Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Інформаційна система онлайн бронювання квитків у глядацькі зали»

08-54.БКР.002.00.000 ПЗ

Виконав студент 4 курсу, групи 1КІ-23мс
спеціальності 123 — «Комп'ютерна інженерія»
освітня програма — «Комп'ютерна інженерія»
Бабур М. С.
(прізвище та ініціали)
Керівник к.т.н. доц. каф. ОТ
Муращенко О. Г.
(прізвище та ініціали)
Рецензент к.т.н. доц. каф. ПЗ
Яремчук Ю.Є
(прізвище та ініціали)

Допущено до захисту
Завідувач кафедри ОТ
д.т.н., проф. Азаров О. Д.

« » _____ 2025 р.

Вінниця ВНТУ — 2025 рік

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки
Освітньо-кваліфікаційний рівень бакалавр
Галузь знань — 12 «Інформаційні технології»
Спеціальність — 123 «Комп'ютерна інженерія»
Освітньо-професійна програма «Комп'ютерна інженерія»

ЗАТВЕРДЖУЮ
Завідувач кафедри ОТ
д.т.н., проф. Азаров О. Д.
«21» березня 2025 року

ЗАВДАННЯ
НА БАКАЛАВРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Бабуру Михайлу Сергійовичу

1 Тема роботи: «Інформаційна система онлайн бронювання квитків у глядацькі зали», керівник роботи Муращенко О. Г. к.т.н., доцент кафедри ОТ, затверджено наказом ВНТУ від «20» березня 2025 року № 97.

2 Термін подання студентом роботи 13.05.2025 р.

3 Вихідні дані до роботи — розташування місць, дані про наявність та статус квитків, інформація про сеанси, дані користувача, інтерактивні інтерфейси для візуалізації місць у залі, системні повідомлення про підтвердження чи скасування бронювання.

4 Зміст пояснювальної записки: вступ, аналіз сучасного стану засобів онлайн бронювання квитків та підходів до їх розробки, розробка інформаційної системи онлайн бронювання квитків, тестування та оцінка ефективності інформаційної системи, висновки.

5 Перелік графічного матеріалу: графічний інтерфейс інформаційної системи онлайн бронювання квитків у глядацькі зали.

6 Ілюстративна частина – інтерфейс веб - додатку

7 Консультанти розділів роботи представлені у таблиці 1.

Таблиця 1 — Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1 – 3	к.т.н., доц. каф. ОТ Муращенко О. Г.	21.03.2025	13.05.2025
Нормоконтроль	ас. каф. ОТ Швець С. І.	14.05.2025	30.05.2025

7 Дата видачі завдання 21.03.2025 р.

8 Календарний план виконання БКР наведений у таблиці 2.

Таблиця 2 — Календарний план

№	Назва та зміст етапу	Термін виконання	Примітка
1.	Постановка задачі роботи	21.03.25	
2.	Загальна характеристика інформаційних систем для бронювання квитків, сфера застосування систем онлайн бронювання квитків	22.03-29.03.25	Вик.
3.	Огляд сучасних платформ онлайн бронювання квитків, технічні підходи до реалізації систем бронювання	30.03-31.03.25	Вик.
4.	Особливості розробки клієнтської частини вебзастосунків бронювання, проблеми та виклики сучасних систем бронювання квитків, перспективи розвитку та актуальність для українського ринку	01.04-05.04.25	Вик.
5.	Розробка інформаційної системи онлайн бронювання квитків	06.04-30.04.25	Вик.
6.	Тестування та оцінка ефективності інформаційної системи	01.05-02.05.25	Вик
7.	Оформлення пояснювальної записки	03.05-12.05.25	Вик.
8.	Перевірка якості виконання БКР	13.05.25	Вик.
9.	Захист БКР	05.06.25	Вик.

Студент Михайло БАБ

Керівник роботи Олександр МУРАЩЕНК

АНОТАЦІЯ

УДК 004.51

Бабур М. С. Інформаційна система онлайн бронювання квитків у глядацькі зали. Бакалаврська кваліфікаційна робота зі спеціальності 123 — Комп'ютерна інженерія, освітня програма — Комп'ютерна інженерія. Вінниця: ВНТУ, 2025.

На укр. мові. 16 назв; рис.: 12; таб.: 6.

У бакалаврській кваліфікаційній роботі розглянуто процес створення інформаційної системи онлайн бронювання квитків у глядацькі зали. Проведено аналіз сучасних підходів до організації електронного бронювання та вебінтерфейсів. Розроблено клієнтську частину застосунку з використанням сучасних технологій веброзробки: JavaScript, Jotai, React, SCSS. Представлено структуру системи, описано її функціональні можливості та інтерфейс. Проведено тестування системи, яке підтвердило відповідність функціональності заданим вимогам. Отримані результати свідчать про ефективність розробленого рішення для автоматизації процесу бронювання квитків.

Ключові слова: інформаційна система, Javascript, Jotai, React, клієнтська частина.

ABSTRACT

Babur M. S. Information system for online ticket booking to the auditorium. Bachelor's qualification thesis in specialty 123 - Computer Engineering, educational program - Computer Engineering. Vinnytsia: VNTU, 2025. The explanatory note contains 79 pages, 12 figures and 16 references.

The bachelor's qualification thesis examines the process of creating an information system for online ticket booking for auditoriums. An analysis of modern approaches to the organization of electronic booking and web interfaces was conducted. The client side of the application was developed using modern web development technologies: JavaScript, Jotai, React, SCSS. The structure of the system is presented, its functionality and interface are described. The system was tested, which confirmed that the functionality meets the specified requirements. The results obtained indicate the effectiveness of the developed solution for automating the ticket booking process.

Keywords: information system, online booking, auditorium, web application, Javascript, Jotai, React, client side, testing, interface.

ЗМІСТ

ВСТУП

У сучасному цифровому середовищі все більшого значення набувають автоматизовані рішення, що спрощують взаємодію користувачів із послугами в режимі онлайн. Однією з таких сфер є продаж та бронювання квитків до глядацьких залів. Для користувачів важливо мати можливість швидко переглянути доступні заходи, обрати бажані місця та здійснити бронювання без необхідності фізичної присутності або телефонного звернення. Традиційні підходи до організації цього процесу вже не відповідають вимогам сучасного користувача, тому виникає потреба у створенні зручних та функціональних інформаційних систем.

Особливо **актуальним** є впровадження таких рішень в Україні, де значна частина культурних установ ще не забезпечена сучасними цифровими сервісами. В умовах поступової диджиталізації сфери послуг розроблення онлайн-систем бронювання квитків є не лише доцільним, а й стратегічно важливим кроком для підвищення якості обслуговування відвідувачів, розвантаження персоналу та розширення аудиторії за рахунок зручності користування.

Основна **мета** цієї бакалаврської кваліфікаційної роботи полягає у створенні інтерактивної інформаційної системи для онлайн бронювання квитків до глядацьких залів, що дозволяє користувачам самостійно обирати події, переглядати доступні місця та здійснювати бронювання через вебінтерфейс.

Для досягнення поставленої мети необхідно розв’язати такі **завдання**:

— проаналізувати предметну область та виявити особливості існуючих програмних рішень у сфері бронювання квитків;

— визначити основні вимоги до функціональності та інтерфейсу майбутньої системи;

— здійснити вибір оптимальних технологій для реалізації клієнтської частини системи;

— спроєктувати архітектуру вебзастосунку з урахуванням зручності користувача;

— реалізувати основні функції інтерфейсу: перегляд заходів, візуалізація схеми залу, вибір і бронювання місць;

— провести тестування працездатності розробленої системи та оцінити її ефективність у типових сценаріях використання.

Об'єкт дослідження — організаційні та програмні процеси, пов’язані з бронюванням квитків у глядацькі зали.

Предмет дослідження — технології та інструменти веброзробки, що застосовуються для створення інтерактивних систем онлайн бронювання.

Структура роботи передбачає три основні розділи. Перший розділ присвячено аналізу предметної галузі та огляду існуючих рішень. У другому розділі описано процес розробки клієнтської частини системи із застосуванням сучасних вебтехнологій. Третій розділ містить результати тестування програмного забезпечення та оцінювання його відповідності заданим вимогам.

1 АНАЛІЗ СУЧАСНОГО СТАНУ ЗАСОБІВ ОНЛАЙН БРОНЮВАННЯ КВИТКІВ ТА ПІДХОДІВ ДО ЇХ РОЗРОБКИ

1.1 Загальна характеристика інформаційних систем для бронювання квитків

Інформаційні системи бронювання квитків є одним із найважливіших елементів сучасної цифрової інфраструктури у сферах культури, розваг, транспорту та туризму. Їх основне призначення полягає в автоматизації процесу вибору, резервування та оплати квитків на події або поїздки, що значно підвищує ефективність роботи організаторів і забезпечує зручність для кінцевих користувачів.

Типова інформаційна система бронювання квитків складається з клієнтської частини (інтерфейс для користувачів), серверної частини (обробка запитів, бізнес-логіка), бази даних (зберігання інформації про події, місця, користувачів, платежі), а також інтерфейсів для взаємодії з платіжними сервісами, поштовими та SMS-шлюзами тощо. Основні функції таких систем включають перегляд афіші, вибір події, вибір місця у залі, здійснення бронювання та оплати, а також отримання електронного квитка.

З технічної точки зору, подібні системи зазвичай реалізуються як вебзастосунки з архітектурою типу “клієнт-сервер”. На клієнті може бути реалізований односторінковий застосунок (SPA), що забезпечує швидкий відгук інтерфейсу та покращений користувацький досвід. Серверна частина відповідає за авторизацію, обробку бронювань, валідацію запитів та підтримку актуального стану доступних місць.

Інформаційні системи бронювання повинні відповідати низці критичних вимог, зокрема, вони мають бути:

— масштабованими, щоб витримувати великі навантаження під час пікових продажів;

— безпечними, для захисту персональних даних користувачів і транзакцій;

— доступними, з підтримкою адаптивного дизайну для мобільних пристроїв;

— інтуїтивно зрозумілими, із простим і логічним інтерфейсом для кінцевого користувача.

Завдяки стрімкому розвитку вебтехнологій, сучасні системи бронювання інтегрують в себе можливості реального часу (наприклад, оновлення статусу місць без перезавантаження сторінки), функції персоналізації, інтеграцію з картами, соціальними мережами та CRM-системами. Також часто використовується візуалізація глядацької зали для точного вибору місця — реалізована за допомогою SVG, Canvas або спеціалізованих JavaScript-бібліотек.

Таким чином, інформаційні системи бронювання квитків є важливим інструментом цифрової трансформації в різних сферах. Їх ефективність напряму впливає на якість взаємодії між організаторами подій та їхньою аудиторією.

1.2 Сфера застосування систем онлайн бронювання квитків

Системи онлайн бронювання квитків отримали широке поширення в різних галузях економіки та культури завдяки своїй здатності автоматизувати процес продажу, зменшити черги, оптимізувати облік і забезпечити комфорт для користувачів. Їх застосування охоплює як великі державні та приватні організації, так і невеликі підприємства, що надають послуги з організації заходів чи транспортних перевезень.

Однією з найбільш розповсюджених сфер використання таких систем є транспортна галузь — зокрема, авіаційні, залізничні, автобусні перевезення та міські маршрути. Користувачі мають змогу швидко знайти потрібний маршрут, обрати зручний рейс, переглянути наявність місць, здійснити оплату та отримати електронний квиток без необхідності фізичної присутності в касі.

Другою значною галуззю є індустрія розваг: кінотеатри, театри, концертні майданчики, музеї, фестивалі, спортивні змагання. У цій сфері інформаційні системи онлайн бронювання виконують не лише функцію продажу, а й дозволяють

реалізувати інтерактивну схему залу для вибору конкретного місця, динамічне ціноутворення, акційні пропозиції, персоналізацію контенту.

Третім прикладом є туристичні послуги — онлайн бронювання екскурсій, відвідування визначних місць, бронювання турів. Тут особливу увагу приділяють інтеграції з іншими системами: розкладом транспорту, погодними умовами, готельними сервісами, картографічними платформами.

Також важливим напрямом є освітні й культурні події, зокрема наукові конференції, семінари, виставки, майстер-класи. У таких випадках системи бронювання дозволяють попередньо реєструвати учасників, збирати статистику, автоматизувати видачу бейджів, керувати потоками відвідувачів.

В умовах сучасної України особливої актуальності набувають державні та соціальні ініціативи, які передбачають прозорий і доступний механізм розподілу квитків на культурні заходи, особливо у громадах, що постраждали від бойових дій чи були вимушено переміщені. Цифрові рішення допомагають організаторам швидко поширювати інформацію про події, обмежувати дублювання заявок і уникати спекуляцій.

Таким чином, універсальність систем онлайн бронювання квитків дозволяє їх ефективно адаптувати до потреб конкретної галузі, підвищуючи рівень організації, прозорість і доступність заходів для широкого кола користувачів.

1.3 Огляд сучасних платформ онлайн бронювання квитків

На сучасному ринку інформаційних технологій представлено чимало платформ, що реалізують функціонал онлайн бронювання квитків. Ці сервіси відрізняються між собою за сферою застосування, функціональними можливостями, масштабованістю, зручністю інтерфейсу та технологічною реалізацією. У цьому підрозділі розглядаються найбільш популярні з них, а також наводяться їх переваги та недоліки.

Concert.ua. Це одна з провідних українських платформ для придбання квитків на концерти, театральні вистави, фестивалі та інші події. Вона забезпечує зручний інтерфейс із можливістю вибору місць на інтерактивній схемі залу. Серед

технічних можливостей — особистий кабінет користувача, електронні квитки з QR-кодами, email-повідомлення та інтеграція з популярними платіжними системами [1].

Недоліки: деякі користувачі відзначають складність у поверненні квитків і технічні збої при великому навантаженні. На рисунку 1.1 зображено зовнішній вигляд головної сторінки платформи.

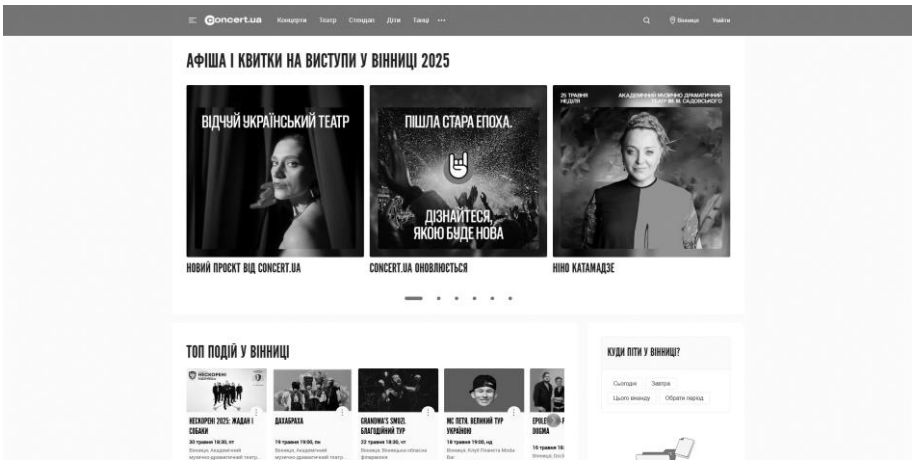

Рисунок 1.1 — Головна сторінка платформи Concert.ua

Karabas.com. Ще один популярний сервіс в Україні, який орієнтований переважно на культурно-розважальні події. Платформа надає детальну інформацію про заходи, підтримує вибір місця, застосування знижок та подарункових сертифікатів. Є мобільна версія сайту та додаток [2]. На рисунку 1.2 зображено зовнішній вигляд головної сторінки сервісу.

