

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:

«Додаток для бронювання квитків у кіно»

Виконав студент групи ІПЗк-23-1
ПЕТРИШИН Андрій Олександрович

Керівник роботи:
ГОДЛЕВСЬКИЙ Юрій Олександрович

Рецензент:
ЄФРЕМОВ Юрій Миколайович

Житомир – 2025

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

«26» лютого 2024 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **ПЕТРИШИН Андрій Олександрович**

Керівник роботи: **ГОДЛЕВСЬКИЙ Юрій Олександрович**

Тема роботи: **«Додаток для бронювання квитків у кіно»**,

затверджена Наказом закладу вищої освіти від **«14» лютого 2024 р., №61/с**

Вихідні дані для роботи: **додаток для бронювання квитків у кіно**

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

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Годлевський Ю.О.	27.02.2024р.	27.02.2024р.
2	Годлевський Ю.О.	27.02.2024р.	27.02.2024р.
3	Годлевський Ю.О.	27.02.2024р.	27.02.2024р.

РЕФЕРАТ

Випускна кваліфікаційна робота бакалавра складається з веб-додатку для бронювання квитків у кіно та пояснювальної записки. Пояснювальна записка до випускної роботи містить 60 сторінок, 35 ілюстрацій та 24 таблиці. Загальний обсяг – 66 сторінок.

Метою роботи є розробка веб-додатку для здійснення та оплати кіно-квитків. Система включає модулі створення та керування фільмами, кінотеатрами, залами, сеансами, користувачами, перегляд афіші, місць та автоматичної обробки зображень в універсальний формат.

У роботі поставлені основні завдання на розробку програмного забезпечення, проведено аналіз готових аналогічних рішень, обрано архітектуру клієнт-сервер, приведено сценарії використання системи, логіку взаємодії між компонентами додатку, діаграма класів та алгоритм роботи модулів, спроектована база даних на основі MongoDB. Клієнтська частина розроблена з використання фреймворку React. Серверна частина розроблена на основі взаємодії Node.js та Express.js.

Додаток має зручний інтуїтивно зрозумілий інтерфейс, який забезпечує комфортний перегляд та здійснення операцій для користувача.

КЛЮЧОВІ СЛОВА: БРОНЮВАННЯ, КВИТОК, АФІША, ПОСТЕР, РЕЙТИНГ, ТРАНЗАКЦІЇ.

					ІПЗ.КР.Б – 121 – 25 – ЗПП				
Змн.	Арк.	№ докум.	Підпис	Дата	Додаток для бронювання квитків у кіно Пояснювальная записка	Лім.	Арк.	Аркушів	
Розроб.		Петришин А.О.							
Керівник		Годлевський Ю.О					3	60	
						Житомирська політехніка група ІПЗк-23-1			
Н. контр.									
Затвердив									

ABSTRACT

The bachelor's qualification thesis consists of a web application for cinema ticket booking and an explanatory report. The explanatory report contains 60 pages, 35 illustrations, and 24 tables. The total volume is 66 pages.

The goal of the project is to develop a web application for purchasing and paying for cinema tickets. The system includes modules for managing movies, cinemas, halls, screenings, users, viewing the movie schedule (poster), seat selection, and automatic image processing into a universal format.

The work defines the main tasks for software development, analyzes existing similar solutions, selects a client-server architecture, presents usage scenarios of the system, interaction logic between application components, a class diagram, module operation algorithms, and a database design based on MongoDB. The client-side is developed using the React framework. The server-side is implemented based on Node.js and Express.js.

The application features a convenient and intuitive interface, providing users with a comfortable browsing and operation experience.

KEYWORDS: BOOKING, TICKET, POSTER, MOVIE POSTER, RATING, TRANSACTIONS.

		Петришин А.О.			ІПЗ.КР.Б – 121 – 25 – 3ПП	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		4

ЗМІСТ

РЕФЕРАТ.....	1
ABSTRACT	4
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ	7
ВСТУП.....	8
РОЗДІЛ 1 АНАЛІЗ НАПРЯМКІВ ЗАСТОСУВАННЯ ДОДАТКУ БРОНЮВАННЯ	10
1.1 Постановка задачі	10
1.2 Аналіз аналогів програмного продукту:	13
1.3. Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення	17
Висновки до першого розділу.	20
РОЗДІЛ 2. ПРОЕКТУВАННЯ СИСТЕМИ БРОНЮВАННЯ КВИТКІВ	21
2.1 Алгоритм роботи та стани програмної системи	21
2.2 Розробка бази даних системи.....	29
2.3 Проектування та реалізація алгоритмів роботи системи.....	35
2.4 Реалізація функціоналу системи бронювання квитків.....	36
Висновки до другого розділу.....	41
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З СИСТЕМОЮ БРОНЮВАННЯ КВИТКІВ.....	42
3.1 Порядок встановлення та налаштування параметрів додатку.	42
3.2 Структура інтерфейсу додатку для бронювання квитків у кіно	43
3.3 Тестування роботи програмного продукту.....	50

Висновки до третього розділу.....	55
ВИСНОВКИ	56
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	57
ДОДАТКИ.....	61

		Петришин А.О.			ІПЗ.КР.Б – 121 – 25 – ЗПП	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		6

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ПЗ – Програмне забезпечення

БД – База даних

API – Application Program Interface

JSON – JavaScript Object Notation

HTTP – HyperText Transfer Protocol

QR – Quick Response

		Петришин А.О.			ІПЗ.КР.Б – 121 – 25 – ЗПП	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		7

ВСТУП

Наразі кіноіндустрія активно використовує веб-технології і стрімко розвивається, що зумовлює створення онлайн додатків для бронювання місць і оплати квитків. Такі системи забезпечують комфорт для користувачів у виборі фільмів та місць і здійсненні транзакцій, знаходячись в будь-якому куточку світу з доступом до мережі Інтернет.

Актуальність полягає в попиті на автоматизовані дистанційні сервіси і додатки в сфері розваг та кіно. Такі системи дозволяють значно швидше обробити велику кількість користувачів, знижуючи навантаження на персонал кінотеатрів, і, в той ж час, надавати відвідувачам зручний і доступний сервіс.

Метою роботи є розробка веб-додатку для бронювання, який надаватиме доступ до комфортного вибору місць на схемі кінозалу та здійснення безпосереднього бронювання, інтуїтивно зрозумілим процесом онлайн оплати та генерацією квитка.

З огляду на мету роботи, можна визначити наступні завдання:

1. Здійснити порівняльний аналіз існуючих аналогічних рішень для бронювання квитків.
2. Обрати та обґрунтувати інструменти та технології проектування і розробки.
3. Спроекувати архітектуру та розробити ключові функціональні модулі системи додатку.
4. Здійснити тестування програмного продукту.

Об'єктом дослідження є охоплення автоматизаційних процесів для забезпечення ефективного та зручного для користувачів бронювання, включаючи в себе системи вибору бажаних місць в кінозалі, їх резервування в режимі реального часу, здійснення онлайн транзакцій, генерації електронних квитків, адміністрування даними з можливістю контролю прав доступу та захисту даних користувачів.

		Петришин А.О.			ІПЗ.КР.Б – 121 – 25 – ЗПП	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		8

Предметом дослідження є структура та функціональність веб-додатків для здійснення бронювання, що включає в себе механізм взаємодії користувачів із системою, обробки бронювань, онлайн-транзакцій, генерації квитків і адміністрування даними додатку.

		Петришин А.О.			ІПЗ.КР.Б – 121 – 25 – ЗПП	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		9

РОЗДІЛ 1 АНАЛІЗ НАПРЯМКІВ ЗАСТОСУВАННЯ ДОДАТКУ БРОНЮВАННЯ

1.1 Постановка задачі

Метою кваліфікаційної роботи бакалавра є створення веб-додатку для здійснення бронювання квитків задля забезпечення швидкого та безпечного доступу користувачів до функціоналу перегляду фільмів та сеансів, вибору місць у залі та онлайн-оплатою квитків. Також розробляється WebAPI сервіс, який дозволяє в майбутньому масштабувати додаток і забезпечує необхідну гнучкість управління даними.

Онлайн-бронювання квитків – це технологія, що дозволяє економити час користувачів, не змушуючи їх чекати в черзі та спрощувати процес керування кінозалами для адміністраторів [1]. Такий підхід є актуальним у світі необхідної автоматизації та цифровізації.

При створенні подібного рішення необхідно враховувати, що додаток повинен містити в собі декілька програмних модулів, що забезпечують необхідний функціонал.

Однією з основних частин є механізм вибору місць на схемі кінозалу, який представляє собою інтерфейс для вибору місць, відображення доступних, тимчасово зайнятих та заброньованих місць.

Іншим важливим елементом є система обробки платежів та електронних квитків. Даний модуль включає систему здійснення онлайн-транзакцій та, при здійсненні успішної оплати, генерацію електронного квитка.

Для забезпечення безпеки користувачів та доступу його до функціоналу додатку впроваджується модуль аутентифікації та авторизації.

Окрему роль виконує і адміністративний модуль, який дає можливості адміністраторам контролювати всі аспекти роботи системи, а саме: редагування, додавання та видалення фільмів, налаштування сеансів, моніторинг здійснених транзакцій.

		Петришин А.О.			ІПЗ.КР.Б – 121 – 25 – ЗПП	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		10

Зв'язок між клієнтською частиною та серверною забезпечується через Web API, який обробляє всі користувацькі запити, реалізує бізнес-логіку та забезпечує можливість інтеграції з зовнішніми сервісами.

В результаті повинно бути створено вебсайт, на якому будь-який користувач зможе самостійно увійти в систему, переглянути сеанси на бажаний фільм, забронювати місце та купити квиток. Система повинна забезпечувати зручну взаємодію, мати інтуїтивно-зрозумілий інтерфейс, високу продуктивність, а також забезпечувати необхідний рівень безпеки даних користувачів і фінансових транзакцій. Також важливим аспектом є підтримка масштабованості для розширення майбутнього функціоналу та можливість інтеграції зовнішніх сервісів за потреби.

Основними етапами розробки є:

Спочатку розробка дизайну вебсайту. Формується загальна концепція дизайну, яка відображає стиль проєкту. Наступними створюються UI-компоненти (кнопки, форми, навігаційні елементи) які забезпечують зручну взаємодію користувача з додатком. Далі макет адаптуються під розміри різноманітних пристроїв – від смартфонів до комп'ютерів, що, в свою, чергу, дозволяє створити зручний інтерфейс, який коректно виглядає на будь-якому пристрої.

Побудова архітектури серверної частини додатку. На серверній стороні розгортається RESTful Web API, який являє собою набір інструкцій, який дозволяє клієнту та серверу взаємодіяти між собою [2]. Він обробляє клієнтські запити і повертає необхідні дані. Всередині бекенду реалізується бізнес-логіка, а саме: перевірка прав доступу, валідація даних, обробка транзакцій; та налаштовується взаємодія з БД.

Наступним етапом є побудова архітектури клієнтської частини додатку. Клієнтська частина організована у вигляді модулів із чуткою структурою файлів та маршрутизацією. Також тут реалізовується логіка вибору місць у залі, перегляду розкладу сеансів у зручному для користувача вигляді

		Петришин А.О.			ІПЗ.КР.Б – 121 – 25 – ЗПП	Арк.
		Годлевський Ю.О.				11
Змн.	Арк.	№ докум.	Підпис	Дата		

та надсилання запитів до серверу для отримання, оновлення чи видалення даних.

На етапі розробки системи авторизації та ролей користувачів створюються механізми реєстрації нових користувачів і входу вже існуючих з використанням облікових даних. В функціонал інтегрується захист чутливої інформації, що являє собою: хешування паролів та токенизацію. Окрім того, налаштовується розподіл прав для звичайних користувачів та адміністраторів.

Далі реалізується логіка бронювання квитків із перевіркою доступності місць у реальному часі задля уникнення одночасних бронювань. Створюється БД для збереження інформації про користувачів, фільми, бронювання, квитки та транзакції. Серверна частина обробляє транзакції оплати та формує електронні квитки, які потім надсилає користувачеві в разі успішного проведення операції.

Після виконується функціональне тестування додатку – від компонентів користувацького дизайну до API-методів – для перевірки коректності роботи кожної функції. До таких методів відносяться: GET – відповідають за отримання даних, POST – за створення та обробку на вказаний ресурс, PUT – оновлення з новими даними та DELETE – для видалення відповідно [3].

Результатом виконання плану дій є створення веб-додатку, за допомогою якого користувачі можуть здійснювати пошук фільмів та наявних кіносеансів, обирати місця в залі та бронювати їх, здійснювати оплату та отримувати згенерований квиток. Після підтвердження транзакції система автоматично оновлює статус місць, що дозволяє уникнути дублювання бронювання.

		Петришин А.О.			ІПЗ.КР.Б – 121 – 25 – ЗПП	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		12

1.2 Аналіз аналогів програмного продукту:

Fandango є одним із найбільших сервісів у США для пошуку фільмів, перегляду рейтингів та бронювання квитків. Платформа надає зручний інтерфейс для користувачів, які хочуть дізнатися про новинки кіно, переглянути трейлери та огляди фільмів, а також придбати квитки на сеанси [4].

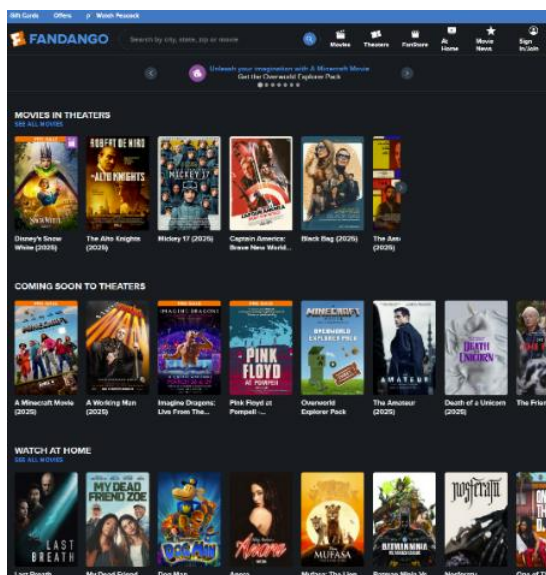


Рис. 1.1. Fandango

Fandango вирізняється надзвичайно зручним та інтуїтивно зрозумілим інтерфейсом. Підтримка інтеграцій з Google Pay та Apple Waller робить процес оплати й збереження квитків максимально простим. Користувачі можуть одразу переглядати оцінки й описи фільмів, отримувати push-повідомлення про зміни сеансів і персональні рекомендації. Мобільний додаток зберігає квитки.

Основним недоліком є фокус на внутрішній ринок, через що інтерфейс і сеанси можуть бути недоступними, а на слабких пристроях система створює велике навантаження.

Multiplex – це популярна українська платформа для бронювання квитків у кінотеатри, яка відрізняється простотою та зручністю використання. Має власну систему бронювання через сайт та мобільний додаток [5].

