МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ ЕКОНОМІКИ І ТЕХНОЛОГІЙ

ННІ/факультет	Інформаційних технологій
Кафедра	Інформатики і прикладного програмного забезпечення
Спеціальність	Інженерія програмного забезпечення
Форма навчання	Денна

КВАЛІФІКАЦІЙНА БАКАЛАВРСЬКА РОБОТА[1]

Мацака Даніїла Дмитровича
(прізвище, ім'я, по батькові здобувача)

на тему

Розробка програмного забезпечення бронювання квитків для кінотеатру
(повна назва теми)

за матеріалами праць провідних спеціалістів з розробки ПЗ та проектування БД
(повна назва бази дослідження)

науковий керівник д.т.н., професор _____ О.С. Зеленський
(наук. ступінь, вчене звання) *(підпис)* *(прізвище, ініціали)*

Робота допущена до захисту в ЕК
Протокол засідання кафедри
від 11.06.2025р. № 12
Завідувач кафедри _____
(підпис)

д.т.н., професор Зеленський О.С.
Наук. ступень, вчене звання *Ініціали, прізвище*

Кривий Ріг – 2025

[1] Назва кваліфікаційної роботи відповідно до ОПП

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ ЕКОНОМІКИ І ТЕХНОЛОГІЙ

ННІ/факультет	Інформаційних технологій
Кафедра	Інформатики і прикладного програмного забезпечення
Спеціальність	Інженерія програмного забезпечення
Форма навчання	Денна

«ЗАТВЕРДЖУЮ»
Завідувач кафедри _____ Зеленський О.С.
(підпис) (Прізвище, ініціали)
«11» червня 2025 року

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ БАКАЛАВРСЬКУ РОБОТУ[2]

1. Тема роботи_Розробка програмного забезпечення бронювання квитків для кінотеатру

Керівник роботи____д.т.н., професор____О.С.Зеленський_____
затверджені наказом закладу вищої освіти від «04» квітня 2025 р. № 222-ст

2. Строк подання здобувачем роботи до «09» червня 2025 р.

3. Зміст кваліфікаційної роботи, об'єкт, предмет та мета дослідження:

Розділ 1. Аналіз предметної області та постановка задачі

Розділ 2. Розробка алгоритму розв'язання задачі

Розділ 3. Організація інформаційного забезпечення

Розділ 4. Розробка програмного забезпечення та веб-сайту

Об'єкт дослідження: Процес електронного бронювання квитків у кінотеатрах

Предмет дослідження: Програмні засоби та інформаційні технології, що забезпечують функціонування автоматизованої системи бронювання квитків.

Мета кваліфікаційної роботи: *Розробка програмного забезпечення для автоматизації процесу бронювання квитків до кінотеатр*.

5. Дата видачі завдання «04» квітня 2025 р.

[2] Назва кваліфікаційної роботи відповідно до ОПП

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів МДР	Строк виконання етапів роботи	Відмітка керівника про виконання етапів (дата, підпис)
1	Підготовка розділу 1	04.04.2025-13.04.2025	
2	Підготовка розділу 2	14.04.2025-26.04.2025	
3.	Підготовка розділу 3	27.04.2025-15.05.2025	
4	Підготовка розділу 4	16.05.2025-08.06.2025	
5	Реєстрація завершеної кваліфікаційної роботи	09.06.2025	Реєстраційний №___ «09»червня 2025 р.
6	Отримання відгуку від наукового керівника	10.06.2025	
7	Подання кваліфікаційної роботи на перегляд завідувачу кафедри	11.06.2025	
8	Отримання зовнішньої рецензії	12.06.2025	
9	Попередній захист кваліфікаційної роботи на кафедрі	13.06.2025	
10	Підготовка до захисту в ЕК	16.06.2025-21.06.2025	

Завдання підготував науковий керівник _____ ___Зеленський О.С.__
(підпис) (прізвище та ініціали)

Завдання одержав _____ ___Мацак Д.Д._____
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

на кваліфікаційну бакалаврську роботу

«Розробка програмного забезпечення бронювання квитків для кінотеатру»

Мацака Данїїла Дмитровича

Кваліфікаційна бакалаврська робота на здобуття освітньо-кваліфікаційного рівня бакалавра за спеціальністю 121 «Інженерія програмного забезпечення» - Державний університет економіки і технологій), Кривий Ріг, 2025.

У даній роботі розроблено інформаційну систему для автоматизації процесу бронювання квитків у кінотеатрі. Система включає функціонал додавання фільмів, управління сеансами та бронюванням місць. Програмний продукт створено з використанням мови програмування C# у середовищі Windows Forms, а для збереження даних застосовано реляційну базу даних Microsoft SQL Server. Для взаємодії з базою даних використано технологію ADO.NET. Реалізовано зручний інтерфейс користувача, який забезпечує ефективну роботу з даними про фільми, сеанси та бронювання.

Ключові слова: бронювання квитків, кінотеатр, база даних, ADO.NET, C#, Windows Forms.

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

ASP.NET	Технологія створення веб-застосунків від Microsoft.
SQL	Мова структурованих запитів до бази даних.
C#	Об’єктно-орієнтована мова програмування.
БД	База даних.
ПЗ	Програмне забезпечення.

ЗМІСТ

ВСТУП

У сучасному світі автоматизація процесів у сфері обслуговування відіграє важливу роль. Однією з таких сфер є кіноіндустрія, зокрема процес бронювання квитків у кінотеатр. З метою оптимізації взаємодії між клієнтом та адміністрацією кінотеатру було створено систему, яка дозволяє бронювати квитки через програму або веб-сайт.

Метою даної кваліфікаційної роботи є розробка інформаційної системи бронювання квитків, яка включає у себе програму для адміністраторів кінотеатру, веб-сайт для користувачів та базу даних для зберігання інформації про зали, сеанси, квитки, клієнтів і бронювання.

Об’єктом дослідження є процес електронного бронювання квитків. Предметом дослідження є розробка програмних засобів для цього процесу.

Система має забезпечити швидке та зручне бронювання квитків, спрощення управління даними, зменшення помилок при обробці замовлень та підвищення ефективності обслуговування клієнтів.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1. Вступ до предметної області

У сучасному цифровому середовищі, де темп життя стрімко зростає, а потреба в автоматизації процесів стає дедалі актуальнішою, особливе значення набувають інформаційні системи, які спрощують повсякденні операції користувачів. Однією з таких повсякденних дій є відвідування кінотеатрів. І хоча похід до кіно - це в першу чергу форма дозвілля, технічна сторона процесу організації перегляду фільмів вимагає чітко продуманого механізму взаємодії між глядачем та кінотеатром.

Традиційні способи замовлення квитків, такі як черги в касі або телефонні дзвінки, є не лише застарілими, але й незручними для сучасного користувача. Із розвитком мобільних технологій, онлайн-сервісів та широкого доступу до Інтернету виникає потреба в ефективних цифрових рішеннях, що дозволяють автоматизувати процес бронювання квитків і покращити взаємодію з клієнтами.

Таким чином, система бронювання квитків у кінотеатр повинна забезпечувати не лише базову функціональність покупки квитка, а й відповідати ряду вимог щодо зручності, швидкодії, безпеки, масштабованості та адаптивності до потреб користувача. Реалізація такої системи є важливим кроком до цифровізації культурної сфери та підвищення ефективності бізнес-процесів кінотеатрів.

1.2. Опис проблеми

Традиційні способи придбання квитків у кінотеатрах, такі як покупка безпосередньо у касі, пов’язані з рядом проблем: обмежений час роботи кас, черги, відсутність можливості вибору місця заздалегідь, а також людський фактор при обробці замовлень. Це викликає незручності у відвідувачів і втрати для кінотеатрів через неоптимальне управління місцями.

Для вирішення цих проблем потрібна система, що дозволить:

- Оперативно переглядати наявні фільми та їхні сеанси;
- Вибирати конкретний сеанс для перегляду;
- Вибирати місця у залі з урахуванням вже заброньованих місць;
- Бронювати квитки онлайн із підтвердженням по телефону;
- Забезпечувати безпеку та валідність введених даних.

Ці проблеми не лише знижують рівень сервісу для кіноглядачів, але й негативно впливають на бізнес-ефективність кінотеатрів: втрачається прибуток, погіршується репутація, підвищується рівень невдоволених клієнтів.

1.3. Постановка задачі

Основна мета дипломного проєкту - розробити інформаційну систему для автоматизації процесу бронювання квитків у кінотеатр, яка дозволяє користувачу самостійно обирати фільм, сеанс, зал, місце у залі та оформлювати бронювання в зручний спосіб.

На основі аналізу предметної області було сформульовано основні задачі для реалізації інформаційної системи бронювання квитків у кінотеатр:

1) Розробити веб-застосунок із сучасним, інтуїтивно зрозумілим інтерфейсом користувача, що забезпечить комфортний вибір фільмів, сеансів та місць у залі.

2) Організувати збереження даних про фільми, зали, сеанси та бронювання в реляційній базі даних.

3) Забезпечити динамічне відображення інформації про заброньовані місця, щоб уникнути конфліктів при бронюванні.

4) Реалізувати механізми валідації даних на стороні сервера і клієнта для коректної обробки замовлень.

5) Створити систему, що дозволяє користувачу підтвердити бронювання, ввівши телефон та номер карти.

1.4. Огляд існуючих рішень

На сучасному ринку існує ряд сервісів, що пропонують онлайн-бронювання квитків: KARABAS, Concert.ua, Multiplex, Planetakino тощо. Однак більшість із них є великими централізованими платформами, які або не підходять для локального використання, або мають обмежену адаптацію до конкретного кінотеатру. Крім того, багато таких систем не дозволяють власнику кінотеатру повноцінно керувати внутрішніми процесами через сторонню інтеграцію або обмежений доступ до налаштувань.

Таким чином, виникає потреба у створенні власного рішення - кастомізованої системи, яка буде адаптована до потреб конкретного закладу, дозволить оптимізувати внутрішні бізнес-процеси і забезпечити високий рівень комфорту для користувача.

1.5. Вибір технологій

Для реалізації проєкту було обрано стек технологій на основі платформи ASP.NET Core MVC:

- ASP.NET Core MVC - фреймворк для створення веб-застосунків із розділенням логіки, представлення та контролерів;
- Entity Framework Core - ORM для роботи з реляційною базою даних SQL Server, що спрощує маніпуляції з даними;
- SQL Server - СУБД для збереження структурованих даних про фільми, зали, сеанси і бронювання;
- Razor Pages - шаблони для генерації HTML із підтримкою динамічного контенту;
- HTML, CSS, JavaScript - технології для розробки інтерфейсу користувача, включно з динамічним вибором місць.

1.6. Основні функціональні вимоги

Проєкт повинен реалізовувати такі функції:

- Перегляд списку фільмів з можливістю фільтрації за датами;
- Перегляд детальної інформації про фільм;
- Відображення розкладу сеансів конкретного фільму;
- Вибір сеансу для бронювання;
- Візуалізація місць у залі із позначенням зайнятих;
- Вибір місця, введення контактної інформації та оформлення бронювання;
- Валідація та обробка помилок.

1.7. Структура даних

У базі даних реалізовано такі основні таблиці (Рис. 1.1):

- Movies - фільми з атрибутами: Id, Title, Genre, DurationMinutes, PosterUrl;
- Halls - зали кінотеатру, містять інформацію про назву та кількість місць;
- Showtimes - сеанси, з прив’язкою до фільму та залу, з часом початку;
- Bookings - бронювання, що містять інформацію про сеанс, номер місця, контактні дані користувача;
- Cinemas - містить інформацію з фільмами на які можна придбати квиток;
- Orders - згенеровані замовлення, забронювані місця які мають відповідні атрибути даних користувача який зробив бронювання.

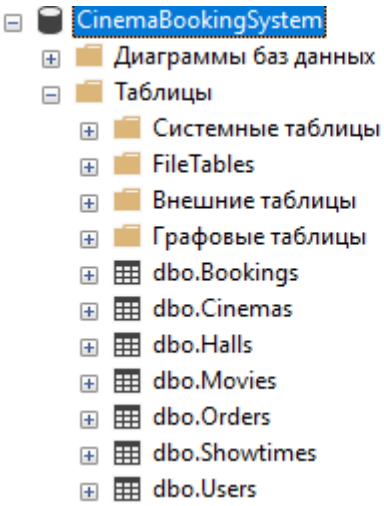

Рис. 1.1. Структура бази даних

1.8. Опис ключових сценаріїв використання

1) Користувач заходить на головну сторінку, де бачить список фільмів з доступними сеансами на найближчі дні.

