МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ КОМП'ЮТЕРНИХ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра економічної кібернетики
та інформатики

МІЖДИСЦИПЛІНАРНА КУРСОВА РОБОТА

на тему:

Інформаційна система для бронювання квитків у театр

Студентки 4 курсу групи СА-41
спеціальності 124 «Системний аналіз»
Татарин В.М,
(прізвище та ініціали)

Керівник

_____

_____

(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Національна шкала _____

Кількість балів: _____Оцінка: ECTS _____

Члени комісії: _____(_____)
прізвище та ініціали підпис

_____(_____)
прізвище та ініціали підпис

_____(_____)
прізвище та ініціали підпис

м. Тернопіль – 2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ КОМП'ЮТЕРНИХ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра економічної кібернетики та інформатики

ЗАВДАННЯ НА МІЖДИСЦИПЛІНАРНУ КУРСОВУ РОБОТУ

група СА -41 курс4

Студентки Татарин В.М.

Тема міждисциплінароної курсової роботи:

Інформаційна система для бронювання квитків у театр

Основні розділи міждисциплінарної курсової роботи:

Вступ

1. Аналіз та вибір засобів розробки

2. Проектування інформаційної системи

3.Реалізація та тестування

Висновки

Рекомендована література:

1. Савчук Т.О. Організація баз даних і знань. − Вінниця: ВДТУ, 2000.

2. Безбородько І.Г. Все про бази даних - Львів 2013, 2014 - 276с.

3. Береза А.М.. Основи створення інформаційних систем: Навч. посібник. К.: КНЕУ, 2001, 214с.

Додатки

Дата видачі завдання "____"_____ 2024р.

Термін представлення роботи на кафедру "____"_____ 2024р.

Керівник міжисциплінарної курсової роботи _____

Структура

1. Титульний лист
2. Завдання на міждисциплінарну курсову роботу
3. Зміст
4. Вступ
5. Основні розділи
6. Висновки
7. Список використаної літератури
8. Додатки

Зміст

ВСТУП

Для українців відвідування театру давно стало невід'ємною частиною дозвілля. З розвитком технологій театральні кіоски перетворилися на сайти та платформи продажу квитків, пропонуючи користувачам можливість швидко придбати квитки та оплатити карткою. У сучасному світі театри не можуть існувати без веб-додатків, мобільних додатків або соціальних мереж. Саме тому інформаційна система для бронювання квитків може бути корисною для тих хто бажає поринути у світ мистецтва.

Метою даної курсової роботи є розробка інформаційної системи для театру, яка дозволяє автоматизувати процеси бронювання квитків, реєстрації користувачів, а також управління репертуаром. Ця система спрямована на підвищення ефективності роботи адміністрації театру, забезпечення зручності для відвідувачів та спрощення взаємодії між працівниками театру.

Об'єктом дослідження є інформаційна система для бронювання квитків у театри. Предметом дослідження є процес розробки інформаційної системи з графічним інтерфейсом користувача та підключеної до неї бази даних. У роботі розглядається концептуальна модель бази даних, основні особливості системи та технічні аспекти її реалізації.

Розроблена інформаційна система стане основою для подальшої автоматизації інших процесів у театрі, таких як управління репертуаром, нарахування заробітної плати, аналіз фінансових показників.

Розділ 1

АНАЛІЗ ТА ВИБІР ЗАСОБІВ РОЗРОБКИ

1.1Аналіз предметної області

У нашій країні існує досить багато театрів з різними напрямками (жанрами), і, природно, вистави, які проводяться, теж різні. Для створення інформаційної системи бронювання квитків з інтерфейсом для користувачів необхідно врахувати багато факторів: наприклад для визначення ціни важливим є театр в якому відбувається виступ, розташування місця в залі (його номер, ряд, категорія), дата та час проведення тощо. Тому перед початком розробки програмного продукту важливо розглянути основні аспекти цієї теми, такі як визначення інформаційної системи та її видів.

Інформаційна система (ІС) — це сукупність взаємозв'язаних компонентів, які забезпечують збирання, обробку, зберігання, аналіз і передачу інформації з метою підтримки прийняття рішень, координації, контролю, аналізу та візуалізації в межах організації чи середовища [1].

Інформаційні системи можуть включати як апаратні засоби (комп'ютери, мережі), так і програмні (додатки, бази даних), а також методи управління інформацією [9].

Далі розглянемо основні аспекти театральної сфери.

Театральне мистецтво охоплює широкий спектр творчих видів діяльності, які об'єднуються для створення живої вистави. Ця спільна форма мистецтва об'єднує такі елементи, як акторська майстерність, режисура, сценографія, дизайн костюмів, освітлення, звук тощо, щоб передати історію або викликати емоції .

Театр (з грецької – місце для видовищ) – 1) рід мистецтва, який відображає дійсність у художніх сценічних образах; 2) видовище, спектакль; 3) приміщення, де відбуваються вистави [5].

Театр має низку особливостей:

- Репертуарна політика: кожен театр має унікальний набір вистав, які можуть бути як прем’єрами, так і регулярно виконуваними постановками.

- Обмеженість ресурсів: кількість місць у залах, час для репетицій та постановок.
- Сезонність: багато театрів працюють із чіткими сезонами (наприклад, осінь-весна).
- Залежність від аудиторії: успіх театру залежить від залучення глядачів, популярності вистав і задоволення клієнтів.

Основні бізнес-процеси театру

1. Розробка репертуару:
    - Вибір постановок.
    - Планування прем’єр і повторів.
    - Залучення режисерів, акторів, технічного персоналу.
2. Організація вистав:
    - Планування розкладу вистав.
    - Визначення залу та місць для вистави.
    - Підготовка декорацій, костюмів, репетиції.
3. Продаж квитків:
    - Формування цінової політики (ціни на квитки залежно від ряду, місця).
    - Продаж квитків онлайн і в касах.
    - Резервування квитків для груп чи VIP-клієнтів.
4. Маркетинг та реклама:
    - Рекламні кампанії (афіші, соцмережі, сайти).
    - Програми лояльності (знижки для постійних клієнтів, акції).
    - Взаємодія зі спонсорами та партнерами.
5. Управління персоналом:
    - Розподіл обов’язків серед працівників.
    - Ведення графіків роботи.
    - Навчання та розвиток персоналу.
6. Аналіз і управління:
    - Моніторинг популярності вистав.

- Оцінка доходів та витрат.
- Планування покращень у роботі театру.

1.2 Огляд існуючих рішень та постановка задачі

Наразі в усьому світі доступні різноманітні сервіси для бронювання квитків на будь-які розважальні заходи, включно з театральними. Важливим кроком у розробці власного програмного продукту є пошук і аналіз існуючих аналогічних продуктів для подальшого формування функціональних вимог. Тому розглянемо переваги та недоліки деяких із них, а саме Concert.ua та Karabas.com.

Concert.ua — популярний в Україні сервіс для бронювання квитків на різноманітні заходи, зокрема вистави, концерти, фестивалі та інші культурні події [3].

Доступний веб-сайт і мобільний додаток. Далі описані основні переваги та недоліки цього сервісу.

Переваги : широкий вибір подій, зручний інтерфейс, доступність у різних містах України, оплата онлайн після реєстрації, можливий перегляд власних трансляцій, квитків, сертифікатів, створення списку бажань користувачів.

Недоліки: можливий обмежений вибір подій у деяких регіонах, можливі технічні проблеми, дещо складніший процес реєстрації в мобільному додатку порівняно з веб-сайтом тієї самої служби та обмеження лише однією мовою.

Karabas.com – це український сервіс для бронювання квитків на культурні та розважальні заходи, серед яких вистави, концерти, фестивалі, спортивні заходи тощо [4].