		Петришин А.О.			ІПЗ.КР.Б – 121 – 25 – ЗПП	Арк.
		Годлевський Ю.О.				13
Змн.	Арк.	№ докум.	Підпис	Дата		

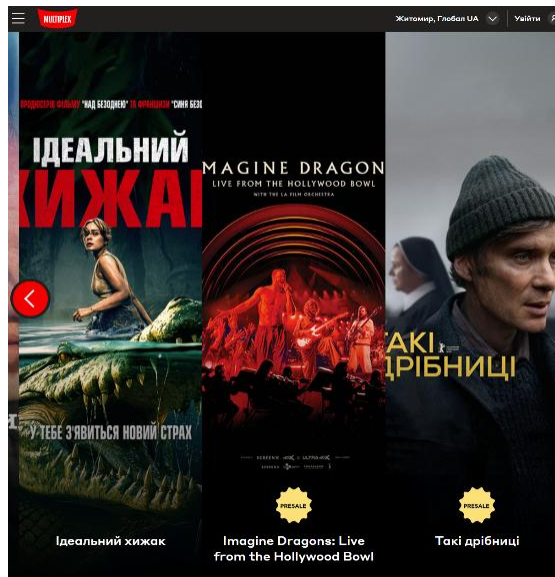


Рис. 1.2. Multiplex

Multiplex пропонує унікальну можливість одразу купувати квитки і обрані попкорн і напої, що виключає необхідність витрат часу на касі. Стабільна система бронювання із простим та чистим інтерфейсом, власні сповіщення та повна підтримка української мови забезпечують комфорт користування для локальних користувачів. Однак недоліком є висока вартість квитків. Також спостерігається часом повільне завантаження сторінок, що може знизити загальне враження від сервісу.

PlanetaKino – це інноваційна платформа для бронювання квитків, яка пропонує унікальні функції для покращення досвіду користувачів [6].



Рис. 1.3. PlanetaKino

		Петришин А.О.			ІПЗ.КР.Б – 121 – 25 – ЗПП	Арк.
		Годлевський Ю.О.				14
Змн.	Арк.	№ докум.	Підпис	Дата		

Дана платформа пропонує передові формати перегляду для більш позитивних вражень. Яскравий та сучасний дизайн, розширений функціонал у застосунку та програма лояльності роблять користування сервісом приємним. Проте складний інтерфейс для новачків та надмірна кількість візуальних ефектів можуть ускладнювати навігацію. Окрім того висока вартість квитків і повільне завантаження можуть стати на заваді швидкому вибору сеансу.

AMC Theatres – провідна кіномережа у США, що пропонує широкий спектр можливостей для кіноглядачів [7].

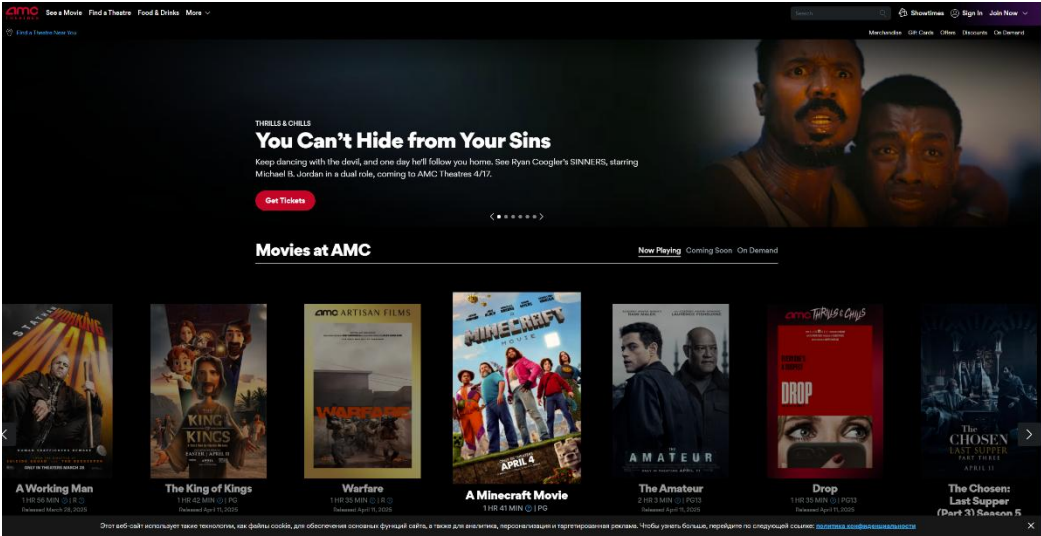


Рис. 1.4. AMC Theatres

AMC Theatres підтримує широкий вибір преміум-форматів для перегляду та має власну програму лояльності під назвою AMC Stubs, яка винагороджує постійних відвідувачів. Додаток вирізняється простотою та зрозумілим для новачків дизайну, дозволяє попередньо замовляти їжу та напої, переглядати рецензії на фільми та трейлери. Квитки мають вигляд QR-кодів (двовимірний штрих-код, за допомогою якого можна кодувати будь-який вид інформації – текст, посиланням, номер телефону тощо [8]). Сервіс орієнтований лише на США, послуги обмежені в менших містах, а мобільний додаток іноді дає збої при оплаті.

Лінія Кіно – відома українська мережа кінотеатрів, що пропонує зручний онлайн-сервіс для перегляду афіш та бронювання [9].

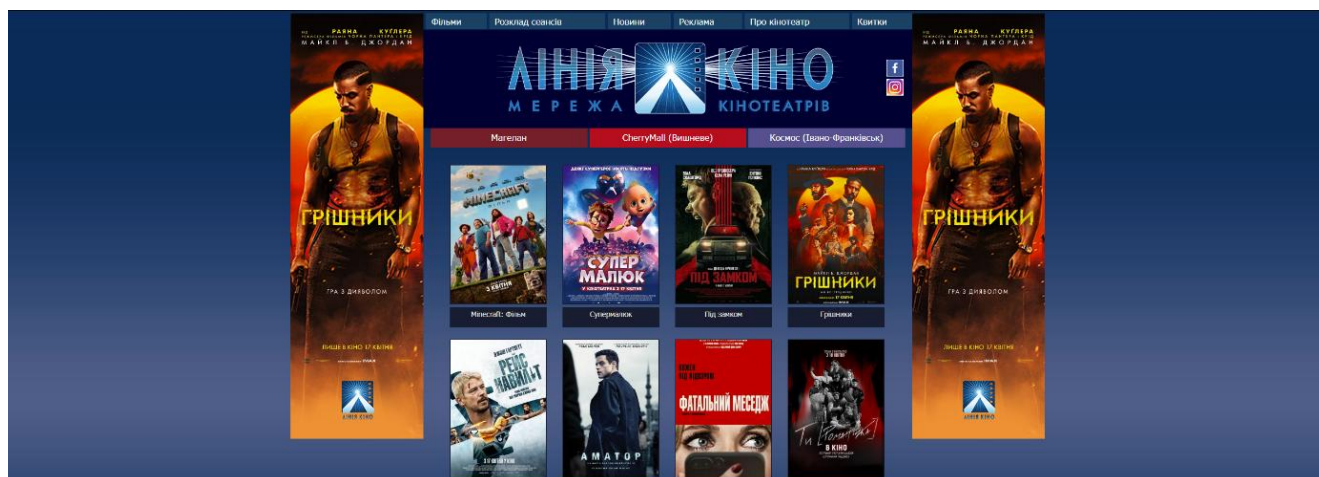


Рис. 1.5. Лінія Кіно

«Лінія Кіно» пропонує базовий та надійний функціонал онлайн-бронювання та можливістю оплати через мобільний застосунок. Регулярні акції та спеціальні пропозиції і зручний вибір місць та перегляд плану зали. Водночас відсутні інноваційні можливості, присутній застарілий та не до кінця адаптований інтерфейс, а також робота лише в мережах власної мережі кінотеатрів, що обмежує привабливість для нових користувачів.

Виходячи з наявних переваг та недоліків, складемо порівняльну таблицю вищезгаданих програмних аналогів (табл. 1.1).

Таблиця 1.1

Таблиця порівняння платформ бронювання квитків

Характеристика	Fandango	Multi-plex	Plane-taKino	AMC Theatres	Лінія Кіно	Додаток
Простота інтерфейсу	+	+	+	+	-	+
Оцінка фільмів	+	-	+	+	-	+
Масштабованість	+	+	+	+	-	+
QR-квитки	+	+	+	+	+	+
Підтримка зовнішніх кінотеатрів	+	-	-	+	-	+
Українська локалізація	-	+	+	-	+	+

Переглянувши існуючі платформи бронювання квитків у кіно, можна виділити основні відмінності, а саме: деякі системи пропонують широкий функціонал та інтеграції сторонніх сервісів; інші забезпечують стабільну робочу роботу, проте мають обмежену персоналізацію та заважкий у освоєнні для нових користувачів інтерфейс; Тоді як, наприклад, PlanetaKino фокусується на візуальних ефектах та нестандартних UX-рішеннях.

Враховуючи отриману в дослідженні інформацію, прийнято рішення створити систему бронювання квитків, яка буде орієнтована на український ринок, а за рівнем функціоналу відповідатиме західним аналогам. Система повинна поєднувати простоту використання та швидкодію, а також бути адаптивною для різних типів пристроїв.

1.3 Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення

Одним з найефективніших підходів для створення веб-додатку є використання архітектури клієнт-сервер. У такій архітектурі клієнт (користувач) взаємодіє з сервером через інтерфейс веб-додатку. Сервер на базі Node.js та Express.js обробляє запити користувачів, зберігає дані в базі даних MongoDB та передає результати на клієнтську сторону, що побудована на React [10].

Для оптимізації процесу роботи з великим обсягом користувацьких запитів та запитів на доступність квитків в реальному часі, особливо у пікові періоди попиту на квитки, застосовується архітектура на основі мікросервісів. Це рішення дозволить модульно розділити систему на кілька компонентів, кожен з яких відповідає за конкретний функціонал, наприклад, бронювання, платіжну систему, вибір місць тощо. Такий підхід забезпечує спрощення розгортання нових функцій без впливу на роботу системи та масштабованість [11].

		Петришин А.О.			ІПЗ.КР.Б – 121 – 25 – ЗПП	Арк.
		Годлевський Ю.О.				17
Змн.	Арк.	№ докум.	Підпис	Дата		

React – сучасний JavaScript-бібліотека для побудови інтерфейсів. Основними перевагами є її компонентна структура, яка дозволяє створювати компоненти, що можна багаторазово використовувати [12]. Також бібліотека забезпечує високу продуктивність завдяки віртуальному DOM. За допомогою розвиненої потужної екосистеми та активної спільноти React легко підтримувати і дозволяє значно пришвидшити розробку [13]. Використання даного рішення дозволяє будувати зручні як користувацький так й адміністративний інтерфейси з інтуїтивною логікою, швидким відгуком та зручною навігацією.

Серверна сторона використовує платформу Node.js у поєднанні з фреймворком Express. Node.js Працює на основі неблокуючої моделі обробки подій. Це надає можливість більш ефективно обробляти велику кількість одночасних запитів на сервер [14]. Також Node.js має велику кількість вже готових модулів, які можна завантажити за допомогою npm [15]. У свою чергу, Express забезпечує потужний інструментарій для роботи і створення RESTful API і маршрутизації, що дозволяє легко структурувати логіку та швидко розширювати функціонал додатку у процесі розробки [16].

Для зберігання даних застосовується БД MongoDB, яка є NoSQL і є оптимальним рішенням для структур із гнучкою схемою, таких як фільми, сеанси, бронювання, користувачі тощо [17]. За допомогою даної бази даних можна швидко оперувати великою кількістю обсягами даних. Окрім того вона підтримує реплікацію та масштабування по горизонталі.

Платіжна система: Для забезпечення можливості онлайн-оплати квитків можна створити власне рішення, що дозволить користувачам здійснювати оплату через популярні платіжні картки.

Також для покращення продуктивності та швидкості роботи, використовується оптимізація статичних файлів, зокрема медіафайли. Тому замість використання CDN (Content Delivery Network) файли зберігатимуться локально на сервері, що зменшить залежність від зовнішніх сервісів та

		Петришин А.О.			ІПЗ.КР.Б – 121 – 25 – ЗПП	Арк.
		Годлевський Ю.О.				18
Змн.	Арк.	№ докум.	Підпис	Дата		

дозволить швидше звертатись до необхідних ресурсів [18]. Для поліпшення ефективності зображення будуть конвертуватись у формат WebP, який забезпечує високу якість зображення при меншому розмірі порівняно з такими форматами як JPEG або PNG. Таке рішення скоротить час завантаження сторінок і значно зменшить навантаження з серверу.

Для забезпечення безпеки особистих даних користувачів та їх платіжної інформації, система повинна використовувати сучасний метод шифрування bcrypt для хешування паролів. Автентифікація реалізована через токени доступу (JWT), що запобігає несанкціонованому доступу до особистої інформації користувача до адміністративних функцій [19; 20].

Вибрана архітектура на основі мікросервісів та клієнт-серверного підходу є оптимальною для виконання вантажних, масштабованих бездодачків. У порівнянні з монолітною структурою, мікросервіси дозволяють масштабувати окремі частини системи без значного впливу на інші частини, що прискорює розробку, спрощує обслуговування та значно зменшує ризики збоїв.

Поєднання Node, MongoDB та React є популярним та надійним рішенням, так як має велику кількість вже готових рішень, прикладів, документацій. Окрім того, стек підтримується величезною спільнотою, що впливає на строки та складність розробки програмного забезпечення. Також це зменшує ймовірність помилок і гарантує довгострокову підтримку та гнучкість при майбутньому розширенні [21].

Таким чином, обране технічне рішення дозволяє забезпечити високу продуктивність та надійність системи бронювання, що, в свою чергу, дозволяє максимально ефективно обробляти велику кількість запитів, розширювати функціональність такої системи і гарантувати захист персональних даних та платіжних даних користувачів.

		Петришин А.О.			ІПЗ.КР.Б – 121 – 25 – ЗПП	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		19

Висновки до першого розділу.

Проведено аналіз предметної області, в результаті якого були визначені основні функціональні і нефункціональні вимоги до веб-додатку. Основна увага приділялась можливості перегляду афіші та інформації про фільми, вибору місць, а також зручному здійсненню бронювання і онлайн-транзакцій. Виділено ключові завдання, які повинна виконувати розроблювана система. Також проведено порівняльний аналіз аналогічних існуючих рішень, який дозволив узагальнити їх сильні та слабкі сторони, що дозволило сформулювати та здійснити вибір відповідних для подальшої реалізації технологій та інструментів, які забезпечать найбільшу ефективність і надійність роботи системи.