2) За допомогою вкладок обирає дату для перегляду сеансів.

3) Натискає на фільм або час сеансу - потрапляє до сторінки вибору місця.

4) Вибирає вільне місце у залі, вводить телефон і номер карти.

5) Підтверджує бронювання - отримує повідомлення про успішне або неуспішне бронювання.

Висновок до розділу

У результаті проведеного аналізу предметної області було визначено основні проблеми та виклики, які постають перед сучасними кінотеатрами при організації процесу бронювання квитків. На сьогоднішній день індустрія розваг зазнає суттєвих змін, зокрема внаслідок цифрової трансформації та зміни поведінки споживачів. Глядачі очікують на зручність, швидкість та інтуїтивність у процесі придбання квитків, що вимагає від кінотеатрів використання сучасних інформаційних систем, які здатні відповідати цим потребам.

Під час аналізу було виявлено, що багато традиційних методів - зокрема, фізичне придбання квитків у касі або попереднє резервування без підтвердження онлайн - мають суттєві обмеження. Серед них: незручність для користувача, затримки у підтвердженні бронювання, висока ймовірність помилок при введенні даних, неможливість гнучкого вибору місця, відсутність історії замовлень і персоналізації. Усе це створює підґрунтя для необхідності розробки автоматизованої системи, яка б вирішувала ці недоліки.

На основі проведеного аналізу було сформульовано основну мету проєкту - створення ефективної, масштабованої та зручної для кінцевого користувача системи бронювання квитків у кінотеатр. Ця система має підтримувати ключовий функціонал, включаючи перегляд фільмів, розкладу сеансів, вибір залу та місця, підтвердження бронювання, введення контактних та платіжних даних, а також можливість адміністрування з боку персоналу закладу.

У рамках аналізу також було визначено основні категорії користувачів системи. З одного боку, це кінцеві користувачі - глядачі, які здійснюють бронювання квитків. З іншого боку - адміністратори кінотеатру, які керують базою фільмів, створюють сеанси, відстежують наявність місць та контролюють стан бронювань. Таким чином, система повинна бути двосторонньою - з окремими інтерфейсами та правами доступу для кожної групи користувачів. Це забезпечить безпеку, зручність і відповідність потребам обох сторін.

У процесі проєктування було сформульовано перелік функціональних вимог, які включають:

- можливість додавання, редагування та видалення фільмів і сеансів;
- візуальне представлення розкладу;
- інтерактивний вибір місць у залі;
- перевірку наявності вільних місць;
- обробку запитів на бронювання;
- збереження інформації про користувачів і замовлення;
- формування звітів для адміністрації;
- забезпечення цілісності та узгодженості даних у базі.

Окрім функціональних вимог, було також окреслено низку нефункціональних характеристик, яких необхідно дотримуватися при реалізації системи: висока швидкодія, зручність інтерфейсу, масштабованість (зокрема у разі розширення мережі кінотеатрів), інформаційна безпека, мінімізація людського фактору, підтримка сучасних браузерів та операційних систем, а також готовність до подальшої інтеграції з мобільними платформами.

Після аналізу вимог і архітектурних варіантів було прийнято рішення на користь гібридної моделі: створення настільного застосунку для адміністратора з використанням технології WinForms та розробка веб-інтерфейсу для кінцевих користувачів на базі ASP.NET Core MVC. Такий підхід дозволяє розділити обов'язки між двома рівнями: внутрішньою системою управління (Back Office) та зовнішнім інтерфейсом (Front Office). Це, у свою чергу, підвищує надійність, безпеку та зручність розробки.

Для зберігання даних обрано реляційну СУБД Microsoft SQL Server, яка відповідає вимогам стабільності, масштабованості та гнучкого проєктування структури бази. Цей вибір обґрунтований також можливістю інтеграції з платформами Microsoft, зокрема через використання ADO.NET, LINQ to SQL або Entity Framework у середовищі C#.

Отже, обрана технологічна платформа - поєднання ASP.NET Core MVC, WinForms, C# і SQL Server - забезпечує не лише високу продуктивність та простоту підтримки, але й створює гнучкий фундамент для подальшого розвитку системи. У майбутньому до неї може бути підключено інші сервіси, наприклад, платіжні шлюзи, повідомлення про сеанси, API зовнішніх партнерів (наприклад, розкладів фільмів) тощо.

Також важливо наголосити, що під час аналізу було враховано потенційні ризики та виклики, які можуть виникнути під час розробки та впровадження системи. До них належать: забезпечення цілісності даних у багатокористувацькому середовищі, необхідність синхронізації між настільним застосунком і веб-інтерфейсом, а також потреба в резервному копіюванні бази даних. Ці питання

планується розглядати у відповідних розділах, присвячених реалізації функціоналу та тестуванню системи.

Таким чином, підсумовуючи результати першого розділу, можна зробити висновок, що проведений аналіз предметної області дозволив сформувати чітке бачення завдань, які має вирішити система бронювання квитків, та визначити необхідні функціональні і технологічні компоненти. Отримані висновки слугуватимуть основою для реалізації наступних етапів розробки, що будуть детально описані у подальших розділах дипломної роботи

РОЗДІЛ 2
РОЗРОБКА АЛГОРИТМУ РОЗВ'ЯЗАННЯ ЗАДАЧІ

У цьому розділі розглянуто логічну та алгоритмічну побудову програмного забезпечення, призначеного для автоматизації процесу бронювання квитків у кінотеатрі. Йдеться про структуру та взаємозв'язок компонентів, які забезпечують функціонування системи, а також про логіку обробки даних, починаючи з введення нової інформації в систему й завершуючи бронюванням місць користувачем.

2.1. Формалізація задачі

Процес розробки будь-якої інформаційної системи розпочинається з чіткої постановки задачі, її формалізації та подальшого опису основних вимог до функціоналу. У межах даного дипломного проєкту задача полягає у створенні системи автоматизації бізнес-процесів кінотеатру, зокрема процесу бронювання квитків. Це завдання охоплює як внутрішні потреби адміністраторів закладу, так і потреби кінцевих користувачів - глядачів, які бажають забронювати квиток онлайн або за допомогою персоналу каси.

Основна мета - створити таку інформаційну систему, яка дозволить значно зменшити ручну працю, підвищити швидкість обслуговування клієнтів, мінімізувати помилки в роботі з розкладом, а також надати зручний та інтуїтивний інтерфейс як для адміністративного персоналу, так і для відвідувачів. Формалізація задачі полягає в описі основних компонентів системи, їхніх функцій та логічних зв'язків між ними, що надалі дозволить реалізувати ефективний програмний продукт.

Задача, яку вирішує система, охоплює кілька ключових бізнес-процесів. Зокрема:

- Додавання та зберігання інформації про фільми, зали та сеанси. Система повинна забезпечувати можливість адміністратору створювати нові записи

про фільми, вказуючи їхню назву, опис, тривалість, постер тощо. Окрім того, адміністратор повинен мати можливість створювати інформацію про зали (із зазначенням кількості рядів і місць), а також формувати розклад показів (сеансів), вказуючи дату, час і пов’язуючи їх із відповідним фільмом та залом.

- Створення місць у залах, їх візуалізація та зміна статусу. Для кожного залу необхідно автоматично або вручну генерувати схему місць. Важливо, щоб ці місця були представлені у вигляді графічної сітки, де кожне місце має певний статус: вільне, заброньоване або недоступне (наприклад, через ремонт або технічні причини). Система має дозволяти змінювати статус місць - як автоматично (при бронюванні), так і вручну (через інтерфейс адміністратора).
- Відображення наявних сеансів та вільних місць для користувача. Глядач має можливість переглядати список доступних сеансів за обраною датою, фільтрувати їх за назвою фільму або часом. При виборі конкретного сеансу користувач бачить візуальне представлення залу та може оцінити, які місця залишаються доступними для бронювання. Цей інтерфейс має бути простим, зрозумілим і пристосованим для роботи з мобільних пристроїв.
- Механізм бронювання квитків через вибір місця. Ключова функціональність системи - можливість для користувача вибрати місце у залі та підтвердити бронювання. Після цього місце переходить у статус «заброньоване» і стає недоступним для інших користувачів. У системі повинна бути реалізована логіка перевірки наявності конфліктів - тобто заборона на подвійне бронювання одного і того ж місця.

Таким чином, формалізована задача системи полягає у створенні інтегрованої програми, яка автоматизує наступні процеси:

1) Адміністрування даних - додавання, редагування та видалення фільмів, залів і сеансів.

2) Побудова структури залів - автоматичне або ручне створення місць, з можливістю їх подальшої візуалізації.

3) Виведення розкладу - формування списку актуальних сеансів для користувачів.

4) Процедура бронювання - вибір місця, фіксація бронювання, недопущення конфліктів.

З логічної точки зору, система повинна підтримувати цілісність даних, синхронізацію між адміністративною і клієнтською частиною, а також дотримання бізнес-правил (наприклад, неможливість забронювати минулий сеанс або місце, яке вже зайняте). Усі операції мають бути чітко формалізовані у вигляді алгоритмів, що визначають умови виконання дій і послідовність кроків.

Оскільки система працює з базою даних, важливо забезпечити належний рівень абстракції між інтерфейсом і збереженими даними. Це дозволить не лише уникнути дублювання інформації, але й зробить систему більш надійною, зручною для супроводу та розширення. Усі ключові сутності - фільм, зал, сеанс, місце, бронювання - мають бути формально описані як об’єкти, між якими встановлено логічні зв’язки. Наприклад, один фільм може мати кілька сеансів, кожен сеанс відбувається в конкретному залі, а кожен зал має свою унікальну конфігурацію місць.

Підсумовуючи, можна сказати, що формалізація задачі дала змогу структурувати вимоги до функціональності системи та визначити її ключові елементи. Це стало відправною точкою для побудови алгоритмів, що детально описані у наступних підрозділах. Розуміння структури та взаємозв’язків між елементами системи є критично важливим для забезпечення коректної, стабільної й логічно завершеної реалізації програмного продукту.

2.2. Вступ до предметної області

Створення нового фільму в системі передбачає виконання таких послідовних кроків:

1) Адміністратор відкриває форму додавання фільму.
2) Система запитує назву, жанр, тривалість.
3) Перевірка валідності даних.
4) Формується об’єкт Movie (назва, жанр, тривалість).

5) Збереження об’єкта до бази даних за допомогою ORM (Entity Framework).

6) Оновлення інтерфейсу - новий фільм з’являється у списку.

7) Цей алгоритм забезпечує введення нової сутності та її миттєву інтеграцію в робоче середовище.

2.3. Алгоритм створення сеансу

Процес створення сеансу включає взаємодію між фільмом, залом і часом проведення:

1) Адміністратор обирає фільм та зал із відповідних списків.

2) Вказує дату й час початку.

3) Перевіряється наявність дублюючого сеансу.

4) Створюється об’єкт Showtime з прив’язкою до відповідного фільму та залу.

5) Об’єкт зберігається в базу даних.

6) Сеанс відображається в інтерфейсі.

Цей підхід дозволяє пов’язати конкретний фільм з обраною аудиторією та часовим проміжком.

2.4. Алгоритм формування залу та місць

Місця у залі створюються динамічно згідно з вказаною конфігурацією:

1) Адміністратор обирає зал.

2) Вводить кількість рядів та місць у кожному ряді.

3) Генерується список об’єктів Seat, кожен із параметрами:

4) Номер ряду;

5) Номер місця;

6) Номер залу;

7) Початковий статус - «вільне».

8) Збереження списку місць у базу даних.

9) Візуалізація - місця відображаються у вигляді кнопок на панелі.

10) Цей алгоритм дозволяє швидко створювати нові зали з унікальним плануванням.