Доступний веб-сайт і мобільний додаток. Далі наведено переваги та недоліки .

Основні переваги : широкий вибір подій, зручний пошук і фільтрація, доступність у різних містах України, вибір мови для сайту та мобільного додатку, функція оплати після реєстрації онлайн.

Недоліки: Можливі перебої в роботі, складний процес реєстрації як сайту, так і програми.

Після виконання дослідження предметної області інформаційної системи для бронювання квитків у театри, та пошуку та аналізу аналогів, їх функцій, наступним кроком є визначення функціональних вимог програмного продукту цієї кваліфікаційної роботи. Інформаційна система бронювання квитків у театри повинна мати функціонал який в першу чергу відповідає темі та задовольняє її вимогам, крім того потрібен зручний та легкий інтерфейс для її користувачів

Функціональні вимоги:

- Збереження та управління даними про вистави, квитки, клієнтів, персонал.
- Резервування та продаж квитків із можливістю відображення статусу місць у реальному часі.
- Автоматичне оновлення інформації про заповнюваність залів після кожного продажу.
- Формування звітів про продажі та популярність вистав.

Інформаційна система для театру повинна враховувати всі аспекти його роботи: від організації вистав до взаємодії з глядачами. Ефективна автоматизація цих процесів дозволить театру:

- Забезпечити точне управління ресурсами (залами, персоналом, розкладом).
- Підвищити рівень обслуговування клієнтів завдяки зручному інтерфейсу для покупки квитків.

- Надати адміністрації театру інструменти для аналізу та стратегічного планування.

Це створить основу для стабільного розвитку театру та збільшення його популярності серед аудиторії

1.3 Огляд і обґрунтування засобів розробки

Вибір відповідних засобів розробки має суттєвий вплив на ефективність роботи системи, її масштабованість, безпеку та зручність для користувачів. Для створення інтуїтивно зрозумілого та стабільного інтерфейсу користувача, а також для забезпечення безперебійної роботи системи в умовах високої навантаженості, важливо правильно обрати як інструменти для розробки програмного забезпечення, так і технології для управління базами даних.

Мова програмування JavaScript

JavaScript — це високорівнева мова програмування, яка здебільшого використовується для розробки веб-додатків. Це одна з провідних мов для створення інтерактивних елементів на веб-сторінках, що дозволяє динамічно взаємодіяти з користувачами без перезавантаження сторінки. JavaScript працює у браузері вашого клієнта та дозволяє вам створювати швидкі та чутливі інтерфейси користувача для ваших веб-додатків. Завдяки своїй універсальності JavaScript також можна використовувати на стороні сервера за допомогою таких технологій, як Node.js, що дозволяє розробляти повноцінні веб-додатки за допомогою однієї мови програмування [5].

Переваги використання JavaScript:

1. Інтерактивність та зручність користувача: Завдяки JavaScript можна реалізувати інтерактивні елементи на веб-сайті, такі як анімовані розкладки, калькулятори для підрахунку вартості квитків, а також швидку обробку вибору місць у театрі в режимі реального часу. Це робить користувацький досвід більш зручним та приємним.

2. Масштабованість та гнучкість: Використовуючи JavaScript на серверній стороні через Node.js, можна створити масштабовану та ефективну серверну інфраструктуру для обробки запитів, керування базою даних і реалізації платіжних систем. Це дозволяє легко адаптувати систему для великих навантажень, що важливо для театральних сайтів з великою кількістю користувачів.

3. Широка підтримка бібліотек і фреймворків: Для JavaScript існує безліч бібліотек і фреймворків, таких як React.js, Vue.js чи Angular, що значно прискорюють розробку та допомагають створювати складні інтерфейси з високим рівнем інтерактивності. Це особливо корисно для розробки адаптивних інтерфейсів, які будуть зручні як на десктопах, так і на мобільних пристроях.

4. Підтримка роботи в реальному часі: JavaScript і Node.js дозволяють створювати системи, що підтримують реальний час. Це важливо для таких функцій, як оновлення доступних місць у залі, повідомлення користувачам про зміни в наявності квитків або статус їх замовлення.

5. Крос-платформенність: Оскільки JavaScript працює в браузерах, система продажу квитків може бути доступною на будь-яких пристроях, не потребуючи окремої адаптації під різні операційні системи. Це забезпечує універсальний доступ до системи як з комп'ютерів, так і з мобільних телефонів, що є важливим аспектом для сучасних веб-сервісів.

Мова розмітки HTML

HTML (HyperText Markup Language) — це стандартна мова розмітки, яка використовується для створення та структурування веб-сторінок. HTML дозволяє визначати структуру веб-вмісту, включаючи заголовки, абзаци, списки, посилання, зображення, таблиці та інші елементи, які складають основу веб-сторінки. Веб-браузер інтерпретує код HTML і відображає веб-сторінки користувачам [6].

Переваги використання HTML для системи продажу квитків театру:

1. Структуризація вмісту: HTML дозволяє чітко структурувати контент веб-сторінки, що особливо важливо для таких елементів, як розклад вистав,

інформація про квитки, деталі театру тощо. Кожен тип вмісту можна оформити за допомогою відповідних тегів, що забезпечує зручний та зрозумілий вигляд сторінки.

2. Адаптивність і сумісність: HTML працює на всіх веб-браузерах та пристроях, що дозволяє створювати систему продажу квитків театру, яка буде доступна як на мобільних телефонах, так і на десктопах.

3. Основи для інтеграції з іншими технологіями: HTML є основою для використання інших веб-технологій, таких як CSS (для стилізації) та JavaScript (для інтерактивності). Це дозволяє створювати функціональні та привабливі веб-додатки для продажу квитків.

Мова стилів CSS

CSS — це каскадні таблиці стилів, які використовуються для візуального форматування документа в мовах розмітки. CSS дозволяє визначати кольори, шрифти, верстку та інші аспекти вигляду сторінки. Основна ідея CSS полягає в відокремленні дизайну документа від його вмісту. CSS відповідає за оформлення і зовнішній вигляд, тоді як HTML зосереджений на змісті та логічній структурі документа.

Переваги використання CSS:

1. Полегшене оформлення та контроль: CSS дає змогу централізовано контролювати зовнішній вигляд веб-сторінки. Це важливо для таких сайтів, як система продажу квитків театру, де кожен елемент (кнопки, таблиці, форми) повинен мати однаковий стиль і вигляд. Завдяки CSS можна легко змінювати вигляд всього сайту, редагуючи лише стилі.

2. Адаптивність і мобільна сумісність: Використовуючи CSS, можна створювати адаптивний дизайн для системи продажу квитків театру, що дозволяє користувачам комфортно здійснювати покупки на будь-яких пристроях. Медіа-запити дозволяють зручно адаптувати сайт під різні розміри екранів, що важливо для користувачів мобільних телефонів.

3. Кастомізація інтерфейсу: CSS дозволяє значно покращити зовнішній вигляд інтерфейсу користувача, використовуючи ефекти, анімації та переходи.

Для системи продажу квитків театру це може включати анімації для кнопок, ефекти при наведенні на елементи, плавне завантаження контенту тощо. Це створює більш приємне враження від взаємодії з сайтом.

Платформа Visual Studio Code

Visual Studio Code (VS Code) — це популярний, безкоштовний та відкритий редактор коду, розроблений компанією Microsoft. Він підтримує широкий спектр мов програмування, таких як JavaScript, Python, Java, C++, HTML, CSS та багато інших. VS Code є потужним інструментом для розробки як веб-додатків, так і програмного забезпечення для різних платформ.

Переваги використання Visual Studio Code:

1. Підтримка веб-технологій: Оскільки система продажу квитків театру буде використовувати веб-технології, такі як HTML, CSS та JavaScript, Visual Studio Code є ідеальним вибором завдяки своїй підтримці цих мов програмування та численним розширенням для веб-розробки.