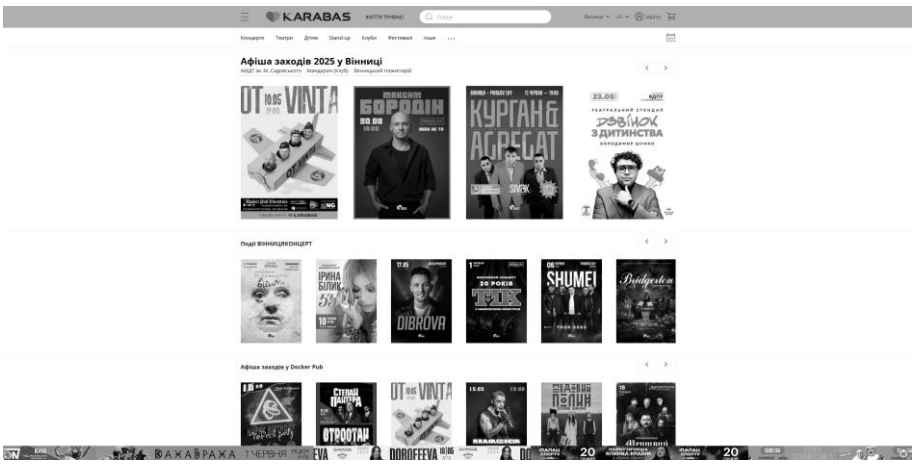

Рисунок 1.2 — Головна сторінка сервісу Karabas.com

Недоліки: інтерфейс може здаватися застарілим, відсутня підтримка динамічного завантаження без повного перезавантаження сторінки (SPA).

Tickets.ua та Busfor.ua. Ці платформи спеціалізуються на транспортному бронюванні (авіаквитки, автобуси, залізничні квитки). Вони забезпечують інтеграцію з міжнародними базами перевізників, динамічне оновлення цін, пошук оптимальних маршрутів [3, 4]. Переваги таких сервісів — швидка обробка запитів, можливість порівняння пропозицій від різних перевізників та наявність мобільних додатків. На рисунку 1.3 зображено зовнішній вигляд головної сторінки Tickets.ua.

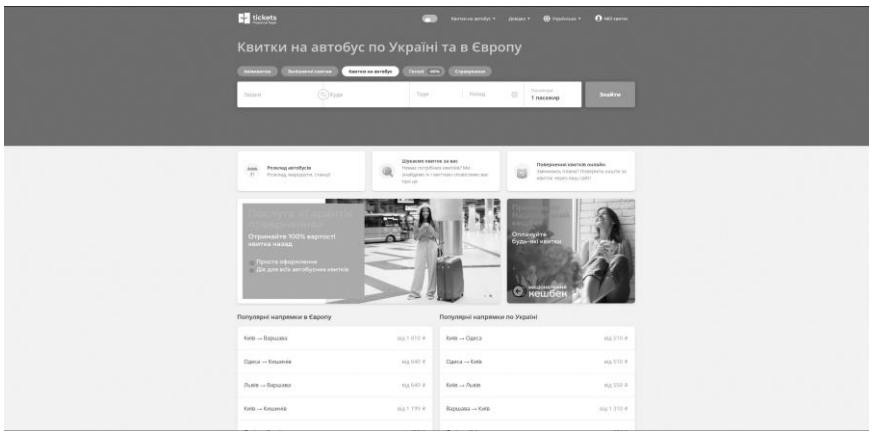

Рисунок 1.3 — Головна сторінка платформи Tickets.ua

Eventbrite. Міжнародна платформа для організації подій. Вона дозволяє створювати події, продавати квитки, інтегрувати з соціальними мережами та надсилати автоматичні повідомлення [5]. На рисунку 1.4 зображено головну сторінку платформи.

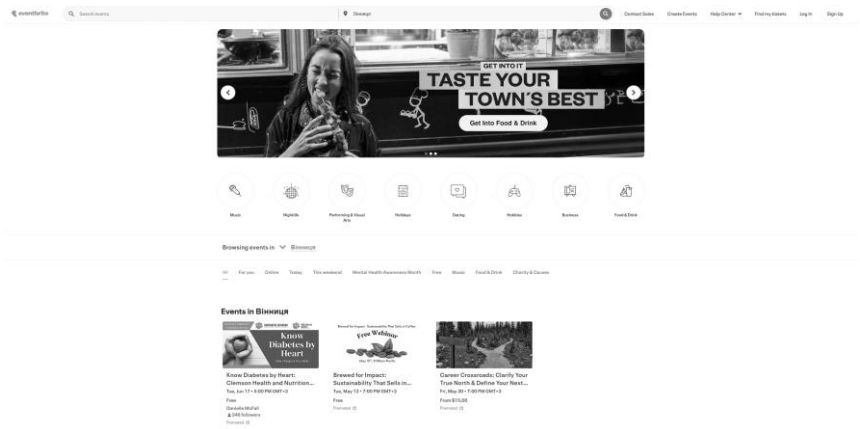

Рисунок 1.4 — Головна сторінка платформи Eventbrite

Недоліки: платна модель для організаторів, відсутність української локалізації.

Загалом можна зробити висновок, що більшість існуючих платформ зосереджені на задоволенні базових потреб користувачів, але не завжди враховують специфіку окремих заходів або потребу в гнучкому конфігуруванні залів. Це відкриває можливості для розробки більш адаптивної, функціональної системи з сучасним дизайном та гнучкою архітектурою, яка буде орієнтована на конкретний сегмент (наприклад, глядацькі зали для освітніх чи культурних подій) та потреби користувачів в Україні.

1.4 Технічні підходи до реалізації систем бронювання

Реалізація систем онлайн-бронювання квитків вимагає врахування специфіки як бізнес-логіки, так і технічної архітектури. Від вибору технологій та підходів до проєктування залежить надійність, масштабованість і зручність використання системи.

Більшість сучасних систем бронювання побудовані за принципом клієнт-серверної архітектури. Клієнтська частина (Front-end) відповідає за взаємодію з користувачем, тоді як серверна (Back-end) — за обробку запитів, доступ до баз даних, логіку бронювання тощо. Комунікація між ними відбувається через API — здебільшого REST або GraphQL [6, 7].

Для створення зручного, адаптивного та інтерактивного інтерфейсу використовуються сучасні JavaScript-фреймворки та бібліотеки. Найбільш поширені з них:

— React — бібліотека для побудови інтерфейсів на основі компонентів, що забезпечує високу продуктивність завдяки віртуальному DOM і активну підтримку спільноти [8];

— Jotai — сучасна бібліотека для керування станом у React-застосунках, вона забезпечує мінімалістичний та продуктивний підхід до управління глобальним і локальним станом без зайвого перевантаження логіки [9];

— SCSS (Sass) — розширення CSS, яке дозволяє використовувати змінні, вкладені правила та інші можливості для створення масштабованих стилів [10];

— React Router DOM — бібліотека для маршрутизації в SPA-застосунках, яка дозволяє реалізувати навігацію без перезавантаження сторінки [11].

На серверному боці часто використовуються такі платформи:

— Node.js — середовище виконання JavaScript на сервері, популярне завдяки високій продуктивності та сумісності з фронтендом;

— Express.js — веб-фреймворк для Node.js, який дозволяє швидко налаштувати маршрути та обробку запитів;

— Firebase або Supabase — хмарні бекенд-рішення з готовими модулями для аутентифікації, зберігання даних і обміну повідомленнями в реальному часі.

Для зберігання інформації про події, користувачів, бронювання та зали використовуються реляційні або документ-орієнтовані бази даних:

— PostgreSQL або MySQL — потужні реляційні СКБД із підтримкою транзакцій, індексів та складних запитів;

— MongoDB — документ-орієнтована база даних, яка добре підходить для гнучких структур даних [12].

Для реалізації вибору місця в залі важливо передбачити інтерактивні компоненти, що дозволяють візуалізувати розташування місць. Це може бути досягнуто за допомогою SVG-графіки або Canvas API, з інтерактивною логікою у React-компонентах [13].

Особлива увага приділяється захисту персональних даних користувачів, безпечній аутентифікації (наприклад, через Firebase Auth), авторизації доступу до бронювань і захисту від несанкціонованих запитів (CSRF, XSS, SQL-ін’єкції).

Успішна система бронювання повинна витримувати високі навантаження під час масових подій. Для цього впроваджують:

— кешування (наприклад, Redis);

— мікросервісну архітектуру для масштабування окремих модулів;

— CDN для прискорення завантаження фронтенду.

Таким чином, технічна реалізація систем онлайн-бронювання квитків потребує комплексного підходу до вибору інструментів, архітектури та механізмів взаємодії, що забезпечують стабільну та зручну роботу застосунку для кінцевого користувача.

1.5 Особливості розробки клієнтської частини вебзастосунків бронювання

Клієнтська частина (front-end) є критично важливою складовою будь-якого вебзастосунку для онлайн-бронювання квитків, адже саме через неї відбувається основна взаємодія користувача з системою. Вона повинна забезпечувати інтуїтивно зрозумілий інтерфейс, високу швидкодію, адаптивність до різних пристроїв, а також чітку візуалізацію залів або схем розсадки.

Одним із провідних інструментів для розробки сучасного клієнтського інтерфейсу є бібліотека React, яка дозволяє реалізовувати компонентний підхід, повторне використання коду та ефективне управління станом інтерфейсу. У зв’язку з потребою в динамічному оновленні інтерфейсу без перезавантаження сторінки, React є оптимальним вибором для систем бронювання, де користувач часто змінює параметри пошуку, фільтрує події або обирає місця.

Для керування станом додатку доцільним є використання Jotai — сучасної бібліотеки управління станом, що дозволяє будувати просту та зрозумілу структуру атомів і забезпечує високу продуктивність. Вона краще підходить для середніх за складністю проєктів, де необхідна гнучкість і швидкий відгук інтерфейсу без надмірної складності.

Стилізування компонентів здійснюється за допомогою SCSS, який підтримує вкладеність, змінні та міксини, що дозволяє створювати масштабовані стилі з високим рівнем підтримки. Це особливо важливо при роботі з інтерфейсами, які мають складну візуальну структуру — наприклад, інтерактивні зали з маркуванням місць, сцен, проходів тощо.

Ще одним важливим аспектом є реалізація інтерактивної схеми залу, що дозволяє користувачу обирати конкретне місце у залі в реальному часі. Для цього часто застосовуються SVG або Canvas API, які дозволяють створювати адаптивні,

масштабовані та клікабельні графічні інтерфейси. У таких компонентах необхідно реалізувати зміну кольору місця при наведенні, відображення його доступності, забронюваності або VIP-статусу.

У клієнтській частині також реалізуються функції:

— реєстрації та аутентифікації користувачів;

— вибору події;

— вибору місця в залі;

— формування замовлення;

— перегляду квитків у особистому кабінеті.

Невід'ємною частиною сучасних фронтенд-застосунків є асинхронна взаємодія з сервером. Застосування бібліотеки Axios або вбудованого fetch дозволяє надсилати запити до REST API, отримувати списки заходів, зберігати замовлення або перевіряти статус бронювання.

Також важливо враховувати адаптивність інтерфейсу, що забезпечує коректне відображення сторінок на різних пристроях — смартфонах, планшетах, ноутбуках. Для цього застосовуються медіазапити, гнучка верстка та фреймворки типу Flexbox або Grid.

Таким чином, успішна реалізація клієнтської частини системи онлайн-бронювання квитків вимагає глибокого розуміння не лише інструментів, але й особливостей користувацької взаємодії, швидкодії, доступності та графічного відображення елементів, що динамічно змінюються.

1.6 Безпека інформаційних систем онлайн-бронювання

Сучасні інформаційні системи онлайн-бронювання квитків працюють із великою кількістю персональних і платіжних даних користувачів, що робить безпеку однією з найважливіших складових їх розробки. Забезпечення надійного захисту інформації запобігає можливим втратам даних, шахрайству та порушенню конфіденційності, що в кінцевому результаті підвищує довіру користувачів та сприяє успішному функціонуванню системи.

В процесі експлуатації таких систем можна виділити ряд потенційних загроз, що становлять найбільший ризик:

— несанкціонований доступ до облікових записів користувачів, недостатньо складні паролі, відсутність багатофакторної аутентифікації або помилки в управлінні сесіями можуть призвести до злому облікових записів;

— атаки SQL-ін’єкції, при яких зловмисник вводить у поля форми спеціально сформовані запити, що дозволяють отримати доступ до бази даних або модифікувати її вміст;

— Cross-Site Scripting (XSS) — впровадження шкідливих скриптів у веб-сторінки, що може використовуватись для викрадення сесій користувачів або крадіжки конфіденційної інформації;

— Cross-Site Request Forgery (CSRF) — примушування користувача виконати небажані дії у системі, використовуючи його активну сесію;

— витік або компрометація платіжних даних, що веде до шахрайства та фінансових втрат.

Для зменшення ризиків системи онлайн-бронювання повинні впроваджувати комплекс заходів безпеки, що включають як технічні, так і організаційні рішення:

— надійна аутентифікація і управління сесіями, застосування складних алгоритмів хешування паролів, таких як bcrypt або Argon2, обов’язкове впровадження багатофакторної аутентифікації (MFA), контроль тривалості сесій та їх безпечне завершення запобігають несанкціонованому доступу;

— захист від SQL-ін’єкцій та XSS, використання параметризованих SQL-запитів, регулярна перевірка і валідація введених користувачем даних, а також застосування Content Security Policy (CSP) для блокування виконання шкідливих скриптів;

— впровадження токенів CSRF, захист від підробки запитів забезпечується через генерацію унікальних токенів, які перевіряються на сервері при кожній зміні стану;

— шифрування даних, використання протоколу HTTPS для захищеної передачі інформації, а також шифрування конфіденційних даних, що зберігаються у базах даних;

— безпечна інтеграція платіжних сервісів, співпраця з перевіреними платіжними шлюзами, що відповідають міжнародним стандартам безпеки (PCI DSS), дозволяє уникнути зберігання платіжної інформації на стороні сервера і значно підвищує безпеку транзакцій;

— регулярні оновлення і аудит безпеки, постійне оновлення програмного забезпечення, бібліотек і середовищ розробки дозволяє усувати виявлені вразливості, проведення періодичних аудиторських перевірок та пентестів допомагає виявляти нові загрози і усувати їх.

В Україні та багатьох інших країнах законодавство вимагає дотримання норм щодо захисту персональних даних (наприклад, Закон України «Про захист персональних даних»). Розробники систем бронювання повинні отримувати інформовану згоду користувачів на обробку їх персональних даних, застосовувати заходи для захисту цих даних від несанкціонованого доступу, втрати або пошкодження, забезпечувати можливість користувачам контролювати свої дані (перегляд, оновлення, видалення), впроваджувати політику конфіденційності, що прозоро інформує про обробку даних.

Дотримання цих вимог є не лише юридичною необхідністю, а й підвищує довіру користувачів до системи.

У процесі розробки системи онлайн-бронювання необхідно застосовувати принципи безпечного програмування з самого початку проєкту. До рекомендацій входять:

— ретельне планування структури системи з урахуванням безпекових вимог;

— використання сучасних фреймворків і бібліотек, які мають вбудовані механізми захисту;

— проведення навчання розробників принципам безпеки та регулярний аудит коду;

— впровадження автоматизованих інструментів для перевірки вразливостей;

— проведення тестування безпеки (пентестів) перед кожним релізом.

Український ринок цифрових послуг активно розвивається, що підвищує вимоги до захищеності інформаційних систем. Надійний захист даних є конкурентною перевагою і гарантує стабільність роботи сервісу навіть у разі спроб атак. Врахування специфіки локального законодавства та потреб користувачів сприяє успішній інтеграції та масштабуванню системи.

1.7 Проблеми та виклики сучасних систем бронювання квитків

Попри широке розповсюдження онлайн-сервісів для бронювання квитків, розробники та користувачі стикаються з низкою проблем, які впливають на якість використання систем та ефективність їхнього функціонування.

Високі вимоги до UX/UI. Сучасний користувач очікує інтуїтивно зрозумілого та візуально привабливого інтерфейсу. Якщо процес бронювання містить зайві кроки, має заплутану навігацію або неадаптований до мобільних пристроїв дизайн, це призводить до зниження лояльності та втрати клієнтів. Особливо складними у реалізації є інтерфейси зі схемами залів, де необхідно забезпечити чіткість візуалізації та простоту у виборі місць.