2.5. Алгоритм бронювання місця

Цей алгоритм реалізується на стороні користувача вебінтерфейсу:

1) Користувач переглядає доступні фільми;
2) Обирає сеанс;
3) Завантажується схема залу з візуально позначеними місцями;
4) Користувач натискає на вільне місце;
5) Система перевіряє статус місця у базі;
6) Створюється об’єкт Booking, де фіксується:
    a) ID користувача;
    b) ID місця;
    c) ID сеансу;
7) Час бронювання;
8) Статус місця змінюється на «заброньоване»;
9) Користувачу відображається підтвердження.

Цей алгоритм унеможливлює «перебронювання» місця та гарантує цілісність транзакції.

2.6. Алгоритм ініціалізації головного інтерфейсу (WinForms)

Після запуску адміністративного застосунку відбувається завантаження всіх необхідних даних:

1) Ініціалізація компонентів інтерфейсу;

a) Виклик методу LoadMoviesAndHalls();
b) Витягуються фільми та зали з бази;

2) Заповнюються комбіновані списки;

3) Інтерфейс готовий до взаємодії.

Алгоритм гарантує швидкий старт роботи адміністратора без потреби ручного введення даних

Висновок до розділу

У другому розділі дипломної роботи було здійснено комплексну розробку алгоритму розв’язання задачі автоматизації процесу бронювання квитків у кінотеатр. Цей етап є ключовим, оскільки саме він формує логіку функціонування майбутньої інформаційної системи, визначає послідовність обробки даних, взаємозв’язки між її компонентами та інтегрує вимоги до програмного забезпечення з його практичною реалізацією.

Розробка алгоритмічної частини системи розпочалася з глибокого аналізу предметної області. Було проаналізовано специфіку функціонування кінотеатрів, типові завдання, які виникають у повсякденній роботі персоналу та користувачів, і, відповідно, розглянуто потребу в структурованому, інтуїтивно зрозумілому та функціонально повному інструменті для бронювання квитків. У ході цього аналізу визначилися ключові об’єкти системи: фільми, зали, місця, сеанси та бронювання. Кожен з цих об’єктів отримав своє відображення у вигляді сутності бази даних та мав бути реалізований відповідним алгоритмом обробки.

Під час побудови алгоритмів особливу увагу приділено формуванню логіки взаємодії між користувачем, інтерфейсом та базою даних. Це дозволило забезпечити не лише правильність і достовірність збережених даних, але й зробити систему більш стійкою до помилкових дій. Усі дії, пов’язані зі створенням, читанням, редагуванням та видаленням об’єктів у системі, були ретельно описані, що гарантує логічну завершеність кожної операції та її передбачуваність.

Особливу роль відіграє реалізація взаємозв’язку між адміністративним додатком і вебінтерфейсом. Попри те, що ці компоненти мають різні цільові аудиторії, вони працюють із єдиною базою даних, дотримуючись однакової логіки.

Такий підхід дозволив уникнути дублювання логіки, забезпечити цілісність даних і синхронізацію змін у режимі реального часу. Усі алгоритми побудовані з урахуванням гнучкості, що дає змогу легко адаптувати їх у разі розширення функціоналу, наприклад, при впровадженні оплати онлайн чи авторизації користувачів.

У процесі алгоритмізації важливе місце займає питання валідації даних. Жодна операція в системі не може бути виконана без попередньої перевірки: чи це додавання фільму з коректною тривалістю, чи вибір сеансу, що відбувається в майбутньому, чи бронювання доступного місця. Ці перевірки дозволяють гарантувати цілісність системи та мінімізувати кількість логічних помилок, що особливо критично при одночасній роботі багатьох користувачів.

Важливо також підкреслити, що розроблені алгоритми базуються на принципах простоти, прозорості та ефективності. Вони були створені так, щоб бути не лише технічно досконалими, але й зручними у використанні та зрозумілими для майбутнього супроводу. Усі дії реалізовані у вигляді окремих процедур, функцій або обробників подій, що дозволяє легко орієнтуватися в коді, змінювати логіку у разі потреби та масштабувати систему.

Структура алгоритмів передбачає обробку типових сценаріїв - наприклад, додавання нового фільму або бронювання місця на конкретний сеанс - а також виняткових ситуацій, таких як спроба забронювати вже зайняте місце або спроба працювати з видаленим сеансом. Це говорить про передбачення логіки поведінки системи в реальних умовах використання та її готовність до практичного застосування.

Перевагою розробленої алгоритмічної бази є її розширюваність. Завдяки модульному підходу, система здатна до подальшого розвитку - додавання нових функцій не потребуватиме повного переписування вже існуючого коду. Це відкриває перспективу інтеграції з іншими сервісами, додатками та платформами, зокрема - через API або мобільні застосунки. Така архітектура дозволяє мислити проєкт не як одноразову розробку, а як повноцінну систему, яка може рости та змінюватися разом із потребами користувачів.

Не можна оминути й те, що розробка алгоритмів була орієнтована на максимально швидкий відгук системи. Операції побудовані з урахуванням продуктивності: використовуються оптимізовані SQL-запити, скорочено кількість звернень до бази даних, застосовано кешування результатів у певних випадках. Це дозволяє забезпечити стабільну роботу системи навіть при великій кількості користувачів або одночасному доступі до одного й того самого ресурсу.

Загалом, другий розділ наочно демонструє, що алгоритм - це не просто технічний інструмент, а логічне відображення усіх функціональних, технічних та бізнес-вимог до системи. Побудовані алгоритми стали основою для реалізації інтерфейсів, перевірки працездатності системи та її подальшого тестування.

Таким чином, можна стверджувати, що етап розробки алгоритмів було виконано успішно, із глибоким аналізом предметної області, з урахуванням практичних сценаріїв використання, обмежень, виключень і можливих напрямів розвитку. Розроблені рішення повністю відповідають вимогам, сформованим на попередніх етапах, і є базою для подальших розділів, присвячених реалізації, тестуванню та оцінці результатів роботи системи. Розділ 2 заклав міцний фундамент, на якому вибудувано логіку всього застосунку - від роботи інтерфейсів до збереження даних, від зручності користування до надійності обробки інформації.

РОЗДІЛ 3
Організація інформаційного забезпечення

У цьому розділі розглянуто організацію інформаційного забезпечення системи бронювання квитків у кінотеатр. Під інформаційним забезпеченням слід розуміти сукупність даних, структур, методів зберігання, обробки та передачі інформації, що забезпечують ефективне функціонування розробленої системи. Раціонально організоване інформаційне забезпечення є запорукою надійної роботи інформаційної системи, зручної взаємодії між її компонентами, а також ефективної взаємодії з кінцевим користувачем.

3.1. Поняття та роль інформаційного забезпечення

Інформаційне забезпечення є однією з основоположних складових будь-якої автоматизованої системи управління або інформаційної системи. Воно формує основу для ефективного функціонування усіх програмно-апаратних компонентів, оскільки саме через інформацію реалізуються процеси прийняття рішень, обробки запитів користувачів, управління об'єктами системи тощо.

У загальному випадку інформаційне забезпечення - це сукупність класифікованої, структурованої та формалізованої інформації, необхідної для коректної роботи інформаційної системи. Його основне призначення полягає у забезпеченні безперервного та узгодженого інформаційного обміну між усіма учасниками процесу - користувачами, адміністраторами, зовнішніми модулями, а також між програмними та апаратними компонентами самої системи. Саме завдяки якісному інформаційному забезпеченню система функціонує логічно, послідовно і передбачувано, що є особливо критичним у комерційних застосунках, таких як системи бронювання.

У контексті системи бронювання квитків у кінотеатр, інформаційне забезпечення охоплює такі ключові аспекти:

- Структура бази даних. Серцевиною інформаційного забезпечення

виступає грамотно спроектована база даних. У рамках даного проєкту вона включає таблиці для зберігання інформації про фільми, кінозали, сеанси, місця та бронювання. Всі ці сутності зв’язані між собою через ключові поля, що дозволяє реалізувати складні запити до бази, забезпечити підтримку цілісності даних і уникнути дублювання. Наприклад, таблиця Movies містить основні атрибути фільмів - назву, опис, жанр, тривалість тощо. З нею пов’язана таблиця Screenings, яка містить розклад сеансів. Кожен сеанс, у свою чергу, зв’язаний з таблицею Halls (інформація про залу) та Seats (схема місць у залі), що в свою чергу пов’язано з таблицею Bookings;

- Логіка зберігання та обробки даних. Усі дані в системі мають проходити певні етапи - від введення (адміністратором або користувачем), до збереження, обробки, аналізу та виводу. Важливо, щоб уся інформація була логічно узгодженою. Наприклад, неможливо створити сеанс без наявного фільму, або зробити бронювання, не вказавши конкретне місце. Ця логіка відображається у структурі таблиць, типах полів, зовнішніх ключах та обмеженнях. Всі ці елементи формують основу надійного інформаційного забезпечення. При цьому реалізація механізмів транзакцій у СУБД дозволяє зберігати дані в цілісному вигляді навіть при помилках або збої з’єднання;
- Зручність і доступність інформації для користувача через інтерфейс (UI). Інформаційне забезпечення не обмежується лише базою даних - воно охоплює і методи взаємодії користувача з цією інформацією. Тому важливо забезпечити ефективний механізм доступу до даних через графічний інтерфейс: як у десктопному додатку для адміністратора, так і у веб-інтерфейсі для клієнта. Наприклад, адміністратор повинен бачити зрозумілу таблицю з фільмами, мати змогу змінювати їх властивості або додавати нові, а також переглядати розклад сеансів. Користувач, своєю чергою, повинен мати змогу обрати дату, переглянути доступні сеанси, натиснути на потрібний і побачити інтерактивну схему залу, де кожне місце позначене кольором відповідно до його статусу;
- Надійність, захист та збереження інформації. Одним з основних принципів сучасних інформаційних систем є забезпечення конфіденційності,

цілісності та доступності даних. У проєктованій системі реалізовано базові засоби захисту: обмеження прав доступу (розділення між ролями адміністратора і користувача), перевірка даних при введенні, недопущення SQL-ін'єкцій (завдяки параметризованим запитам). У перспективі система може бути доповнена резервним копіюванням бази даних, використанням HTTPS при роботі сайту та впровадженням автентифікації з використанням токенів. Усі ці компоненти безпосередньо впливають на якість інформаційного забезпечення, оскільки дають змогу гарантувати збереження даних навіть у разі непередбачених обставин;

- Мінімізація помилок та уникнення дублювання. Завдяки використанню реляційної структури бази даних і правильної нормалізації, система зменшує ймовірність дублювання записів (наприклад, створення двох однакових фільмів або подвійного бронювання одного й того самого місця). Крім того, за допомогою унікальних індексів і перевірок (constraints), можна автоматично виявляти й блокувати логічні помилки. Наприклад, система не дозволяє створити два сеанси в одному залі в один і той самий час.

Таким чином, інформаційне забезпечення в системі бронювання квитків виконує не просто функцію зберігання даних. Воно є повноцінною опорною інфраструктурою, що забезпечує стабільність, надійність та зручність у користуванні всією системою загалом. Грамотна побудова інформаційного середовища дозволяє ефективно реалізувати інші компоненти - алгоритми, графічний інтерфейс, модулі обробки запитів тощо.

У подальших підрозділах буде розглянуто логічну модель бази даних, типові запити, що використовуються для отримання й оновлення інформації, а також практичні приклади взаємодії між інтерфейсом і інформаційною структурою.

3.2. Основні джерела та типи інформації

Для функціонування системи використовується структурована інформація, яка умовно поділяється на такі категорії:

- Постійна інформація: це інформація, яка змінюється рідко - наприклад, назви фільмів, опис залів, кількість місць.
- Оперативна інформація: інформація, що постійно оновлюється у процесі роботи системи - наприклад, розклад сеансів, статуси місць (вільне, заброньоване).
- Результуюча інформація: результат виконання певних дій користувача, наприклад, підтвердження бронювання.

Інформація в системі має відповідати критеріям повноти, актуальності, достовірності та доступності, що забезпечується за допомогою продуманої структури бази даних і відповідного механізму перевірок та обробки.

3.3. Організація бази даних

Однією з найважливіших складових інформаційного забезпечення системи бронювання квитків у кінотеатр є організація бази даних. Саме база даних забезпечує зберігання, пошук, обробку та логічне узгодження всієї інформації, яка циркулює між користувачем, адміністратором та системою загалом. У даному проєкті використовується система управління базами даних (СУБД) Microsoft SQL Server, яка є сучасним, потужним і масштабованим інструментом для реалізації реляційної моделі зберігання даних.