		Петришин А.О.			ІПЗ.КР.Б – 121 – 25 – ЗПП	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		20

РОЗДІЛ 2. ПРОЕКТУВАННЯ СИСТЕМИ БРОНЮВАННЯ КВИТКІВ

2.1 Алгоритм роботи та стани програмної системи

Веб-додаток для бронювання квитків у кіно надає зручний та інтуїтивний вебінтерфейс, через який користувачі можуть переглядати афішу, бронювати місця та взаємодіяти із системою. Основна функціональність залежить від активної участі користувачів. Вони поділяються на дві основні категорії: користувачів, які бронюють квитки та переглядають фільми, і адміністратори, які керують контентом, розкладами сеансів та обробкою платежів. Кожне бронювання містить деталі щодо сеансу, місць, ціни, статусу оплати та дозволяє отримати електронний квиток з QR-кодом. Для ілюстрації взаємодії між ролями та системними функціями створено діаграму варіантів використання (рис. 2.1).

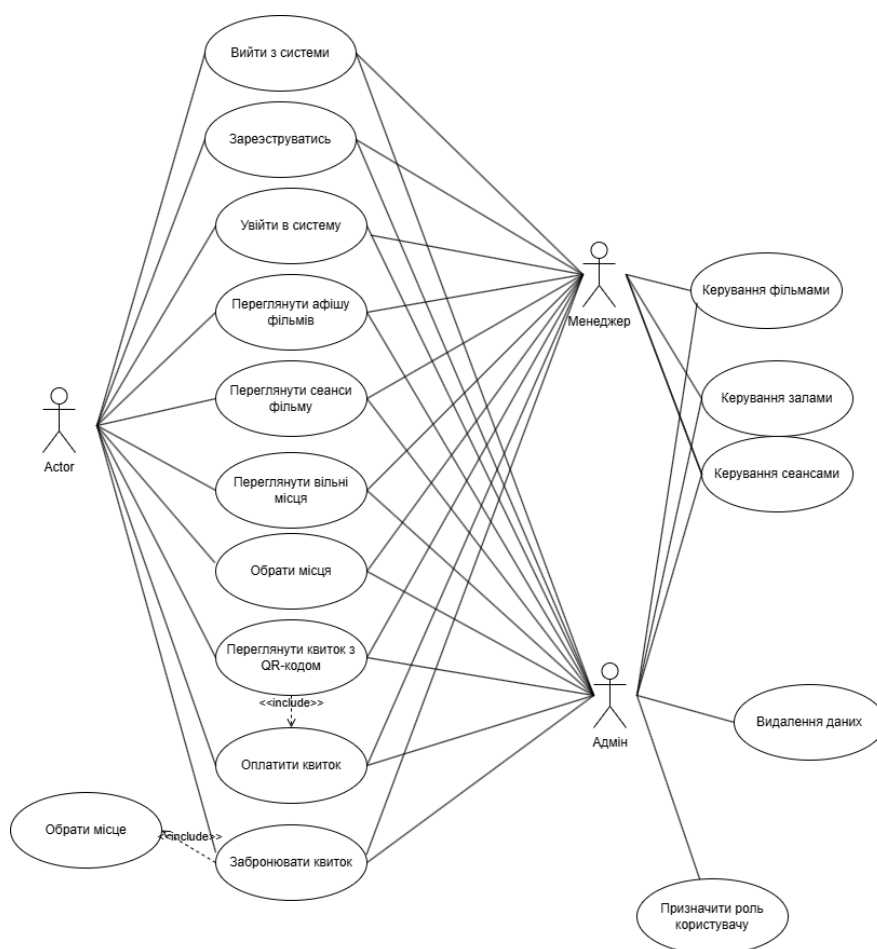


Рис. 2.1. Діаграма варіантів використання

Опис діаграми варіантів викосриатння наведено в таблицях 2.1 – 2.10.

Таблиця 2.1

Опис прецеденту «Вхід до системи»

Назва прецеденту	Вхід до системи
Актори	Користувач, Адміністратор, Менеджер
Короткий опис	Користувач вводить облікові дані для входу до системи

Таблиця 2.2

Опис прецеденту «Реєстрація»

Назва прецеденту	Реєстрація
Актори	Користувач
Короткий опис	Користувач створює обліковий запис, вказуючи ім'я, email та пароль для подальшого входу до системи

Таблиця 2.3

Опис прецеденту «Переглянути афішу фільмів»

Назва прецеденту	Переглянути афішу фільмів
Актори	Користувач, Менеджер, Адміністратор
Короткий опис	Користувач переглядає доступні фільми: назву, постер, описі рейтинг

Таблиця 2.4

Опис прецеденту «Переглянути сеанси фільмів»

Назва прецеденту	Переглянути сеанси фільмів
Актори	Користувач, Менеджер, Адміністратор
Короткий опис	Користувач обирає фільм і бачить доступні дати, часи показу, зал і доступні місця

Таблиця 2.5

Опис прецеденту «Обрати місце»

Назва прецеденту	Обрати місце
Актори	Користувач, Менеджер, Адміністратор
Короткий опис	Користувач на схемі кінозалу обирає доступні місця для перегляду фільму

Таблиця 2.6

Опис прецеденту «Оплатити квиток»

Назва прецеденту	Оплатити квиток
Актори	Користувач, Менеджер, Адміністратор
Короткий опис	Після вибору місця користувач здійснює оплату за квиток онлайн

Таблиця 2.7

Опис прецеденту «Переглянути квиток з QR-кодом»

Назва прецеденту	Переглянути квиток з QR-кодом
Актори	Користувач, Менеджер, Адміністратор
Короткий опис	Користувач переглядає квитки з QR-кодом на окремій сторінці
Включає	Оплатити квиток

Таблиця 2.8

Опис прецеденту «Забронювати квиток»

Назва прецеденту	Забронювати квиток
Актори	Користувач, Менеджер, Адміністратор
Короткий опис	Користувач бронює обране місце на обраний сеанс.
Включає	Оплатити квиток

Таблиця 2.9

Опис прецеденту «Керування фільмами»

Назва прецеденту	Керування фільмами
Актори	Менеджер, Адміністратор
Короткий опис	Додавання, редагування даних про фільми

Таблиця 2.10

Опис прецеденту «Керування залами»

Назва прецеденту	Керування залами
Актори	Менеджер, Адміністратор
Короткий опис	Додавання, редагування даних про зали

Таблиця 2.11

Опис прецеденту «Керування сеансами»

Назва прецеденту	Керування сеансами
Актори	Менеджер, Адміністратор
Короткий опис	Додавання, редагування даних про зали

Таблиця 2.12

Опис прецеденту «Призначити роль користувачу»

Назва прецеденту	Керування залами
Актори	Адміністратор
Короткий опис	Адміністратор змінює роль користувача на «Менеджер» або «Користувач»

Таблиця 2.13

Опис прецеденту «Видалення даних»

Назва прецеденту	Керування сеансами
Актори	Адміністратор
Короткий опис	Адміністратор може видаляти будь-які дані системи

Головна ціль веб-додатку для бронювання квитків у кіно – створити зручне та надійне онлайн-середовище, яке дозволяє глядачам швидко знаходити потрібні фільми, обирати сеанси й місця та одразу здійснити оплату, після чого отримати QR-квиток. Система спрямована на забезпечення простого та зрозумілої взаємодії між адміністраторами і глядачами: перші керують кінотеатрами, залами, розкладами сеансів, а другі – без зайвих кроків бронюють й оплачують квитки

Основна мета полягає в тому, щоб створити середовище, яке робить процес вибору фільму, бронювання місць і оплати максимально простим та зручним для глядачів, надаючи адміністраторам ефективні інструменти керування сеансами, залами й розкладом. Завдяки QR-квиткам, які автоматизовано генеруються після оплати, миттєвому оновленню статусу бронювання та оплати і гнучким налаштуванням залів, система забезпечує безперервний досвід як для кіномеханіків так і для кіноманів.

Під час створення системи важливо враховувати потреби різних типів користувачів, а також функціональні та нефункціональні вимоги до системи.

Для адміністратора важливим є можливість повного керування системою, що включає в себе керування інформацією про зали, фільми і кінотеатри, налаштування розкладу сеансів, моніторингу стану бронювань і платежів, а також статусу доступу користувачів.

Клієнту система повинна забезпечити зручний та безпечний процес авторизації, перегляд фільми та інформації про них, вибір доступних сеансів, можливість вибору місця, створення бронювання та онлайн-оплати. Окрім того, після здійснення транзакції, користувач має миттєво отримувати електронний квиток.

Функціональні вимоги передбачають, що система повинна надавати можливість швидко створювати обліковий запис та здійснювати вхід. Інтерфейс повинен бути інтуїтивно зрозуміли, щоб користувач міг з

		Петришин А.О.			ІПЗ.КР.Б – 121 – 25 – ЗПП	Арк.
		Годлевський Ю.О.				25
Змн.	Арк.	№ докум.	Підпис	Дата		

легкістю знаходити необхідну йому інформацію. Процес здійснення бронювання повинен бути зручним та швидким, а також супроводжуватись автоматичною генерацією електронного квитка. Адміністраторам ж надається окремий функціонал керування системою.

До нефункціональних вимог входить забезпечення системою стабільної та безперебійної її роботи. Дизайн простий та адаптивний і однаково зручний на будь-якому пристрої. Усі дані, такі як паролі користувачів, повинні зберігатись безпечно. Система повинна підтримувати одночасно велику кількість користувачів і мати достатній об'єм сховища для зберігання даних про кінотеатри, фільми, тощо.

Серверна частина розгортається на машині під керуванням Ubuntu 20.04 або новішої версії для забезпечення стабільності безпеки та гнучкості у налаштуванні середовища. Альтернативним рішенням може слугувати розгортання системи за допомогою Windows Server – операційної системи корпоративного рівня, яка може запропонувати зручний інтерфейс взаємодії і сумісність з багатьма програмними продуктами, в тому числі і для адміністрування.

Для обробки запитів і реалізації бізнес-логіки застосовується середовище Node.js версії 18 або вище разом із фреймворком Express.js. Усі дані зберігаються в MongoDB, що розгортається на окремому сервері та має зв'язок із сервером за допомогою HTTP запитів.

Клієнтська частина працює в будь-якому сучасному веб-браузері (Chrome, Firefox, Edge тощо) з увімкненим JavaScript, що дозволяє коректно відображати адаптивний інтерфейс, виконувати валідацію форм та здійснювати асинхронні запити до серверного API.

Для стабільної роботи системи необхідно забезпечити мінімум 4 ГБ оперативної пам'яті на сервері та щонайменше 25 ГБ вільного місця на SSD-диску для зберігання БД, логів і тимчасових файлів [22;23].

		Петришин А.О.			ІПЗ.КР.Б – 121 – 25 – ЗПП	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		26

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

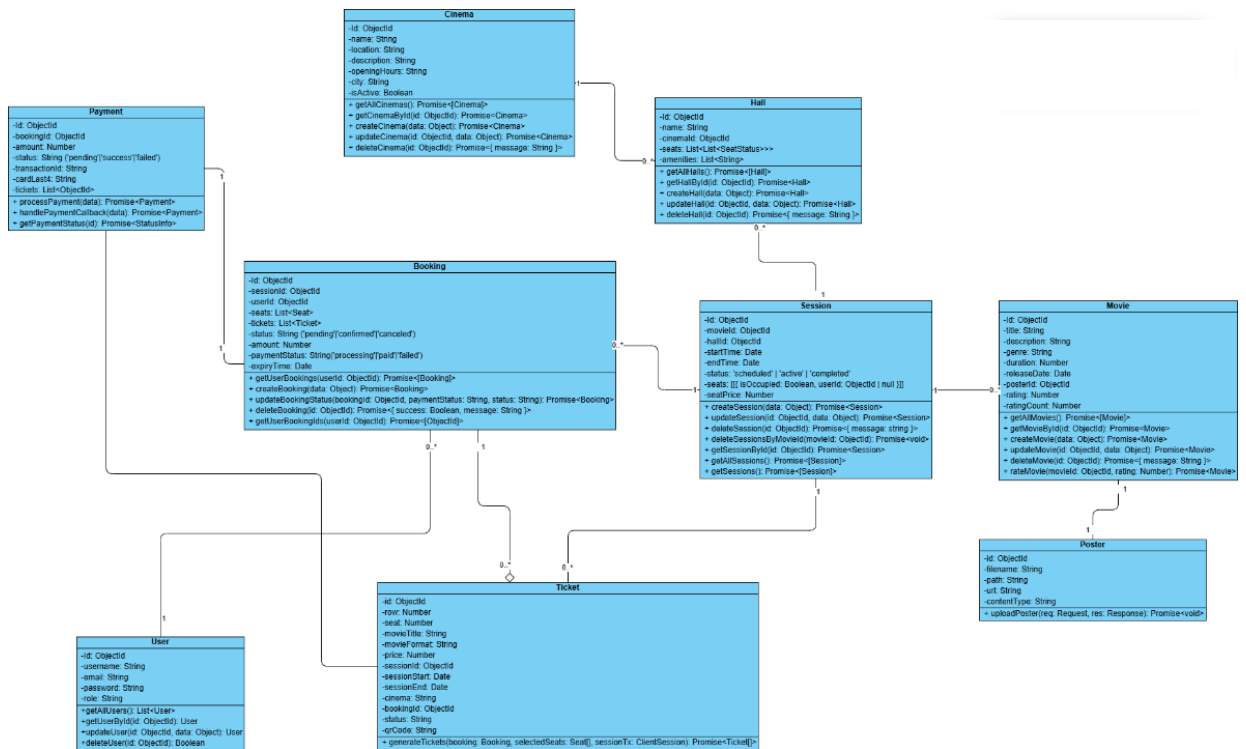


Рис. 2.2. Діаграма класів

Опишемо діаграму класів (рис. 2.2.).

User – клас, що відповідає за облікові записи у системі. У даного класу є операції: register, яка приймає дані користувача, перевіряє їх коректність та створює нового користувача у БД, а login, в свою чергу, звіряє email та пароль з тими, що є в базі та повертає токен авторизації разом з інформацією про користувача. Таким чином, User вирішує хто має доступ до користування системою і видає необхідні права у вигляді токenu.

Booking – клас, який керує переглядом бронювань, створенням та їх редагуванням. За допомогою функції getUserBookings можна отримати бронювання всіх користувачів, в той час як createBooking створює новий запис бронювання для обраних місць та конкретного сеансу та включає в себе перевірку чи вони ще вільні та встановлення таймеру видалення бронювання. При оплаті користувачем, активується метод updateBookingStatus, який відповідає за видалення таймеру та оновлення

статусу броні і оплати, які підтверджують успішність здійсненої транзакції. Для видалення використовується `deleteBooking`.

`Session` – це клас, що відповідає за розклад і параметри показів. Адміністратор може керувати даними сеансів за допомогою наступних функцій: `createSession` – метод, що дозволяє додавати до БД запис сеансу, копіюючи необхідні дані з класу `Hall`, `getAllSessions` – для перегляду всіх доступних сеансів, а для зручного перегляду з додатковими даними про фільм, зал та кінотеатр використовується `getSessions`, `updateSession` слугує для оновлення окремих полів або всього сеансу, а для видалення – `deleteSession`.

`Movie` – місце зберігання даних про фільм. Для отримання переліку стрічок, включаючи і URL постеру, викликається `getAllMovies`, а для отримання детальної інформації про окремий фільм – `getMovieById`. Додавання ж нового фільму здійснюється через `createMovie`, оновлення через `updateMovie`, а видалення – `deleteMovie`, що дозволяє адміністратору керувати каталогом доступних фільмів.