У випадках, коли декілька користувачів бронюють квитки одночасно, виникає ризик конфлікту — одне й те саме місце може бути вибране двома особами. Щоб уникнути цього, система має реалізовувати механізми блокування місця на час бронювання, а також регулярно оновлювати дані про доступність у реальному часі, наприклад через WebSocket або короткі інтервали оновлення через REST-запити.

Недостатня масштабованість. При високому навантаженні (наприклад, у день початку продажу квитків на популярну подію) багато сервісів не витримують навантаження. Виникають проблеми із завантаженням сторінок, збої при оплаті або помилки під час підтвердження бронювання. Ці виклики вимагають ретельного

проєктування архітектури, використання балансування навантаження та резервування систем.

Безпека персональних даних та платежів. Онлайн-бронювання передбачає збір та обробку конфіденційної інформації: ПІБ, контактних даних, платіжних реквізитів. Необхідно забезпечити відповідність вимогам захисту даних, реалізувати SSL-шифрування, автентифікацію користувачів, захист від CSRF/XSS-атак тощо. Особливу увагу слід приділити інтеграції з платіжними системами, які повинні бути сертифіковані за стандартами безпеки (наприклад, PCI DSS) [14].

Відсутність єдиних стандартів API. Інтеграція з зовнішніми системами (сервіси продажу, афіші, CRM) часто ускладнена через різноманіття структур API та документацій. Кожна система має свої протоколи, формати запитів і типи автентифікації, що ускладнює об’єднання декількох джерел у межах одного сервісу.

Підтримка різних типів заходів. Системи повинні бути достатньо гнучкими, щоб обслуговувати різні формати подій: концерти, театральні вистави, фестивалі, конференції тощо. Це потребує адаптивної схеми залу, змінної логіки розрахунку вартості (наприклад, залежно від сектора або дати), можливості для групових замовлень або комбінованих квитків.

Адміністрація залів або організатори не надають достатньо детальної інформації, або зміни (у схемі зали, часі початку події тощо) вносяться із запізненням. Це створює труднощі у підтримці актуальності даних на платформі.

Вимоги до доступності. Багато систем не враховують потреби користувачів із порушенням зору або інших категорій, що обмежує доступ до сервісу. Дотримання принципів вебдоступності (WCAG) є важливим фактором для розробників [15].

У сукупності ці проблеми формують виклики, які мають вирішуватися як на етапі проєктування, так і під час супроводу та масштабування інформаційної системи онлайн-бронювання. Їхнє усвідомлення дозволяє обрати оптимальні технологічні рішення та уникнути критичних недоліків у майбутній розробці.

1.8 Перспективи розвитку та актуальність для українського ринку

Ринок онлайн-бронювання квитків в Україні продовжує активно розвиватися під впливом цифровізації суспільства, зростання популярності масових заходів та зростаючого попиту на зручні вебзастосунки для купівлі квитків. Останні роки продемонстрували суттєвий зсув у поведінці споживачів — дедалі більше українців віддають перевагу онлайн-сервісам для здійснення повсякденних дій, включаючи бронювання місць у театрах, кінотеатрах, на концертах чи конференціях.

Попри наявність окремих комерційних платформ, український ринок потребує більш гнучких і адаптивних рішень, орієнтованих на локальні потреби: підтримку української мови, інтеграцію зі специфічними для регіону платіжними системами (наприклад, LiqPay, Portmone), врахування специфіки малих та середніх організаторів подій. Багато локальних заходів (наприклад, студентські вистави, регіональні фестивалі, освітні конференції) досі не мають зручних каналів продажу квитків через онлайн.

Крім того, сучасні загрози, пов’язані з безпекою, переміщенням населення в межах країни та вимушеною цифровізацією через воєнні дії, стимулюють потребу в надійних, захищених та легко масштабованих платформах. В умовах обмежених ресурсів організатори подій потребують рішень із відкритим кодом або недорогих інструментів, які можна адаптувати під свої потреби без значних інвестицій.

Окрему увагу слід приділити розвитку можливостей для інтеграції зі схемами глядацьких залів, що особливо актуально для концертних майданчиків, театрів та стадіонів. Досі не всі системи, присутні на українському ринку, пропонують зручну функцію візуального вибору місць — це створює додаткові труднощі для користувачів і обмежує функціональність сервісу.

Також варто відзначити зростання інтересу до технологій реального часу (WebSocket), які дають змогу оперативно оновлювати інформацію про доступність квитків, а також впроваджувати чат-підтримку або функціонал взаємодії з користувачами під час реєстрації. Використання сучасних фреймворків (React) та

бібліотек стану (наприклад, Jotai) дозволяє створювати масштабовані SPA-застосунки з високим рівнем взаємодії.

У перспективі очікується подальше зростання конкуренції в галузі, що стимулює інновації та поліпшення якості сервісів. Водночас це створює запит на кваліфікованих фахівців, здатних розробляти вебзастосунки з урахуванням специфіки українського ринку.

Таким чином, розробка сучасного вебзастосунку для онлайн-бронювання квитків, який враховує потреби українських користувачів, має високу актуальність як з технічної, так і з соціальної точки зору. Такий проєкт може стати основою для подальшого масштабування, інтеграції з різними типами заходів та забезпечення якісного цифрового сервісу для населення.

1.9 Узагальнення вимог до інформаційної системи онлайн-бронювання

На основі проведеного аналізу сучасних рішень у сфері онлайн-бронювання квитків, технічних підходів до їх реалізації та вимог користувачів, можна сформувати узагальнені вимоги до інформаційної системи, що забезпечить ефективне, надійне та зручне бронювання місць у глядацьких залах.

Усі вимоги до інформаційної системи можна умовно поділити на дві основні категорії: функціональні та нефункціональні. Перші описують, що саме повинна робити система, а другі — як вона повинна це робити.

Функціональні вимоги визначають перелік основних можливостей, які повинна мати система для забезпечення повного життєвого циклу бронювання. До них належать:

— перегляд списку доступних фільмів та їх описів, користувач повинен мати змогу переглядати актуальний репертуар кінотеатру з детальною інформацією про кожен фільм, включаючи опис, жанр, тривалість, акторський склад та постер;

— вибір сеансу та залу, після вибору фільму система повинна відобразити всі доступні сеанси з відповідною інформацією про час, залу та доступність місць;

— вибір місця в залі, необхідною є візуалізація плану глядацької зали з можливістю обрати вільне місце. Зайняті місця мають бути позначені відповідним чином;

— оформлення бронювання, користувач повинен мати змогу підтвердити своє бронювання, після чого система зберігає інформацію про обрані місця та сеанс;

— перегляд квитків, користувач має мати доступ до історії своїх бронювань або придбаних квитків з можливістю їх перегляду;

— особистий кабінет користувача, система повинна мати профіль користувача з можливістю редагування особистих даних та перегляду історії взаємодії з системою;

— обмеження переходу до оплати, без вибору місць та сеансу користувач не повинен мати можливість перейти до етапу бронювання, що запобігає помилкам і плутанині.

Ці вимоги є базовими для реалізації мінімально життєздатної системи онлайн-бронювання. У перспективі вони можуть бути доповнені такими можливостями, як реєстрація, оплата онлайн, створення промокодів тощо.

Нефункціональні вимоги відповідають за якість роботи системи та її взаємодію з користувачами. До основних із них належать:

— інтуїтивний інтерфейс, система повинна мати зручний, зрозумілий та адаптивний дизайн, який легко сприймається користувачами різного віку та рівня цифрової грамотності;

— швидкодія, усі переходи між сторінками повинні виконуватись без перезавантаження сторінки, що забезпечує високу швидкість роботи та кращий досвід користування;

— масштабованість, архітектура застосунку має дозволяти безболісне розширення функціоналу або інтеграцію нових компонентів у майбутньому;

— можливість багаторазового використання компонентів, компонентно-орієнтований підхід має забезпечити повторне використання елементів інтерфейсу (наприклад, картка фільму, кнопка бронювання) у різних частинах системи;

— надійність та стабільність, система має бути стійкою до збоїв, не допускати подвійного бронювання місць і коректно обробляти виключні ситуації;

— мінімальні вимоги до ресурсів, система повинна швидко працювати навіть на пристроях з низькою обчислювальною потужністю або при обмеженій швидкості Інтернету;

— безпека, навіть у простій реалізації повинні бути базові обмеження доступу до логічно залежних сторінок (наприклад, неможливість перейти до оплати без вибору місць);

— тестованість, компоненти системи мають бути ізольованими і незалежними, що дозволяє проводити ручне або автоматизоване тестування з мінімальними витратами.

Виконання перелічених вимог забезпечує належну якість реалізованого продукту, його конкурентоспроможність, а також зручність користування. Вимоги, сформовані у цьому підрозділі, стали основою для проєктування архітектури, розробки користувацького інтерфейсу та організації логіки роботи вебзастосунку в подальших етапах дипломної роботи.

2 РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ ОНЛАЙН БРОНЮВАННЯ КВИТКІВ

2.1 Вибір технологій для розробки клієнтської частини

Для розробки клієнтської частини інформаційної системи онлайн-бронювання квитків було обрано чотири основні технології: React, SCSS (Sass), Jotai, та React Router DOM. Кожна з цих технологій була вибрана з урахуванням вимог до функціональності системи, зручності розробки та підтримки в майбутньому.

React є популярною JavaScript-бібліотекою для побудови інтерфейсів користувача, яка дозволяє створювати складні веб-застосунки, що оновлюються в реальному часі. Однією з основних переваг React є використання віртуального DOM, що дозволяє мінімізувати кількість маніпуляцій із реальним DOM і, як наслідок, підвищити продуктивність додатка. Враховуючи, що інформаційна система передбачає високий рівень взаємодії з користувачем, React дозволяє ефективно реалізувати компонентну архітектуру. Це сприяє модульності коду, що в свою чергу забезпечує зручність тестування та обслуговування додатка.

Для опису стилів інтерфейсу було обрано SCSS — препроцесор CSS, який надає зручні можливості для організації стилів за допомогою змінних, вкладених правил, міксинів та інших функцій. Вибір SCSS обумовлений його здатністю значно підвищити ефективність і масштабованість процесу розробки стилів, особливо для великих і складних проектів. SCSS дозволяє зберігати структуру стилів організованою та легко змінюваною, що є важливим для забезпечення консистентності інтерфейсу та швидкого внесення змін у разі необхідності.

Jotai — це бібліотека для управління станом додатка, що була обрана через свою простоту і мінімалістичний підхід. Вона дозволяє створювати атоми стану та забезпечує ефективне їх оновлення без надмірної складності. Однією з основних переваг Jotai є її здатність інтегруватися безпосередньо з компонентами React, що дозволяє реалізувати зручне й ефективне управління станом. Оскільки проект не передбачає надмірної складності в управлінні даними, Jotai є оптимальним

вибором для легкого і швидкого впровадження функціоналу.

Для реалізації маршрутизації в додатку було обрано бібліотеку React Router DOM. Вона дозволяє створювати багатосторінкові інтерфейси, що є необхідним для організації навігації між різними секціями інформаційної системи, такими як головна сторінка, сторінка фільмів, сторінка вибору місця, оплата тощо. React Router DOM забезпечує безперервну навігацію без перезавантаження сторінки, що створює плавний та інтуїтивно зрозумілий досвід для користувачів. Оскільки ця бібліотека є стандартним рішенням для управління маршрутизацією в React-застосунках, вона була вибрана через свою надійність, популярність та підтримку спільноти.

2.2 Структура програмного продукту

Структура програмного продукту є основою для організації та управління кодом, що дозволяє забезпечити зручне тестування, масштабованість і подальший розвиток проекту. Вона визначає, як розділяється функціональність між різними компонентами і як ці компоненти взаємодіють між собою. У рамках даного проєкту структура програми побудована з урахуванням використання React як основного інструменту для розробки інтерфейсу користувача та організації компонентної архітектури.

У лістингу 2.1 представлено структуру директорії під назвою src, де зберігається основна частина програмної реалізації проєкту:

Лістинг 2.1 — Структура директорії src

```
/src
    /assets
    /components
    /pages
    /shared
    index.js
    App.js
```

/src/assets/ містить усі статичні ресурси, необхідні для проєкту, такі як зображення, шрифти та інші файли, що використовуються в інтерфейсі користувача. Структурування цієї директорії забезпечує зручний доступ до

ресурсів, що дозволяє швидко оновлювати або замінювати їх без впливу на основну логіку програми.

У директорії /src/components зберігаються усі компоненти, що відповідають за рендеринг окремих елементів інтерфейсу користувача. Вони можуть бути як незалежними компонентами, так і компонентами вищого рівня, які складаються з декількох маленьких компонентів. Усі компоненти є переписуваними та тестованими, що підвищує гнучкість і повторне використання в різних частинах програми.

Директорія /src/pages/ містить файли, які визначають основні сторінки програми. Кожен файл в цій директорії відповідає за рендеринг конкретної сторінки веб-застосунку, наприклад: головна сторінка, сторінка фільму, профіль користувача, оплата тощо. Всі ці компоненти є контейнерами для компонентів з /components, забезпечуючи таким чином зв'язок між ними та структуровану навігацію в програмі.

Директорія /src/shared містить загальні утиліти та допоміжні функції, які використовуються в різних частинах програми. Це може бути, наприклад, конфігурація загальних стилів, функції для валідації форм, обробники помилок, чи будь-які інші функції, які потребують доступу в кількох компонентах або сторінках.

Файл index.js є точкою входу програми. Тут здійснюється ініціалізація React-додатку та підключення головного компонента App.js до DOM-елементу на сторінці.

Компонент App.js є кореневим компонентом програми, який містить структуру навігації та рендерить різні сторінки залежно від маршруту. Використовуючи React Router DOM, він керує переходами між різними частинами застосунку.

Отже, структура проєкту дозволяє зберігати високий рівень організації та модульності коду, що є необхідним для ефективної розробки та подальшої підтримки додатку. Вона дозволяє швидко оновлювати окремі частини системи, зберігаючи цілісність проекту.

2.3 Програмна реалізація сторінок клієнтської частини

У даному підрозділі описується реалізація основних сторінок клієнтської частини інформаційної системи онлайн-бронювання квитків. Сторінки є важливими елементами інтерфейсу, які забезпечують користувачеві зручний доступ до основних функціональних можливостей додатку, таких як перегляд фільмів, вибір місць у залі та оплата квитків. Реалізація кожної сторінки базується на використанні компонентів React, що дозволяє досягти високої динамічності та ефективної взаємодії користувача з інтерфейсом. Дизайн системи орієнтовано на підхід mobile first, що забезпечує оптимальну роботу додатку на мобільних пристроях перед тим, як адаптувати його для більших екранів. Це дозволяє досягти високої зручності користування додатком незалежно від розміру екрана пристрою.

2.3.1 Головна сторінка

Головна сторінка є першою сторінкою, з якою взаємодіє користувач після входу до інформаційної системи онлайн-бронювання квитків. Вона виконує функцію оглядового хабу, що знайомить користувача з актуальними фільмами, які доступні для перегляду, а також з майбутніми прем'єрами, новинами та сервісами, пов’язаними з кіноіндустрією.

У файлі /src/pages/Home/Home.jsx реалізовано компонент, який відповідає за рендеринг цієї сторінки. На початку файлу сторінки імпортуються стилі, компоненти та необхідні ресурси (зображення, тестові дані), частина імпортів представлена у лістингу 2.2. Повний лістинг файлу наведено у ДОДАТКУ Д.

Лістинг 2.2 — Частина імпортів в компоненті Home.jsx

```
import styles from './index.module.scss'
import promoImg from '../../assets/Promo.png'
import NowPlayingList from "../../components/NowPlayingList/NowPlayingList";
import mockMovies from "../../shared/mock/films";
import HomeSection from "../../components/HomeSection/HomeSection";
```

Головна частина компонента — це функція Home, яка повертає JSX-розмітку. Вона обгортає сторінку в з класом profile, що стилізується за допомогою SCSS-модуля. Кожна секція контенту винесена в окремий компонент HomeSection,

що дозволяє логічно структурувати інформацію. У лістингу 2.3 представлено JSX-структуру головної сторінки.