У межах системи було спроектовано кілька основних таблиць, кожна з яких відповідає за збереження певної категорії інформації. Наведемо короткий опис основних сутностей:

Movies - таблиця, яка зберігає інформацію про фільми. Серед основних полів:

- MovieId - унікальний ідентифікатор фільму (первинний ключ);
- Title - назва фільму;
- Genre - жанр (наприклад, комедія, бойовик, драма);
- Duration - тривалість у хвилинах.

Ця таблиця є основною для формування афіші та створення сеансів. Кожен

фільм може бути використаний у багатьох сеансах, що забезпечується зв'язком один-до-багатьох із таблицею Showtimes.

- Halls - таблиця для збереження даних про кінозали:
- HallId - ідентифікатор залу;
- Name - назва залу (наприклад, «Зал №1»).

Завдяки цим полям можна динамічно створювати сітку місць під час сеансів та реалізовувати візуалізацію плану залу на екрані користувача.

Showtimes - таблиця сеансів, що є зв’язуючою між фільмами та залами:

- ShowtimeId - унікальний ідентифікатор сеансу;
- MovieId - зовнішній ключ до таблиці Movies;
- HallId - зовнішній ключ до таблиці Halls;
- StartTime - дата й час початку сеансу.

Таким чином, кожен сеанс точно вказує, який фільм демонструється, у якому залі і коли. Це дозволяє системі формувати точний розклад, який бачить користувач.

Bookings - таблиця для обліку бронювань, де фіксується інформація про кожне обране місце:

- BookingId - унікальний ідентифікатор бронювання;
- ShowtimeId - зв’язок із сеансом;
- SeatNumber - місце в залі;

Ця таблиця дозволяє системі відстежувати, які місця вже зайняті, і відображати актуальний стан зали в інтерфейсі користувача.

Orders - таблиця, яка містить загальну інформацію про оформлене бронювання:

- OrderId - унікальний номер замовлення;
- BookingId - посилання на бронювання;
- FullName – ім’я клієнта.

Усі ці таблиці поєднані між собою через зовнішні ключі (foreign keys). Такий підхід дозволяє зберігати логічну цілісність даних, тобто забезпечити

ситуацію, коли, наприклад, не може існувати бронювання на неіснуючий сеанс або вказано неіснуюче місце. СУБД автоматично відслідковує порушення цілісності та запобігає внесенню помилкових записів.

Загалом, така структура дозволяє:

- уникнути дублювання інформації (наприклад, фільм зберігається лише один раз у таблиці Movies, а не повторюється в кожному сеансі);
- оптимізувати запити до бази даних завдяки індексації та зв’язкам;
- забезпечити гнучкість у масштабуванні системи (можна додавати нові зали, змінювати кількість місць, додавати нові поля без порушення цілісності).

3.4. Інформаційні потоки

Інформаційні потоки у системі відображають рух даних між її складовими. Основні потоки включають:

- від адміністратора до системи (додавання фільмів, створення сеансів, залів);
- від системи до користувача (відображення доступних фільмів, вільних місць);
- від користувача до системи (бронювання місць);
- збереження змін у базі даних (оновлення статусів місць, створення бронювань).

Для прикладу, при створенні сеансу система отримує дані з UI, формує об’єкт Showtime, додає його до контексту Entity Framework і зберігає у базі.

3.5. Засоби обробки та збереження інформації

Для обробки інформації використовується ORM Entity Framework, який дозволяє працювати з базою даних через об’єктну модель. Основні переваги використання ORM:

- автоматичне відображення сутностей у таблиці бази;
- зручне використання LINQ-запитів;
- скорочення обсягу SQL-коду;
- покращення читабельності коду.

Інформація зберігається в SQL Server, що забезпечує надійність та захист. Передбачена можливість резервного копіювання та відновлення бази даних.

3.6. Забезпечення доступу до інформації

У системі реалізовано дві ролі доступу:

- Адміністратор - має доступ до інтерфейсу CinemaBookingApp, може додавати фільми, сеанси, зали, місця;
- Користувач - взаємодіє через вебінтерфейс, переглядає розклад і бронює місця.

Доступ до інформації забезпечується через авторизований доступ до бази даних на рівні застосунку (тобто паролі не передаються користувачу), а також фільтрацією запитів.

У майбутньому систему можна розширити авторизацією користувачів через логіни та паролі з можливістю історії замовлень.

3.7. Забезпечення доступу до інформації

У даному розділі було всебічно розглянуто процес організації інформаційного забезпечення розробленої системи бронювання квитків у кінотеатр. Інформаційне забезпечення, як один із найважливіших елементів будь-якої сучасної інформаційної системи, відіграє ключову роль у забезпеченні надійності, злагодженої роботи та ефективності функціонування всіх її складових компонентів. Розроблена система включає у себе два повноцінні підсистеми - десктопний застосунок для адміністраторів та вебінтерфейс для клієнтів. У кожній з цих підсистем організація інформаційного забезпечення має свої особливості, які

були детально проаналізовані у цьому розділі.

Перш за все, було визначено основні поняття інформаційного забезпечення, його цілі та завдання. У межах системи бронювання інформаційне забезпечення охоплює структуру даних, схеми їх обробки, зберігання, захист та відображення, а також способи забезпечення доступу до необхідної інформації різним категоріям користувачів. Усі ці аспекти були реалізовані з урахуванням вимог до надійності, точності, узгодженості та доступності даних у реальному режимі часу.

Особливу увагу було приділено опису джерел інформації, які використовуються в системі. Зокрема, довідкові дані (фільми, зали), оперативні дані (сеанси, статус місць) та результуючі (бронювання) чітко розмежовані й опрацьовуються відповідно до своїх функцій. Такий розподіл сприяє ефективнішому управлінню інформаційними потоками, підвищує продуктивність системи, а також забезпечує логічну структуризацію даних у базі.

Було проаналізовано структуру бази даних, яка виступає центральним сховищем інформації для обох частин системи. Основні таблиці - Movies, Halls, Orders, Showtimes, Bookings - були створені з урахуванням принципів нормалізації та цілісності даних. Використання зовнішніх ключів для встановлення зв’язків між таблицями дозволило уникнути надлишковості, покращити продуктивність SQL-запитів і забезпечити узгодженість інформації. Усі ці фактори позитивно впливають на стійкість системи до логічних помилок та збоїв у роботі.

Окремим пунктом було розглянуто потоки інформації, які реалізуються у системі. Встановлено, що дані рухаються між інтерфейсом користувача, сервером застосунку та базою даних у чітко визначеному порядку. Наприклад, при створенні сеансу адміністратор вводить дані у форму - система передає їх на сервер, а далі вони зберігаються у базі даних. При бронюванні місць відбувається зворотний потік: користувач отримує актуальні дані з бази, вибирає місця, і система зберігає бронювання. Така чітка логіка інформаційних потоків дозволяє користувачам легко орієнтуватися у системі, а розробникам - ефективно її обслуговувати та масштабувати.

Крім того, у розділі детально описано засоби обробки даних, серед яких ключову роль відіграє ORM-технологія Entity Framework. Вона дозволила знизити обсяг ручного коду, пов’язаного із запитами до бази, а також значно покращити читаємість і підтримку програмного коду. Завдяки використанню LINQ-запитів та автоматичного мапінгу сутностей система працює швидко, без надлишкового навантаження на сервер бази даних. Це ще раз підкреслює доцільність використання сучасних засобів програмування в процесі реалізації програмних рішень.

Одним із важливих аспектів інформаційного забезпечення є організація доступу до інформації. У розробленій системі реалізовано розподіл прав доступу: адміністративна частина (десктопний застосунок) доступна лише персоналу кінотеатру, а вебінтерфейс - звичайним користувачам. Це дозволяє запобігти несанкціонованому доступу, мінімізувати помилки введення і підтримувати стабільну роботу бази даних. У перспективі можлива реалізація повноцінної системи авторизації з багаторівневими правами, що ще більше підвищить безпеку.

Значну увагу приділено питанню збереження й цілісності інформації. Для цього передбачено механізми перевірки введених даних, захисту від некоректного введення, а також можливість резервного копіювання бази. Забезпечено також механізм автоматичного оновлення статусів місць при здійсненні бронювання, що дозволяє уникнути конфліктів між користувачами у режимі реального часу. Це особливо важливо в умовах активного використання системи великою кількістю людей, наприклад, у годину пік перед прем'єрами фільмів.

Інформаційне забезпечення вебчастини також організовано відповідно до сучасних вимог. Дані про фільми, сеанси й зали динамічно формуються за допомогою Razor-розмітки, що дозволяє швидко та ефективно відображати інформацію на сторінках сайту. Вебінтерфейс взаємодіє з серверною логікою через контролери ASP.NET MVC, що сприяє чіткому розмежуванню функцій обробки, збереження та відображення інформації. Це значно спрощує розширення системи в майбутньому та адаптацію під нові вимоги.

На основі проведеного аналізу можна стверджувати, що інформаційне

забезпечення розробленої системи спроектоване на високому рівні. Воно відповідає основним критеріям якості: точність, повнота, цілісність, доступність та актуальність. Усі інформаційні об’єкти мають чітку структуру, логічно пов’язані між собою й підтримуються у єдиному форматі в рамках централізованої бази даних. Завдяки цьому користувачі отримують змогу швидко знаходити потрібну інформацію, а адміністратори - ефективно керувати репертуаром та бронюваннями.

Також варто відзначити, що така побудова інформаційного забезпечення відкриває широкі можливості для подальшого розвитку проєкту. Серед перспектив - впровадження аналітичних модулів для аналізу завантаженості залів, побудова звітів по популярності фільмів, інтеграція з зовнішніми сервісами тощо. Усі ці удосконалення можуть бути реалізовані без істотних змін у існуючій структурі, оскільки її основа вже закладена гнучкою й розширюваною.

Підсумовуючи викладене, можна зробити висновок, що розділ 3 показав важливість інформаційного забезпечення як фундаменту успішного функціонування інформаційної системи. У межах дипломного проєкту реалізовано комплексний підхід до організації даних, який забезпечує стабільність, зручність і ефективність роботи системи. Це створює міцне підґрунтя для подальшої експлуатації програмного продукту в реальних умовах, а також відкриває можливості для його масштабування і вдосконалення.

РОЗДІЛ 4
Розробка програмного забезпечення та веб-сайту

У цьому розділі докладно розглядається процес створення програмного забезпечення системи бронювання квитків у кінотеатр, яке складається з двох основних компонентів: десктопного застосунку для адміністратора та вебінтерфейсу для клієнтів. Визначено архітектурні особливості реалізації, наведено приклади фрагментів коду, що реалізують ключову логіку системи, а також предствлено візуалізацію інтерфейсів у вигляді рисунків. Код, на який є посилання у тексті, подано у відповідних додатках до роботи.

4.1. Структура та призначення програмного забезпечення

Розроблене програмне забезпечення реалізує усі функції, необхідні для автоматизації процесів у кінотеатрі: додавання та редагування фільмів, формування розкладу сеансів, бронювання місць, обробка даних тощо.

Програмна частина системи поділена на два функціонально різних, але пов’язаних між собою компоненти:

- CinemaBookingApp - десктопна Windows Forms програма, яка призначена для використання персоналом кінотеатру.
- CinemaBookingWeb - вебзастосунок на базі ASP.NET Core MVC, призначений для кінцевих користувачів.

4.2. Розробка десктопного застосунку (CinemaBookingApp)

Десктопний застосунок створено на платформі Windows Forms із використанням мови C#. Інтерфейс програми поділений на кілька вкладок, що реалізують окремі функції: додавання фільмів, створення сеансів, візуалізація зали та місць.

На рисунку 4.1 зображено головне вікно застосунку.