`Poster` – клас, відповідальний за конвертацію зображень у необхідний полегшений формат та завантаження самого файлу і даних про нього у БД. Метод, який виконує описані операції `uploadPoster` перевіряє, чи файл передано, за потреби конвертує його у формат `webp` і зберігає локально на сервері, після чого заносить атрибути (`filename`, `path`, `url`) постеру до бази. Таким чином, завдяки цьому `Movie` може прив'язати до себе зображення для показу користувачам.

`Ticket` – клас електронного квитка, який створюється після успішної оплати і викликом методу `generateTickets`, який приймає сам об'єкт бронювання, обрані місця та інформацію про сеанс, а саме: фільм, зал та кінотеатр), вираховує ціну, генерує QR-код з об'єктом, що містить в собі всі дані квитка, і створює відповідний запис у колекції. За бажанням,

		Петришин А.О.			ІПЗ.КР.Б – 121 – 25 – ЗПП	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		28

користувач може отримати всі свої квитки, викликавши метод `getUserTickets`, яка повертає список на основі бронювань.

`Payment` – клас, який опрацьовує фінансові транзакції. Метод `processPayment` відкриває саму транзакцію в базі та починає перевірку на наявність самого бронювання і після цього викликає `PaymentGateway` (платіжний шлюз) і при успішній оплаті, створює запис у БД, викликає у `Booking` функцію зміну статусу на підтверджений і викликає метод створення квитка у `Ticket` (коли платіжна система надсилає `callback`, працює метод `handlePaymentCallback`, оновлюючи статус оплати та бронювання). Метод `getPaymentStatus` дає можливість в будь-який момент дізнатись успішність списання коштів з додатковою інформацією.

`Hall` – клас, що описує кінозал, і в якому зберігаються назва зали, масив сидінь `seats` і `amenities` (3D, IMAX тощо). Даний клас також містить CRUD методи для взаємодії з ним: для створення – `createHall`, для отримання інформації про зали `getAllHalls` та `getHallById`, для оновлення одного, або багатьох полів – `updateHall`, а для видалення – `deleteHall`.

`Cinema` – клас, що містить інформацію про кінотеатр, включаючи такі поля, як: назва кінотеатру, його адреса, контактні дані, години роботи (початок та кінець робочого дня) і місто, де він розташований. Для створення, редагування, перегляду та видалення використовуються ті ж за сенсом CRUD методи, які є ідентичними до інших класів.

2.2 Розробка бази даних системи

Для збереження файлів та інформації у додатку використовується MongoDB, документоорієнтована NoSQL-СУБД.

База даних системи складається із дев'яти таблиць:

- `users` — зберігає користувачів;
- `movies` — зберігає фільми;
- `posters` — зберігає метадані постерів;
- `cinemas` — зберігає кінотеатри;

		Петришин А.О.			ІПЗ.КР.Б – 121 – 25 – ЗПП	Арк.
		Годлевський Ю.О.				29
Змн.	Арк.	№ докум.	Підпис	Дата		

- halls — зберігає зали;
- sessions — зберігає сеанси показу;
- bookings — зберігає бронювання;
- tickets — зберігає квитки;
- payments — зберігає платежі.

Таблиця 2.9

Опис таблиці User

Поле	Тип	Призначення
_id	ObjectId	Унікальний ідентифікатор документа
username	String, required, unique	Логін користувача
email	String, required, unique	Email користувача
password	String, required	Хеш пароля
role	String, enum	Роль («user» / «admin» / «manager»), за замовчуванням «user»
createdAt	Date	Дата створення запису
updatedAt	Date	Дата останнього оновлення

Таблиця 2.10

Опис таблиці Movie

Поле	Тип	Призначення
_id	ObjectId	Унікальний ідентифікатор документа
title	String, required	Назва фільму
description	String	Опис фільму
genre	String, required	Жанр фільму
duration	Number, required	Тривалість у хвилинах
releaseDate	Date, required	Дата виходу
posterId	ObjectId	Посилання на документ у колекції posters
rating	Number	Рейтинг (1–10)
ratingCount	Number	Кількість користувачів, що оцінили
createdAt	Date	Дата створення запису
updatedAt	Date	Дата останнього оновлення

Таблиця 2.11

Опис таблиці posters

Поле	Тип	Призначення
_id	ObjectId	Унікальний ідентифікатор документа
filename	String, required	Ім'я файлу на сервері
path	String, required	Шлях до файлу у файловій системі
url	String, required	Публічна URL-адреса
contentType	String, required	MIME-тип вмісту (наприклад image/webp)

Таблиця 2.12

Опис таблиці cinemas

Поле	Тип	Призначення
_id	ObjectId	Унікальний ідентифікатор документа
name	String, required	Назва кінотеатру
location	String, required	Адреса чи опис локації
description	String	Опис кінотеатру
contactInfo	String	Контактні дані
openingHours	String	Години роботи
city	String	Місто
isActive	Boolean	Статус активності (за замовчуванням true)
createdAt	Date	Дата створення запису
updatedAt	Date	Дата останнього оновлення

Таблиця 2.13

Опис таблиці halls

Поле	Тип	Призначення
_id	ObjectId	Унікальний ідентифікатор документа
name	String, required	Назва залу
cinemaId	ObjectId, required	Посилання на документ у колекції cinemas
seats	[[Boolean]], required	Матриця місць (ряд × місце), false = вільне
amenities	[String]	Список додаткових зручностей у залі
createdAt	Date	Дата створення запису
updatedAt	Date	Дата останнього оновлення

Таблиця 2.14

Опис таблиці sessions

Поле	Тип	Призначення
_id	ObjectId	Унікальний ідентифікатор документа
movieId	ObjectId, required	Посилання на документ у колекції movies
hallId	ObjectId, required	Посилання на документ у колекції halls
startTime	Date, required	Час початку сеансу
endTime	Date, required	Час завершення сеансу
status	String, enum	Статус (scheduled,active,completed), де-фолт scheduled
seats	[[Boolean]], required	Поточний стан місць (копія початкового плану залу)
seatPrice	Number, required, default 100	Ціна за одне місце
createdAt	Date	Дата створення запису
updatedAt	Date	Дата останнього оновлення

Таблиця 2.15

Опис таблиці bookings

Поле	Тип	Призначення
_id	ObjectId	Унікальний ідентифікатор документа
sessionId	ObjectId, required	Посилання на документ у колекції sessions
userId	ObjectId, required	Посилання на документ у колекції users
seats	[{row: Number, seat: Number}]	Перелік заброньованих місць
tickets	[ObjectId]	Масив посилань на документи у колекції tickets
status	String, enum	Статус бронювання
Amount	Number	Ціна за обрані місця
paymentStatus	String, enum	Статус оплати (processing,paid,failed)
expiryTime	Date	Дата, до якої діє резервування
createdAt	Date	Дата створення бронювання
updatedAt	Date	Дата останнього оновлення

Таблиця 2.16

Опис таблиці tickets

Поле	Тип	Призначення
_id	ObjectId	Унікальний ідентифікатор документа
row	Number, required	Номер ряду
seat	Number, required	Номер місця
movieTitle	String, required	Назва фільму
movieFormat	String, required	Формат фільму
price	Number, required	Ціна квитка
sessionId	ObjectId, required	Посилання на документ у колекції sessions
sessionStart	Date, required	Дата початку сеансу
sessionEnd	Date, required	Дата кінця сеансу
cinema	String, required	Назва кінотеатру
bookingId	ObjectId, required	Посилання на документ у колекції bookings
status	String, enum	Статус квитка (reserved,confirmed,canceled)
qrCode	String, required	QR-код квитка
createdAt	Date	Дата створення квитка
updatedAt	Date	Дата останнього оновлення

Таблиця 2.17

Опис таблиці payments

Поле	Тип	Призначення
_id	ObjectId	Унікальний ідентифікатор документа
paymentId	String, required, unique	Унікальний зовнішній ідентифікатор платежу
bookingId	ObjectId, required	Посилання на документ у колекції bookings
amount	Number, required	Сума платежу
status	String, enum	Статус платежу (pending,success,failed)
transactionId	String, required	Ідентифікатор внутрішньої транзакції

cardLast4	String, required	Останні 4 цифри картки
tickets	[ObjectId]	Масив посилань на документи у колекції tickets
createdAt	Date	Дата створення запису
updatedAt	Date	Дата останнього оновлення

В контексті роботи з MongoDB кожен документ є автономним «пакетом» даних, який зберігається в колекції як BSON-структура. Колекція виступає як таблиця в базі, але без жорсткої схеми, тобто додаткові поля можна додавати динамічно, а структура документів може відрізнятися між записами [24]. Коли створюється новий документ mongoose пакує його в BSON, підписує поточним часом у полях createdAt і updatedAt та відправляє на первинний вузол кластера [25].

Цей вузол фіксує операцію в локальному журналі та поширює запис іншим вузлам (вторинним) завдяки реплікації. Якщо шардинг налаштований, то документ потрапляє до певного шарду ключа (наприклад userId), що дозволяє горизонтально масштабувати навантаження.

Усередині кластеру Mongo кожен блок-журнал містить послідовність операцій вставки, оновлення або видалення документів/ Вторинні вузли застосовують ці операції у своїй копії даних, гарантуючи, що всі репліки синхронізовані. Якщо первинний вузол виходить з ладу, один із вторинних автоматично обирається новим первинним, і записи не втрачаються [26;27].

Для звернення до даних документів використовуються CRUD-методи Mongoose (find, findById, findOneAndUpdate) [28]. Ці методи формують відповідний запит до кластера та отримують результат із урахуванням налаштувань читання – з перминного вузла або з готовиз до читання вторинних.

Таким чином MongoDB поєднує гнучкість безсхемної моделі з високою доступністю та масштабованістю, а Mongoose забезпечує контроль структури й автоматичне управління часовими мітками.

2.3 Проектування та реалізація алгоритмів роботи системи

Додаток є простим у використанні, має основні функції, які забезпечують інтуїтивну та комфортну взаємодію користувачів із додатком. Алгоритм роботи з веб-додатком зображено на рисунку 2.3, де надано опис та пояснення кожному етапу.

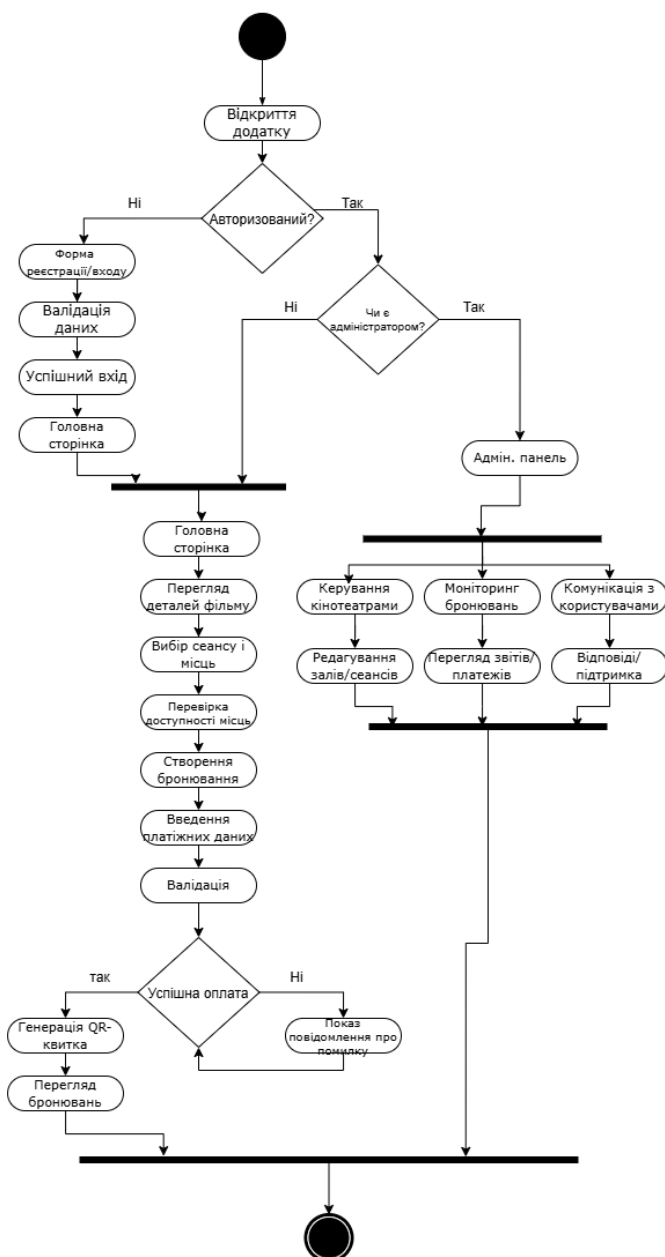


Рис. 2.3. Діаграма активності

Після відкриття додатку система перевіряє чи користувач авторизований і, якщо ні, то відкривається форма реєстрації/входу, після введення даних відбувається валідація даних і користувач потрапляє на головну сторінку. У випадку, якщо користувач авторизований, система перевіряє роль користувача, на наявність ролі адміністратора.

У разі авторизації адміністратора – адміністратор переходить до адмін. панелі, де доступні такі функції як: керування кінотеатрами, з можливістю редагувати зали та сеанси, моніторинг бронювань та перегляд звітів по платежам, комунікація з користувачем, відповіді на звернення й підтримку.

У разі авторизації звичайного користувача або після входу – користувач потрапляє на головне сторінку, де може переглянути наявні фільми та подивитись деталі фільму, обрати сеанс і місце. Після перевірки доступності обраного місця – створюється бронювання та відкривається сторінка введення платіжних даних і, якщо валідація успішна, то оплата вважається успішною, в наслідок чого генерується QR-квиток і з’являється можливість перегляду своїх бронювань. Однак, якщо оплата не пройшла – система повідомляє про помилку і дає можливість ввести дані коректно.

2.4 Реалізація функціоналу системи бронювання квитків

Розглянемо основні етапи реалізації функціоналу бронювання квитків у кінотеатрі. Для цього користувач повинен спочатку зареєструвати новий обліковий запис, після чого отримає змогу переглядати доступні сеанси фільмів й обирати зручне для себе місце в залі. Процес вибору місць є першим важливим кроком у послідовності бронювання. Саме він дозволяє фіксувати обрані користувачем позиції на схемі залу перед здійсненням тимчасового бронювання, подальшою оплатою та підтвердженням.