Лістинг 2.3 — JSX-структура головної сторінки

```
export default function Home() {
  return (
    <div className={styles.profile}>
      <HomeSection title="Now playing">
        <NowPlayingList items={mockMovies}/>
      </HomeSection>
      <HomeSection title="ComingSoon">
        <ComingSoonList items={upcomingShort}/>
      </HomeSection>
      <HomeSection title="Promo">
        <img className={styles.promoImg} src={promoImg} alt=""/>
      </HomeSection>
      <HomeSection title="Service">
        <ServiceList/>
      </HomeSection>
      <HomeSection title="Movie news">
        <MovieNewsList/>
      </HomeSection>
    </div>
  )
}
```

У структурі сторінки реалізовано п’ять основних блоків:

— Now playing – показує фільми, що наразі демонструються, компонент NowPlayingList отримує дані через проп items={mockMovies};

— ComingSoon – майбутні прем’єри, що передаються в компонент ComingSoonList;

— Promo – промобанер, представлений зображенням promoImg;

— Service – список сервісів у вигляді компонента ServiceList;

— Movie news – новини кіно, які виводить компонент MovieNewsList.

Такий підхід дозволяє побудувати головну сторінку як композицію окремих незалежних частин, що легко масштабуються, змінюються та перевикористовуються. Завдяки компонентному підходу React забезпечується ефективна організація коду, зручна підтримка та адаптація під різні пристрої, що особливо актуально для mobile-first дизайну проєкту.

2.3.2 Сторінка зі списками прем’єр та анонсованих фільмів

Дана сторінка призначена для відображення двох категорій фільмів — поточних прем’єр (Now Playing) та анонсованих (Upcoming). Вона дозволяє користувачеві легко перемикатися між цими списками, переглядати деталі фільмів, а також переходити до детальної сторінки фільму.

Основна логіка сторінки реалізована в компоненті Movie, у якому використовується React-хук useState для збереження стану перемикача між режимами показу [16]. Повний лістинг файлу наведено у ДОДАТКУ Е.

У компоненті зберігається стан showUpcoming, який вказує, який із двох списків (анонсовані чи поточні прем’єри) потрібно відображати. Перемикання реалізовано через власний компонент Switch, який викликає функцію handleSwitchToggle. Програмну реалізацію наведено у лістингу 2.4.

Лістинг 2.4 — Змінна стану та логіка перемикання

```
const [showUpcoming, setShowUpcoming] = useState(true)
const movies = !showUpcoming ? upcoming : nowPlaying;
const handleSwitchToggle = (newState) => {
   setShowUpcoming(newState)
}
```

Користувачі можуть натискати на картки фільмів, щоб перейти до сторінки з детальною інформацією про фільм. Для цього використовується useNavigate з бібліотеки react-router-dom і глобальне сховище selectedFilmAtom (Jotai), через яке зберігаються дані обраного фільму. Реалізація представлена у лістингу 2.5.

Лістинг 2.5 — Обробник натискання на картку фільму

```
const handleClick = (film) => {
   if (showUpcoming) {
      setSelectedFilm(film);
      navigate('/filmpage');
   }
}
```

Відображення фільмів реалізовано за допомогою методу map, що генерує список карток з назвою, рейтингом, тривалістю, жанром та датою виходу (залежно від режиму перегляду). Програмна реалізація рендерингу наведено у лістингу 2.6.

Лістинг 2.6 — Рендеринг списку фільмів

```
<div className={`${styles.grid} ${showUpcoming ? styles.gridThin : styles.gridWide}
${styles.red}`}>
   {movies.map((movie, index) => (
      <div key={movie.id || index} className={styles.card} onClick={() => handleClick(movie)}>
         <img src={movie.image} alt={movie.title} className={styles.poster} />
         <div className={styles.info}>
            <h2>{movie.title}</h2>
            {showUpcoming && <p className={styles.itemInfo}>{movie.rating}</p>}
            {showUpcoming && (
               <p className={styles.itemInfo}>
                  <img src={clock} alt="" /> {movie.duration}
               </p>
            )}
            <p className={styles.itemInfo}>
               <img src={genre} alt="" /> {movie.genre}
            </p>
            {!showUpcoming && (
               <p className={styles.itemInfo}>
                  {movie.releaseDate}
               </p>
            )}
         </div>
      </div>
   ))}
</div>
```

Візуально картки фільмів розміщуються у сітці (grid), яка динамічно змінює свій стиль залежно від типу списку. Компонент забезпечує адаптивний інтерфейс, орієнтований на mobile-first підхід, що покращує зручність користування з мобільних пристроїв.

2.3.3 Сторінка перегляду фільму

Сторінка перегляду фільму (FilmPage.jsx) надає користувачеві детальну інформацію про вибраний фільм. Вона слугує проміжною ланкою між ознайомленням із фільмом та переходом до вибору місця в залі. Основне завдання цієї сторінки — презентувати фільм, показати його характеристики, опис сюжету та інформацію про кінотеатр, у якому він демонструється.

На початку компонента імпортуються стилі, маршрутизатор useNavigate з бібліотеки react-router-dom, хук useAtomValue з Jotai, а також глобальний атом

selectedFilmAtom, через який отримується інформація про вибраний фільм. Програмна реалізація наведена у лістингу 2.7.

Лістинг 2.7 — Імпорти та отримання даних фільму

```
import styles from './index.module.scss'
import { useNavigate } from "react-router-dom"
import { useAtomValue } from 'jotai'
import { selectedFilmAtom } from '../../shared/store/selectedFilm-store'
export default function FilmPage() {
   const film = useAtomValue(selectedFilmAtom)
   const navigate = useNavigate()
```

Для захисту від помилки при відсутності даних про фільм, реалізована перевірка: якщо film не визначено, компонент повертає null, тим самим нічого не рендерячи. Реалізацію та повний лістинг файлу наведено у ДОДАТКУ Ж.

Основна структура компонента розділена на три ключові частини: шапку (header) з постером, накладкою (overlay) та заголовком, блок деталей фільму, який включає жанр, мову, вікові обмеження, сюжет, а також інформацію про кінотеатр та кнопку продовження, яка веде користувача до етапу вибору місця, для цього використовується функція handleContinueClick, що викликає navigate('/selectplace'). Реалізацію шапки представлено у лістингу 2.8, а блоку деталей та кнопки продовження у лістингу 2.9.

Лістинг 2.8 — Відображення основної інформації про фільм

```
<div className={styles.header}>
   <img src={film.image} className={styles.poster} />
   <div className={styles.overlay}>
      <div className={styles.mainInfo}>
         <h1 className={styles.title}>{film.title}</h1>
         <p>{film.duration} • {film.date}</p>
         <div className={styles.rating}>
            <span> {film.rating}</span>
            <button>▶ Watch trailer</button>
         </div>
```

Лістинг 2.9 — Блок з деталями фільму

```
<div className={styles.details}>
   <p><span className={styles.textType}>Movie genre:</span> {film.genre}</p>
   <p><span className={styles.textType}>Censorship: </span>{film.censorship}</p>
   <p><span className={styles.textType}>Language: </span>{film.language}</p>
   <div className={styles.storyLine}>
```

```
      <h3>Storyline</h3>
      <p>{film.story}</p>
    </div>
    <h3>Cinema</h3>
    <div className={styles.cinemaBox}>
      <h4>{film.cinema.name}</h4>
      <p>{film.cinema.distance} • {film.cinema.address}</p>
    </div>
</div>
const handleContinueClick = () => {
   navigate('/selectplace')
}
<button className={styles.continueBtn} onClick={handleContinueClick}>Continue</button>
```

Сторінка реалізована відповідно до принципів адаптивного дизайну з орієнтацією на mobile-first, що дозволяє зручно переглядати інформацію навіть на мобільних пристроях. Компонент отримує дані з глобального сховища Jotai, що спрощує управління станом вибраного фільму та забезпечує плавний перехід між сторінками.

2.3.4 Сторінка вибору місця

Сторінка SelectPlace дозволяє користувачеві обрати зручні місця для перегляду фільму, встановити дату та час сеансу, а також побачити загальну вартість вибраних квитків. Вона є ключовим етапом взаємодії з користувачем перед оплатою.

На початку компонента імпортуються стилі, зображення екрану (Frame.png), хелпери (seatRows, availableTime, basePrice), маршрутизатор, та атоми з глобального стану (Jotai), за допомогою яких зберігається інформація про фільм, вибрані місця та сеанс. Програмну реалізацію та повний лістинг файлу наведено у ДОДАТКУ И.

У лістингу 2.10 наведено збереження локальних станів, а саме:

— selectedSeats, масив вибраних місць;

— selectedDate, дата сеансу;

— selectedTime, час сеансу.

Лістинг 2.10 — Локальні стани та глобальні атоми

```
const [selectedSeats, setSelectedSeats] = useState([])
const [selectedDate, setSelectedDate] = useState('10')
const [selectedTime, setSelectedTime] = useState('14:15')
const [reservedSeats] = useAtom(reservedSeatsAtom)
const [, setSelectedSession] = useAtom(selectedSessionAtom)
const [, setSelectedSeatsAtom] = useAtom(selectedSeatsAtom)
const [film] = useAtom(selectedFilmAtom)
const navigate = useNavigate()
```

Інформація про зарезервовані місця береться з глобального стану reservedSeatsAtom залежно від film.id, обраної дати і часу. Функція toggleSeat обробляє натискання на місця: якщо місце не зарезервоване, воно додається або видаляється з вибраного списку. Ціни на місця варіюються в залежності від ряду. Визначення ціни виконується функцією getSeatPrice, що враховує індекс ряду. Загальна сума квитків розраховується за вибраними місцями. Вищенаведені функції наведено у лістингу 2.11.

Лістинг 2.11 — Отримання зарезервованих місць

```
const reservedForSession = film?.id
   ? reservedSeats?.[film.id]?.[selectedDate]?.[selectedTime] || []
   : []
const toggleSeat = (seat) => {
   if (reservedForSession.includes(seat)) return
   setSelectedSeats((prev) =>
      prev.includes(seat) ? prev.filter((s) => s !== seat) : [...prev, seat]
   )
}
const getSeatPrice = (rowIndex) => 900 - basePrice * rowIndex
const total = selectedSeats.reduce((sum, seat) => {
   const rowIndex = seatRows.findIndex((row) => row.includes(seat))
   return sum + getSeatPrice(rowIndex)
}, 0)
```

Після натискання кнопки Continue обрані місця зберігаються в глобальному сховищі (selectedSeatsAtom), формується об’єкт сеансу (selectedSessionAtom) та виконується перехід на сторінку оплати. Програмна реалізація наведена у лістингу 2.12.

Лістинг 2.12 — Перехід до оплати

```
const handleContinueClick = () => {
   if (!film || selectedSeats.length === 0) return
   setSelectedSeatsAtom((prev) => ({
      ...prev,
```

```
      [film.id]: { date: selectedDate, time: selectedTime, seats: selectedSeats }
    }))
    setSelectedSession({
      date: selectedDate,
      time: selectedTime,
      seats: selectedSeats,
      price: total
    })
    navigate('/payment')
}
```

Основна розмітка компонента включає:

— зображення екрану;

— сітку місць із відповідною логікою візуалізації зарезервованих, вибраних та доступних місць;

— блок легенди (кольори місць);

— вибір дати (7 днів вперед);

— вибір часу сеансу;

— нижню панель з підсумком вартості.

Розмітку наведено у лістингу 2.13.

Лістинг 2.13 — Рендеринг місць у сітці

```
<div className={styles.seats}>
  {seatRows.map((row, rowIndex) => (
    <div key={rowIndex} className={styles.row}>
      {row.map((seat) => (
        <div
          key={seat}
          className={`${styles.seat}
                                ${reservedForSession.includes(seat) ? styles.reserved : ''}
            ${selectedSeats.includes(seat) ? styles.selected : ''}
          `}
          onClick={() => toggleSeat(seat)}
        >
          {seat}
        </div>
      ))}
    </div>
  ))}
</div>
```

Інтерфейс реалізовано у відповідності до принципів зручності користувача: інтуїтивний вибір місця, кольорова індикація, підрахунок вартості в реальному часі. Це суттєво покращує досвід взаємодії з системою онлайн-бронювання квитків.

2.3.5 Сторінка оплати

Сторінка Payment відповідає за фінальний етап взаємодії користувача з системою — оплату вибраних квитків. Користувач бачить підсумкову інформацію про замовлення, вибирає платіжний сервіс і вводить реквізити картки.

На початку компонента імпортуються необхідні стилі, зображення платіжних сервісів, хелпери та атоми стану для роботи з обраними квитками, фільмом, сеансом та забронюваними місцями. Реалізацію та повний лістинг файлу наведено у ДОДАТКУ К.

Для роботи з даними використовуються глобальні атоми, що наведені у лістингу 2.14, а саме:

— selectedFilmAtom – обраний фільм;

— selectedSessionAtom – вибраний сеанс (дата, час, місця, ціна);

— selectedSeatsAtom – обрані місця;

— reservedSeatsAtom – вже зарезервовані місця;

— myTicketsAtom – список придбаних квитків.

Лістинг 2.14 — Отримання стану з атомів

```
const [film] = useAtom(selectedFilmAtom)
const [session] = useAtom(selectedSessionAtom)
const [, setMyTickets] = useAtom(myTicketsAtom)
const [, setSelectedSeatsAtom] = useAtom(selectedSeatsAtom)
const [reservedSeats, setReservedSeats] = useAtom(reservedSeatsAtom)
const navigate = useNavigate()
```

Після натискання кнопки "Pay now" виконується функція handlePayClick, що наведена у лістингу 2.15. Вона додає новий квиток до списку myTicketsAtom, оновлює reservedSeatsAtom — зарезервовані місця для даного сеансу, очищає обрані місця з selectedSeatsAtom, перенаправляє користувача на сторінку /ticket.

Лістинг 2.15 — Обробка натискання кнопки “Pay now”

```
const handlePayClick = () => {
   if (!film || !session) return
   const newTicket = {
      film,
      date: session.date,
      time: session.time,
      price: session.price,
      seats: session.seats,
   }
   setMyTickets(prev => [...prev, newTicket])
   setReservedSeats(prev => ({
      ...prev,
      [film.id]: {
         ...prev[film.id],
         [session.date]: {
            ...prev[film.id]?.[session.date],
            [session.time]: [
               ...(prev[film.id]?.[session.date]?.[session.time] || []),
               ...session.seats
            ]
         }
      }
   }))
   setSelectedSeatsAtom({})
   navigate('/ticket')
}
```

Візуально сторінка складається з:

— списку платіжних сервісів у вигляді логотипів;

— форми введення платіжних реквізитів (номер картки, CVV, дати);

— виводу фінальної вартості;

— кнопки для підтвердження оплати.

У лістингу 2.16 наведено розмітку форми.

Лістинг 2.16 — Розмітка форми

```
<form className={styles.form}>
   <label>
      <p>Card Number</p>
      <input type="text" placeholder="Enter your card number" />
   </label>
   <div className={styles.row}>
      <label>
         <p>CVV</p>
         <input type="text" placeholder="..." />
      </label>
```

```
      <label>
        <p>Date</p>
        <input type="text" placeholder="MM/YYYY" />
      </label>
    </div>
</form>
```

Отже, дизайн сторінки відповідає загальному стилю застосунку, підтримує мобільну адаптацію та забезпечує користувачеві простий і зрозумілий процес завершення покупки.

2.3.6 Сторінка профілю

Сторінка профілю реалізує інтерфейс для відображення інформації про користувача, надаючи можливість керувати особистими даними та налаштуваннями акаунту. Вона включає такі функціональні можливості, як перегляд квитків, історія платежів, зміна пароля, налаштування мови, а також увімкнення/вимкнення біометричної автентифікації (Face ID або Touch ID). Вся інформація про користувача на цій сторінці представлена у зручному та інтуїтивно зрозумілому форматі.