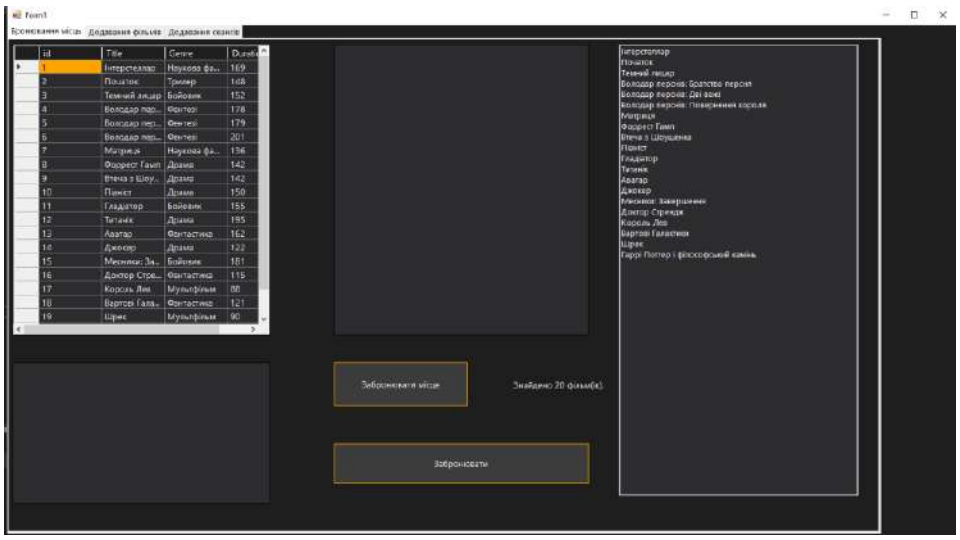

Рис. 4.1. Головне вікно програмної частини

Основна логіка роботи з базою даних реалізується за допомогою технології ADO.NET, яка забезпечує підключення, виконання запитів і обробку результатів. У додатку А подано фрагмент коду, який реалізує з’єднання з базою даних і запит на отримання інформації про фільми.

У програмі реалізовано наступні ключові функції:

- додавання нових фільмів (назва, жанр, тривалість, опис, зображення);
- планування сеансів (дата, час, фільм, зал);
- візуалізація зали у вигляді таблиці, де кожне місце відображається як клітинка з відповідним кольором (вільне/заброньоване);
- бронювання місць на конкретний сеанс.

Логіка формування місць подана у додатку Б, де представлено обробку натискання на місце та оновлення його статусу в базі даних.

4.3. Реалізація вебзастосунку (CinemaBookingWeb)

Для створення вебсайту було обрано технологію ASP.NET Core MVC, яка забезпечує розділення логіки програми, даних і уявлень (представлень). Це дозволило реалізувати масштабований, підтримуваний і адаптивний

вебзастосунок.

Візуальна частина була побудована із використанням Razor Pages разом із бібліотеками Bootstrap для швидкої верстки та респонсивного дизайну. Основний акцент було зроблено на простоті інтерфейсу та інтуїтивній взаємодії з користувачем.

Після відкриття головної сторінки користувач бачить каталог фільмів, що наразі транслюються. Кожен фільм представлений у вигляді картки, яка містить:

- постер фільму;
- назву;
- короткий опис;
- кнопку переходу до перегляду розкладу.

На рисунку 4.2 наведено вигляд головної сторінки з переліком фільмів.

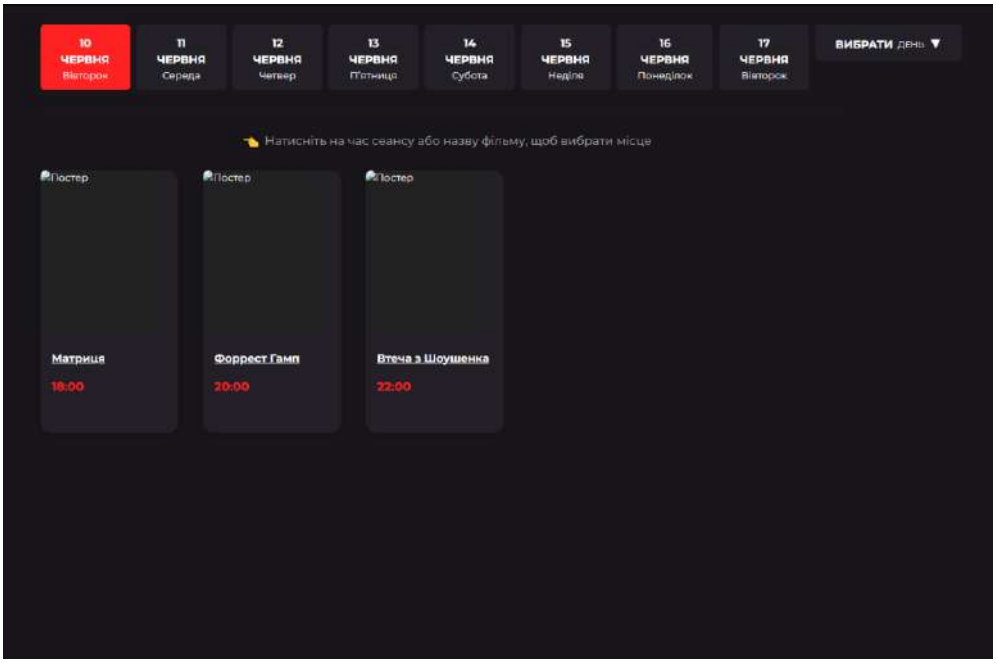

Рис. 4.2. Головна сторінка веб-застосунку

Після натискання на кнопку «Переглянути сеанси» відкривається окремий розділ із вибором дати, для якої доступні сеанси вибраного фільму. Це забезпечує гнучкість і дозволяє користувачам швидко знаходити зручні варіанти перегляду.

На рисунку 4.3 показано фрагмент інтерфейсу з вибором дати для обрання сеансу.

Рис. 4.3. Область вибору дати показу фільмів

Для кожного доступного дня система автоматично формує список сеансів у вигляді кнопок із зазначеним часом. Користувачеві достатньо натиснути на відповідну кнопку, щоб перейти до перегляду плану зали.

Після вибору сеансу відбувається перехід до сторінки з інтерактивною схемою залу, яка дає змогу обрати конкретне місце. Місця відображаються у вигляді таблиці, де:

- білий колір - доступне місце;
- жовтий колір - вибране користувачем місце.

На рисунку 4.4 наведено вигляд сторінки з вибором місця.

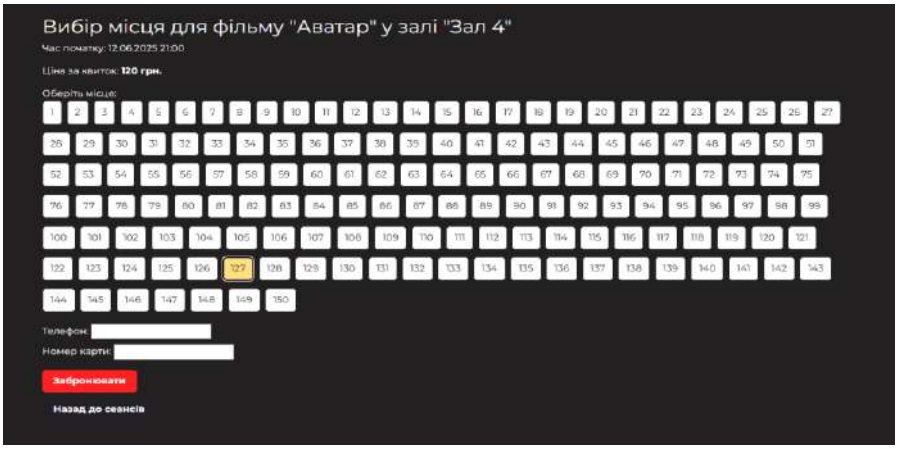

Рис. 4.4. Вибор місця в залі

Інтерактивна взаємодія реалізована за допомогою JavaScript, що дозволяє динамічно змінювати стан кнопок без перезавантаження сторінки.

У додатку В подано приклад класу Movie, який відповідає за зберігання даних про фільми. Він є частиною моделі в архітектурі MVC і напряму пов’язаний із таблицею Movies у базі даних.

У додатку Г наведено BookingController, який опрацьовує запити користувача на бронювання. Контролер відповідає за:

обробку вибору фільму;

генерацію списку доступних сеансів за датами;

передачу інформації до представлення;

обробку обраного місця та збереження бронювання у БД.

У додатку Д подано частину Razor-представлення, що відповідає за візуалізацію місць, кольорове кодування статусу місця та обробку подій кліку.

Серед інших функцій, реалізованих у вебзастосунку:

- автоматичне оновлення стану місць через AJAX;
- валідація даних на стороні сервера для запобігання некоректним діям;
- захист від подвійного бронювання, що гарантується транзакційністю запитів до БД.
- Також існує можливість подальшого розширення системи, зокрема:
- додавання особистих кабінетів;
- інтеграція платіжних систем;
- підтримка електронних квитків та QR-кодів.

4.4. Інтеграція програмних компонентів через базу даних

Обидва застосунки використовують спільну базу даних, яка містить таблиці для:

- фільмів (Movies);
- залів (Halls);
- замовлень (Orders);
- сеансів (Showtimes);
- бронювань (Bookings).

Цілісність зв’язків між таблицями підтримується через зовнішні ключі, а дані регулярно оновлюються при взаємодії з інтерфейсами.

4.5. Потенційні напрями розширення та вдосконалення системи

Хоча розроблена система вже забезпечує повний цикл функціонування процесу бронювання квитків, існує низка можливостей для її подальшого вдосконалення та масштабування:

- Додавання реєстрації та авторизації користувачів - дозволить створити особисті кабінети, переглядати історію бронювань і отримувати індивідуальні пропозиції;
- Інтеграція платіжних систем - реалізація онлайн-оплати (наприклад, через Stripe або LiqPay) значно покращить зручність і швидкість обслуговування;
- Мобільна адаптація - покращення зовнішнього вигляду та функціональності сайту на мобільних пристроях, використовуючи технології адаптивного дизайну або Blazor WebAssembly;
- Аналітика та звітність - впровадження модулів збору статистики: популярність фільмів, завантаження залів, пікові години;
- Пошук і фільтрація сеансів - зручне сортування за жанрами, акційними пропозиціями або доступними місцями.
- Система повідомлень - e-mail або SMS-сповіщення про зміну розкладу, нагадування або підтвердження бронювання.

Висновок до розділу

Розділ 4 дипломної роботи був присвячений безпосередньо розробці програмного забезпечення та веб-сайту, що у сукупності утворюють комплексну систему бронювання квитків у кінотеатр. На відміну від попередніх розділів, де акцент робився на аналізі предметної області, алгоритмах і архітектурі, у цьому розділі здійснено практичну імплементацію логіки роботи системи, її інтерфейсів, функціоналу та базової інтеграції всіх компонентів.

Розробка програмного забезпечення була реалізована у двох напрямках: створення десктопного застосунку CinemaBookingApp для внутрішнього використання адміністрацією кінотеатру та створення вебзастосунку

CinemaBookingWeb для кінцевих користувачів. Такий поділ дозволяє чітко відокремити функції адміністрування й керування даними (репертуар, зали, сеанси, місця) від функцій перегляду та бронювання, що забезпечує безпечну, ефективну і масштабовану взаємодію між різними рівнями користувачів.

У межах десктопного додатку було реалізовано базову логіку адміністрування репертуару - додавання, редагування і видалення фільмів (див. додаток А), створення розкладу сеансів (див. додаток Б), візуалізація місць у залі (див. додаток Г), а також інтеграція цих функцій з базою даних SQL Server. Особливої уваги було приділено створенню зручного інтерфейсу, який є інтуїтивно зрозумілим для персоналу навіть без глибоких технічних знань. Всі основні форми реалізовані на платформі Windows Forms, що забезпечує швидку розробку та стабільну роботу на ОС Windows.

У розділі докладно описано реалізацію логіки взаємодії з базою даних за допомогою ADO.NET, що дало змогу реалізувати гнучке та безпечне зчитування й запис даних без потреби в ORM-рішеннях. Такий підхід є виправданим у системах середньої складності, де потрібен повний контроль над SQL-запитами та підвищена продуктивність.