2. Інтеграція з іншими інструментами та сервісами: Для інтеграції з базами даних або сторонніми API (наприклад, для платіжних систем або перевірки доступних місць у залі) VS Code дозволяє налаштовувати безліч розширень, які значно спрощують процес інтеграції та взаємодії з іншими технологіями.

3. Простота налаштування і використання: VS Code має простий у використанні інтерфейс, що дозволяє швидко налаштувати середовище для веб-розробки. Його гнучкість дозволяє розробникам зосередитись на креативному процесі розробки, а не на складних налаштуваннях середовища.

Розділ 2

ПРОЕКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1 Проектування бази даних

Проектування бази даних є основним етапом у розробці будь-якої інформаційної системи. Воно передбачає створення структури даних, яка забезпечить ефективне зберігання, організацію та доступ до інформації. Процес проектування бази даних складається з кількох етапів, які включають аналіз вимог, розробку логічної та фізичної моделей, нормалізацію даних, а також оптимізацію продуктивності та забезпечення безпеки. Одним з перших етапів є концептуальне проектування, на якому формується уявлення про дані, що зберігатимуться в базі, а також їх взаємозв'язки [7].

Концептуальна модель бази даних — це абстрактне представлення структури бази даних, яке описує основні сутності, їхні атрибути та зв’язки між ними без прив’язки до будь-якої конкретної технології чи реалізації. Основна мета концептуальної моделі — представити бізнес-процеси та вимоги до даних на високому рівні, дозволяючи зрозуміти основні компоненти системи, не заглиблюючись у технічні аспекти [8].

В рамках проектування бази даних концептуальне проектування є критичним етапом, оскільки на цьому етапі визначаються основні сутності, атрибути та зв'язки між даними. Концептуальна модель дозволяє відобразити предметну область у вигляді абстрактної схеми, яка не залежить від конкретних технологій і СУБД. Вона фокусується на тому, що і як має бути збережено, представляючи високорівневу структуру даних, яка визначає, які об'єкти і процеси будуть підтримуватися в системі.

Основні елементи концептуальної моделі даних:

1. Сутності – представлення основних об’єктів або понять у системі, що потрібно відслідковувати.
2. Атрибути – характеристики або властивості кожної сутності.
3. Взаємозв’язки – описують взаємозв’язки між сутностями.

4. Ключі – ідентифікатори, які унікально ідентифікують кожен запис в сутності.

5. Класифікації та ієрархії – групування сутностей або атрибутів у класи та створення ієрархій.

6. Асоціації – визначення взаємозв’язків між сутностями та їх характер.

База даних “Театр” (theater) має 6 сутностей (таблиць), вони наступні: “Вистава” (Performance), “Роль у виставі” (RoleInPlay), “Людина” (Person), “Квиток” (Ticket), “Сеанс” (Session), “Зал” (Hall). Кожна з зазначених сутностей має власні атрибути, які необхідно ретельно дослідити. Саме тому наступною дією буде виявлення цих властивостей (атрибутів), їх опис (дані які вони зберігають, наявність ключа), а також типів даних, які зберігаються в них.

Атрибут сутності – це іменована характеристика, що є деякою властивістю сутності.

Атрибути, які належать сутності “Вистава” (Performance) наведено у табл. 2.1.

Таблиця 2.1 – опис сутності “Вистава”

Атрибут	Тип даних	Опис
Id	int	Зберігає індекс вистави. Первинний ключ
Name	nvarchar(100)	Зберігає назву вистави
Genre	nvarchar(50)	Зберігає жанр вистави
Category	nvarchar(50)	Зберігає категорію вистави

Опис сутності “Зал” (Hall) наведено у табл. 2.2.

Таблиця 2.2 – опис сутності “Зал”

Атрибут	Тип даних	Опис
Id	int	Зберігає індекс залу. Первинний ключ

Name	Nvarchar(50)	Зберігає назву залу
NumberOfSeats	int	Зберігає кількість місць

Опис сутності “Людина” (Person) складається з атрибутів розглянутих у табл. 2.3.

Таблиця 2.3 – опис сутності “Людина”

Атрибут	Тип даних	Опис
Id	int	Зберігає індекс людини. Первинний ключ
Name	Nvarchar(50)	Зберігає ім’я людини
Surname	Nvarchar(50)	Зберігає прізвище людини
Position	Nvarchar(50)	Зберігає посаду людина
Email	Nvarchar(50)	Зберігає електронну пошту
Phone	Nvarchar(30)	Зберігає номер телефону

Наступна сутність - “Роль у виставі ” (RoleInPlay), пов’язує дві інші описані вище: “Вистава” (Performance) та “Людина ” (Person). Дослідження її атрибутів наведено у табл. 2.4.

Таблиця 2.4 – опис сутності “Роль у виставі”

Атрибут	Тип даних	Опис
Id	int	Зберігає індекс ролі. Первинний ключ
Descr	Nvarchar(100)	Зберігає опис ролі у виставі
PerformId	int	Зберігає індекс вистави. Зовнішній ключ
PersonId	int	Зберігає індекс людини. Зовнішній ключ

Наступна таблиця (сутність), “Сеанс” (Session), так само як “ Роль у виставі” пов’язує дві інші сутності, а саме “Вистава” (Performance) та “Зал” (Hall). Опис цієї сутності наведено у табл. 2.5.

Таблиця 2.5 – опис сутності “Сеанс”

Атрибут	Тип даних	Опис
Id	int	Зберігає індекс сеансу. Первинний ключ
DateAndTime	datetime	Зберігає дату та час сеансу
HallId	int	Зберігає індекс залу. Зовнішній ключ
PerformId	int	Зберігає індекс вистави. Зовнішній ключ

Останньою шостою сутністю є “Квиток” (ticket), вона містить інформацію про всі квитки на вистави, а також пов’язує такі сутності як: “Сеанс” (Session) та “Людина” (Person). Її атрибути наведено у табл. 2.6

Таблиця 2.6 – опис сутності “Квиток”

Атрибут	Тип даних	Опис
Id	int	Зберігає індекс квитка. Первинний ключ
Price	int	Зберігає ціну квитка
NumOfSeat	int	Зберігає номер місця
State	Nvarchar(50)	Зберігає стан квитка
Quantity	int	Зберігає кількість квитків
SessionId	int	Зберігає індекс сеансу. Зовнішній ключ

PersonId	int	Зберігає індекс людини. Зовнішній ключ

Перетворення концептуальної моделі в логічну модель зазвичай виконується за формальними правилами. Цей крок можна в значній мірі автоматизувати.

На етапі логічного проєктування враховується специфіка конкретної моделі даних, але може не враховуватися специфіка конкретної СУБД.

Логічна модель є основою бази даних, вона повинна відображати взаємозв'язки між реляційними таблицями. Між реляційними таблицями можуть бути наступні типи зв'язків 1:1, 1:Б та Б:Б. Найбільш поширеним зв'язком є зв'язок 1:Б. Зв'язок 1:1 зустрічається рідше, тому що дані між якими існує такий тип зв'язку в переважній більшості випадків входять до складу однієї реляційної таблиці. Зв'язок Б:Б безпосередньо не підтримується в реляційних СУБД. Для реалізації такого зв'язку необхідно створювати додаткову реляційну таблицю, яка буде відігравати роль зв'язкової. Зв'язкова таблиця має обов'язково містити первинні ключі таблиць, між якими встановлюється зв'язок.

Побудова логічної моделі реляційної бази даних включає в себе декілька кроків. Ці кроки можуть бути описані за допомогою сутностей, їх атрибутів та відносин між ними.