Для реалізації сторінки використовуються стилі SCSS для оформлення елементів, а також SVG-іконки для візуалізації функціональних блоків. Код компонента організовано таким чином, що він дозволяє легко підтримувати та оновлювати профіль користувача в майбутньому. Також передбачена інтеграція з маршрутизацією для переходу до інших сторінок, таких як сторінка квитків.

Імпорти, компоненти профілю та повний лістинг файлу наведено у ДОДАТКУ Л.

Компонент профілю складається з двох основних частин. Перша частина — шапка профілю, що містить зображення аватара користувача, інформацію про користувача: ім’я, номер телефону та email, кнопку редагування профілю. Друга частина — меню налаштувань користувача, що включає кнопки для переходу на сторінки "Мої квитки", "Історія платежів", "Зміна мови", "Зміна пароля" та перемикач для включення/вимикання біометричної автентифікації.

Програмна реалізація шапки профілю та меню налаштувань наведено у лістингах 2.17 та 2.18 відповідно.

Лістинг 2.17 — Структура шапки профілю

```
<div className={styles.header}>
  <img src={avatar} alt="avatar" className={styles.avatar} />
  <div className={styles.userInfo}>
    <h2>Angelina</h2>
    <p className={styles.userdata}><img src={call} alt=""/>(704) 555–0127</p>
    <p className={styles.userdata}><img src={sms} alt=""/>angelina@example.com</p>
  </div>
  <span className={styles.edit}></span>
</div>
```

Кнопки в меню налаштувань відповідають за навігацію по додатку. Наприклад, кнопка "My ticket" здійснює перехід на сторінку з квитками.

Лістинг 2.18 — Меню налаштувань

```
<div className={styles.list}>
  <button onClick={() => navigate("/ticket")}><img src={ticket} alt=""/>My ticket</button>
  <button><img src={cart} alt=""/>Payment history</button>
  <button><img src={translate} alt=""/>Change language</button>
  <button><img src={lock} alt=""/>Change password</button>
  <div className={styles.toggleRow}>
    <img src={face} alt=""/>Face ID / Touch ID
    <label className="switch">
      <input type="checkbox"/>
      <span className="slider"></span>
    </label>
  </div>
</div>
```

На сторінці також є інтерактивний перемикач для активації/деактивації функцій Face ID або Touch ID, що дозволяє користувачу налаштувати додаткові параметри безпеки для входу в систему. Програмна реалізація наведена у лістингу 2.19.

Лістинг 2.19 — Меню налаштувань

```
<div className={styles.toggleRow}>
  <img src={face} alt=""/>Face ID / Touch ID
  <label className="switch">
    <input type="checkbox"/>
    <span className="slider"></span>
  </label>
```

Весь дизайн профілю є зручним для користувача, а також адаптованим до мобільних пристроїв, що забезпечує оптимальний досвід користування на різних платформах.

2.4 Напрями подальшого розвитку системи

Хоча розроблена інформаційна система онлайн-бронювання квитків уже виконує всі базові функції, її архітектура дозволяє легко розширювати функціонал, вдосконалювати інтерфейс та впроваджувати нові технології. Надалі система може еволюціонувати з демонстраційного прототипу в повноцінний комерційний продукт, що обслуговує реальні бізнес-процеси. Нижче наведено ключові напрями розвитку та вдосконалення.

Підключення до серверної частини (Backend). На поточному етапі система використовує мок-дані для демонстрації функціоналу. Наступним логічним кроком є реалізація серверної частини, що дозволить зберігати реальні дані користувачів, бронювань та фільмів. Технологічно це може бути реалізовано на таких платформах, як:

— Node.js + Express — легкий сервер для REST API;

— Firebase — як хмарна платформа для зберігання даних, автентифікації та хостингу;

— PostgreSQL / MongoDB — як основа для бази даних, залежно від обраної моделі (реляційна чи документна).

Це забезпечить збереження інформації про замовлення та історію користувача, централізоване оновлення розкладу фільмів, відображення лише актуальних сеансів із доступними місцями.

Реалізація автентифікації та авторизації. Наразі система імітує роботу авторизованого користувача. Для безпечного доступу до персональних даних потрібно впровадити повноцінну систему реєстрації та входу. Для цього можна використати:

— JWT (JSON Web Token) — для захищеної передачі токенів між клієнтом і сервером;

— Firebase Authentication — як спрощене рішення з підтримкою email/пароля, Google, Facebook тощо;

— захищені маршрути — обмеження доступу до сторінок профілю та оплати лише авторизованим користувачам.

Платіжні системи. Справжній функціонал онлайн-придбання квитків вимагає інтеграції з платіжними сервісами. Потенційні варіанти:

— LiqPay (локальний сервіс від ПриватБанку);

— Stripe / PayPal – глобальні сервіси з повною документацією та SDK для інтеграції.

Після проведення оплати користувач має отримувати електронний квиток з QR-кодом для сканування при вході.

Генерація та верифікація електронних квитків. Для кожного придбаного квитка можна генерувати:

— унікальний QR-код, який зберігає ID замовлення та верифікується при вході;

— можливість експорту квитка у форматі PDF або надсилання на електронну пошту;

— валідаційний механізм на мобільному пристрої адміністратора або у терміналі контролю.

Розширення профілю користувача. Особистий кабінет можна вдосконалити, додавши повну історію замовлень, можливість завантаження квитків, повідомлення про майбутні сеанси, налаштування профілю (зміна паролю, електронної пошти, тощо).

Адміністративна панель. Для повноцінного керування системою потрібен окремий інтерфейс для адміністратора. Його функції можуть включати додавання/редагування фільмів, керування розкладом сеансів, перегляд аналітики (кількість бронювань, завантаженість залів), редагування конфігурацій залів (кількість місць, розміщення тощо).

Аналіз попиту та рекомендації. На основі дій користувачів система може формувати персоналізовані рекомендації:

— список рекомендованих фільмів;

— повідомлення про новинки, що відповідають вподобанням користувача;

— аналітика популярних сеансів для кращого планування розкладу.

Багатомовність та локалізація. Система вже має готовність до багатомовності завдяки бібліотеці react-i18next, але поки що реалізована лише українська версія. Майбутнє розширення передбачає додавання:

— англійської, польської чи інших мов для іноземних користувачів;

— переклад повідомлень, кнопок, сповіщень та повідомлень про помилки.

Реалізація push-сповіщень. Push-сповіщення дозволять інформувати користувачів про наближення сеансу, зміни у розкладі, нові надходження та знижки. Це може бути реалізовано через Firebase Cloud Messaging (FCM), Web Push API для браузерів.

Оптимізація продуктивності. Із зростанням кількості користувачів важливо забезпечити стабільну роботу застосунку. Можливі технічні вдосконалення:

— розбиття коду на чанки (code splitting);

— оптимізація зображень;

— мемоізація компонентів React;

— використання серверного рендерингу (наприклад, через Next.js).

Мобільна адаптація та мобільний застосунок. Зважаючи на те, що значна частина користувачів здійснює бронювання квитків саме зі смартфонів, доцільно приділити увагу адаптивності інтерфейсу та можливості створення окремого мобільного застосунку. Потенційні напрями:

— покращення адаптивної верстки для різних типів пристроїв (смартфони, планшети);

— використання React Native для створення кросплатформеного мобільного застосунку з повторним використанням логіки з веб-версії;

— інтеграція з нативними можливостями пристроїв, зокрема для зчитування QR-кодів, push-сповіщень або збереження квитків у мобільному гаманці.

Це дозволить значно підвищити зручність користування сервісом у дорозі або без доступу до ПК.

Моніторинг та масштабування. У процесі розвитку проєкту, особливо після запуску в продакшн, важливо передбачити засоби моніторингу:

— використання сервісів типу Sentry, LogRocket або Firebase Performance Monitoring для збору логів, відстеження помилок і продуктивності;

— впровадження систем аналітики (Google Analytics, Matomo) для розуміння поведінки користувачів;

— горизонтальне масштабування серверної частини при зростанні навантаження — наприклад, через контейнеризацію (Docker) та хмарні сервіси (Heroku, Vercel, AWS).

Перспективи вдосконалення інформаційної системи охоплюють як технічні аспекти (автентифікація, оплати, бази даних), так і розширення функціоналу для користувачів та адміністраторів. Гнучка архітектура на основі React дозволяє поступово впроваджувати зміни, не порушуючи існуючу логіку. Це створює передумови для розвитку системи у повноцінний онлайн-сервіс, готовий до інтеграції в реальні бізнес-процеси.

3 ТЕСТУВАННЯ ТА ОЦІНКА ЕФЕКТИВНОСТІ ІНФОРМАЦІЙНОЇ СИСТЕМИ

3.1 Мета та методологія тестування

Метою тестування є перевірка функціональності, зручності використання та стабільності роботи клієнтської частини інформаційної системи онлайн-бронювання квитків. Тестування дозволяє виявити можливі помилки в логіці роботи інтерфейсу, оцінити його адаптивність до різних пристроїв та забезпечити відповідність очікуванням кінцевих користувачів.

У межах даної роботи застосовувалося ручне тестування як основний метод перевірки коректності роботи сторінок. Такий підхід дозволяє наочно перевірити реакцію інтерфейсу на дії користувача, визначити можливі проблеми з версткою, переходами між сторінками, коректністю відображення даних тощо.

Ключовими аспектами тестування стали:

— перевірка працездатності інтерфейсу у популярному веб-браузері Google Chrome;

— адаптивне тестування — оцінка коректності відображення інтерфейсу на пристроях із різною роздільною здатністю екрану (ноутбуки, планшети, смартфони), відповідно до принципу mobile first;

— UI-тестування — перевірка візуального відображення компонентів (списки, кнопки, зображення, підписи), правильності стилізації згідно з дизайном та відсутності графічних дефектів;

— перевірка функціональної логіки — тестування сценаріїв користувача: перегляд фільму, вибір місця, оплата, навігація між сторінками, взаємодія з інтерфейсом тощо.

Таким чином, тестування охоплює всі основні елементи клієнтської частини системи, забезпечивши всебічну перевірку її працездатності до етапу впровадження.

3.2 Середовище тестування

Тестування клієнтської частини інформаційної системи онлайн-бронювання квитків проводилося у локальному середовищі розробки, яке було попередньо налаштоване для імітації типової роботи веб-застосунку у браузері.

Основні компоненти тестового середовища:

— операційна система: Windows 11;

— браузер для тестування — Google Chrome (v136.0.7103.93;

— інструменти розробки та запуску проєкту — Node.js (v22.14.0), npm (v10.9.2);

— редактор коду — Visual Studio Code (v1.100) з відповідними розширеннями для React, SCSS і перевірки коду.

З метою тестування сценаріїв взаємодії з інтерфейсом були використані фіктивні (mock) дані, збережені у відповідних файлах проєкту, що дозволило імітувати реальні дії користувача без необхідності підключення до серверної частини.

Середовище тестування дозволяє ефективно перевірити повну функціональність клієнтського інтерфейсу в умовах, наближених до реальних.

3.3 Результати тестування сторінок клієнтської частини

3.3.1 Головна сторінка

Мета тестування головної сторінки — перевірка коректного відображення динамічного вмісту (списків фільмів, зображень, секцій), працездатності компонентів та відповідності верстки стандартам адаптивного дизайну.

На рисунку 3.1 зображено початковий вигляд головної сторінки при завантаженні сервісу. Сторінка завантажується без помилок у консолі браузера, працює відображення тестових даних із розділів "Now Playing", "ComingSoon", "Promo", "Service", "Movie news", компонент HomeSection успішно відображає заголовок секції та вкладений вміст, компоненти NowPlayingList, ComingSoonList, ServiceList і MovieNewsList правильно рендерять фільми й іншу інформацію з

відповідних мок-файлів, адаптивність верстки перевірено: елементи адекватно масштабується під роздільну здатність екранів мобільних пристроїв і десктопів, усі посилання, що ведуть на інші сторінки (якщо реалізовані), працюють коректно.

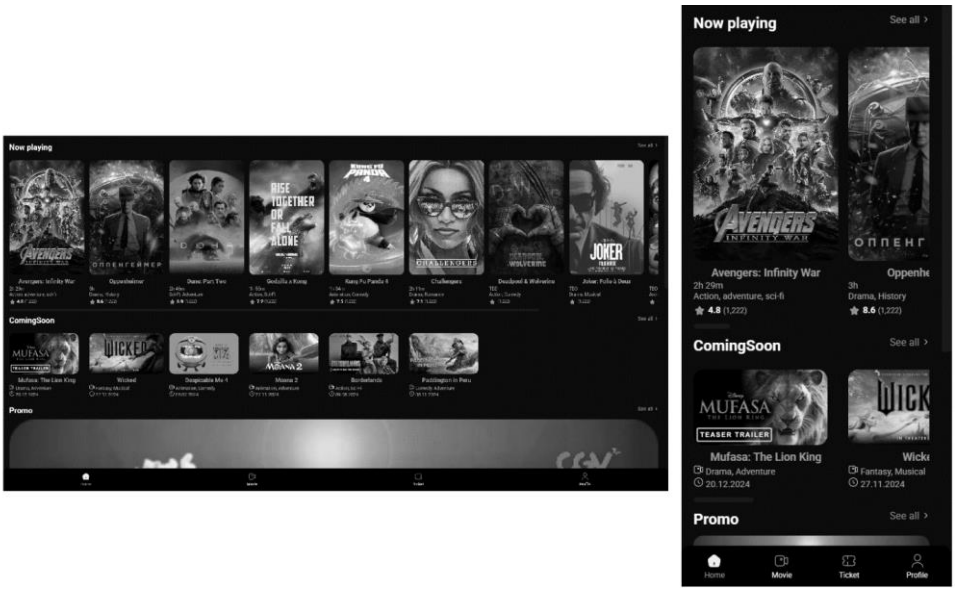

Рисунок 3.1 — Головна сторінка з адаптивним дизайном

Отже, головна сторінка успішно пройшла всі етапи тестування. Усі компоненти завантажуються й відображаються коректно, сторінка адаптивна та не має функціональних або візуальних помилок.

3.3.2 Сторінка з фільмами

Сторінка з фільмами надає користувачам можливість ознайомитися з переліком фільмів, які наразі демонструються у кінотеатрі, а також переглянути фільми, що з’являться найближчим часом. На рисунку 3.2 зображено вміст відповідного поділу на категорії.

Користувач має можливість взаємодіяти з фільмами: натискання на картку фільму веде до його детального перегляду. Елементи сторінки добре адаптуються під мобільні пристрої, автоматично змінюючи розміри і розташування залежно від ширини екрана. Жодних критичних помилок або збоїв в роботі компонентів не виявлено.

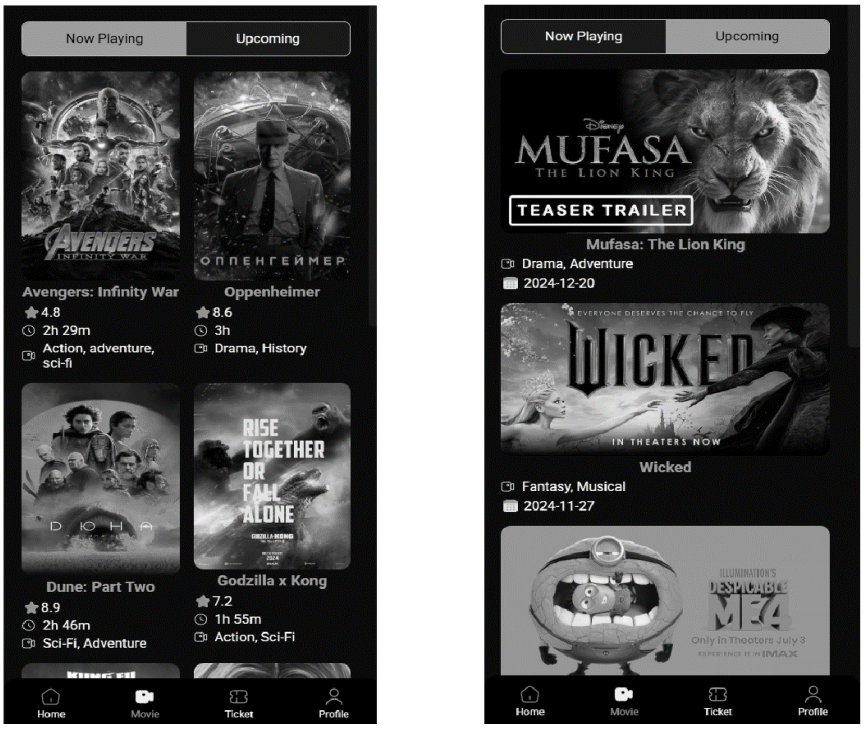

Рисунок 3.2 — Сторінка з фільмами, що йдуть та анонсованими

Отже, сторінка з фільмами відповідає вимогам функціональності, зручності та адаптивності. Вона ефективно демонструє наявний контент та забезпечує логічну навігацію до інших сторінок застосунку.