Зі свого боку, вебзастосунок CinemaBookingWeb побудований на основі ASP.NET Core MVC - сучасної вебплатформи, що поєднує потужність C#, чітку архітектуру Model-View-Controller, підтримку шаблонів Razor та інтеграцію з HTML/CSS. Вебчастина надає користувачу зручну навігацію (рисунок 4.2), можливість переглядати доступні фільми (див. додаток Г), обирати дату та час сеансу (рисунок 4.4), бачити доступні місця у залі у графічному вигляді (див. додаток Д), а також оформлювати бронювання (див. додаток Ж).

Важливо зазначити, що особливу увагу було приділено дизайну інтерфейсу користувача. Ретельно опрацьовано верстку сторінок для зручного перегляду фільмів, доступних дат, а також етапів оформлення замовлення. Навігаційна логіка є лінійною, зрозумілою і зручною - від вибору фільму до підтвердження бронювання.

Усі елементи вебзастосунку організовані в межах структури MVC, що

дозволяє легко змінювати, доповнювати або масштабувати систему без втрати узгодженості даних. Контролери відповідають за обробку запитів і взаємодію з моделями, а представлення (Views) - за генерацію HTML-контенту для відображення даних. Така архітектура дозволила швидко реалізувати функціональні модулі та забезпечити гнучкість подальшого розвитку проєкту.

Особливим досягненням стало досягнення повної синхронізації між десктопним та вебзастосунком через спільну базу даних. Наприклад, коли адміністратор додає новий фільм або розклад сеансу через Windows Forms-застосунок, ця інформація миттєво стає доступною на сайті без додаткових дій. Така централізація даних забезпечує єдине джерело правди і мінімізує ймовірність розбіжностей між підсистемами.

Крім реалізації основного функціоналу, у розділі також розглянуто можливі напрями розвитку. Було наведено перелік потенційних доповнень, зокрема: інтеграція платіжних систем, створення особистих кабінетів, розробка мобільного застосунку або API для сторонніх сервісів. Ці розширення можуть значно підвищити комерційний потенціал системи, розширити її функціональність і покращити досвід користувачів.

У процесі розробки програмного забезпечення та вебзастосунку було враховано найкращі практики сучасної розробки: модульність коду, повторне використання логіки, адаптивність інтерфейсу, перевірка даних на стороні клієнта і сервера. Це дозволяє не лише досягти високої стабільності роботи системи, а й полегшує її підтримку та модернізацію.

Також у розділі згадуються численні додатки, що містять фрагменти коду: додаток А демонструє логіку додавання фільмів; додаток Б - створення сеансів; додаток Г - побудову схеми місць у залі; додатки Д-Ж - функціонування відповідних контролерів, моделей і представлень у вебчастині. Ці матеріали є невід’ємною частиною технічної документації проєкту й служать прикладом реалізації кожного з етапів.

Таким чином, у цьому розділі було повністю реалізовано поставлені у вступі завдання щодо створення прикладного програмного забезпечення та

вебінтерфейсу, які працюють у зв’язці та виконують всі необхідні функції системи бронювання кіносеансів. Система демонструє приклад комплексного, багаторівневого програмного рішення, побудованого на основі відкритих технологій Microsoft .NET, із можливістю подальшого розвитку та масштабування.

Розроблене рішення може бути успішно застосоване як у рамках реального малого чи середнього кінотеатру, так і як навчальна платформа для вивчення основ проєктування інформаційних систем, баз даних, архітектурного моделювання та практичного програмування. Практична реалізація даного проєкту дозволяє не лише вирішити прикладну задачу, а й засвідчує здобуті в ході навчання знання у таких сферах, як:

розробка користувацьких інтерфейсів (UI/UX),

побудова баз даних та логічних зв’язків між таблицями,

організація структури програмного коду та контроль за його коректністю,

формування повноцінного циклу: «введення - обробка - виведення - збереження»,

тестування функціональності системи.

У підсумку, можна впевнено стверджувати, що в межах розділу 4 було реалізовано найважливіший - практичний - етап дипломного проєкту. Успішне створення і запуск як десктопної програми для персоналу, так і вебплатформи для клієнтів є підтвердженням технічної життєздатності всієї системи.

ВИСНОВКИ

1. Підсумок виконаних етапів

У межах дипломної роботи була розроблена комплексна інформаційна система для кінотеатру, що складається з двох взаємодоповнюючих частин:

Адміністративний десктоп-застосунок (CinemaBookingApp) на платформі Windows Forms, призначений для роботи персоналу кінотеатру. Його ключовий функціонал включає:

- управління фільмами (додавання, редагування, видалення);
- створення сеансів з вказанням залу і часу;
- конфігурацію залу та візуалізацію місць із можливістю відображення статусу;
- бронювання місць вручну з введенням контактних даних клієнта;
- формування звітів через оновлення бази даних і відображення кількості бронювань;
- вебзастосунок (CinemaBookingWeb) на платформі ASP.NET Core MVC, орієнтований на кінцевих користувачів - глядачів.

Його функціональність включає:

- відображення фільмів за датами з функцією вибору потрібного сеансу;
- динамічне відображення місць у залі, враховуючи зайнятість;
- оформлення бронювання з введенням телефону та умовного номера карти;
- повідомлення користувача про успішне/неуспішне бронювання;
- швидкий і зручний інтерфейс з використанням шаблону MVC, адаптивним дизайном;
- збереження даних (бронювання/замовлення) у спільній базі даних SQL Server.

Кожна зі складових системи пройшла етапи аналізу, проектування, реалізації та тестування, що дозволило досягти наступних результатів:

1) Аналіз предметної області - формалізовано потреби кінотеатрів та

глядачів, визначено користувацькі ролі, встановлено вимоги до інтерфейсу та архітектури;

2) Архітектура - обрано клієнт-серверну модель з єдиною базою даних, що дозволяє централізовано керувати даними та забезпечувати актуальність на обох рівнях системи;

3) Реалізація десктоп-застосунку - за допомогою WinForms та C#, забезпечено функціональність CRUD-операцій та інтерфейс, зручний для адміністраторів;

4) Розробка вебінтерфейсу - використано ASP.NET Core MVC та Razor для створення адаптивного UX, що дозволяє користувачам бронювати бронювання онлайн;

5) Створення бази даних - у SQL Server реалізовано структуру з таблицями Movies, Halls, Showtimes, Bookings, Orders із взаємозв'язками, що забезпечує цілісність даних;

6) Тестування - виконано модульне, інтеграційне та функціональне тестування, як для десктоп-застосунку, так і для веб-сайту; система показала стабільність, здатність до навантаження та інтуїтивність;

7) Продуктивність і безпека - перевірено одночасне бронювання, тестовано валідацію форм, захист від некоректного введення, оптимізовано запити;

8) Документування та презентація - створено рисунки, таблиці та код-додатки, що підтверджують складність і обґрунтованість рішення.

Таким чином, робота підтвердила відповідність заявленим цілям зі вступу та продемонструвала реальні результати як теоретичних, так і практичних аспектів.

2. Оцінка досягнення цілей

2.1. Аналітичний блок

Проведено глибоке дослідження процесів роботи кінотеатру та поведінки кінцевих користувачів. Визначено, що головне - це:

- швидкість доступу до сеансів;
- гнучкий вибір дати та фільму;

- прозора система бронювання з відсутністю конфліктів;
- адміністраторський контроль репертуару.

Формалізація вимог до обох типів користувачів дозволила правильно побудувати архітектуру і інтерфейс.

2.2. Архітектурне рішення

Застосовано клієнт-серверний підхід із централізованою базою даних SQL Server:

- CDI-модель забезпечує одноразове зберігання даних та їх актуальність на обох рівнях додатків;
- чіткий розподіл ролей та взаємодії між компонентами сприяє підтримці та тестуванню;
- архітектура готова до масштабування - наприклад, підключення реплік або хмарної БД.

2.3. Реалізація Admin-застосунку

WinForms + C# + ADO.NET реалізувалися функціонально повно:

- CRUD-операції з іде чи без IDE рівня складності;
- перевірка валідації, обробка помилок, повідомлення оператора;
- UX-функції - підказки, підсвітлення кнопок, протоколи дій тощо.

2.4. Реалізація Web-інтерфейсу

ASP.NET MVC допоміг досягти:

- відповідності REST-парадигмі (контролери, методи GET/POST);
- розділення відповідальностей - моделі, контролери, представлення;
- адаптивного frontend-ування з Bootstrap/кастомним CSS;
- UX динаміку - вибір форматів, живі підказки, JS-події на сторінках.

2.5. База даних

Ґрунтовна проєктування структури:

- всі таблиці мають PK, FK, індекси;
- моделювання багатозв’язку між сеансом і бронюванням;
- підтримка інформації для звітів зносостійкої схеми.

2.6. Тестування

- модульне: окремі методи перевірені на коректність;
- інтеграційне: взаємодія компонентів показала узгодженість;
- end-to-end: від додавання фільму до бронювання;
- надійність: стрес-тестування підтвердило стабільність, відсутність витоків або зависань;
- ергономіка: інтерфейс інтуїтивний навіть для нетехнічних користувачів.

2.7. Технологічна цілісність

Використані інструменти: Visual Studio, EF Core, SQL Server, WinForms, HTML, CSS, JS, ASP.NET MVC - довели свою доцільність для створення системи автоматизації.

3. Інтеграція результатів і ефективність

3.1. Адмін та користувацька складові

- система має двоканальну архітектуру:
- адміністративний інструмент - централізоване керування;
- публічний веб-інтерфейс - зручний інструмент для клієнтів.

3.2. Узгодженість даних

Завдяки єдиній базі даних інтерфейси демонструють однакову інформацію:

- зміни у адмін-застосунку миттєво відображаються на сайті;
- бронювання з сайту одразу блокуються у адмін-інтерфейсі;
- привабливо для реального використання.

3.3. Продуктивність

- SQL-запити оптимізовані для читання;
- EF Core/ADO.NET працюють швидко при середньому навантаженні;
- сайт відображає сторінки <1с, бронювання займає <2–3с;
- десктоп-частина працює локально з майже миттєвим відгуком.

4. Обмеження та ризики

4.1. Безпека

- поки що немає функції login/password - лише тимчасова авторизація оператора;
- ввід номера карти без токенізації - не відповідає PCI DSS;
- плани на future: OAuth, шифрування, HSTS, HTTPS.

4.2. Масштабованість

- SQL Server достатньо для невеликих середовищ, але при завантаженні потрібне sharding/Cloud;
- реляційна модель не підтримує offline caching.

Поетапне розширення можливостей:

Таблиця 1.1

Майбутній функціонал	Опис
Реєстрація\авторизація користувачів	Індивідуальні кабінети, історія бронювань, ролі (користувач, адмін).
Онлайн-оплата	Stripe, LiqPay, Monobank - інтеграція з реальними платіжними системами.
Повідомлення	SMS або Email-підтвердження бронювання.
Адаптивність	Mobile-first дизайн, SPA/PWA через Blazor WebAssembly або Angular.
Інтеграція	API для мобільного додатку, партнерських платформ (агрегаторів фільмів).

6. Науково-практична цінність:

- Демонстрація комплексного підходу від аналізу до тестування;
- Доведення ефективності платформи .NET + SQL Server для індустріальних застосунків;
- Методологічний інструмент для навчання інформаційним системам;

- Базовий прототип для комерціалізації - готовий для MVP.

7. Висновок

Дипломний проєкт цілком виправдав поставлені цілі:

- виконано глибокий аналіз і формалізовано вимоги;
- створено ефективну архітектуру та інструментарій;
- реалізовано, протестовано і готово до запуску двокомпонентне рішення;
- обґрунтовані напрями подальшого розвитку.

Система можна успішно впроваджувати в кінотеатрах, використовувати в навчальних цілях та вдосконалювати для комерційного використання.