Модель «сутність-зв'язок» (ER-модель) — модель даних запропонована П. Ченом, яка дозволяє описувати концептуальні схеми за допомогою узагальнених конструкцій блоків. ER-модель — це мета-модель даних, тобто засіб опису моделей даних. Хоча існує багато моделей представлення знань, моделі сутності-зв’язку є одним із найбільш практичних інструментів для послідовного представлення даних, незалежно від програмного забезпечення, яке реалізує дані. Важливо, що всі існуючі моделі даних (ієрархічні, мережеві, реляційні, об’єктні) можуть бути згенеровані з моделі «сутність-зв’язок», що робить її найпоширенішою моделлю даних [8].

Основні переваги ER-моделей:

- Наочність

- ER-моделі реалізовані в багатьох системах автоматизованого проектування баз даних
- Моделі дозволяють проектувати бази даних з великою кількістю об’єктів і атрибутів

Основні елементи ER-моделей:

- Об’єкти (сутності)
- Атрибути об’єктів
- Зв’язки між об’єктами

В описаній вище базі даних між усіма її сутностями (таблицями) наявний один тип зв’язку – один до багатьох.

Зв'язок "один до багатьох" (1:N) в контексті баз даних описує відношення між двома сутностями, де одному запису з першої сутності відповідає кілька записів з другої сутності, але кожному запису з другої сутності відповідає лише один запис з першої сутності.

Цей зв'язок встановлюється за допомогою ключового поля у першій таблиці (сутності), та не ключового поля у другій. Зазвичай першу таблицю називають батьківською, а другу – дочірньою.

Прикладом зв'язку "один до багатьох" є відношення між двома сутностями: “Квиток” (Ticket) та “Людина” (Person), яке зображено на рис. 2.1.

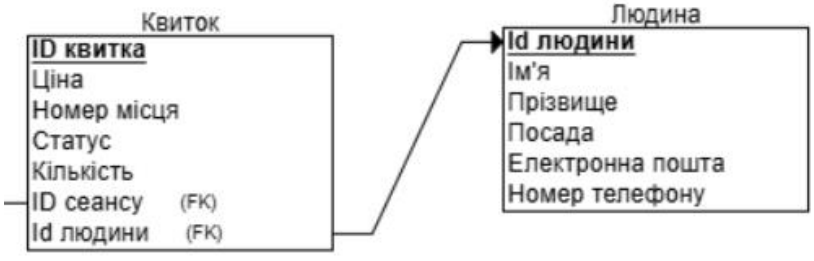

Рисунок 2.1 – Зв'язок між сутностями “Квиток” та “Людина”

Так як вище були досліджені всі вісім сутностей, їх атрибути та зв'язки між ними, була розроблена ER модель, яка наведена на рис. 2.2.

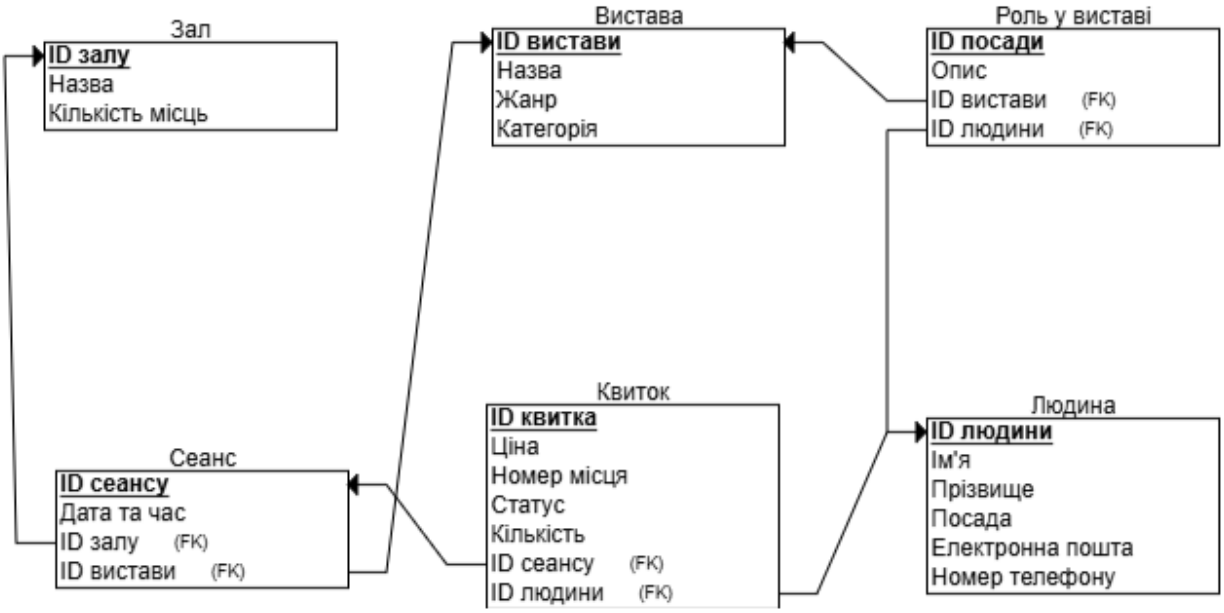

Рисунок 2.2 – ER модель БД “Театр”

Розділ 3

РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ

3.1 Розробка клієнтської частини

В попередніх розділах було проведено дослідження, вибрані необхідні інструменти розробки, спроектовано структуру та здійснено моделювання програмного продукту, а також визначено його функціонал. Це надає можливість розробити всі компоненти створюваного застосунку.

Клієнтська частина веб-застосунку була розроблена з використанням мови розмітки HTML, каскадних таблиць стилів CSS та мови програмування JavaScript. Головна мета полягала у створенні інтуїтивно зрозумілого та функціонального інтерфейсу для взаємодії користувачів із системою. Було реалізовано такі основні форми:

1. Форма реєстрації користувача

Форма реєстрації дозволяє новим користувачам створити обліковий запис у системі. Вона складається з таких елементів:

- Поля вводу для введення імені користувача, прізвища, електронної пошти та пароля.
- Поле для підтвердження пароля, яке забезпечує контроль його правильності.
- Кнопка "Зареєструватися", яка викликає функцію перевірки введених даних.

Функціональні особливості:

- Якщо поле пароля та підтвердження пароля не співпадають, відображається повідомлення про помилку.
- У разі успішної реєстрації виводиться повідомлення: *"Реєстрація успішна!"*.
- Якщо користувач із таким іменем вже існує, відображається повідомлення: *"Користувач із таким ім'ям вже існує."*

Ця форма інтегрується з масивом користувачів у JavaScript для збереження зареєстрованих облікових записів. Функція для реєстрації нових користувачів зображена на рис. 3.1

```
function handleRegistration() {
  const registrationForm = document.getElementById("registration-form");

  if (!registrationForm) {
      console.error("Форма реєстрації не знайдена!");
      return;
  }

  registrationForm.addEventListener("submit", (e) => {
      e.preventDefault();
      const username = document.getElementById("reg-username").value.trim();
      const password = document.getElementById("reg-password").value;
      const confirmPassword = document.getElementById("reg-confirm-password").value;
      const message = document.getElementById("register-message");
      if (message) {
          message.textContent = "";
          message.className = "message";
      }
      if (users.some((user) => user.username === username)) {
          message.textContent = "Користувач із таким ім'ям вже існує.";
          message.classList.add("error");
          return;
      }
      if (password !== confirmPassword) {
          message.textContent = "Паролі не співпадають.";
          message.classList.add("error");
          return;
      }
      users.push({ username, password });
      message.textContent = "Реєстрація успішна!";
      message.classList.add("success");

      registrationForm.reset();
  });
}
```

Рис. 3.1 – Програмний код функції handleRegistration

2. Форма входу користувача

Форма входу дозволяє зареєстрованим користувачам отримати доступ до функціоналу системи. Вона містить:

- Поля вводу для введення імені користувача та пароля.
- Кнопку "Увійти", яка перевіряє введені дані.