3.3.3 Сторінка одного фільму

Сторінка перегляду одного фільму надає користувачу докладну інформацію про обраний фільм, включаючи його назву, жанр, опис, вік обмеження, та можливість перейти до вибору місць. Вона є важливою ланкою між загальним переліком фільмів та подальшим процесом бронювання квитків. На рисунку 3.3 зображено зовнішній вигляд сторінки після обрання фільму.

Сторінка коректно рендериться за наявності вибраного фільму. Відображаються всі ключові дані фільму: назва, опис, жанр, вікове обмеження, оцінка користувачів сервісу, а також постер. Усі дані динамічно завантажуються з глобального стану. Кнопка “Continue” доступна і працює, виконується перехід до сторінки вибору місць. Усі елементи адаптовані під мобільний інтерфейс. Жодних помилок при передачі або відсутності даних не виявлено.

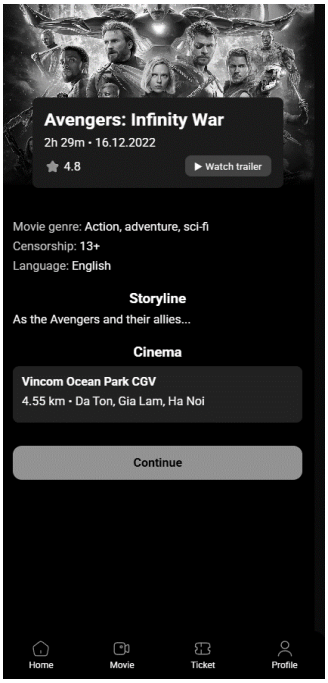

Рисунок 3.3 — Сторінка з обраним фільмом

Сторінка перегляду фільму відповідає функціональним вимогам до відображення даних та користувацької взаємодії. Завдяки використанню глобального стану сторінка динамічно підлаштовується під обраний фільм.

3.3.4 Сторінка вибору місця

Сторінка вибору місця є ключовим етапом взаємодії користувача з системою бронювання. Вона дозволяє обрати конкретні місця у залі, дату та час сеансу, після чого перейти до оплати. На рисунку 3.4 зображено початковий стан сторінки, коли тільки була натиснута кнопка “Continue” та після обрання декількох місць.

Сторінка коректно завантажується і відображає схему зали, доступні та зайняті місця. Користувач може обирати і скасовувати вибір місця кліком по ньому. Зайняті місця не піддаються взаємодії. Після кожного вибору ціна квитків автоматично перераховується відповідно до ряду місця. Користувач може обрати дату серед 7 доступних днів та час серед визначених сеансів. Після натискання кнопки “Continue” інформація про обрані місця, дату і час зберігається у глобальні стани, після чого відбувається перехід на сторінку оплати. Якщо місця не обрані — перехід не виконується (захист від помилок). Сторінка повністю

адаптована під мобільні пристрої, елементи зручно масштабуються. Після того як оплата здійснена, місця стають забронюваними, як це зображено на рисунку 3.5.

Рисунок 3.4 — Початковий стан сторінки з бронюванням квитків у глядацьку залу та після обрання декількох місць

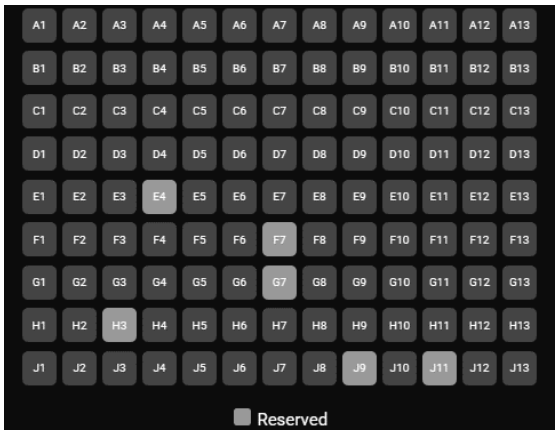

Рисунок 3.5 — Заброньовані місця

Функціональність сторінки вибору місця реалізовано повноцінно та стабільно. Логіка взаємодії з місцями, вибору дати й часу, підрахунку вартості та переходу до наступного етапу відповідає вимогам до сучасної системи онлайн-бронювання. Гнучкість завдяки використанню React та Jotai забезпечує коректне збереження стану та швидкий відгук інтерфейсу.

3.3.5 Сторінка оплати

Сторінка оплати є завершальним етапом взаємодії користувача із системою. Вона дозволяє обрати зручний для користувача метод оплати та ввести реквізити

банківської карти для завершення бронювання квитків, як це зображено на рисунку 3.6.

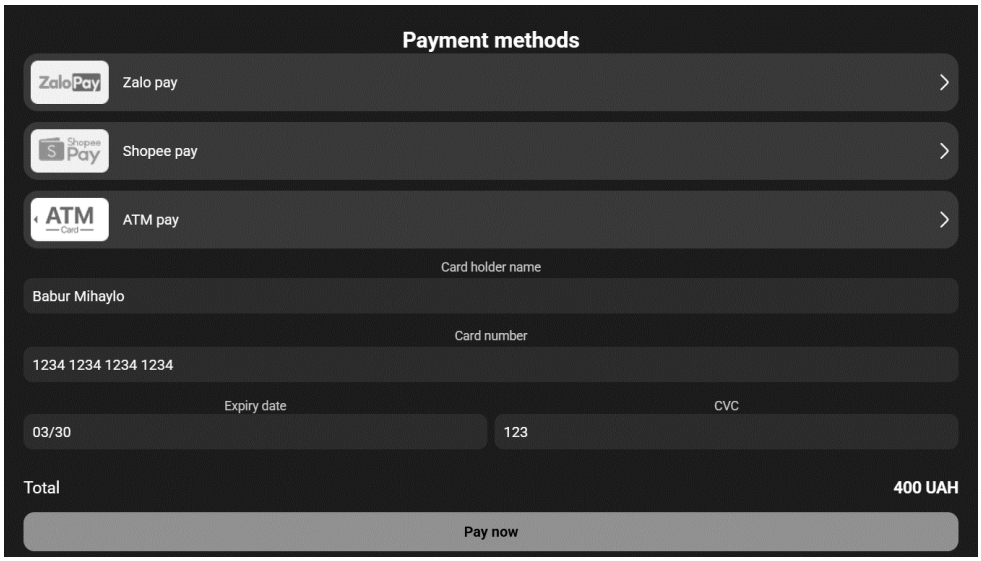

Рисунок 3.6 — Сторінка оплати

Загальна вартість замовлення динамічно розраховується на основі отриманих параметрів. Після натискання кнопки “Pay now” при заповнених полях виконується перенаправлення на головну сторінку. Інтерфейс сторінки зберігає коректне відображення на екранах різної ширини.

Отже, сторінка оплати виконує усі необхідні функції. Інтерфейс простий, інтуїтивно зрозумілий, що забезпечує зручний фінальний крок для користувача в процесі бронювання.

3.3.6 Сторінка профілю користувача

Сторінка профілю забезпечує користувача персоналізованим інтерфейсом для перегляду та керування своїми налаштуваннями безпеки, перегляду придбаних квитків. На рисунку 3.7 зображено зовнішній вигляд сторінки профілю користувача.

Сторінка коректно завантажується, перемикання між вкладками виконується без перезавантаження сторінки, завдяки внутрішній логіці React-компонентів. Усі елементи адаптивні, інтерфейс залишався зрозумілим на екранах різної ширини.

Реакція на натискання кнопок та перемикачів миттєва — інтерфейс реагує без затримок.

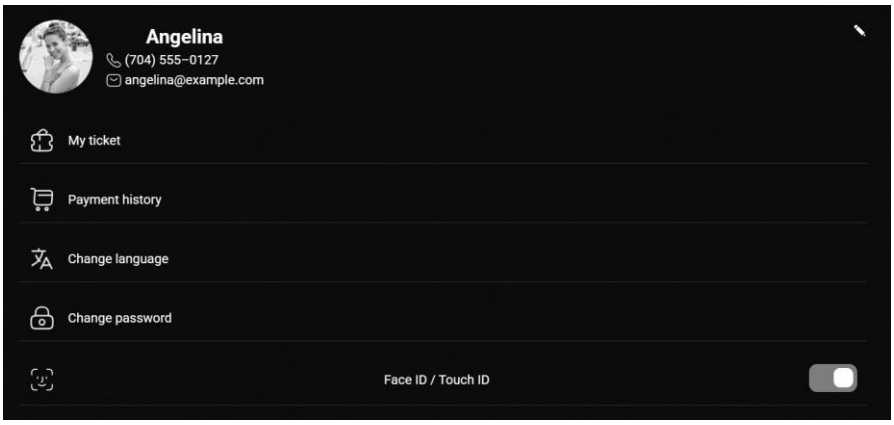

Рисунок 3.7 — Сторінка профілю користувача

При натисненні на поле “My ticket”, виконується перехід на сторінку придбаних квитків, на рисунку 3.8 зображено результат.

Рисунок 3.8 — Сторінка з придбаними квитками

Сторінка профілю реалізована як окремий компонент з вкладками, кожна з яких відповідає за певний аспект. Незважаючи на відсутність бекенда, усі зміни зберігаються на рівні браузера, що дозволяє протестувати логіку взаємодії з користувачем. Інтерфейс зрозумілий та зручний у використанні, що є позитивним чинником для майбутнього розширення функціональності.

3.4 Порівняння з аналогічними системами

Під час розробки інформаційної системи онлайн-бронювання квитків було проаналізовано декілька існуючих аналогічних рішень, таких як Concert.ua, Karabas.com, Tickets.ua, а також міжнародні сервіси на зразок Eventbrite. Метою

аналізу було визначення функціональних можливостей, принципів побудови інтерфейсу, зручності користування, а також технічних підходів до реалізації.

Більшість комерційних платформ мають схожі функції: перегляд списку подій або фільмів, вибір місця, оплата квитків, реєстрація користувача, а також особистий кабінет. Вони забезпечують інтеграцію з базами даних фільмів або подій, підтримку електронної пошти та мобільні додатки.

У порівнянні з цими рішеннями, розроблена система зосереджена на простоті інтерфейсу, мінімалістичному дизайні та легкій структурі застосунку, що робить її ідеальною для подальшої адаптації під різні потреби. Однією з ключових переваг є компонентна архітектура React, яка забезпечує гнучкість розширення системи та полегшує підтримку коду. Завдяки використанню бібліотеки Jotai для управління станом, система працює швидко і стабільно, що є перевагою над більш традиційними підходами, що використовують глобальні Redux-подібні рішення.

Крім того, застосунок розроблено так, щоб у майбутньому легко інтегрувати нові модулі — наприклад, платіжні шлюзи, базу даних реальних фільмів, механізми реєстрації користувачів тощо. Це забезпечує масштабованість та адаптивність системи, які є критично важливими характеристиками для комерційного використання.

Таким чином, у порівнянні з існуючими аналогами, запропоноване рішення демонструє високу гнучкість, простоту та можливість подальшого розвитку, що робить його конкурентоспроможним на ринку онлайн-сервісів бронювання квитків.

3.5 Оцінка ефективності інформаційної системи

Ефективність розробленої інформаційної системи онлайн-бронювання квитків була оцінена за рядом критеріїв, зокрема за стабільністю роботи, відповідністю функціональним вимогам, зручністю користувацького інтерфейсу, адаптивністю до різних пристроїв, а також за потенціалом для масштабування та подальшого вдосконалення. Проведене ручне функціональне тестування охопило всі ключові компоненти системи, які взаємодіють між собою в межах єдиної

логічної архітектури.

Функціональна повнота. Усі основні функції, закладені на етапі проєктування, були реалізовані та перевірені в умовах моделювання реальних сценаріїв використання. Серед них: перегляд списку фільмів, розподілених за категоріями (поточні та майбутні покази), вибір конкретного сеансу, взаємодія з інтерфейсом вибору місць у залі, оформлення бронювання, перегляд статусу замовлення та доступ до особистого профілю користувача. Перевірка логіки переходів показала, що система правильно обробляє дії користувача, зберігаючи необхідний контекст між сторінками.

Кожна сторінка функціонує відповідно до свого призначення, а перевірка маршрутизації показала, що неможливо здійснити перехід, наприклад, до сторінки оплати без попереднього вибору місця і сеансу. Такий підхід дозволяє уникнути логічних помилок під час навігації та забезпечує цілісність користувацького досвіду.

Оцінка інтерфейсу користувача. Інтерфейс системи є логічно організованим, з чітким поділом функціональних блоків. Завдяки використанню React та SCSS досягнуто високого рівня адаптивності дизайну, що забезпечує коректне відображення елементів на різних розмірах екранів, включаючи мобільні пристрої. Інтерактивні компоненти, такі як вибір місць у залі чи перемикання між вкладками профілю, працюють плавно та швидко. Перехід між сторінками реалізовано без повного перезавантаження сторінки (через механізми клієнтської маршрутизації React Router DOM), що позитивно впливає на швидкість та зручність користування.

Візуальний стиль дотримується єдиної концепції — всі елементи оформлено у відповідній кольоровій гамі, використано єдині шрифти, розміри кнопок і полів форм відповідають вимогам сучасного UX-дизайну. Це сприяє інтуїтивному сприйняттю системи навіть користувачами без спеціальної підготовки.

Тестування проводилося в межах звичайного використання без залучення серверної частини, з використанням мок-даних. Усі сторінки завантажуються швидко, а взаємодія з ними відбувається без помітних затримок. Система реагує на події користувача миттєво — зокрема, оновлення стану вибраного місця чи

перемикання між вкладками в профілі відбувається без збоїв. Це стало можливим завдяки легкій компонентній архітектурі та використанню бібліотеки Jotai для управління станом.

Особливу увагу приділено стабільності роботи в складних сценаріях — наприклад, при швидкому переході між сторінками, спробах повторного оформлення замовлення або очищенні локального сховища. У всіх випадках система демонструвала передбачувану поведінку без втрати даних чи порушення логіки.

Гнучкість архітектури та масштабованість. Застосунок побудований за компонентним принципом, де кожна частина системи ізольована в окремому модулі. Це спрощує не лише тестування, а й подальшу підтримку та розвиток проєкту. Використання атомарного стейту (Jotai) дозволяє зберігати дані про вибрані сеанси, місця, параметри користувача у зручному форматі, який легко масштабувати, наприклад, при підключенні зовнішніх API або збереженні в базі даних.

Система готова до подальшої інтеграції з серверною частиною. Впровадження backend-компонентів дозволить реалізувати повний цикл обслуговування: від реєстрації користувача та входу, до перевірки платіжної інформації та генерації електронних квитків. Архітектура клієнтської частини вже враховує такі перспективи розвитку, тому масштабування не потребуватиме кардинального переписування коду.

3.6 Підсумки тестування та оцінки ефективності інформаційної системи

У розділі 3 було проведено комплексне тестування та оцінку ефективності розробленої інформаційної системи онлайн-бронювання квитків. Кожна сторінка застосунку — від головної до сторінки профілю — була проаналізована з погляду коректності роботи функціоналу, зручності користувацького інтерфейсу, адаптивності до різних пристроїв і стабільності при взаємодії з іншими компонентами системи.