Кінцевим результатом стала практична демонстрація того, яким має бути сучасне програмне рішення - технологічно міцне, логічно продумане, дружнє до користувачів, безпечне та готове до майбутнього.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Microsoft Learn. ASP.NET Core MVC documentation - https://learn.microsoft.com/en-us/aspnet/core/mvc/

2. Microsoft Learn. SQL Server documentation - https://learn.microsoft.com/en-us/sql/sql-server/

3. Microsoft Learn. Windows Forms documentation - https://learn.microsoft.com/en-us/dotnet/desktop/winforms/

4. Bootstrap 5 Documentation - https://getbootstrap.com/docs/5.3/getting-started/introduction/

5. Price J. C# 12 and .NET 8 – Modern Cross-Platform Development. - Packt Publishing, 2024. — 560 p.

6. SQLBolt. Learn SQL with simple, interactive exercises - https://sqlbolt.com/

ДОДАТКИ

Додаток А

Лістинг елементів головної форми «Form1.Designer.cs»

```
using System;
using System.Data;
using System.Data.SqlClient;
using System.Windows.Forms;

namespace CinemaBookingApp
{
    partial class Form1
    {
        /// <summary>
        /// Обов'язкова змінна конструктора.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        /// Звільнити всі використовувані ресурси.
        /// </summary>
        /// <param name="disposing">true, якщо керовані ресурси повинні бути видалені; інакше false.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }
```

```
        #region Код, автоматично створений конструктором форм Windows

        /// <summary>
        /// </summary>
        private void InitializeComponent()
        {
            this.dgvMovies = new System.Windows.Forms.DataGridView();
            this.dgvShowtimes = new System.Windows.Forms.DataGridView();
            this.dgvSeats = new System.Windows.Forms.DataGridView();
            this.btnBook = new System.Windows.Forms.Button();
            this.lblStatus = new System.Windows.Forms.Label();
            this.txtTitle = new System.Windows.Forms.TextBox();
            this.txtDuration = new System.Windows.Forms.TextBox();
            this.txtGenre = new System.Windows.Forms.TextBox();
            this.label1 = new System.Windows.Forms.Label();
            this.label2 = new System.Windows.Forms.Label();
            this.label3 = new System.Windows.Forms.Label();
            this.btnAddMovie = new System.Windows.Forms.Button();
            this.dtpStartTime = new System.Windows.Forms.DateTimePicker();
            this.btnAddShowtime = new System.Windows.Forms.Button();
            this.tabControl1 = new System.Windows.Forms.TabControl();
            this.tabPage1 = new System.Windows.Forms.TabPage();
            this.listBoxMovies = new System.Windows.Forms.ListBox();
            this.tabPage2 = new System.Windows.Forms.TabPage();
            this.tabPage3 = new System.Windows.Forms.TabPage();
            this.label4 = new System.Windows.Forms.Label();
            this.cbMoviesForShowtime = new System.Windows.Forms.Com-
boBox();
```

```
            this.btnBookSeat = new System.Windows.Forms.Button();
            ((System.ComponentModel.ISupportInitialize)(this.dgvMov-
ies)).BeginInit();
            ((System.ComponentModel.ISupportInitialize)(this.dgvShow-
times)).BeginInit();
            ((System.ComponentModel.ISupportInitial-
ize)(this.dgvSeats)).BeginInit();
            this.tabControl1.SuspendLayout();
            this.tabPage1.SuspendLayout();
            this.tabPage2.SuspendLayout();
            this.tabPage3.SuspendLayout();
            this.SuspendLayout();
            //
            // dgvMovies
            //
            this.dgvMovies.ColumnHeadersHeightSizeMode = System.Win-
dows.Forms.DataGridViewColumnHeadersHeightSizeMode.AutoSize;
            this.dgvMovies.Location = new System.Drawing.Point(6, 6);
            this.dgvMovies.Name = "dgvMovies";
            this.dgvMovies.Size = new System.Drawing.Size(351, 354);
            this.dgvMovies.TabIndex = 0;
            this.dgvMovies.SelectionChanged += new Sys-
tem.EventHandler(this.dgvMovies_SelectionChanged);
            //
            // dgvShowtimes
            //
            this.dgvShowtimes.ColumnHeadersHeightSizeMode = System.Win-
dows.Forms.DataGridViewColumnHeadersHeightSizeMode.AutoSize;
```

```
            this.dgvShowtimes.Location = new System.Drawing.Point(444, 6);
            this.dgvShowtimes.Name = "dgvShowtimes";
            this.dgvShowtimes.Size = new System.Drawing.Size(347, 354);
            this.dgvShowtimes.TabIndex = 1;
            //
            // dgvSeats
            //
            this.dgvSeats.ColumnHeadersHeightSizeMode = System.Win-
dows.Forms.DataGridViewColumnHeadersHeightSizeMode.AutoSize;
            this.dgvSeats.Location = new System.Drawing.Point(6, 393);
            this.dgvSeats.Name = "dgvSeats";
            this.dgvSeats.Size = new System.Drawing.Size(351, 171);
            this.dgvSeats.TabIndex = 2;
            //
            // btnBook
            //
            this.btnBook.Location = new System.Drawing.Point(444, 393);
            this.btnBook.Name = "btnBook";
            this.btnBook.Size = new System.Drawing.Size(183, 53);
            this.btnBook.TabIndex = 3;
            this.btnBook.Text = "Забронювати місце";
            this.btnBook.UseVisualStyleBackColor = true;
            //
            // lblStatus
            //
            this.lblStatus.AutoSize = true;
            this.lblStatus.ForeColor = System.Drawing.Color.PaleGreen;
            this.lblStatus.Location = new System.Drawing.Point(685, 413);
```

```
this.lblStatus.Name = "lblStatus";
this.lblStatus.Size = new System.Drawing.Size(35, 13);
this.lblStatus.TabIndex = 4;
this.lblStatus.Text = "label1";
//
// txtTitle
//
this.txtTitle.Location = new System.Drawing.Point(113, 95);
this.txtTitle.Name = "txtTitle";
this.txtTitle.Size = new System.Drawing.Size(122, 20);
this.txtTitle.TabIndex = 5;
//
// txtDuration
//
this.txtDuration.Location = new System.Drawing.Point(113, 144);
this.txtDuration.Name = "txtDuration";
this.txtDuration.Size = new System.Drawing.Size(122, 20);
this.txtDuration.TabIndex = 6;
//
// txtGenre
//
this.txtGenre.Location = new System.Drawing.Point(113, 192);
this.txtGenre.Name = "txtGenre";
this.txtGenre.Size = new System.Drawing.Size(122, 20);
this.txtGenre.TabIndex = 7;
//
// label1
//
```

```
this.label1.AutoSize = true;
this.label1.Location = new System.Drawing.Point(54, 95);
this.label1.Name = "label1";
this.label1.Size = new System.Drawing.Size(39, 13);
this.label1.TabIndex = 8;
this.label1.Text = "Назва";
//
// label2
//
this.label2.AutoSize = true;
this.label2.Location = new System.Drawing.Point(30, 144);
this.label2.Name = "label2";
this.label2.Size = new System.Drawing.Size(63, 13);
this.label2.TabIndex = 9;
this.label2.Text = "Тривалість";
//
// label3
//
this.label3.AutoSize = true;
this.label3.Location = new System.Drawing.Point(57, 195);
this.label3.Name = "label3";
this.label3.Size = new System.Drawing.Size(36, 13);
this.label3.TabIndex = 10;
this.label3.Text = "Жанр";
//
// btnAddMovie
//
this.btnAddMovie.Location = new System.Drawing.Point(33, 254);
```

```
            this.btnAddMovie.Name = "btnAddMovie";
            this.btnAddMovie.Size = new System.Drawing.Size(202, 36);
            this.btnAddMovie.TabIndex = 11;
            this.btnAddMovie.Text = "Додати фільм";
            this.btnAddMovie.UseVisualStyleBackColor = true;
            this.btnAddMovie.Click += new System.EventHandler(this.btnAdd-
Movie_Click_1);
            //
            // dtpStartTime
            //
            this.dtpStartTime.CustomFormat = "yyyy-MM-dd HH:mm";
            this.dtpStartTime.Format = System.Windows.Forms.DateTimePicker-
Format.Custom;
            this.dtpStartTime.Location = new System.Drawing.Point(96, 44);
            this.dtpStartTime.Name = "dtpStartTime";
            this.dtpStartTime.Size = new System.Drawing.Size(235, 20);
            this.dtpStartTime.TabIndex = 12;
            //
            // btnAddShowtime
            //
            this.btnAddShowtime.Location = new System.Drawing.Point(96, 155);
            this.btnAddShowtime.Name = "btnAddShowtime";
            this.btnAddShowtime.Size = new System.Drawing.Size(235, 50);
            this.btnAddShowtime.TabIndex = 13;
            this.btnAddShowtime.Text = "Додати сеанс";
            this.btnAddShowtime.UseVisualStyleBackColor = true;
            this.btnAddShowtime.Click += new Sys-
tem.EventHandler(this.btnAddShowtime_Click);
```

```
//
// tabControl1
//
this.tabControl1.Controls.Add(this.tabPage1);
this.tabControl1.Controls.Add(this.tabPage2);
this.tabControl1.Controls.Add(this.tabPage3);
this.tabControl1.Location = new System.Drawing.Point(-2, -2);
this.tabControl1.Name = "tabControl1";
this.tabControl1.SelectedIndex = 0;
this.tabControl1.Size = new System.Drawing.Size(1193, 621);
this.tabControl1.TabIndex = 14;
//
// tabPage1
//
this.tabPage1.Controls.Add(this.btnBookSeat);
this.tabPage1.Controls.Add(this.listBoxMovies);
this.tabPage1.Controls.Add(this.dgvMovies);
this.tabPage1.Controls.Add(this.dgvShowtimes);
this.tabPage1.Controls.Add(this.dgvSeats);
this.tabPage1.Controls.Add(this.btnBook);
this.tabPage1.Controls.Add(this.lblStatus);
this.tabPage1.Location = new System.Drawing.Point(4, 22);
this.tabPage1.Name = "tabPage1";
this.tabPage1.Padding = new System.Windows.Forms.Padding(3);
this.tabPage1.Size = new System.Drawing.Size(1185, 595);
this.tabPage1.TabIndex = 0;
this.tabPage1.Text = "Бронювання місць";
this.tabPage1.UseVisualStyleBackColor = true;
```

```
//
// listBoxMovies
//
this.listBoxMovies.FormattingEnabled = true;
this.listBoxMovies.Location = new System.Drawing.Point(832, 6);
this.listBoxMovies.Name = "listBoxMovies";
this.listBoxMovies.Size = new System.Drawing.Size(327, 550);
this.listBoxMovies.TabIndex = 5;
//
// tabPage2
//
this.tabPage2.Controls.Add(this.btnAddMovie);
this.tabPage2.Controls.Add(this.txtTitle);
this.tabPage2.Controls.Add(this.txtDuration);
this.tabPage2.Controls.Add(this.txtGenre);
this.tabPage2.Controls.Add(this.label3);
this.tabPage2.Controls.Add(this.label1);
this.tabPage2.Controls.Add(this.label2);
this.tabPage2.Location = new System.Drawing.Point(4, 22);
this.tabPage2.Name = "tabPage2";
this.tabPage2.Padding = new System.Windows.Forms.Padding(3);
this.tabPage2.Size = new System.Drawing.Size(1185, 595);
this.tabPage2.TabIndex = 1;
this.tabPage2.Text = "Додавання фільмів";
this.tabPage2.UseVisualStyleBackColor = true;
//
// tabPage3
//
```

```
this.tabPage3.Controls.Add(this.label4);
this.tabPage3.Controls.Add(this.cbMoviesForShowtime);
this.tabPage3.Controls.Add(this.dtpStartTime);
this.tabPage3.Controls.Add(this.btnAddShowtime);
this.tabPage3.Location = new System.Drawing.Point(4, 22);
this.tabPage3.Name = "tabPage3";
this.tabPage3.Padding = new System.Windows.Forms.Padding(3);
this.tabPage3.Size = new System.Drawing.Size(1185, 595);
this.tabPage3.TabIndex = 2;
this.tabPage3.Text = "Додавання сеансів";
this.tabPage3.UseVisualStyleBackColor = true;
//
// label4
//
this.label4.AutoSize = true;
this.label4.Location = new System.Drawing.Point(41, 90);
this.label4.Name = "label4";
this.label4.Size = new System.Drawing.Size(43, 13);
this.label4.TabIndex = 15;
this.label4.Text = "Фільм:";
//
// cbMoviesForShowtime
//
this.cbMoviesForShowtime.FormattingEnabled = true;
this.cbMoviesForShowtime.Location = new System.Drawing.Point(96,
87);
this.cbMoviesForShowtime.Name = "cbMoviesForShowtime";
this.cbMoviesForShowtime.Size = new System.Drawing.Size(234, 21);
```

```
            this.cbMoviesForShowtime.TabIndex = 14;
            //
            // btnBookSeat
            //
            this.btnBookSeat.Location = new System.Drawing.Point(444, 492);
            this.btnBookSeat.Name = "btnBookSeat";
            this.btnBookSeat.Size = new System.Drawing.Size(347, 48);
            this.btnBookSeat.TabIndex = 0;
            this.btnBookSeat.Text = "Забронювати";
            //
            // Form1
            //
            this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.ClientSize = new System.Drawing.Size(1296, 626);
            this.Controls.Add(this.tabControl1);
            this.Name = "Form1";
            this.Text = "Form1";
            ((System.ComponentModel.ISupportInitialize)(this.dgvMovies)).En-
dInit();
            ((System.ComponentModel.ISupportInitialize)(this.dgvShow-
times)).EndInit();
            ((System.ComponentModel.ISupportInitialize)(this.dgvSeats)).En-
dInit();
            this.tabControl1.ResumeLayout(false);
            this.tabPage1.ResumeLayout(false);
            this.tabPage1.PerformLayout();
            this.tabPage2.ResumeLayout(false);
```

```
    this.tabPage2.PerformLayout();
    this.tabPage3.ResumeLayout(false);
    this.tabPage3.PerformLayout();
    this.ResumeLayout(false);

}

#endregion

private System.Windows.Forms.DataGridView dgvMovies;
private System.Windows.Forms.DataGridView dgvShowtimes;
private System.Windows.Forms.DataGridView dgvSeats;
private System.Windows.Forms.Button btnBook;
private System.Windows.Forms.Label lblStatus;
private System.Windows.Forms.DateTimePicker dtpStartTime;
private System.Windows.Forms.Button btnAddShowtime;
private System.Windows.Forms.TextBox txtTitle;
private System.Windows.Forms.TextBox txtDuration;
private System.Windows.Forms.TextBox txtGenre;
private System.Windows.Forms.Label label1;
private System.Windows.Forms.Label label2;
private System.Windows.Forms.Label label3;
private System.Windows.Forms.Button btnAddMovie;
private System.Windows.Forms.Button btnBookSeat;
private TabControl tabControl1;
private TabPage tabPage1;
private TabPage tabPage2;
private TabPage tabPage3;
```

```
        private Label label4;
        private ComboBox cbMoviesForShowtime;
        private ListBox listBoxMovies;
    }
}
```