		Петришин А.О.			ІПЗ.КР.Б – 121 – 25 – ЗПП	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		36

```

socket.on('toggle-seat', async ({ sessionId, row, seat }) => {
  const seatKey = `${row}-${seat}`;
  let locks = seatLocks.get(sessionId) || {};
  const Session = mongoose.model('Session');
  const session = await Session.findById(sessionId);
  if (!session) {
    socket.emit('seat-error', 'Сеанс не знайдено');
    return;
  }

  if (session.seats[row][seat].isOccupied) {
    socket.emit('seat-error', 'Місце вже зайняте');
    return;
  }

  const now = Date.now();
  Object.keys(locks).forEach(key => {
    if (now - locks[key].timestamp > 300000) {
      delete locks[key];
      io.to(sessionId).emit('seat-unlocked', {
        row: Number(key.split('-')[0]),
        seat: Number(key.split('-')[1])
      });
    }
  });
});

```

Рис. 2.4. Вибір місць для сеансу

Коли користувач обрав місце та натиснув відповідну кнопку – формується тимчасове бронювання з часом закінчення дії. На початку сутність має статус pending до моменту підтвердження оплати (рис. 2.5.).

```

const createBooking = async (req, res) => {
  try {
    const { sessionId, seats } = req.body;
    if (!mongoose.Types.ObjectId.isValid(sessionId)) { ...
    }
    const session = await Session.findById(sessionId);
    if (!session) { ...
    }

    const isBusy = seats.some(({ row, seat }) => session.seats[row][seat]?.isOccupied);
    if (isBusy) { ...
    }

    seats.forEach(({ row, seat }) => {
      session.seats[row][seat].isOccupied = true;
      session.seats[row][seat].userId = req.user.userId;
    });
    await session.save();

    const expiryTime = new Date(Date.now() + 5 * 60 * 1000);

    const booking = await Booking.create({
      sessionId,
      userId: req.user.userId,
      seats,
      status: 'pending',
      paymentStatus: 'processing',
      amount: seats.length * session.seatPrice,
      expiryTime
    });

    res.status(201).json({ success: true, data: booking, bookingId: booking._id });
  } catch (err) {
    console.error('[createBooking] Error:', err);
    res.status(500).json({ success: false, error: err.message });
  }
};

```

Рис. 2.5. Створення бронювання

		Петришин А.О.			ІПЗ.КР.Б – 121 – 25 – ЗПП	Арк.
		Годлевський Ю.О.				37
Змн.	Арк.	№ докум.	Підпис	Дата		

Як тільки створилось бронювання – користувач потрапляє на сторінку оплати, де вводить банківські дані для здійснення онлайн транзакції. При правильному форматі введених відвідувачем даних і наявності бронювання, створюється запис про оплату. (див. рис. 2.6).

```
const processPayment = async (req, res) => {
  const sessionTx = await mongoose.startSession();
  sessionTx.startTransaction();

  try {
    const { bookingId, amount, cardDetails, selectedSeats } = req.body;

    const booking = await Booking.findById(bookingId).session(sessionTx);
    if (!booking) throw new Error('Бронювання не знайдено');

    const session = await validateSessionForPayment(booking.sessionId, sessionTx);

    const gateway = new PaymentGateway();
    const paymentResp = await gateway.processPayment(cardDetails, amount);
    if (!paymentResp.success) throw new Error(paymentResp.message);

    const payment = new Payment({
      bookingId: booking._id,
      amount,
      status: 'success',
      paymentMethod: 'card',
      paymentId: paymentResp.paymentId,
      paymentDate: new Date(),
      cardLast4: cardDetails.number.slice(-4),
      transactionId: paymentResp.paymentId,
      seats: selectedSeats
    });
    await payment.save({ session: sessionTx });
  }
}
```

Рис. 2.6. Опрацювання оплати

Після успішно здійсненого платежу, змінюється статус оплати бронювання (PaymentStatus) на paid та статус безпосередньо бронювання. Також прибирається поле дати закінчення дії бронювання (expiryTime) (рис. 2.7).

```
const updateBookingStatus = async (bookingId, status, paymentStatus) => {
  await Booking.findByIdAndUpdate(bookingId, {
    status,
    paymentStatus
  });
};
```

Рис. 2.7. Оновлення статусу бронювання

Коли транзакція вважається успішно завершеною, а бронювання має статус «сплачено», генерується електронний квиток, який має відображати користувачу інформацію про сеанс, фільм, зручності, кінотеатр, дату сеансу, місце та ціну. До наданої інформації створюється QR-код, в який записується об'єкт з тими ж даними, які має квиток (рис. 2.8.).

```

const generateTickets = async (booking, selectedSeats, sessionTx) => {
  console.log('generateTickets called');
  try {
    const session = await Session.findById(booking.sessionId)
      .populate('movieId')
      .populate({ path: 'hallId', populate: { path: 'cinemaId' } })
      .session(sessionTx);

    if (!session) {
      console.log('Session not found!');
      throw new Error('Session not found for ticket generation');
    }

    const hall = session.hallId;
    const movie = session.movieId;
    const cinema = hall.cinemaId;
    const format = hall.amenities;
    const startTime = session.startTime;
    const endTime = session.endTime;
    const seatPrice = session.seatPrice;

    const ticketsData = await Promise.all(selectedSeats.map(async seat => {
      const displayRow = seat.row + 1;
      const displaySeat = seat.seat + 1;
      const ticketUuid = uuidv4();
      const qrPayload = {
        movie: movie.title,
        format: format,
        row: displayRow,
        seat: displaySeat,
        sessionId: session._id.toString(),
        bookingId: booking._id.toString(),
        ticketId: ticketUuid
      };
      const qrCodeDataURL = await QRCode.toDataURL(JSON.stringify(qrPayload));
      return {
        row: displayRow,
        seat: displaySeat,
        movieTitle: movie.title,
        movieFormat: format,
        price: seatPrice,
        sessionId: session._id,
        sessionStart: startTime,
        sessionEnd: endTime,
        cinema: cinema.name,
        bookingId: booking._id,
        status: 'reserved',
        qrCode: qrCodeDataURL
      };
    }));
  }
});

```

Рис. 2.8. Генерація квитка та QR-коду

В результаті виконання всіх операцій користувач може переглядати свої квитки на окремій сторінці. Усі наступні дії (перевірка квитка, відображення у списку на вході в зал тощо) реалізовані за допомогою унікальних ідентифікаторів.

		Петришин А.О.			ІПЗ.КР.Б – 121 – 25 – ЗПП	Арк.
		Годлевський Ю.О.				39
Змн.	Арк.	№ докум.	Підпис	Дата		

Підсумовуючи, створено діаграму компонентів додатку бронювання квитків у кіно (рис. 2.9.) і визначено основні компоненти.

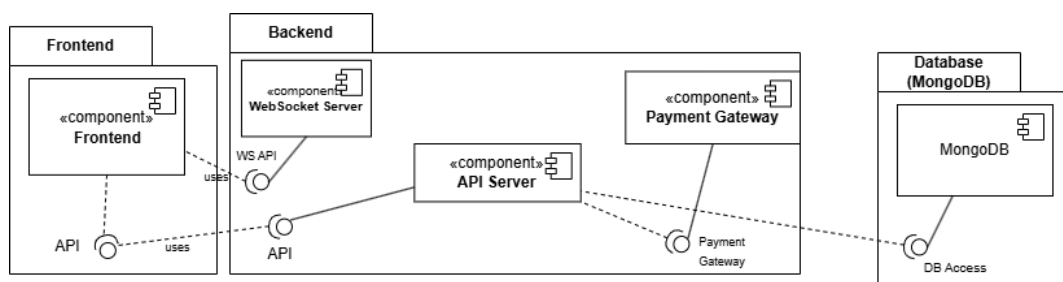


Рис. 2.9. Діаграма компонентів

Згідно з розробленою діаграмою компонентів система складається з декількох ключових компонентів, а саме: клієнтська частина, серверна частина, в яку входять модулі для роботи з WebSocket, API і платіжна система (Payment Gateway), та БД MongoDB.

Клієнтська частина відповідає за взаємодію з користувачем і забезпечує відображення інформації про фільми, зали, зали тощо, а також здійснювати бронювання та оплату. Всі необхідні актуальні дані вона отримує за допомогою API запитів до серверу, проте для здійснення бронювання окремо тримає зв'язок через WebSocket-з'єднання, що дає змогу обирати місце в реальному часі з іншими користувачами.

Серверна частина ж обробляє всі запити з клієнту за допомогою API-серверу, який виконує безпосередньо бізнес логіку та тримає зв'язок з іншими частинами системи. Саме через API-сервер здійснюється взаємодія з БД, звідки сервер отримує необхідну інформацію, змінює її та видаляє за потреби. Додатково підключено WebSocket-сервер, який організовує двосторонній постійний зв'язок між клієнтом й сервером.

Також важливою частиною є платіжний шлюз – Payment Gateway, який являє собою локальний внутрішній модуль. Він дозволяє протестувати повний цикл бронювання та оплати без інтеграцій із зовнішніми платіжними сервісами. У рамках серверної логіки компонент викликається під час обробки запитів на оплату отримує дані картки і суму оплати з фронтенду і викликає опрацювання оплати, який знаходиться в окремому контролері, і,

якщо при опрацюванні платіжних даних не виникло помилок, то штучно повертає успішну відповідь з унікальним ідентифікатором транзакції (paymentId).

База даних реалізована на основі хмарного сервісу MongoDB і зберігає на власних серверах всі об'єкти системи. Компонент діаграми MongoDB взаємодіє з бекендом за допомогою API-серверу.

В підсумку, можна сказати, що представлена діаграма компонентів демонструє розмежування відповідальності на основі модульної системи. Зокрема, використання емуляції платіжного шлюзу дозволяє виконувати тестування та розробку у безпечному середовищі, яке не залежить від зовнішніх сервісів, що забезпечує гнучкість додатку з можливістю розширення в подальшій розробці і інтеграції із справжньою платіжною системою.

Висновки до другого розділу

В процесі проектування додатку бронювання квитків у кінотеатрах було проаналізовано всі можливі сценарії використання додатку відповідно до його функціональних вимог. На основі цього було спроектовано логічну структуру системи та визначено, як її частини взаємодіють між собою.

Було створено схему бази даних, яка забезпечує збереження основної інформації – профілів користувачів, переліку фільмів, розкладу сеансів, статусів бронювання, транзакцій оплати та параметрів квитків.

Було описано алгоритм роботи з додатком та описані послідовні пункти варіантів його використання.

Окрім того наведений опис основного функціоналу додатку, який охоплює вибір фільму та сеансу, резервування доступного місця, створення бронювання, здійснення оплати, зміну статусів бронювання після успішної транзакції та формування електронного квитка.

		Петришин А.О.			ІПЗ.КР.Б – 121 – 25 – ЗПП	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		41

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З СИСТЕМОЮ БРОНЮВАННЯ КВИТКІВ

3.1 Порядок встановлення та налаштування параметрів додатку

Для встановлення і налаштування додатку необхідно провести поетапне розгортання, яке включає в себе розгортання основних компонентів системи і забезпечення коректної взаємодії їх між собою. Для візуального відображення архітектури було розроблено діаграму розгортання (рис 3.1)

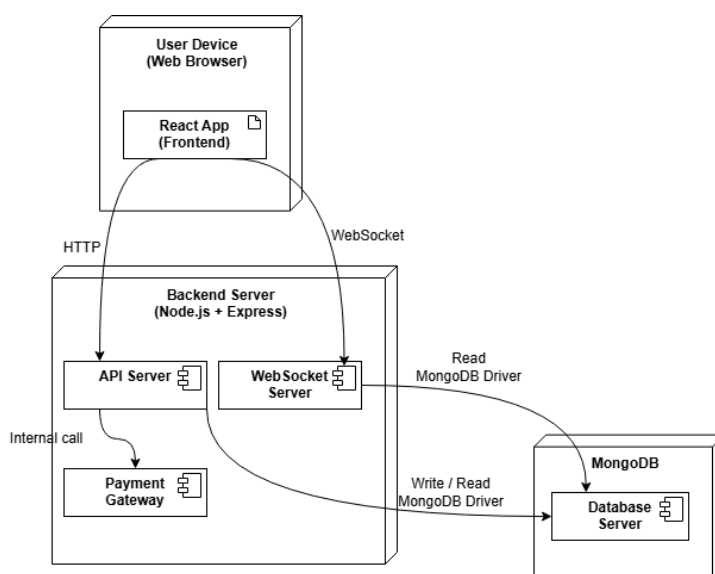


Рис. 3.1. Діаграма розгортання додатку для бронювання

Система складається з трьох компонентів, а саме: клієнтського пристрою, серверу додатку та серверу базу даних. Кожен виконує свій ряд специфічних завдань у роботі додатку і взаємодіє з іншими компонентами.

Клієнтська частина, що виконує React-застосунок, працює у веб-браузері користувача, який взаємодіє з його допомогою із серверною частиною, використовуючи для цього HTTP-запити для отримання даних про фільми, сеанси, квитки тощо та з'єднання WebSocket для можливості обирати місця у реальному часі.

Серверна частина, яка побудована на основі Node.js та Express.js, включає в себе ряд ключових модулів: API Server, який відповідальний за обробку запитів зі сторони фронтенду, реалізує бізнес-логіку і взаємодіє з

БД; Payment Gateway, який симулює роботу платіжної системи, у відповідь на запит з API Server, він передає згенеровані дані (ідентифікатор транзакції, останні 4 цифри картки та ім'я платника), після чого API Server зберігає дані оплати, включаючи інформацію, що була надана з платіжного шлюзу; WebSocket Server, який забезпечує зв'язок між сервером та клієнтом у реальному часі для оновлення статусів місць у кінозалах.

MongoDB Server – окремий сервер бази даних, взаємодія з яким здійснюється через відповідні запити з серверного боку (API Server, WebSocket Server). Інформація, що міститься в базі, включає дані про користувачів, бронювання, сеанси, платежі та інші сутності.

Взаємодія компонентів включає такі протоколи:

- HTTP-запити (між клієнтом та API частиною серверу);
- WebSocket-з'єднання (між клієнтом та WebSocket частиною);
- Внутрішні виклики методів (Internal call між платіжним шлюзом та API Server);
- MongoDB Driver (між БД та API/WebSocket).

Отже, для правильної та повноцінної роботи системи бронювання квитків, необхідно розгорнути всі основні компоненти: сервер додатку, веб-додаток та базу даних, з урахуванням налаштування їх взаємодії. Так як структура є модульною, це полегшує супровід і масштабування, окрім того гарантує гнучкість всієї системи.

3.2 Структура інтерфейсу додатку для бронювання квитків у кіно

Інтерфейс додатку організований з акцентом на забезпечення зручності користувача у виконанні всіх необхідних йому дій – від перегляду списку фільмів на афіші до успішної здійсненої транзакції. Сторінки системи пов'язані між собою та дозволяють переходити послідовно до наступної.

На головній сторінці відображається перелік кінофільмів, які наявні в базі даних системи, кожен з них представлений у вигляді картки, на якій присутні фільму та назва. Для збільшення комфорту відвідувачів реалізована фільтрація за жанром і роками випуску та пошукове поле (рис 3.2).