Функціональні особливості:

- Якщо ім'я користувача відсутнє у системі, виводиться повідомлення: *"Користувача з таким ім'ям не знайдено."*
- Якщо пароль неправильний, відображається повідомлення: *"Невірний пароль."*
- У разі успішного входу виводиться повідомлення: *"Вхід успішний!"*

Ця форма також інтегрується із масивом користувачів, де здійснюється пошук введеного імені та перевірка пароля. Код який ілюструє функцію для входу зареєстрованих користувачів наведено на рис.3.2.

```
223  function handleLogin() {
224      const loginForm = document.getElementById("login-form");
225      if (!loginForm) {
226          console.error("Форма входу не знайдена!");
227          return;
228      }
229      loginForm.addEventListener("submit", (e) => {
230          e.preventDefault();
231          const username = document.getElementById("login-username").value.trim();
232          const password = document.getElementById("login-password").value;
233          const message = document.getElementById("login-message");
234          if (message) {
235              message.textContent = "";
236              message.className = "message";
237          }
238          const user = users.find((user) => user.username === username);
239          if (!user) {
240              message.textContent = "Користувача з таким ім'ям не знайдено.";
241              message.classList.add("error");
242              return;
243          }
244          if (user.password !== password) {
245              message.textContent = "Невірний пароль.";
246              message.classList.add("error");
247              return;
248          }
249          message.textContent = "Вхід успішний!";
250          message.classList.add("success");
251      });
252  }
```

Рис 3.2 – Код функції handleLogin

3. Форма бронювання квитків

Ця форма надає користувачам можливість забронювати квитки на вистави. Вона включає:

- Вибір вистави зі списку доступних.
- Поле для автоматичного заповнення назви залу відповідно до вибраної вистави.
- Вибір дати, часу та місць у схемі залу.
- Поля для введення імені, електронної пошти та вибору способу оплати.

Функціональні особливості:

- Динамічне відображення схеми залу на основі обраної вистави.
- Заборона вибору зайнятих місць.
- Відображення повідомлення про успішне бронювання з переліком обраних місць.

Код функції для бронювання квитків зображено на рис 3.3

```
function setupBookingForm() {
  const playSelect = document.getElementById("play");
  const hallInput = document.getElementById("hall");
  const seatingChart = document.getElementById("seating-chart");
  playSelect.addEventListener("change", () => {
    const selectedPlay = playSelect.value;
    if (selectedPlay) {
      const hall = performances[selectedPlay].hall;
      hallInput.value = hall === "large" ? "Великий зал" : "Малий зал";
      generateSeatingChart(halls[hall]);
    } else {
      hallInput.value = "";
      seatingChart.innerHTML = "";
    }
  });
  document.getElementById("booking-form").addEventListener("submit", (e) => {
    e.preventDefault();
    const selectedSeats = Array.from(
      document.querySelectorAll("#seating-chart .selected")
    ).map((seat) => seat.textContent);
    if (selectedSeats.length === 0) {
      alert("Будь ласка, оберіть місце!");
      return;
    }
    alert(`Ваше бронювання успішно оформлено!\nОбрані місця: ${selectedSeats.join(", ")}`);
  });
}
```

Рис 3.3 – Код функції setupBookingForm

3.2 Тестування

В цьому підрозділі здійснюється тестування застосунку з метою перевірки відповідності функціональним вимогам, встановленим під час аналізу предметної області, а також його працездатності та надійності. Основні його етапи наведені нижче на рис. 3.4 - 3.10.

1. Тест реєстрації користувача.

При реєстрації можливі декілька варіантів подій: введення даних користувача який вже існує, невірне введення паролю та успішне створення облікового запису. В перших двох випадках з'являється вікно повідомлення про помилку, у третьому – про успішну операцію. Форма реєстрації та описані результати зображені на рис. 3.4 – 3.6 відповідно.