Результати тестування підтвердили працездатність основних модулів системи: вибір фільму та сеансу, вибір місця, бронювання та перегляд квитків. Інтерфейс забезпечує інтуїтивну навігацію, а компонентна структура застосунку сприяє гнучкому масштабуванню. Проведена оцінка ефективності підтвердила відповідність системи поставленим вимогам щодо функціональності, продуктивності, зручності та потенціалу до розширення.

Порівняння з аналогами показало, що запропонована система, хоч і не містить поки що інтеграції з платіжними сервісами чи реєстрацією користувача, має чітку структуру та технічну базу для впровадження цих можливостей у майбутньому. Система продемонструвала конкурентні переваги в плані простоти, гнучкості та готовності до подальшого розвитку.

Таким чином, проведене тестування та аналіз дозволяють зробити висновок, що створена система є ефективною, стійкою до помилок, зручною у використанні та має широкі перспективи для практичного застосування та вдосконалення.

ВИСНОВКИ

У межах бакалаврської кваліфікаційної роботи було створено інформаційну систему онлайн-бронювання квитків у глядацькі зали, яка забезпечує ефективну взаємодію між користувачем та цифровою платформою. Система реалізує ключовий функціонал: перегляд доступних фільмів, вибір сеансу, вибір місця у залі, оформлення бронювання, перегляд придбаних квитків та керування профілем користувача.

У процесі розробки було проведено аналіз сучасних тенденцій веброзробки та принципів побудови інформаційних систем, визначено вимоги до функціональних можливостей застосунку, обґрунтовано вибір технологій реалізації. Створено інтерфейс користувача з використанням бібліотеки React, забезпечено стилізацію з допомогою SCSS, реалізовано маршрутизацію за допомогою React Router DOM, а також управління станом через бібліотеку Jotai. Логіка застосунку структурована таким чином, щоб забезпечити зрозумілу навігацію та зручність взаємодії користувача із системою.

Окрему увагу приділено збереженню обраних сеансів і місць, а також перевірці коректності роботи застосунку. Проведено ручне тестування основних сторінок із оцінкою стабільності, зручності інтерфейсу, продуктивності та відповідності вимогам. Проведена оцінка ефективності підтвердила функціональну повноту, масштабованість та потенціал системи до подальшого розвитку.

Практична цінність роботи полягає в створенні основи для впровадження автоматизованого процесу бронювання квитків, що може бути адаптований для потреб кінотеатрів, театрів та інших закладів культури.

Отримані результати свідчать про досягнення поставлених цілей і виконання усіх завдань дослідження. Запропонована система має перспективу подальшого розширення шляхом впровадження серверної частини, авторизації користувачів, електронних платежів та підтримки різноманітних видів подій.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Афіша і квитки на виступи у Вінниці 2025 [Електронний ресурс]. — Режим доступу: https://concert.ua/uk/vinnytsia

2. Афіша заходів 2025 у Вінниці [Електронний ресурс]. — Режим доступу: https://vinnytsia.karabas.com/?_gl=1*1e5lnqt*_gcl_au*NTczOTM5MTMuMTc0NjU1MDA4Ng..

3. Найкращі авіаквитки за найнижчими цінами [Електронний ресурс]. — Режим доступу: https://surl.lu/lqupcs

4. Квитки на автобус [Електронний ресурс]. — Режим доступу: https://busfor.ua/

5. Events in Вінниця [Електронний ресурс]. — Режим доступу: https://www.eventbrite.com/

6. Повний огляд REST: нюанси, поради, приклади [Електронний ресурс]. — Режим доступу: https://dou.ua/forums/topic/50364/

7. Introduction to GraphQL [Електронний ресурс]. — https://graphql.org/learn/

8. React [Електронний ресурс]. — Режим доступу: https://react.dev/reference/react#react

9. Documentation [Електронний ресурс]. — Режим доступу: https://jotai.org/docs

10. Documentation [Електронний ресурс]. — Режим доступу: https://sass-lang.com/documentation/

11. React Router Home [Електронний ресурс]. — Режим доступу: https://reactrouter.com/home

12. What is MongoDB? [Електронний ресурс]. — Режим доступу: https://www.mongodb.com/docs/manual/

13. Canvas API [Електронний ресурс]. — Режим доступу: https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API

14. Document Library [Електронний ресурс]. — Режим доступу: https://www.pcisecuritystandards.org/document_library/?category=pcidss&document=pci_dss

15. Web Content Accessibility Guidelines (WCAG) 2.2 [Електронний ресурс]. — Режим доступу: https://www.w3.org/TR/WCAG22/

16. useState [Електронний ресурс]. — Режим доступу: https://react.dev/reference/react/useState

ДОДАТОК А

Технічне завдання

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

ЗАТВЕРДЖУЮ
Завідувач кафедри ОТ
д.т.н., проф. Азаров О. Д.
«21» березня 2025 року

ТЕХНІЧНЕ ЗАВДАННЯ
на виконання бакалаврської кваліфікаційної роботи
«Інформаційна система онлайн бронювання квитків у глядацькі зали»
08-54.БКР.002.00.000.ТЗ

Науковий керівник: доцент к.т.н.
_____ Муращенко О. Г.

Студент групи 1КІ-23мс
_____ Бабур М. С.

ВНТУ 2025

1 Підстава для виконання бакалаврської дипломної роботи (БКР)

1.1 З огляду на зростаючу популярність онлайн-бронювання квитків на культурно-розважальні заходи, а також потребу у зручних, доступних і технологічно сучасних рішеннях для організаторів подій, актуальним є створення вебзастосунку, який забезпечує користувачам можливість швидкого та інтуїтивного вибору місць у залі, безпечної оплати та отримання електронного квитка. В умовах цифрової трансформації та зростаючої кількості локальних ініціатив в Україні виникає потреба у власних гнучких платформах, здатних адаптуватися під потреби малих і середніх організаторів заходів.

1.2 Наказ про затвердження теми БКР.

2 Мета БКР і призначення розробки

2.1 Мета роботи — розробка вебзастосунку для онлайн-бронювання квитків у глядацькі зали, який забезпечує користувачам можливість перегляду доступних заходів, вибору місць на інтерактивній схемі залу, бронювання та отримання квитків.

2.2 Призначення розробки — автоматизація процесу бронювання квитків на масові заходи, які проходять у глядацьких залах. Підвищення ефективності організації подій, зниження навантаження на касовий персонал та покращення взаємодії з відвідувачами.

3 Вихідні дані для виконання БКР

3.1 Розташування місць.

3.2 Дані про наявність та статус квитків.

3.3 Інформація про сеанси, дані користувача.

3.4 Інтерактивні інтерфейси для візуалізації місць у залі.

3.5 Системні повідомлення про підтвердження чи скасування бронювання.

4 Вимоги до виконання БКР

Головна вимога — створити повноцінну клієнтську частину вебзастосунку для онлайн-бронювання квитків у глядацькі зали, яка включає: інтерфейс користувача, інтерактивну схему залу, систему вибору місць і бронювання, з використанням сучасних технологій фронтенду (React, SCSS, Jotai), а також демонстрацію функціональної і стабільної роботи цього застосунку.

5 Етапи БКР та очікувані результати

Етапи роботи та очікувані результати наведено у Таблиці А.1.

Таблиця А.1 — Етапи БКР

№	Назва та зміст етапу	Термін виконання	Примітка
1.	Постановка задачі роботи	21.03.25	
2.	Загальна характеристика інформаційних систем для бронювання квитків, сфера застосування систем онлайн бронювання квитків	22.03-29.03.25	
3.	Огляд сучасних платформ онлайн бронювання квитків, технічні підходи до реалізації систем бронювання	30.03-31.03.25	
4.	Особливості розробки клієнтської частини вебзастосунків бронювання, проблеми та виклики сучасних систем бронювання квитків, перспективи розвитку та актуальність для українського ринку	01.04-05.04.25	
5.	Розробка інформаційної системи онлайн бронювання квитків	06.04-30.04.25	
6.	Тестування та оцінка ефективності інформаційної системи	01.05-02.05.25	
7.	Оформлення пояснювальної записки	03.05-12.05.25	
8.	Перевірка якості виконання БКР	13.05.25	
9.	Захист БКР	05.06.25	

6 Матеріали, що подаються до захисту БКР

До захисту подаються: пояснювальна записка БКР, ілюстративні матеріали, протокол попереднього захисту БКР на кафедрі, відгук наукового керівника, анотації до БКР українською та іноземною мовами, довідка про відповідність оформлення БКР діючим вимогам.

7 Порядок контролю виконання та захисту БКР

Виконання етапів графічної та розрахункової документації БКР контролюється науковим керівником згідно зі встановленими термінами. Захист БКР відбувається на засіданні Екзаменаційної комісії, затвердженої наказом ректора.

8 Вимоги до оформлювання та порядок виконання БКР

8.1 При оформлювання БКР використовуються:

— ДСТУ 3008: 2015 «Звіти в сфері науки і техніки. Структура та правила оформлювання»;

— ДСТУ 8302: 2015 «Бібліографічні посилання. Загальні положення та правила складання»;

— міждержавний ГОСТ 2.104-2006 «Єдина система конструкторської документації. Основні написи»;

— методичні вказівки до виконання бакалаврських кваліфікаційних робіт зі спеціальності 123 – «Комп’ютерна інженерія» (освітня програма «Комп’ютерна інженерія»). Кафедра обчислювальної техніки ВНТУ 2023;

— документами на які посилаються у вище вказаних.

8.2 Порядок виконання БКР викладено в «Положення про кваліфікаційні роботи на першому (бакалаврському) рівні вищої освіти СУЯ ВНТУ-03.02.02-П.001.01:21»

ДОДАТОК Б

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи:

«Інформаційна система онлайн бронювання квитків у глядацькі зали»

Тип роботи: Бакалаврська кваліфікаційна робота

(БДР, МКР)

Підрозділ кафедра обчислювальної техніки

(кафедра, факультет)

Показники звіту подібності StrikePlagiarism

Оригінальність 99% Схожість 1%

Аналіз звіту подібності (відмітити потрібне):

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

Особа, відповідальна за перевірку _____ Захарченко С. М.

(підпис) (прізвище, ініціали)

Ознайомлені з повним звітом подібності, який був згенерований системою StrikePlagiarism щодо роботи.

Автор роботи Бабур М.С.

Керівник роботи Муращенко О. Г.

ДОДАТОК В

Графічна частина

СТРУКТУРНА СХЕМА ІНФОРМАЦІЙНОЇ СИСТЕМИ ОНЛАЙН БРОНЮВАННЯ КВИТКІВ У ГЛЯДАЦЬКІ ЗАЛИ

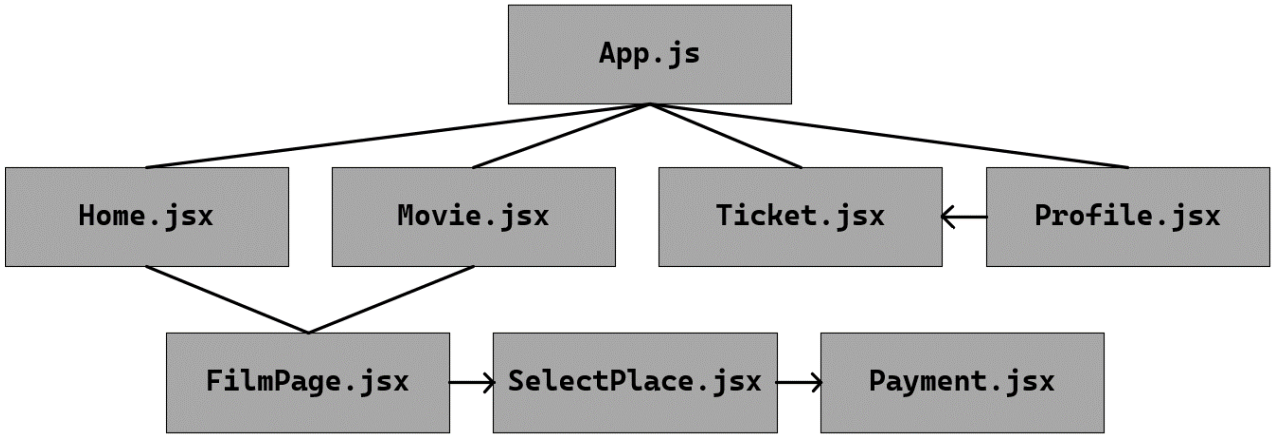

Рисунок В.1 — Структурна схема інформаційної системи онлайн бронювання квитків у глядацькі зали