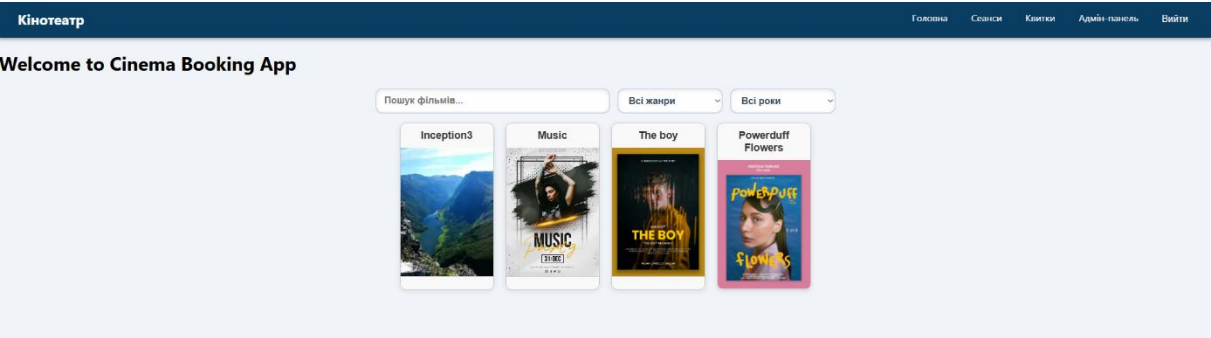


Рис. 3.2. Головна сторінка додатку

Користувач може перейти на сторінку з детальною інформацією про фільм, клацнувши на одну із запропонованих на головній сторінці карток, де міститься повну інформацію: опис, назву, рік випуску, жанр, тривалість у хвилинух, рейтинг та окреме поле для оцінки фільму з кількістю тих користувачів, які вже залишили свій голос. це продемонстровано на рис. 3.3.

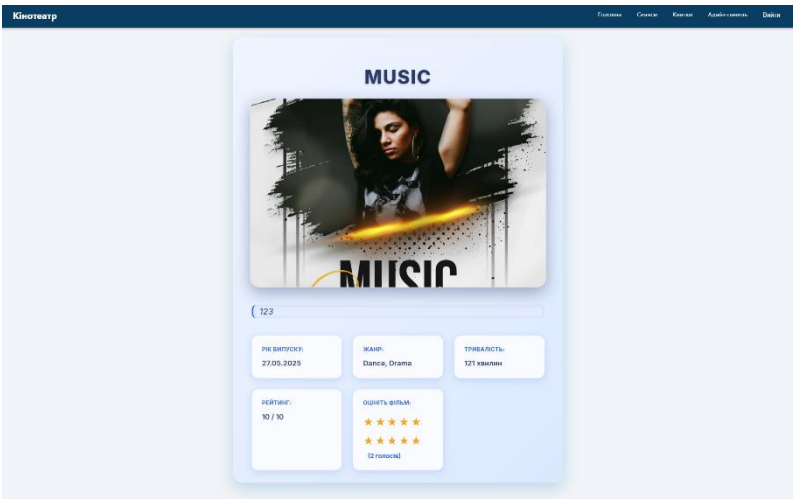


Рис. 3.3. Сторінка фільму

Після перегляду всіх бажаних фільмів на афіші користувач має можливість перейти на сторінку сеансів. На ній у вигляді компактних карток відображається інформація про самі сеанси і включають в себе супутню інформацію, таку як: назва фільму, жанр, тривалість, назву кінотеатру та

адресу, використовувані технології та зручності кінозалу, години роботи кінотеатру, початок і закінчення сеансу, які вираховуються на основі тривалості фільму у хвиликах, а також кількість вільних місць. Окрім того, на даній сторінці присутня фільтрація за кінотеатрами та фільмом, за допомогою чого користувач зможе швидко знайти сеанс за потрібними параметрами (див. рис. 3.4).

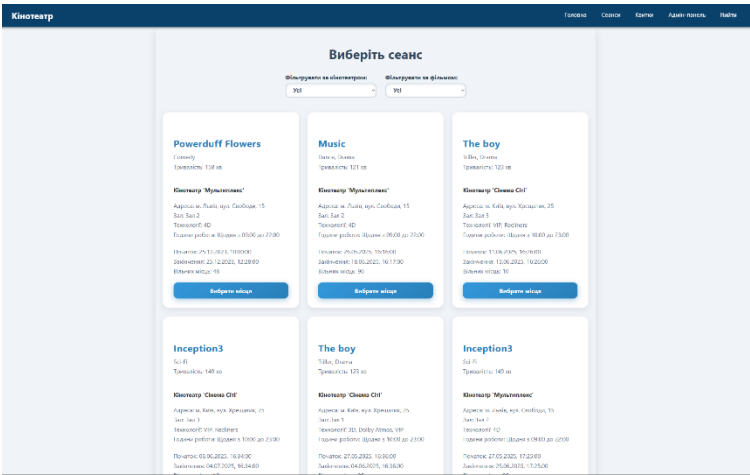


Рис. 3.4. Сторінка сеансів

Однак, при натисканні на кнопку переходу вибору місць, система автоматично перевірить авторизацію користувача та перенаправить його на сторінку «Увійти», де необхідно ввести у відповідні поля форми електронну адресу та пароль, які були вказані під час реєстрації та натиснути відповідну кнопку «Увійти». За відсутності облікового запису відвідувач може скористатись посиланням під формою та перейти на сторінку реєстрації. Зовнішній вигляд сторінки зображений на рисунку 3.5.

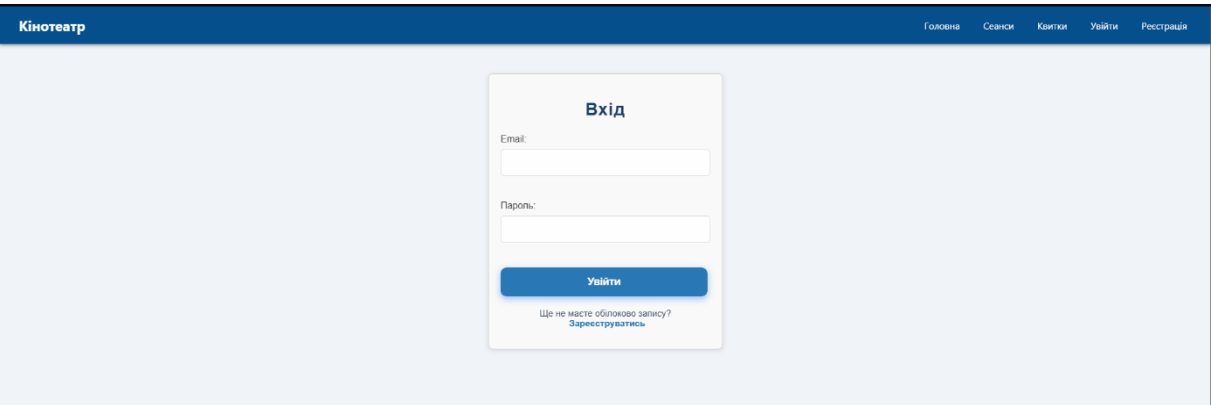


Рис. 3.5. Сторінка входу

Сторінка для додавання нового користувача у систему має ідентичний вигляд, за винятком необхідності ввести ім'я користувача у додаткове поле реєстраційної форми. Також під всіма полями для введення присутнє посилання на сторінку входу, якщо вже є наявний обліковий запис у системі (рис. 3.6).

Рис. 3.6. Сторінка реєстрації

Після здійснення успішної авторизації, користувач отримає змогу як ставити рейтинг фільму, так і вибору місця. На сторінці обраного сеансу представлено зовнішній схематичний вигляд кінотеатру згідно із попередньо заданими параметрами (кількість рядів та місць в рядах). Для зручності додано інформацію про сеанс, яка містить назву фільму, дату, початок сеансу та ціну квитку. Під час вибору місця, знизу відображається кількість обраних місць та кнопка «Забронювати», яка автоматично підраховує загальну суму оплати квитків. Стан місць відображається в залежності від під'єднаних користувачів, тобто користувач, який вибрав місце змінить його статус на «тимчасово зайняте», про що свідчатиме зелений колір на обраному місці, для іншого ж користувача, який обирає в тому ж сеансі, це місце позначатиметься жовтим кольором та відповідно зайняти його вже не вийде. Також, якщо місце заброньовано і тимчасово заброньовано, то місце стає недоступним і позначається сірим. Зовнішній вигляд сторінки без змін статусу за їх зміною продемонстровано на рисунках 3.7 та 3.8 відповідно.

		Петришин А.О.			ІПЗ.КР.Б – 121 – 25 – ЗПП	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		46

Рис. 3.7. Сторінка вибору місць (змін статусу)

Рис. 3.8. Сторінка вибору місць (зі змінами статусу)

Коли користувач забронював місце, система послідовно відправляє його на сторінку здійснення оплати, де відображаються всі ключові деталі замовлення, такі як кількість місць та сума сплати у встановленій заздалегідь валюті (гривня), назва фільму, дата та час здійснення операції оплати. Для підвищення комфорту та зрозумілості, над формою введення платіжних даних знаходиться візуальне відображення картки платника, куди автоматично переносяться з полів введення такі дані як: ім'я власника картки, номер картки, термін дії та CVV-код, при введенні останнього поля картка перевертається, імітуючи зовнішній її вигляд з іншої сторони. Зовнішній вигляд сторінки із вже введеними платіжними даними продемонстрований на рис. 3.9.

Оплата квитків

The boy

11 червня 2025 р. о 16:26

Деталі замовлення

Кількість місць: 1

До сплати: 120 ₴

Дані картки

Ім'я власника картки:

Номер картки:

Термін дії:

CVV/CVC:

[Підтвердити оплату](#)

Рис. 3.9. Сторінка оплати

За умови коректного введення платіжної інформації, наступною сторінкою стане та, що підтверджує факт успішно проведеної транзакції, з відображенням місць, які були куплені, назви фільму, датою та часом сеансу, загальною сумою сплати та пропозицією перейти на головну сторінку додатку (див. рис. 3.10).

Оплата пройшла успішно

Фільм: The boy

Дата та час: 11.06.2025, 16:26:00

Обрані місця: 1-1

Загальна сума: 120 грн.

[На головну](#)

Рис. 3.10. Сторінка успішної оплати

Так як квитки вже згенерувались у процесі опрацювання оплати, користувач може самостійно обрати сторінку «Квитки» у навігаційному меню, де у зручному вигляді буде представлений список квитків, які були

придбані з детальною інформацією про фільм, зручностями залу, вказаними назвою кінотеатру та датою і часом сеансу, номерами ряду та місця, ціною самого сплаченого квитка і статусу (рис. 3.11). Також до цієї інформації додається QR-код, який можна побачити одразу на квитку, або збільшити для більш зручного його перегляду та представлення персоналу кінотеатру (рис. 3.12).

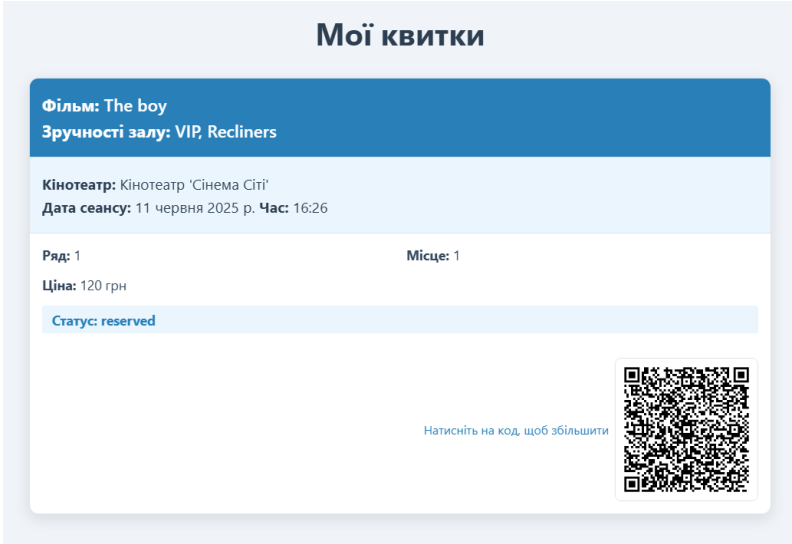


Рис. 3.11. Сторінка «Квитки»

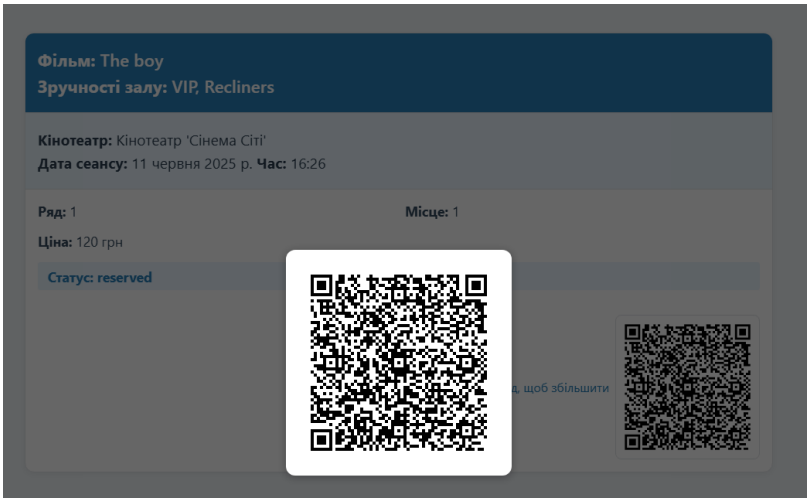


Рис. 3.12. Сторінка «Квитки» з QR-кодом

Окрім стандартних операцій, персонал кінотеатру, який має доступ до керування даними системи, після входу, можуть зайти до адміністративної панелі, пункт переходу на яку з’являється у навігаційному меню. На сторінці адміністрування можна перемикатись між різними вкладками, а саме: фільми, сеанси, зали, кінотеатри, користувачі, бронювання. Кожна з вкладок має необхідну ключову інформацію, перенесення даних до полів у

разі редагування та інтуїтивно зрозумілі пункти керування (створити, редагувати, видалити). На рис. 3.13 продемонстровано вигляд однієї із

Назва	Жанр	Тривалість	Дата	Постер	Дії
Inception3	Sci-Fi	149	16.07.2010		Редагувати Видалити
Music	Dance, Drama	121	27.05.2025		Редагувати Видалити
The boy	Triller, Drama	123	20.05.2025		Редагувати Видалити
Powerduff Flowers	Comedy	158	27.05.2025		Редагувати Видалити

ключових вкладок даної сторінки

Рис. 3.13. Адміністративна панель

3.3 Тестування роботи програмного продукту

В процесі розробки ПЗ виникає необхідність у виявленні явних та неочевидних дефектів, а також в їх усуненні і, для того, щоб забезпечити надійність та якість додатку, проведено тестування за допомогою локального інструменту Postman, який дозволяє здійснювати тестування програмного продукту, здійснюючи API-запити на сервер. Таким чином, можна здійснити аналіз роботи системи, імітуючи реальні сценарії користування без необхідності у розгортанні у відкритому доступі.

Postman – популярний інструмент для тестування RESTful API, що дозволяє імітувати роботу клієнту й надсилати HTTP-запити до серверу, перевіряти відповіді та відстежувати помилки за їх безпосередньої наявності [29]. За допомогою зрозумілого графічного інтерфейсу можна легко формувати запити, змінювати параметри та переглядати відповіді, які надходять з сервера.