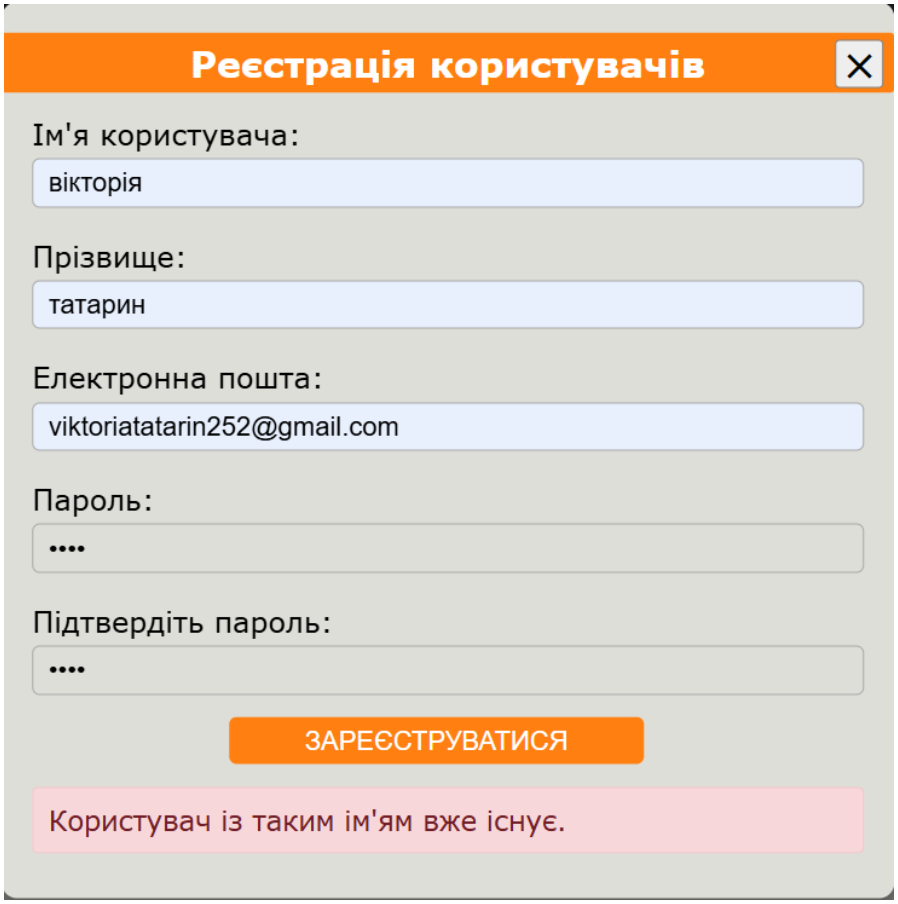

Рис 3.4 - Повідомлення про наявність користувача з цими даними

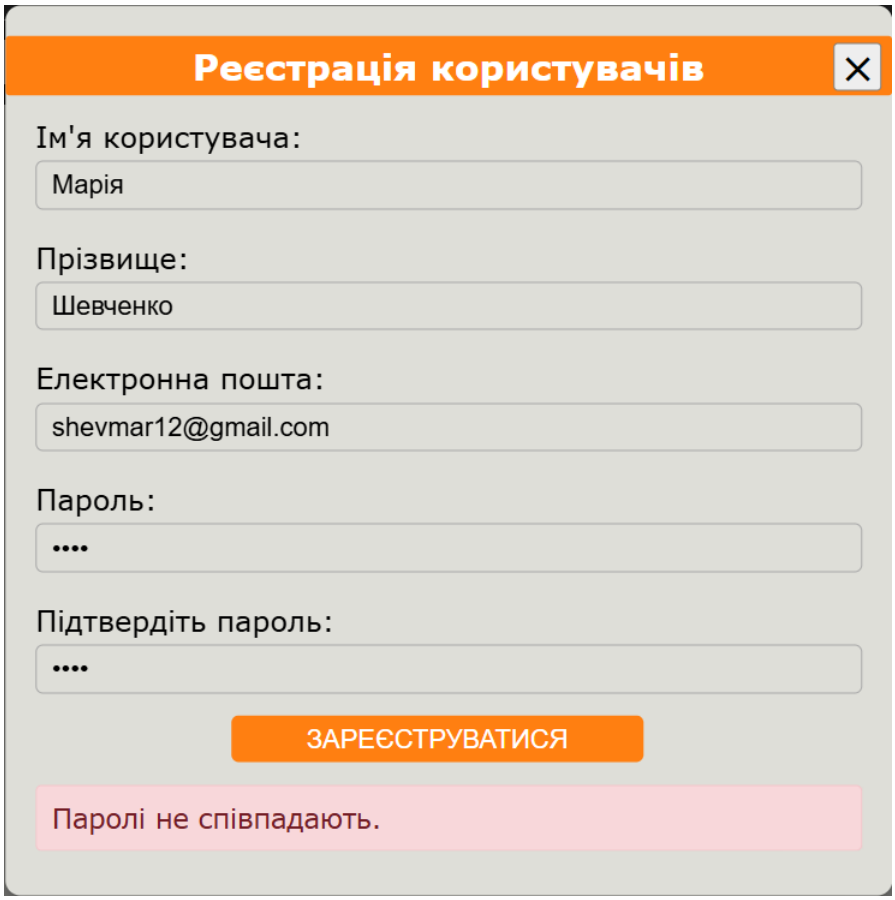

Рис 3.5 - Повідомлення про некоректний ввід паролю

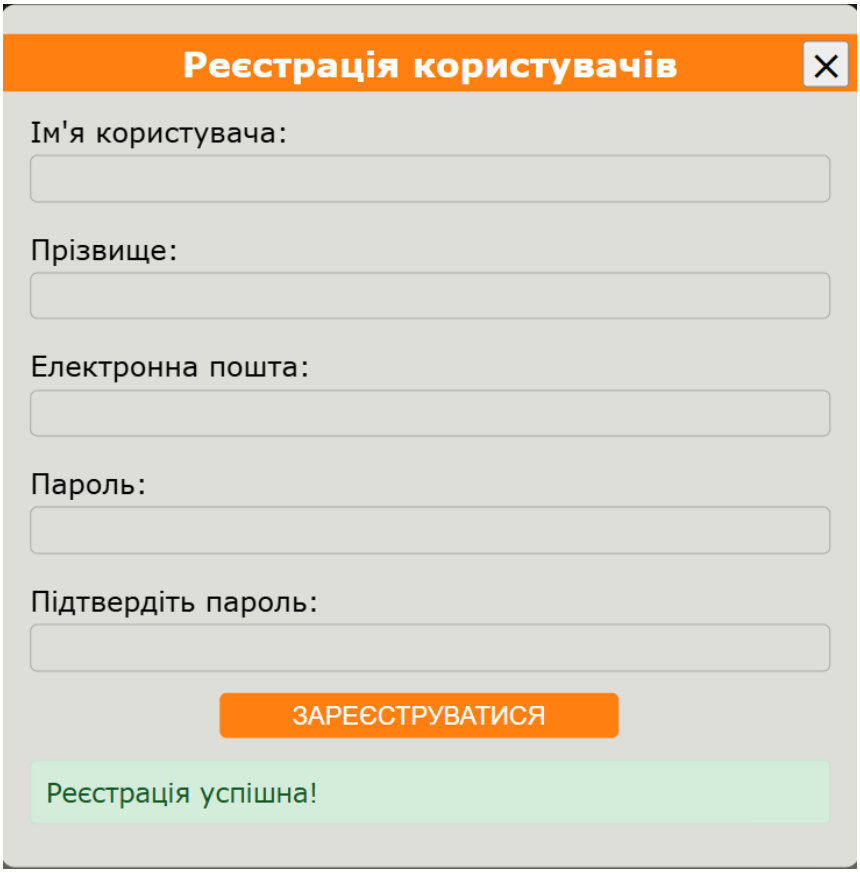

Рис 3.6 - Повідомлення про успішну реєстрацію

2. Тест авторизації користувача.

При авторизації можуть статися наступні варіанти подій: введення неправильних даних та успішний вхід до облікового запису. В першому випадку з’являється вікно повідомлення про помилку, у другому – про успішну операцію. Описані результати останніх двох і зображені на рис. 3.7 – 3.8 відповідно.

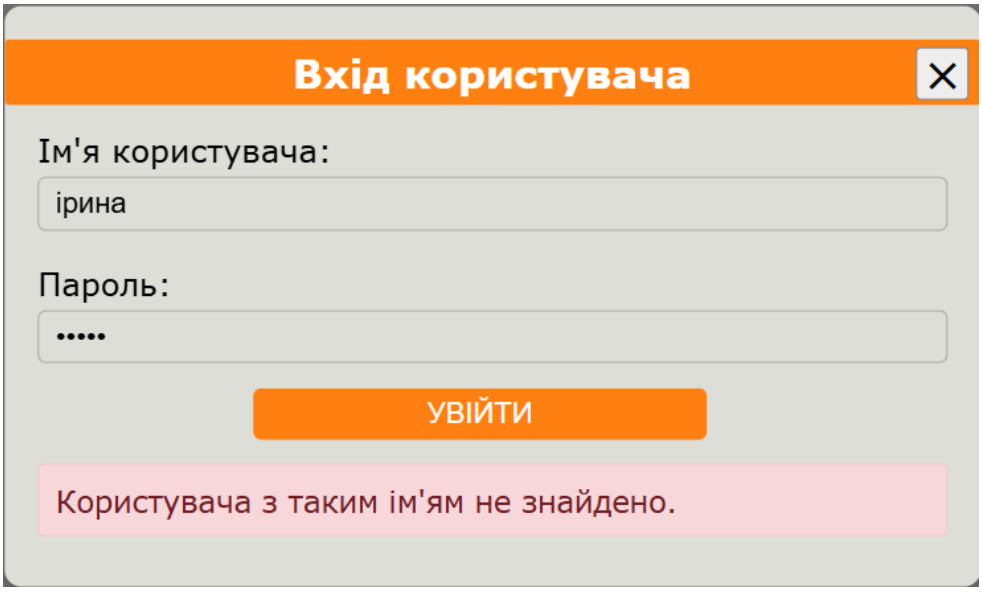

Рис 3.7 - Повідомлення про помилку при введенні даних

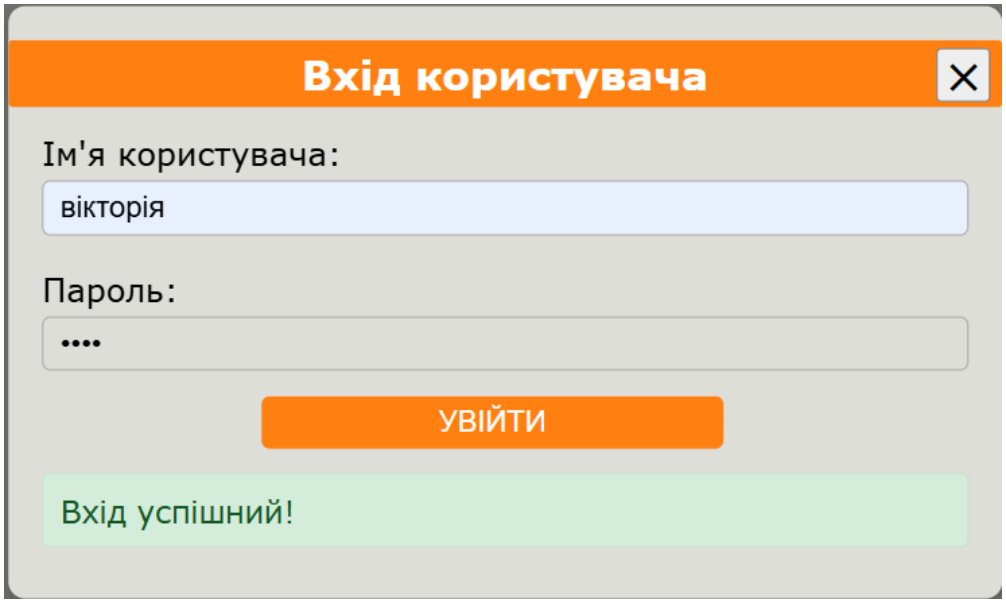

Рис. 3.8 – Повідомлення про успішну авторизацію

3. Тест здійснення броні квитка у театр.