ДОДАТОК Г

Ілюстративна частина

ІНТЕРФЕЙС КОРИСТУВАЧА ІНФОРМАЦІЙНОЇ СИСТЕМИ ОНЛАЙН БРОНЮВАННЯ КВИТКІВ У ГЛЯДАЦЬКІ ЗАЛИ

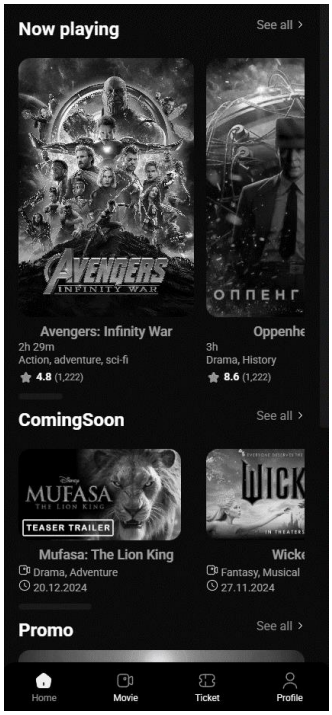

Рисунок Г.1 — Головна сторінка

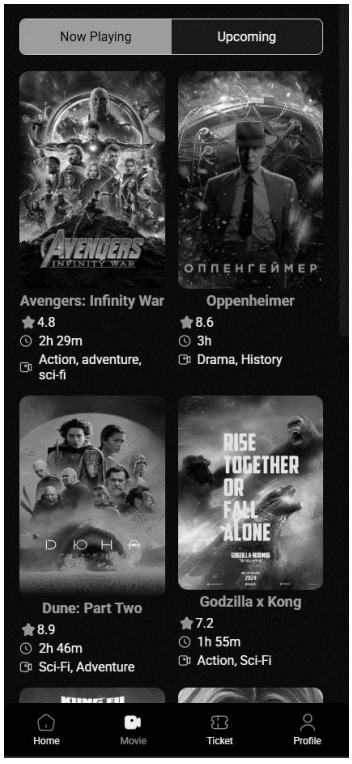

Рисунок Г.2 — Сторінка з фільмами, що йдуть та анонсованими

Рисунок Г.3 — Сторінка з придбаними квитками

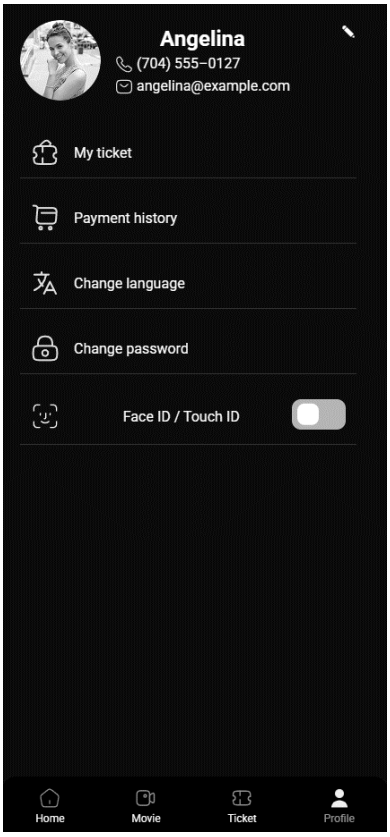

Рисунок Г.4 — Сторінка профілю користувача

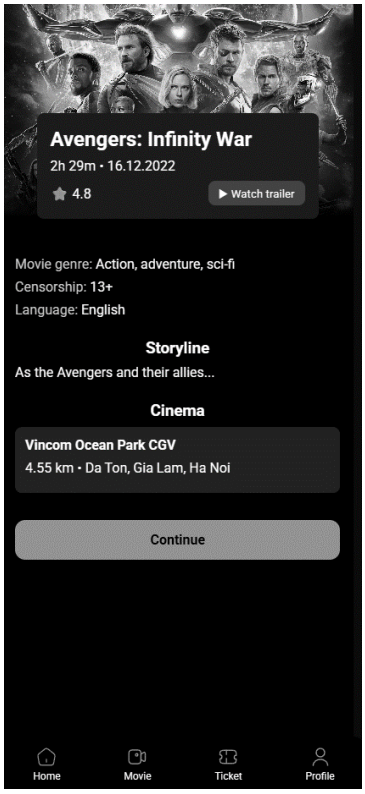

Рисунок Г.5 — Сторінка з вибраним фільмом

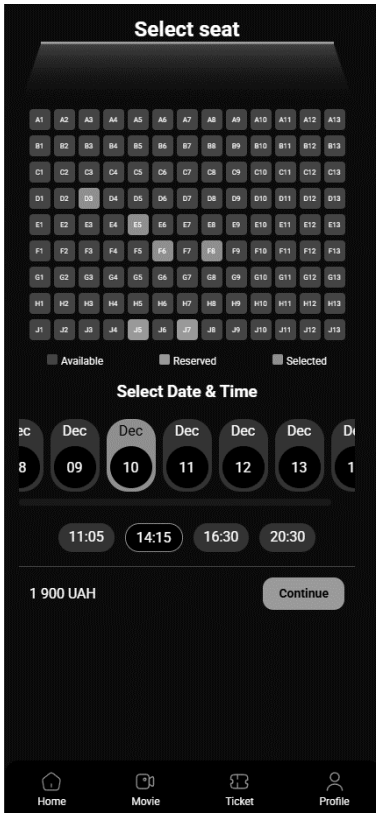

Рисунок Г.6 — Сторінка з бронюванням місць

ДОДАТОК Д

Лістинг файлу Home.jsx

```
import styles from './index.module.scss'
import promoImg from '../../assets/Promo.png'
import NowPlayingList from "../../components/NowPlayingList/NowPlayingList";
import mockMovies from "../../shared/mock/films";
import HomeSection from "../../components/HomeSection/HomeSection";
import upcomingShort from "../../shared/mock/upcoming";
import ComingSoonList from "../../components/ComingSoonList/ComingSoonList";
import ServiceList from "../../components/ServiceList/ComingSoonList";
import MovieNewsList from "../../components/MovieNewsList/ComingSoonList";

export default function Home() {
   return (
      <div className={styles.profile}>
         <HomeSection title="Now playing">
            <NowPlayingList items={mockMovies}/>
         </HomeSection>
         <HomeSection title="ComingSoon">
            <ComingSoonList items={upcomingShort}/>
         </HomeSection>
         <HomeSection title="Promo">
            <img className={styles.promoImg} src={promoImg} alt=""/>
         </HomeSection>
         <HomeSection title="Service">
            <ServiceList/>
         </HomeSection>
         <HomeSection title="Movie news">
            <MovieNewsList/>
         </HomeSection>
      </div>
   )
}
```

ДОДАТОК Е

Лістинг файлу Movie.jsx

```
import { useState } from 'react'
import styles from './index.module.scss'
import nowPlaying from "../../shared/mock/films"
import upcoming from "../../shared/mock/upcoming"
import clock from "../../assets/clock.svg"
import genre from "../../assets/genre.svg"
import Switch from "../../components/Switch/Switch";
import {useNavigate} from "react-router-dom";
import {selectedFilmAtom} from "../../shared/store/selectedFilm-store";
import { useSetAtom } from 'jotai';

export default function Movie() {
   const [showUpcoming, setShowUpcoming] = useState(true)
   const movies = !showUpcoming ? upcoming : nowPlaying;
   const setSelectedFilm = useSetAtom(selectedFilmAtom)
   const navigate = useNavigate()

   const handleSwitchToggle = (newState) => {
      setShowUpcoming(newState)
   }

   const handleClick = (film) => {
      if (showUpcoming) {
         setSelectedFilm(film);
         navigate('/filmpage');
      }
   }

   return (
      <div className={styles.moviePage}>
         <Switch leftText="Now Playing" rightText="Upcoming" onToggle={handleSwitchToggle} />

         <div className={`${styles.grid} ${showUpcoming ? styles.gridThin : styles.gridWide}
${styles.red}`}>
            {movies.map((movie, index) => (
               <div key={movie.id || index} className={styles.card}
onClick={()=>handleClick(movie)}>
                  <img src={movie.image} alt={movie.title} className={styles.poster} />
                  <div className={styles.info}>
                     <h2>{movie.title}</h2>
                     {showUpcoming && <p className={styles.itemInfo}>☆{movie.rating}</p>}
                     {showUpcoming && (
                        <p className={styles.itemInfo}>
                           <img src={clock} alt="" /> {movie.duration}
                        </p>
                     )}
                     <p className={styles.itemInfo}>
                        <img src={genre} alt="" /> {movie.genre}
```

```
                    </p>
                    {!showUpcoming && (
                      <p className={styles.itemInfo}>
                         {movie.releaseDate}
                      </p>
                    )}
                  </div>
                </div>
              ))}
            </div>
          </div>
        )
}
```

ДОДАТОК Ж

Лістинг файлу FilmPage.jsx

```
import styles from './index.module.scss'
import { useNavigate } from "react-router-dom"
import { useAtomValue } from 'jotai'
import { selectedFilmAtom } from '../../shared/store/selectedFilm-store'

export default function FilmPage() {
   const film = useAtomValue(selectedFilmAtom)
   const navigate = useNavigate()

   const handleContinueClick = () => {
      navigate('/selectplace')
   }

   if (!film) return null

   return (
      <div className={styles.page}>
         <div className={styles.header}>
            <img src={film.image} className={styles.poster} />
            <div className={styles.overlay}>
               <div className={styles.mainInfo}>
                  <h1 className={styles.title}>{film.title}</h1>
                  <p>{film.duration} • {film.date}</p>
                  <div className={styles.rating}>
                     <span>☆ {film.rating}</span>
                     <button>▶ Watch trailer</button>
                  </div>
               </div>
            </div>
         </div>

         <div className={styles.details}>
            <p><span className={styles.textType}>Movie genre:</span> {film.genre}</p>
            <p><span className={styles.textType}>Censorship: </span>{film.censorship}</p>
            <p><span className={styles.textType}>Language: </span>{film.language}</p>

            <div className={styles.storyLine}>
               <h3>Storyline</h3>
               <p>{film.story}</p>
            </div>

            <h3>Cinema</h3>
            <div className={styles.cinemaBox}>
               <h4>{film.cinema.name}</h4>
               <p>{film.cinema.distance} • {film.cinema.address}</p>
            </div>
         </div>
```

```
            <button className={styles.continueBtn} onClick={handleContinueClick}>Continue</button>
        </div>
    )
}
```

ДОДАТОК И

Лістинг файлу SelectPlace.jsx

```
import { useEffect, useState } from 'react'
import styles from './index.module.scss'
import frame from '../../assets/Frame.png'
import { useNavigate } from 'react-router-dom'
import { availableTime, basePrice, seatRows } from '../../shared/mock/seatsInfo'
import { useAtom } from 'jotai'
import { selectedSessionAtom, selectedFilmAtom } from '../../shared/store/selectedFilm-store'
import { reservedSeatsAtom, selectedSeatsAtom } from '../../shared/store/selectedSeats-store'

export default function SelectPlace() {
    const [selectedSeats, setSelectedSeats] = useState([])
    const [selectedDate, setSelectedDate] = useState('10')
    const [selectedTime, setSelectedTime] = useState('14:15')

    const [reservedSeats] = useAtom(reservedSeatsAtom)
    const [, setSelectedSession] = useAtom(selectedSessionAtom)
    const [, setSelectedSeatsAtom] = useAtom(selectedSeatsAtom)
    const [film] = useAtom(selectedFilmAtom)

    const navigate = useNavigate()

    const reservedForSession = film?.id
        ? reservedSeats?.[film.id]?.[selectedDate]?.[selectedTime] || []
        : []

    const toggleSeat = (seat) => {
        if (reservedForSession.includes(seat)) return
        setSelectedSeats((prev) =>
            prev.includes(seat) ? prev.filter((s) => s !== seat) : [...prev, seat]
        )
    }

    const getSeatPrice = (rowIndex) => 900 - basePrice * rowIndex

    const total = selectedSeats.reduce((sum, seat) => {
        const rowIndex = seatRows.findIndex((row) => row.includes(seat))
        return sum + getSeatPrice(rowIndex)
    }, 0)

    const handleContinueClick = () => {
        if (!film || selectedSeats.length === 0) return
        setSelectedSeatsAtom((prev) => ({
            ...prev,
            [film.id]: { date: selectedDate, time: selectedTime, seats: selectedSeats }
        }))
        setSelectedSession({
            date: selectedDate,
            time: selectedTime,
```

```
      seats: selectedSeats,
      price: total
    })
    navigate('/payment')
  }

  return (
    <div className={styles.page}>
      <h2>Select seat</h2>
      <img src={frame} alt="screen" />

      <div className={styles.seats}>
        {seatRows.map((row, rowIndex) => (
          <div key={rowIndex} className={styles.row}>
            {row.map((seat) => (
              <div
                key={seat}
                className={`${styles.seat}
                  ${reservedForSession.includes(seat) ? styles.reserved : ''}
                  ${selectedSeats.includes(seat) ? styles.selected : ''}
                `}
                onClick={() => toggleSeat(seat)}
              >
                {seat}
              </div>
            ))}
          </div>
        ))}
      </div>

      <div className={styles.legend}>
        <div><span className={styles.available}></span>Available</div>
        <div><span className={styles.reserved}></span>Reserved</div>
        <div><span className={styles.selected}></span>Selected</div>
      </div>

      <h3 className={styles.selectDateTime}>Select Date & Time</h3>
      <div className={styles.dateList}>
        {[...Array(7)].map((_, i) => {
          const day = (8 + i).toString().padStart(2, '0')
          return (
            <div
              key={day}
              className={`${styles.dateItem} ${day === selectedDate ? styles.active : ''}`}
              onClick={() => setSelectedDate(day)}
            >
              <p className={styles.month}>Dec</p>
              <div className={styles.day}>
                <p className={styles.dayText}>{day}</p>
              </div>
            </div>
          )
```

```
            ))}
          </div>

          <div className={styles.timeList}>
            {availableTime.map((t) => (
              <div
                key={t}
                className={`${styles.timeItem} ${t === selectedTime ? styles.active : ''}`}
                onClick={() => setSelectedTime(t)}
              >
                {t}
              </div>
            ))}
          </div>

          <div className={styles.footer}>
            <div>{total.toLocaleString()} UAH</div>
            <button onClick={handleContinueClick}>Continue</button>
          </div>
        </div>
    )
}
```

ДОДАТОК К

Лістинг файлу Payment.jsx

```
import styles from './index.module.scss'
import Zalo from "../../assets/Zalo.png"
import Shopee from "../../assets/Shopee.png"
import ATM from "../../assets/ATM.png"
import arrow from "../../assets/arrow-right.svg"
import { useAtom } from 'jotai'
import { myTicketsAtom } from '../../shared/store/myTickets-store'
import { selectedFilmAtom, selectedSessionAtom } from '../../shared/store/selectedFilm-store'
import { useNavigate } from 'react-router-dom'
import {reservedSeatsAtom, selectedSeatsAtom} from "../../shared/store/selectedSeats-store";

export default function Payment() {
    const [, setMyTickets] = useAtom(myTicketsAtom)
    const [, setReservedSeats] = useAtom(reservedSeatsAtom)
    const [, setSelectedSeats] = useAtom(selectedSeatsAtom)
    const [film] = useAtom(selectedFilmAtom)
    const [session] = useAtom(selectedSessionAtom)
    const navigate = useNavigate()

    const handlePay = () => {
        if (!film || !session.date || !session.time || session.seats.length === 0) return

        const ticket = {
            date: {
                day: session.date,
                time: session.time,
            },
            info: {
                ...film
            },
            price: session.price,
        }

        setMyTickets((prev) => [...prev, ticket])

        // Додаємо заброньовані місця в reservedSeatsAtom
        setReservedSeats((prev) => {
            const updated = { ...prev }
            if (!updated[film.id]) {
                updated[film.id] = {}
            }
            if (!updated[film.id][session.date]) {
                updated[film.id][session.date] = {}
            }
            updated[film.id][session.date][session.time] = [
                ...(updated[film.id][session.date][session.time] || []),
                ...session.seats,
            ]
```

```
        return updated
      })

      // Очищаємо обрані місця в selectedSeatsAtom
      setSelectedSeats({})

      navigate('/ticket');
    }

    return (
      <div className={styles.page}>
        <h2>Payment methods</h2>

        <div className={styles.serviceMethods}>
          <div className={styles.serviceItem}>
            <div className={styles.Zalo}>
              <img src={Zalo} alt=""/>
              <p>Zalo pay</p>
            </div>
            <img src={arrow} alt=""/>
          </div>
          <div className={styles.serviceItem}>
            <div className={styles.Shopee}>
              <img src={Shopee} alt=""/>
              <p>Shopee pay</p>
            </div>
            <img src={arrow} alt=""/>
          </div>
          <div className={styles.serviceItem}>
            <div className={styles.ATM}>
              <img src={ATM} alt=""/>
              <p>ATM pay</p>
            </div>
            <img src={arrow} alt=""/>
          </div>
        </div>

        <div className={styles.section}>
          <label>Card holder name</label>
          <input type="text" placeholder="Nguyen Van A"/>
        </div>

        <div className={styles.section}>
          <label>Card number</label>
          <input type="text" placeholder="1234 5678 9012 3456" maxLength={19}/>
        </div>

        <div className={styles.row}>
          <div className={styles.cardInfo}>
            <label>Expiry date</label>
            <input className={styles.expiryDate} type="text" placeholder="MM/YY"
maxLength={5}/>
```

```
            </div>
            <div className={styles.cardInfo}>
              <label>CVC</label>
              <input className={styles.cvv} type="text" placeholder="123" maxLength={3}/>
            </div>
          </div>

          <div className={styles.total}>
            <span>Total</span>
            <b>{session.price.toLocaleString()} UAH</b>
          </div>

          <button className={styles.payButton} onClick={handlePay}>Pay now</button>
        </div>
    )
}
```

ДОДАТОК Л

Лістинг файлу Profile.jsx

```
import styles from './index.module.scss'
import avatar from '../../assets/avatar.png'
import ticket from '../../assets/ticket-2.svg'
import cart from '../../assets/shopping-cart.svg'
import translate from '../../assets/translate.svg'
import lock from '../../assets/lock.svg'
import face from '../../assets/FaceID.svg'
import call from '../../assets/call.svg'
import sms from '../../assets/sms.svg'
import "./style.css";
import {useNavigate} from "react-router-dom";
export default function Profile() {
   const navigate = useNavigate()
   return (
      <div className={styles.profile}>
         <div className={styles.header}>
            <img src={avatar} alt="avatar" className={styles.avatar} />
            <div className={styles.userInfo}>
               <h2>Angelina</h2>
               <p className={styles.userdata}><img src={call} alt=""/>(704) 555–0127</p>
               <p className={styles.userdata}><img src={sms} alt=""/>angelina@example.com</p>
            </div>
            <span className={styles.edit}>✎</span>
         </div>

         <div className={styles.list}>
            <button  onClick={() => navigate("/ticket")}><img src={ticket} alt=""/>My ticket</button>
            <button><img src={cart} alt=""/>Payment history</button>
            <button><img src={translate} alt=""/>Change language</button>
            <button><img src={lock} alt=""/>Change password</button>
            <div className={styles.toggleRow}>
               <img src={face} alt=""/>Face ID / Touch ID
               <label className="switch">
                  <input type="checkbox"/>
                     <span className="slider"></span>
               </label>
            </div>
         </div>
      </div>
   )
}
```