}
</button>

```

		Фріз О.І.			ІІЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
		Чижмотря О.Г.				75
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    </button>

    <button
      onClick={onPayLater}
      disabled={processing}
      className="w-full mt-3 border border-gray-300 hover:border-gray-
400 text-gray-600 hover:text-gray-800 py-3 rounded-xl font-medium transition
disabled:opacity-50"
    >
      Оплатити пізніше
    </button>

    <p className="text-center text-xs text-gray-400 mt-3">
      Тестова оплата • Картка: 4111 1111 1111 1111
    </p>
  </div>
</div>
);
};

export default PaymentModal;

```

```

-----

import React from "react";
import { Navigate } from "react-router-dom";
import { useAuth } from "../context/AuthContext";

const ProtectedRoute = ({ children }) => {
  const { user, loading } = useAuth();

  if (loading) return <div className="text-white p-
10">Завантаження...</div>;
  if (!user) {
    localStorage.setItem("redirectAfterLogin", window.location.pathname);
    return <Navigate to="/login" state={{ from: window.location.pathname }}
/>;
  }
  if (user.role === "admin") return <Navigate to="/admin" />;

  return children;
};

export default ProtectedRoute;

```

```

-----

export const EMAIL_REGEX = /^[a-zA-Z0-9._%+\-]+@[a-zA-Z0-9.\-]+\.[a-zA-
Z]{2,}$/;
export const NAME_REGEX = /^[a-zA-Za-яA-Яiİïë€г'\'- ]+$/;

```

		Фріз О.І.			ПІЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмотря О.Г.				76
Змн.	Арк.	№ докум.	Підпис	Дата		

```

export const validateName = (name) => {
  if (!name.trim()) return "Ім'я обов'язкове";
  if (name.trim().length < 2) return "Ім'я мінімум 2 символи";
  if (name.trim().length > 50) return "Ім'я максимум 50 символів";
  if (!NAME_REGEX.test(name.trim())) return "Ім'я може містити лише літери, пробіл, дефіс";
  return null;
};

export const validateEmail = (email) => {
  if (!email.trim()) return "Email обов'язковий";
  if (!EMAIL_REGEX.test(email.trim())) return "Введіть коректний email (наприклад: user@mail.com)";
  return null;
};

export const validatePassword = (password) => {
  if (!password) return "Пароль обов'язковий";
  if (password.length < 6) return "Пароль мінімум 6 символів";
  if (password.length > 64) return "Пароль максимум 64 символи";
  if (!/[a-zA-Za-яA-Яiİİİēē]/.test(password)) return "Пароль повинен містити хоча б одну літеру";
  return null;
};

export const validateMovieTitle = (value) => {
  if (!value?.trim()) return "Назва обов'язкова";
  if (value.trim().length < 2) return "Назва занадто коротка (мінімум 2 символи)";
  if (value.trim().length > 200) return "Назва занадто довга (максимум 200 символів)";
  return null;
};

export const validateDuration = (value) => {
  if (value === "" || value === null || value === undefined) return "Тривалість обов'язкова";
  const n = Number(value);
  if (!Number.isInteger(n) || n <= 0) return "Тривалість має бути цілим числом > 0";
  if (n > 600) return "Тривалість не може перевищувати 600 хв";
  return null;
};

export const validateReleaseDate = (value) => {
  if (!value) return "Дата виходу обов'язкова";
  return null;
};

export const validateScreeningEndDate = (endDate, releaseDate) => {

```

		Фріз О.І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмотря О.Г.				77
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    if (!endDate) return null;
    if (releaseDate && endDate <= releaseDate) return "Дата закінчення має
бути після дати виходу";
    return null;
};

export const validateGenres = (genres) => {
    if (!genres || genres.length === 0) return "Оберіть хоча б один жанр";
    return null;
};

export const validateImageUrl = (url) => {
    if (!url) return "Завантажте постер";
    return null;
};

export const validateMovieId = (value) => {
    if (!value) return "Оберіть фільм";
    return null;
};

export const validateHallId = (value) => {
    if (!value) return "Оберіть залу";
    return null;
};

export const validateStartTime = (value) => {
    if (!value) return "Вкажіть дату та час сеансу";
    if (new Date(value) <= new Date()) return "Сеанс не може бути в минулому";
    return null;
};

export const validatePrice = (value) => {
    if (value === "" || value === null || value === undefined) return "Ціна
обов'язкова";
    const n = Number(value);
    if (isNaN(n) || n <= 0) return "Ціна має бути більше 0";
    if (n > 10000) return "Ціна не може перевищувати 10 000 ₾";
    return null;
};

export const validateCinemaName = (value) => {
    if (!value?.trim()) return "Назва обов'язкова";
    if (value.trim().length < 2) return "Назва занадто коротка";
    return null;
};

export const validateCinemaLocation = (value) => {
    if (!value?.trim()) return "Адреса обов'язкова";
    return null;
};

```

		Фріз О.І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмотря О.Г.				78
Змн.	Арк.	№ докум.	Підпис	Дата		

```

export const validateLatitude = (value) => {
  if (value === "" || value === null || value === undefined) return null;
  const n = Number(value);
  if (isNaN(n) || n < -90 || n > 90) return "Широта: від -90 до 90";
  return null;
};

export const validateLongitude = (value) => {
  if (value === "" || value === null || value === undefined) return null;
  const n = Number(value);
  if (isNaN(n) || n < -180 || n > 180) return "Довгота: від -180 до 180";
  return null;
};

```

		Фріз О.І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмотря О.Г.				79
Змн.	Арк.	№ докум.	Підпис	Дата		