Для того щоб здійснити бронь потрібно вибрати бажану виставу, після чого у вікні бронювання можна вибрати необхідні параметри для квитка: дата та час виступу, номер місця, ряду, та спосіб оплати.

При здійсненні броні можливі декілька варіантів подій: вказані параметри неправильні (у випадку незаповненості деяких полів) та бронювання здійснено успішно. Описані результати зображені на рис. 3.9 – 3.10 відповідно.

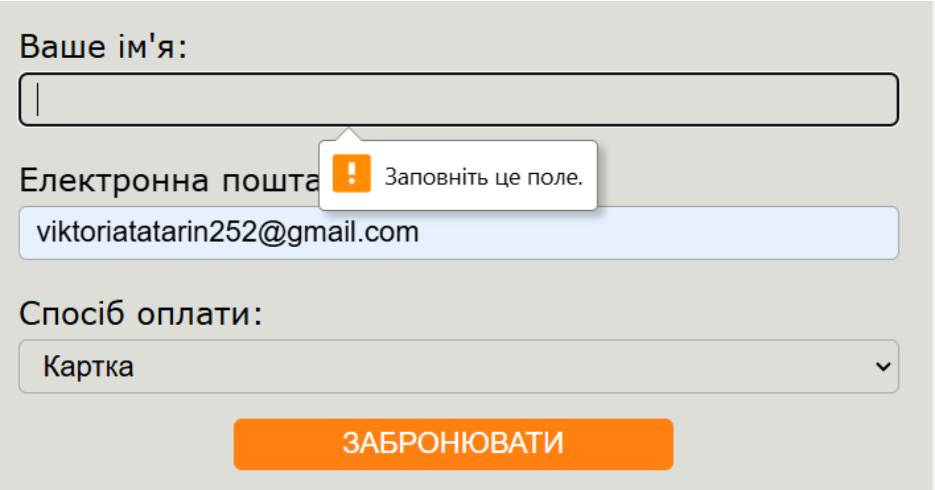

Рис 3.9 - Повідомлення про помилку при введенні даних

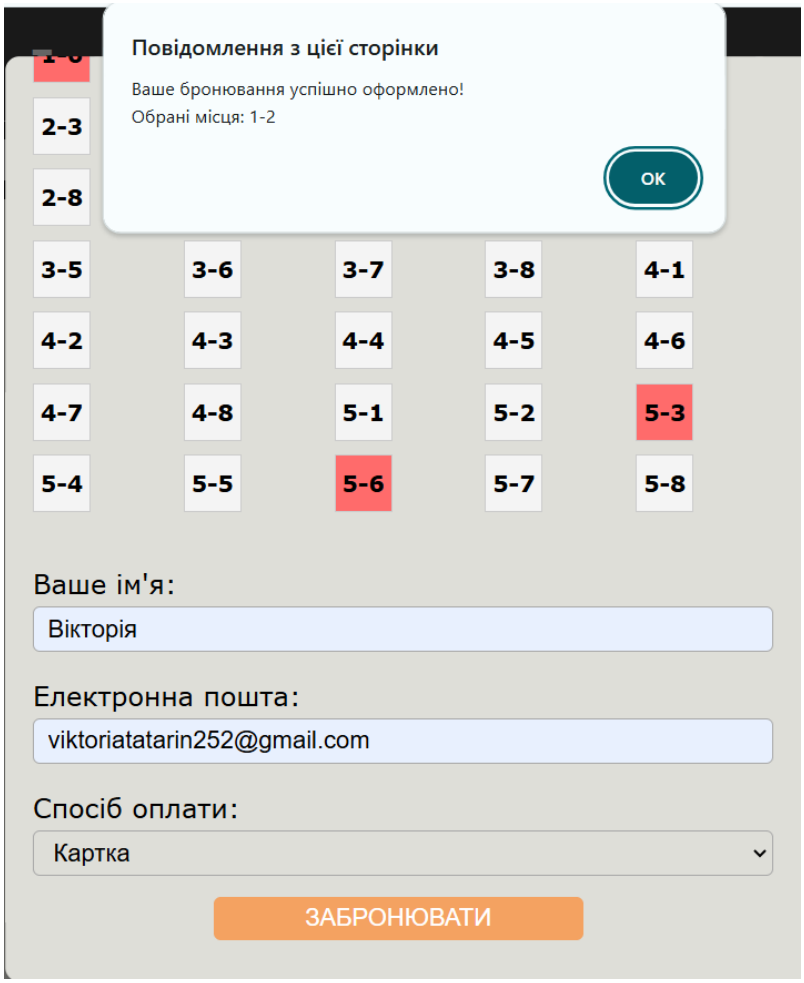

Рис. 3.10 – Повідомлення про успішне бронювання квитка

Проаналізувавши результати тестів можна зробити висновок що розроблений додаток відповідає визначеним функціональним вимогам, а також є працездатним, надійним, і має зручний інтерфейс користувача.

ВИСНОВКИ

Метою курсової роботи була розробка інформаційної системи для театру, яка забезпечує автоматизацію процесів взаємодії з відвідувачами, включаючи реєстрацію користувачів, їх авторизацію, а також бронювання квитків на вистави.

У результаті проведеної роботи була розроблена інформаційна система, яка містить зрозумілий та легкий у використанні інтерфейс користувача. Крім того була розроблена база даних “Театр”, в якій наявна інформація про театри, вистави та користувачів застосунку. Також інформаційна система містить функціонал для реєстрації, авторизації, здійснення та скасування броні на квиток, запису про стан квитка до бази даних, а також перегляду персональних даних.

Після отримання та аналізів результатів роботи можна зробити висновок що всі поставлені завдання успішно виконані, і мета досягнута. Розроблена інформаційна система є працездатною, надійною, містить необхідний функціонал та є легкою у використанні.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Бутенко Т.А., Сирий В.М. Інформаційні системи та технології[Електронний ресурс]: навч. посібник Харків, 2020. С. 34-53 URL: http://surl.li/qofmv

2. Словник театрознавчих термінів і понять [Електронний ресурс]: / укладачі Савченко І., Ліпницька І. Київ, 2021. 144 с. URL: https://enpuir.npu.edu.ua/bitstream/handle/123456789/37502/Slovnyk_Savchenko_Lipnytska.pdf?sequence=1

3. concert.ua. URL: https://concert.ua/uk

4. karabas. URL: https://karabas.com/ua/

5. Чому JavaScript — перспективна мова програмування? URL: https://dou.ua/forums/topic/35184/

6. Переваги HTML. URL: https://uk.education-wiki.com/1711441-advantages-of-html

7. Савчук Т.О. Організація баз даних і знань. − Вінниця: ВДТУ, 2000.

8. Безбородько І.Г. Все про бази даних - Львів 2013, 2014 - 276с.