Для здійснення запиту варто враховувати перевірки на роль користувача (з токена аутентифікації) і дозволяє чи забороняє виконання певних дій.

Роль admin має повний доступ до операцій, таких як create, read, update, delete, update-user, в той час як роль manager не має права здійснювати видалення даних та змінювати права користувачів (update-user). Інші ролі або користувачі неавторизовані отримують відмову у доступі з кодом 401 або 403.

На рисунку 3.14 продемонстровано виконання запиту до локального серверу з метою входу до системи бронювання. Для цього необхідно надати такі дані користувача: електронну адресу (email) та пароль (password), вказати заголовок, що ми передаємо JSON (текстовий формат для зберігання і обміну даними в універсальному вигляді), тобто вказуємо тип контенту (Content-Type) [30]. Як результат отримуємо відповідь, яка включає в себе інформацію про користувача із згенерованим токеном.

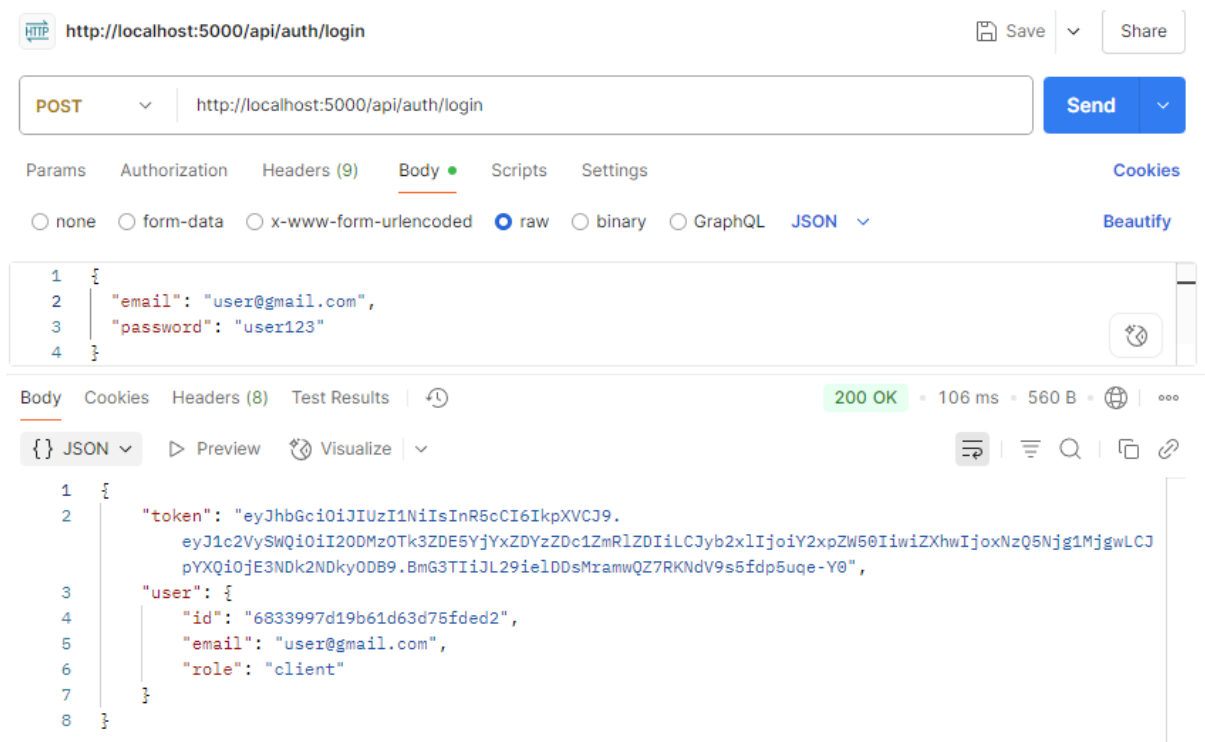


Рис. 3.14. Тестування входу

Також варто перевірити реєстраційні дії, для цього необхідно дещо змінити дані, а саме додати до вже існуючих електронної пошти та паролю ім'я користувача, і змінити шлях запиту. Відповіддю сервера стане статус успішності, повідомлення та інформація про новоствореного користувача. На рисунку 3.15 візуально зображено результат HTTP-запиту.

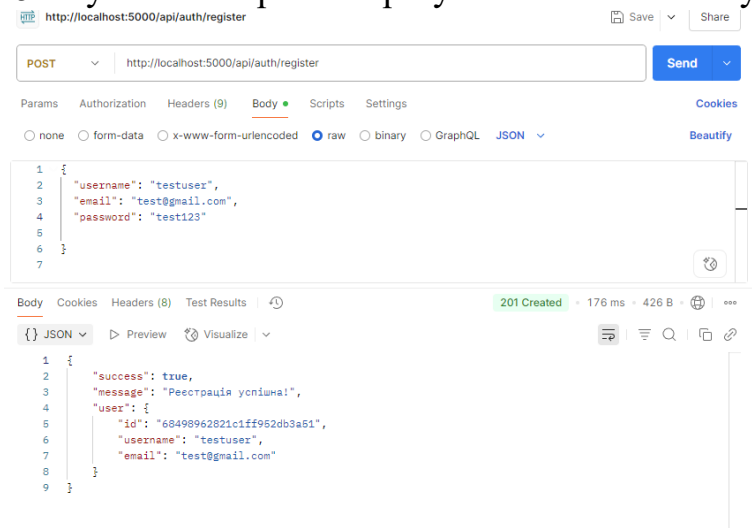


Рис. 3.15. Тестування реєстрації

Також необхідно протестувати ряд запитів на отримання будь-яких даних. Для здійснення тестування таких запитів будемо отримувати дані про фільми та кінотеатри (рис. 3.16 та 3.17 відповідно). Метод запиту буде GET (отримання) без супутніх додаткових параметрів та заголовків. Як результат отримаємо JSON документ з ключовою інформацією про фільми та кінотеатри.

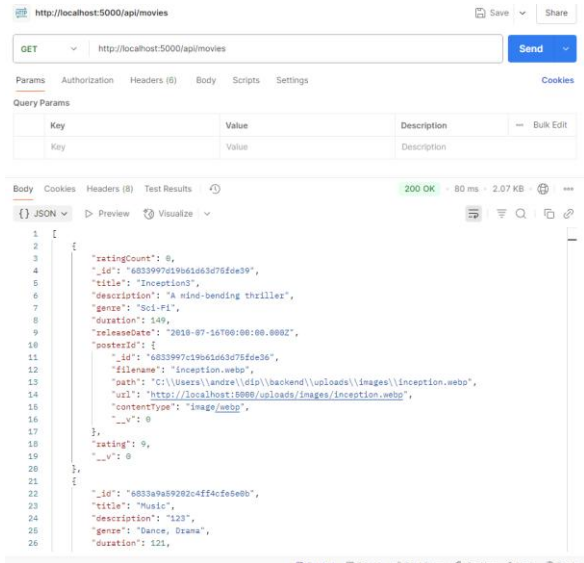


Рис. 3.16. Тестування отримання фільмів

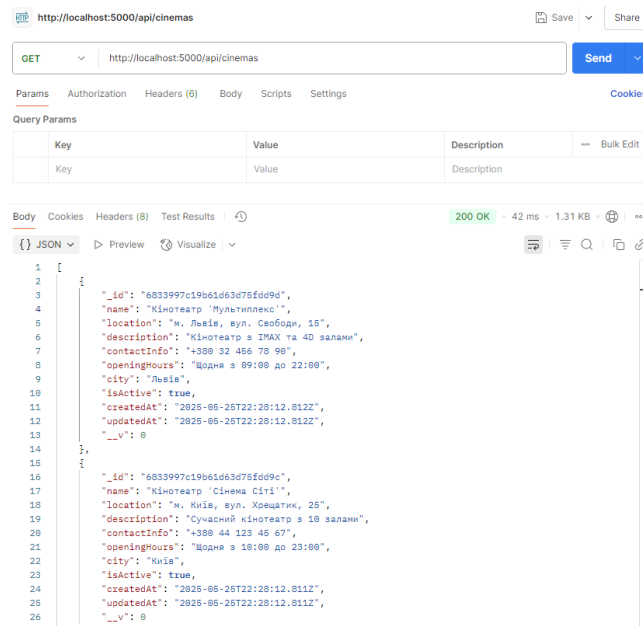


Рис. 3.17. Тестування отримання кінотеатрів

Окрім отримання всього списку даних конкретної категорії, важливо й отримувати детальну інформацію про один з об'єктів списку задля економії часу та зниження навантаження на сервер. Для здійснення вищеописаного запиту використаємо шлях до сеансів системи і додаємо ідентифікатор об'єкту та отримаємо тільки один конкретний екземпляр сеансу із супутньою інформацією про зал кінотеатру, місця з детальним описом і статусом (див. рис. 3.18).

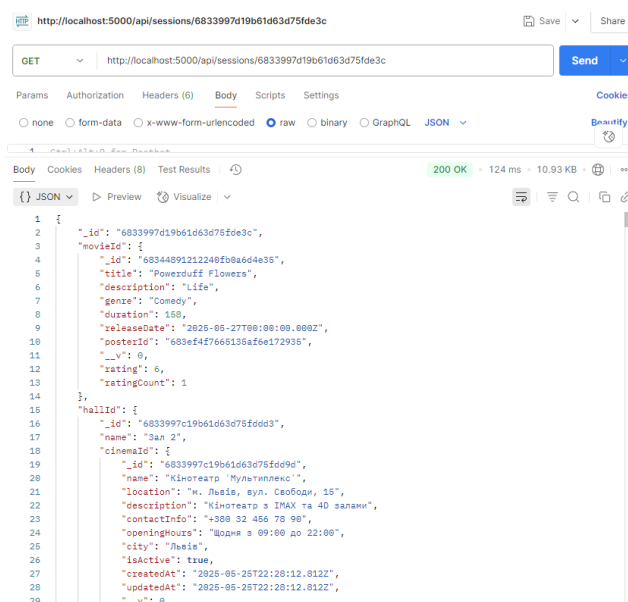


Рис. 3.18. Тестування отримання сеансів за ідентифікатором

Варто також перевірити можливість змінювати інформацію одного з об'єктів, наприклад кінотеатрів. Для цього потрібно також в шляху вказувати ідентифікатор екземпляру, додати необхідні для серверу заголовки, а саме: тип контенту та авторизацію. В тіло запиту додаємо інформацію до необхідного ключа, яку хочем змінити. Відповіддю сервера буде JSON об'єкту з назвою, ідентифікатором та іншою інформацією. Наглядно процес та результат HTTP-запиту зображено на рис. 3.19.

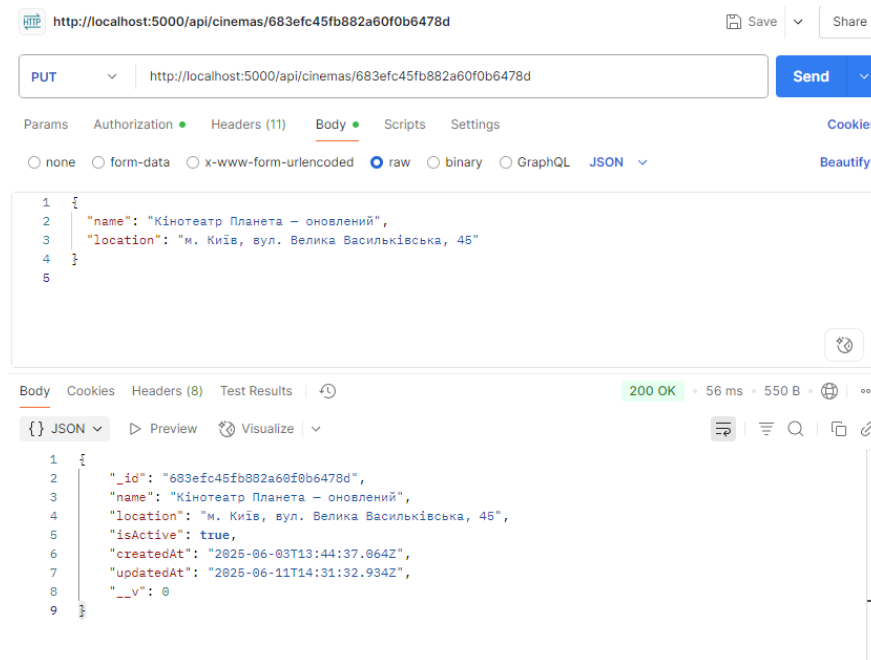


Рис. 3.19. Тестування оновлення даних про кінотеатр

Іноді виникає необхідність у видаленні непотрібних або застарілих даних, для чого також важливим є тестування функціоналу за допомогою HTTP-запитом типу DELETE. Скористаємось новим тестовим створеним користувачем та видалимо його. Спочатку проведемо видалення під роллю manager, для цього в пункти «Авторизація» змінимо токен. В такому випадку результатом стане код 403, який свідчить про заборону на здійснення дії (рис 3.20). Далі проведемо видалення за допомогою токена адміністратора. Результатом такого запиту стане успішне видалення користувача із системи, що зображено на рисунку 3.21.

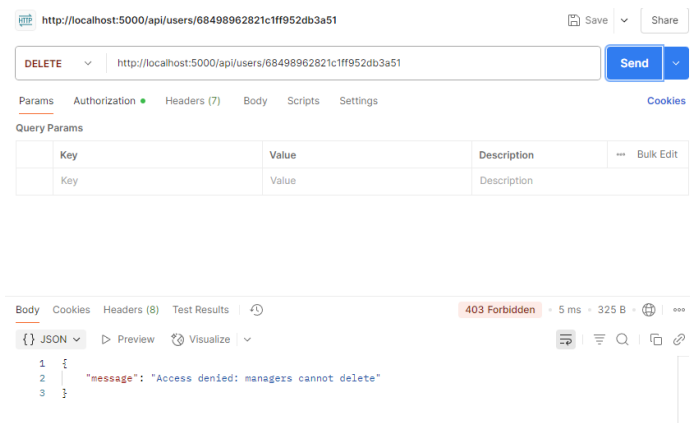


Рис. 3.20. Тестування видалення з обмеженими правами

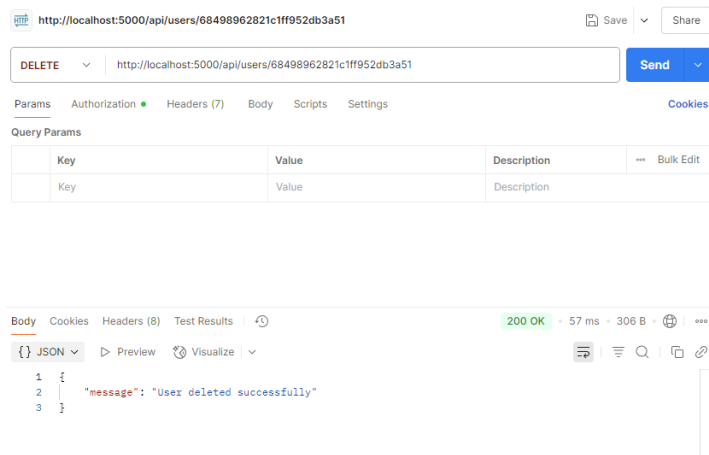


Рис. 3.21. Тестування видалення з повними правами

Висновки до третього розділу

В даному розділі була надана схема розгортання додатку бронювання з детальним описом кожного із компонентів системи. Окрім цього, розглянуто та продемонстровано взаємодію елементів між собою, що дозволило зрозуміти архітектуру веб-додатку.