9. Береза А.М.. Основи створення інформаційних систем: Навч. посібник. К.: КНЕУ, 2001, 214с.

ДОДАТОК А
Програмний код

```
document.addEventListener("DOMContentLoaded", () => {
  const modal = document.getElementById("modal");
  const modalBody = document.getElementById("modal-body");
  const closeButton = document.querySelector(".close-button");
  const tabs = document.querySelectorAll(".tab");

  const users = []; // Масив для збереження зареєстрованих користувачів
  const halls = {
    small: { rows: 3, cols: 5 }, // Малий зал: 3 ряди по 5 місць
    large: { rows: 5, cols: 8 }, // Великий зал: 5 рядів по 8 місць
  };
  const performances = {
    play1: { name: "Мартин Боруля", hall: "large" },
    play2: { name: "Гуцулка Ксеня", hall: "small" },
  };
  // Форми для вставки
  const forms = {
    booking: `
    <h2>Бронювання квитків</h2>
    <form id="booking-form">
      <label for="play">Виберіть виставу:</label>
      <select id="play" name="play" required>
        <option value="">-- Оберіть виставу --</option>
        <option value="play1">Мартин Боруля</option>
        <option value="play2">Гуцулка Ксеня</option>
      </select>

      <label for="hall">Зал:</label>
      <input type="text" id="hall" name="hall" readonly>

      <label for="date">Дата:</label>
      <input type="date" id="date" name="date" required>
      <label for="time">Час:</label>
```

```
  <select id="time" name="time" required>
    <option value="">-- Оберіть час --</option>
    <option value="18:00">18:00</option>
    <option value="20:00">20:00</option>
  </select>

  <label>Схема залу:</label>
  <div id="seating-chart"></div>

  <label for="name">Ваше ім'я:</label>
  <input type="text" id="name" name="name" required>

  <label for="email">Електронна пошта:</label>
  <input type="email" id="email" name="email" required>

  <label for="payment">Спосіб оплати:</label>
  <select id="payment" name="payment" required>
    <option value="card">Картка</option>
    <option value="cash">Готівка</option>
  </select>

  <button type="submit">Забронювати</button>
 </form>
`,
 registration: `
  <h2>Реєстрація користувачів</h2>
    <form id="registration-form">
      <label for="username">Ім'я користувача:</label>
      <input type="text" id="reg-username" name="username" required>
       <label for="surname">Прізвище:</label>
      <input type="text" id="surname" name="surname" required>
      <label for="email">Електронна пошта:</label>
      <input type="email" id="email" name="email" required>
      <label for="password">Пароль:</label>
      <input type="password" id="reg-password" name="password" required>
```

```
        <label for="confirm-password">Підтвердіть пароль:</label>
        <input type="password" id="reg-confirm-password" name="confirm-password" required>
        <button type="submit">Зареєструватися</button>
        <div id="register-message" class="message"></div>
      </form>
  `,
  repertoire: `
  <h2>Вхід користувача</h2>
     <form id="login-form">
       <label for="login-username">Ім'я користувача:</label>
       <input type="text" id="login-username" name="login-username" required>
       <label for="login-password">Пароль:</label>
       <input type="password" id="login-password" name="login-password" required>
       <button type="submit">Увійти</button>
       <div id="login-message" class="message"></div>
     </form>
    `,
};

// Відкрити модальне вікно
tabs.forEach((tab) => {
 tab.addEventListener("click", () => {
   const target = tab.dataset.modal;
   modalBody.innerHTML = forms[target];
   modal.style.display = "block";
   document.body.classList.add("modal-open");

   if (target === "booking") {
     setupBookingForm();
   }
   if (target === "registration") {
     handleRegistration();
   }
   if(target=="repertoire"){
     handleLogin();
```

```
    }
  });
});

// Закрити модальне вікно
closeButton.addEventListener("click", () => {
  modal.style.display = "none";
  document.body.classList.remove("modal-open");
});

window.addEventListener("click", (e) => {
  if (e.target === modal) {
    modal.style.display = "none";
    document.body.classList.remove("modal-open");
  }
});

// Налаштування для форми бронювання
function setupBookingForm() {
  const playSelect = document.getElementById("play");
  const hallInput = document.getElementById("hall");
  const seatingChart = document.getElementById("seating-chart");
  playSelect.addEventListener("change", () => {
    const selectedPlay = playSelect.value;
    if (selectedPlay) {
      const hall = performances[selectedPlay].hall;
      hallInput.value = hall === "large" ? "Великий зал" : "Малий зал";
      generateSeatingChart(halls[hall]);
    } else {
      hallInput.value = "";
      seatingChart.innerHTML = "";
    }
  });
  document.getElementById("booking-form").addEventListener("submit", (e) => {
    e.preventDefault();
```

```
    const selectedSeats = Array.from(
      document.querySelectorAll("#seating-chart .selected")
    ).map((seat) => seat.textContent);
    if (selectedSeats.length === 0) {
      alert("Будь ласка, оберіть місце!");
      return;
    }
    alert(`Ваше бронювання успішно оформлено!\nОбрані місця: ${selectedSeats.join(", ")}`);
  });
}

// Функція для створення схеми залу
function generateSeatingChart(hall) {
  const seatingChart = document.getElementById("seating-chart");
  seatingChart.innerHTML = ""; // Очищення схеми залу

  const { rows, cols } = hall;
  for (let row = 0; row < rows; row++) {
    for (let col = 0; col < cols; col++) {
      const seat = document.createElement("div");
      seat.textContent = `${row + 1}-${col + 1}`;
      if (Math.random() > 0.8) {
        seat.classList.add("occupied");
      }
      seatingChart.appendChild(seat);
    }
  }

  seatingChart.addEventListener("click", (e) => {
    if (e.target.tagName === "DIV" && !e.target.classList.contains("occupied")) {
      e.target.classList.toggle("selected");
    }
  });
}
function handleRegistration() {
```

```
    const registrationForm = document.getElementById("registration-form");

    if (!registrationForm) {
        console.error("Форма реєстрації не знайдена!");
        return;
    }
//Функція для реєстрації
    registrationForm.addEventListener("submit", (e) => {
        e.preventDefault();
        const username = document.getElementById("reg-username").value.trim();
        const password = document.getElementById("reg-password").value;
        const confirmPassword = document.getElementById("reg-confirm-password").value;
        const message = document.getElementById("register-message");
        if (message) {
            message.textContent = "";
            message.className = "message";
        }
        if (users.some((user) => user.username === username)) {
            message.textContent = "Користувач із таким ім'ям вже існує.";
            message.classList.add("error");
            return;
        }
        if (password !== confirmPassword) {
            message.textContent = "Паролі не співпадають.";
            message.classList.add("error");
            return;
        }
        users.push({ username, password });
        message.textContent = "Реєстрація успішна!";
        message.classList.add("success");

        registrationForm.reset();
    });
}
// Функція входу
```

```
function handleLogin() {
    const loginForm = document.getElementById("login-form");
    if (!loginForm) {
        console.error("Форма входу не знайдена!");
        return;
    }
    loginForm.addEventListener("submit", (e) => {
        e.preventDefault();
        const username = document.getElementById("login-username").value.trim();
        const password = document.getElementById("login-password").value;
        const message = document.getElementById("login-message");
        if (message) {
            message.textContent = "";
            message.className = "message";
        }
        const user = users.find((user) => user.username === username);
        if (!user) {
            message.textContent = "Користувача з таким ім'ям не знайдено.";
            message.classList.add("error");
            return;
        }
        if (user.password !== password) {
            message.textContent = "Невірний пароль.";
            message.classList.add("error");
            return;
        }
        message.textContent = "Вхід успішний!";
        message.classList.add("success");
    });
}
})
```