Було надано опис інтерфейсу програмного забезпечення, розглянуто всі функціональні можливості розробленого продукту, залежно від різноманітних потреб різних користувачів. Для кожного типу користувача представлено свій доступний набір функцій.

Також проведено тестування з використанням Postman для перевірки справності роботи API додатку. Враховуючи різноманітні зміни в тілі та інших параметрах запитів та результати проведення тестування, можна зробити висновок в справності функціонування ендпоінтів додатку.

ВИСНОВКИ

Під час виконання випускної кваліфікаційної роботи бакалавра було спроектовано та реалізовано веб-додаток для здійснення онлайн-бронювання квитків у кіно. Початковий етап включав в себе вивчення предметної області та порівняльний аналіз готових аналогічних рішень, на основі чого сформовано уявлення про структуру програмного продукту та виявлено актуальні вимоги до нього.

На основі результатів дослідження визначено основні вимоги до функціоналу, структури програмного продукту та взаємодії користувачів із системою, що передбачало формування технічних та користувацьких вимог.

Проектування додатку для бронювання включало моделювання повноцінної модульної архітектури та бази даних, беручи до уваги можливі сценарії використання програмного продукту та взаємодію ключових компонентів між собою.

Надано схему розгортання веб-додатку та описано взаємодію його компонентів. Наведено детальний опис реалізованого користувацького інтерфейсу, який є інтуїтивно зрозумілим користувачам різних типів.

Результатом кваліфікаційної роботи є готова до впровадження система для здійснення бронювання квитків у кіно, яка відповідає користувацьким та технічним вимогам, забезпечує необхідний комфорт користування програмним продуктом, зокрема перегляд афіші фільмів, вибір сеансу та місця, здійснення онлайн-транзакцій, а для персоналу кінотеатрів – керування інформацією, що містяться в базі даних, за допомогою зручного та зрозумілого інтерфейсу та надає стабільну і безперебійну роботи всього функціоналу.

		Петришин А.О.			ІПЗ.КР.Б – 121 – 25 – ЗПП	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		56

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. ПЕРЕВАГИ ОНЛАЙН-БРОНЮВАННЯ - Сервіс онлайн-запису клієнтів для бізнесу від EserviaBeauty [Електронний ресурс] - Режим доступу до ресурсу: <https://eserviabeauty.com/perevahy-onlayn-bronyuvannya/>
2. Що таке API? – QALight [Електронний ресурс] - Режим доступу до ресурсу: <https://qalight.ua/baza-znaniy/shho-take-api/>
3. What are The Different Types of APIs and Protocols? | Kong Inc. [Електронний ресурс] - Режим доступу до ресурсу: <https://konghq.com/blog/learning-center/different-api-types-and-use-cases>
4. Movie Tickets & Movie Times | Fandango. [Електронний ресурс] - Режим доступу до ресурсу: <https://www.fandango.com>
5. Multiplex – Зараз у кіно, Афіша – Кінотеатр Multiplex | Київ, Харків, Дніпро, Херсон і вся Україна. [Електронний ресурс] - Режим доступу до ресурсу: <https://multiplex.ua>
6. Планета Кіно: дивитися фільми онлайн – кіно онлайн у хорошій якості. [Електронний ресурс] - Режим доступу до ресурсу: <https://planetakino.ua>
7. AMC Theatres – movie times, movie trailers, buy tickets and gift cards. [Електронний ресурс] - Режим доступу до ресурсу: <https://www.amctheatres.com>
8. Що таке QR-код та як його використовувати вчителю? – TeachHub [Електронний ресурс] - Режим доступу до ресурсу: <https://teach-hub.com/scho-take-qr-kod-ta-yak-joho-vykorystovuvaty-vchytelyu/>
9. Лінія кіно – дивись кіно, смакуй попкорн :). [Електронний ресурс] - Режим доступу до ресурсу: <https://liniakino.com>
10. Client-Server Model – GeeksforGeeks [Електронний ресурс] - Режим доступу до ресурсу: <https://www.geeksforgeeks.org/system-design/client-server-model/>

11. What are the benefits of a microservices architecture? [Электронный ресурс] - Режим доступа до ресурсу: <https://about.gitlab.com/blog/2022/09/29/what-are-the-benefits-of-a-microservices-architecture/>
12. Index | Node.js v24.2.0 Documentation [Электронный ресурс] - Режим доступа до ресурсу: <https://nodejs.org/docs/latest/api/>
13. Benefits And Reasons To Choose Reactjs For Your Project [Электронный ресурс] - Режим доступа до ресурсу: <https://www.peerbits.com/blog/reasons-to-choose-reactjs-for-your-web-development-project.html>
14. Asynchronous Programming in Node.js: Event-driven architecture and non-blocking I/O | by WebClues Infotech | Medium Documentation [Электронный ресурс] - Режим доступа до ресурсу: <https://webcluesinfo.medium.com/asynchronous-programming-in-node-js-event-driven-architecture-and-non-blocking-i-o-41ac8ce52fc4>
15. Node.js — An introduction to the npm package manager [Электронный ресурс] - Режим доступа до ресурсу: <https://nodejs.org/en/learn/getting-started/an-introduction-to-the-npm-package-manager>
16. Express - Node.js web application framework [Электронный ресурс] - Режим доступа до ресурсу: <https://expressjs.com/>
17. MongoDB 101: A Practical Guide to NoSQL Databases | by Amit Verma | Medium [Электронный ресурс] - Режим доступа до ресурсу: <https://medium.com/%40amitvsolutions/mongodb-101-a-practical-guide-to-nosql-databases-4bc47c36cb9e>
18. Is CDN really necessary when most traffic will be "local"? Lighthouse score : r/webhosting [Электронный ресурс] - Режим доступа до ресурсу: https://www.reddit.com/r/webhosting/comments/ixkb9j/is_cdn_really_necessary_when_most_traffic_will_be/

19. bcrypt – npm Project [Электронный ресурс] - Режим доступа до ресурсу: <https://www.npmjs.com/package/bcrypt>
20. JSON Web Token Introduction - jwt.io [Электронный ресурс] - Режим доступа до ресурсу: <https://jwt.io/introduction>
21. Mern Stack Advantages And Disadvantages [Электронный ресурс] - Режим доступа до ресурсу: <https://www.skillvertex.com/blog/mern-stack-advantages-and-disadvantages/>
22. Understanding Windows Server System Requirements for Optimal Performance - WafaiCloud Blogs [Электронный ресурс] - Режим доступа до ресурсу: <https://wafaicloud.com/blog/understanding-windows-server-system-requirements-for-optimal-performance/>
23. System requirements - Ubuntu Server documentation [Электронный ресурс] - Режим доступа до ресурсу: <https://documentation.ubuntu.com/server/reference/installation/system-requirements/>
24. BSON Types - Database Manual - MongoDB Docs Disadvantages [Электронный ресурс] - Режим доступа до ресурсу: <https://www.mongodb.com/docs/manual/reference/bson-types/>
25. Mongoose v8.15.2: Schemas [Электронный ресурс] - Режим доступа до ресурсу: <https://mongoosejs.com/docs/guide.html#timestamps>
26. Replication - Database Manual - MongoDB Docs [Электронный ресурс] - Режим доступа до ресурсу: <https://www.mongodb.com/docs/manual/replication/>
27. Everything You Need to Know About MongoDB Oplogs | by Priyansh Khodiyar | Data Platform Engineering | Medium [Электронный ресурс] - Режим доступа до ресурсу: <https://medium.com/data-platform-engineering/everything-you-need-to-know-about-mongodb-oplogs-feebae3bd257>
28. Mongoose v8.15.2: Model [Электронный ресурс] - Режим доступа до ресурсу: <https://mongoosejs.com/docs/api/model.html>

29. What is Postman? Postman API Platform [Електронний ре-сурс] -
Режим доступу до ресурсу: <https://www.postman.com/product/what-is-postman/>

30. What is JSON? | Oracle Україна [Електронний ресурс] - Ре-жим
доступу до ресурсу: <https://www.oracle.com/ua/database/what-is-json/>

		Петришин А.О.			ІПЗ.КР.Б – 121 – 25 – 3ПП	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		60

ДОДАТКИ

Додаток А – Код опрацювання оплати

```
const processPayment = async (req, res) => {
  const sessionTx = await mongoose.startSession();
  sessionTx.startTransaction();

  try {
    const { bookingId, amount, cardDetails, selectedSeats } = req.body;

    const booking = await Booking.findById(bookingId).session(sessionTx);
    if (!booking) throw new Error('Бронювання не знайдено');

    const session = await validateSessionForPayment(booking.sessionId, sessionTx);

    const gateway = new PaymentGateway();
    const paymentResp = await gateway.processPayment(cardDetails, amount);
    if (!paymentResp.success) throw new Error(paymentResp.message);

    const payment = new Payment({
      bookingId: booking._id,
      amount,
      status: 'success',
      paymentMethod: 'card',
      paymentId: paymentResp.paymentId,
      paymentDate: new Date(),
      cardLast4: cardDetails.number.slice(-4),
      transactionId: paymentResp.paymentId,
      seats: selectedSeats
    });
    await payment.save({ session: sessionTx });

    await updateBookingStatus(booking._id, 'paid', 'confirmed');
    await markSeatsAsOccupied(booking, selectedSeats, sessionTx);

    const tickets = await generateTickets(booking, selectedSeats, sessionTx);

    await sessionTx.commitTransaction();
    res.status(200).json({ success: true, payment, tickets });

  } catch (err) {
    await sessionTx.abortTransaction();
    console.error('[processPayment] Error:', err);
    res.status(500).json({ error: err.message });
  } finally {
    sessionTx.endSession();
  }
}
```

Додаток Б – Код створення бронювання

```
const createBooking = async (req, res) => {
  try {
    const { sessionId, seats } = req.body;
    if (!mongoose.Types.ObjectId.isValid(sessionId)) {
      return res.status(400).json({ success: false, error: 'Invalid session ID' });
    }
    const session = await Session.findById(sessionId);
    if (!session) {
      return res.status(404).json({ success: false, error: 'Session not found' });
    }

    const isBusy = seats.some(({ row, seat }) => session.seats[row][seat]?.isOccupied);
    if (isBusy) {
      console.error('[createBooking] 400')
      return res.status(400).json({ success: false, error: 'Some seats are already occupied' });
    }
    seats.forEach(({ row, seat }) => {
      session.seats[row][seat].isOccupied = true;
      session.seats[row][seat].userId = req.user.userId;
    });
    await session.save();

    const expiryTime = new Date(Date.now() + 5 * 60 * 1000);

    const booking = await Booking.create({
      sessionId,
      userId: req.user.userId,
      seats,
      status: 'pending',
      paymentStatus: 'processing',
      amount: seats.length * session.seatPrice,
      expiryTime
    });

    res.status(201).json({ success: true, data: booking, bookingId: booking._id });
  } catch (err) {
    console.error('[createBooking] Error:', err)
    res.status(500).json({ success: false, error: err.message });
  }
};
```

Додаток В – Код генерації електронного квитка

```
const generateTickets = async (booking, selectedSeats, sessionTx) => {
  console.log('generateTickets called');
  try {
    const session = await Session.findById(booking.sessionId)
      .populate('movieId')
      .populate({ path: 'hallId', populate: { path: 'cinemaId' } })
      .session(sessionTx);

    if (!session) {
      console.log('Session not found!');
      throw new Error('Session not found for ticket generation');
    }

    const hall = session.hallId;
    const movie = session.movieId;
    const cinema = hall.cinemaId;
    const format = hall.amenities;
    const startTime = session.startTime;
    const endTime = session.endTime;
    const seatPrice = session.seatPrice;

    const ticketsData = await Promise.all(selectedSeats.map(async seat => {
      const displayRow = seat.row + 1;
      const displaySeat = seat.seat + 1;
      const ticketUuid = uuidv4();
      const qrPayload = {
        movie: movie.title,
        format: format,
        row: displayRow,
        seat: displaySeat,
        sessionId: session._id.toString(),
        bookingId: booking._id.toString(),
        ticketId: ticketUuid
      };
      const qrCodeDataURL = await QRCode.toDataURL(JSON.stringify(qrPayload));
      return {
        row: displayRow,
        seat: displaySeat,
        movieTitle: movie.title,
        movieFormat: format,
        price: seatPrice,
        sessionId: session._id,
        sessionStart: startTime,
        sessionEnd: endTime,
        cinema: cinema.name,
        bookingId: booking._id,
        status: 'reserved',
        qrCode: qrCodeDataURL
      };
    }));

    const tickets = await Ticket.insertMany(ticketsData, { session: sessionTx });
    console.log(`Inserted ${tickets.length} tickets`);
    return tickets;
  } catch (error) {
    console.error('Error in generateTickets:', error);
    throw error;
  }
};
```

Додаток Г – Код оновлення статусу бронювання

```
const updateBookingStatus = async (bookingId, status, paymentStatus) => {  
  await Booking.findByIdAndUpdate(bookingId, {  
    status,  
    paymentStatus  
  });  
};
```

```

socket.on('toggle-seat', async ({ sessionId, row, seat }) => {
  const seatKey = `${row}-${seat}`;
  let locks = seatLocks.get(sessionId) || {};
  const Session = mongoose.model('Session');
  const session = await Session.findById(sessionId);
  if (!session) {
    socket.emit('seat-error', 'Сеанс не знайдено');
    return;
  }

  if (session.seats[row][seat].isOccupied) {
    socket.emit('seat-error', 'Місце вже зайняте');
    return;
  }

  const now = Date.now();
  Object.keys(locks).forEach(key => {
    if (now - locks[key].timestamp > 300000) {
      delete locks[key];
      io.to(sessionId).emit('seat-unlocked', {
        row: Number(key.split('-')[0]),
        seat: Number(key.split('-')[1])
      });
    }
  });

  if (locks[seatKey]) {
    if (locks[seatKey].socketId === socket.id) {
      delete locks[seatKey];
      io.to(sessionId).except(socket.id).emit('seat-unlocked', { row, seat });
      socket.emit('seat-unlocked', { row, seat });
    } else {
      socket.emit('seat-error', 'Місце заблоковане іншим користувачем');
      return;
    }
  } else {
    locks[seatKey] = { socketId: socket.id, timestamp: now };
    io.to(sessionId).except(socket.id).emit('seat-locked', { row, seat });
    socket.emit('seat-locked', { row, seat });
  }

  seatLocks.set(sessionId, locks);
});

```