Додаток Б

Лістинг вхідної точки програми «Program.cs »

```
using System;
using System.Data;
using System.Data.SqlClient;
using System.Windows.Forms;

namespace CinemaBookingApp
{
   public partial class Form1 : Form
   {
      private string connectionString = "Server=DESKTOP-TV91AFR\\SQLEXPRESS;Database=CinemaBookingSystem;Trusted_Connection=True;";

      public DataGridView DgvMovies => dgvMovies;
      public DataGridView DgvShowtimes => dgvShowtimes;
      public ComboBox CbMoviesForShowtime => cbMoviesForShowtime;
      public ListBox ListBoxMovies => listBoxMovies;
      public Label LblStatus => lblStatus;
      public Button BtnAddMovie => btnAddMovie;
      public Button BtnAddShowtime => btnAddShowtime;
      public TextBox TxtTitle => txtTitle;
      public TextBox TxtGenre => txtGenre;
      public TextBox TxtDuration => txtDuration;
      public DateTimePicker DtpStartTime => dtpStartTime;
      public Button BtnBookSeat => btnBookSeat;

      public Form1()
```

```
        {
            InitializeComponent();
            this.dgvShowtimes.SelectionChanged += dgvShowtimes_Selection-
Changed;
            this.dgvMovies.SelectionChanged += dgvMovies_SelectionChanged;
            this.btnAddShowtime.Click += btnAddShowtime_Click;
            this.btnAddMovie.Click += btnAddMovie_Click_1;
            this.Load += Form1_Load;
            this.listBoxMovies.SelectedIndexChanged += listBoxMovies_Selecte-
dIndexChanged;

            this.btnBookSeat.Click += btnBookSeat_Click;
            ApplyTheme();
            AddButtonHoverEffect(btnAddMovie);
            AddButtonHoverEffect(btnAddShowtime);
            AddButtonHoverEffect(btnBookSeat);
            AddButtonHoverEffect(btnBook);
        }

        private void ApplyTheme()
        {
            this.BackColor = System.Drawing.Color.FromArgb(30, 30, 30);
            this.Font = new System.Drawing.Font("Segoe UI", 10F);

            // DataGridView
            foreach (var dgv in new[] { dgvMovies, dgvShowtimes, dgvSeats })
            {
```

```
            dgv.BackgroundColor = System.Drawing.Color.FromArgb(45, 45, 48);
            dgv.DefaultCellStyle.BackColor = System.Drawing.Color.FromArgb(45, 45, 48);
            dgv.DefaultCellStyle.ForeColor = System.Drawing.Color.White;
            dgv.DefaultCellStyle.SelectionBackColor = System.Drawing.Color.Orange;
            dgv.DefaultCellStyle.SelectionForeColor = System.Drawing.Color.Black;
            dgv.ColumnHeadersDefaultCellStyle.BackColor = System.Drawing.Color.FromArgb(30, 30, 30);
            dgv.ColumnHeadersDefaultCellStyle.ForeColor = System.Drawing.Color.White;
            dgv.EnableHeadersVisualStyles = false;
            dgv.GridColor = System.Drawing.Color.Gray;
        }

        // ListBox
        listBoxMovies.BackColor = System.Drawing.Color.FromArgb(45, 45, 48);
        listBoxMovies.ForeColor = System.Drawing.Color.White;

        // Labels
        foreach (var lbl in new[] { label1, label2, label3, label4, lblStatus })
            lbl.ForeColor = System.Drawing.Color.White;

        // TextBox
        foreach (var tb in new[] { txtTitle, txtDuration, txtGenre })
```

```
            {
                tb.BackColor = System.Drawing.Color.FromArgb(45, 45, 48);
                tb.ForeColor = System.Drawing.Color.White;
                tb.BorderStyle = BorderStyle.FixedSingle;
            }
            // ComboBox
            cbMoviesForShowtime.BackColor = System.Drawing.Color.FromArgb(45, 45, 48);
            cbMoviesForShowtime.ForeColor = System.Drawing.Color.White;

            // DateTimePicker
            dtpStartTime.CalendarMonthBackground = System.Drawing.Color.FromArgb(45, 45, 48);
            dtpStartTime.CalendarForeColor = System.Drawing.Color.White;
            dtpStartTime.BackColor = System.Drawing.Color.FromArgb(45, 45, 48);
            dtpStartTime.ForeColor = System.Drawing.Color.White;

            // Кнопки
            foreach (var btn in new[] { btnAddMovie, btnAddShowtime, btnBookSeat, btnBook })
            {
                btn.BackColor = System.Drawing.Color.FromArgb(60, 60, 60);
                btn.ForeColor = System.Drawing.Color.White;
                btn.FlatStyle = FlatStyle.Flat;
                btn.FlatAppearance.BorderColor = System.Drawing.Color.Orange;
                btn.FlatAppearance.BorderSize = 1;
            }
```

```
            // TabControl і TabPages
            tabControl1.BackColor = System.Drawing.Color.FromArgb(30, 30, 30);
            tabPage1.BackColor = System.Drawing.Color.FromArgb(30, 30, 30);
            tabPage2.BackColor = System.Drawing.Color.FromArgb(30, 30, 30);
            tabPage3.BackColor = System.Drawing.Color.FromArgb(30, 30, 30);
        }

        private void AddButtonHoverEffect(Button btn)
        {
            btn.MouseEnter += (s, e) => btn.BackColor = System.Drawing.Color.Orange;
            btn.MouseLeave += (s, e) => btn.BackColor = System.Drawing.Color.FromArgb(60, 60, 60);
        }
        private void Form1_Load(object sender, EventArgs e)
        {
            LoadMovies();
            LoadMoviesToComboBox();
            LoadMoviesToListBox();
        }
        private void LoadMovies()
        {
            using (SqlConnection connection = new SqlConnection(connectionString))
            {
                try
                {
                    connection.Open();
```

```
                string query = "SELECT id, Title, Genre, DurationMinutes FROM
Movies";
                SqlDataAdapter adapter = new SqlDataAdapter(query, connection);
                DataTable table = new DataTable();
                adapter.Fill(table);
                dgvMovies.DataSource = table;
                lblStatus.Text = $"Знайдено {table.Rows.Count} фільм(ів).";
            }
            catch (Exception ex)
            {
                MessageBox.Show("Помилка при завантаженні фільмів: " +
ex.Message);
            }
        }
    }
    private void LoadMoviesToComboBox()
    {
        cbMoviesForShowtime.Items.Clear();
        using (SqlConnection connection = new SqlConnection(connection-
String))
        {
            try
            {
                connection.Open();
                string query = "SELECT id, Title FROM Movies";
                SqlCommand command = new SqlCommand(query, connection);
                SqlDataReader reader = command.ExecuteReader();
                while (reader.Read())
```

```
                {
                    cbMoviesForShowtime.Items.Add(new ComboBoxItem
                    {
                        Id = Convert.ToInt32(reader["id"]),
                        Title = reader["Title"].ToString()
                    });
                }
                if (cbMoviesForShowtime.Items.Count > 0)
                    cbMoviesForShowtime.SelectedIndex = 0;
            }
            catch (Exception ex)
            {
                MessageBox.Show("Помилка при завантаженні фільмів у список: " + ex.Message);
            }
        }
    }
    private void LoadMoviesToListBox()
    {
        listBoxMovies.Items.Clear();
        using (SqlConnection connection = new SqlConnection(connectionString))
        {
            try
            {
                connection.Open();
                string query = "SELECT id, Title FROM Movies";
                SqlCommand command = new SqlCommand(query, connection);
```

```
                SqlDataReader reader = command.ExecuteReader();
                while (reader.Read())
                {
                    listBoxMovies.Items.Add(new ListBoxMovieItem
                    {
                        Id = Convert.ToInt32(reader["id"]),
                        Title = reader["Title"].ToString()
                    });
                }
                listBoxMovies.DisplayMember = "Title";
            }
            catch (Exception ex)
            {
                MessageBox.Show("Помилка при завантаженні фільмів у список для бронювання: " + ex.Message);
            }
        }
    }
    private void AddMovie(string title, int durationMinutes, string genre)
    {
        string query = "INSERT INTO Movies (Title, DurationMinutes, Genre) VALUES (@Title, @DurationMinutes, @Genre)";

        using (SqlConnection connection = new SqlConnection(connectionString))
        {
            SqlCommand command = new SqlCommand(query, connection);
            command.Parameters.AddWithValue("@Title", title);
```

```
            command.Parameters.AddWithValue("@DurationMinutes", dura-
tionMinutes);
            command.Parameters.AddWithValue("@Genre", genre);

            connection.Open();
            command.ExecuteNonQuery();
        }
    }
    private void btnAddMovie_Click_1(object sender, EventArgs e)
    {
        try
        {
            string title = txtTitle.Text.Trim();
            string genre = txtGenre.Text.Trim();
            if (!int.TryParse(txtDuration.Text.Trim(), out int duration))
            {
                MessageBox.Show("Введи коректну тривалість у хвилинах.");
                return;
            }
            if (string.IsNullOrEmpty(title) || string.IsNullOrEmpty(genre))
            {
                MessageBox.Show("Будь ласка, заповни всі поля.");
                return;
            }

            AddMovie(title, duration, genre);
            MessageBox.Show("Фільм додано!");
```

```
            LoadMovies();
            LoadMoviesToComboBox();
            LoadMoviesToListBox();
        }
        catch (Exception ex)
        {
            MessageBox.Show("Помилка при додаванні: " + ex.Message);
        }
    }

    private void LoadShowtimes(int movieId)
    {
        using (SqlConnection connection = new SqlConnection(connection-
String))
        {
            try
            {
                connection.Open();
                string query = "SELECT Id, StartTime FROM Showtimes WHERE
MovieId = @MovieId";
                SqlCommand command = new SqlCommand(query, connection);
                command.Parameters.AddWithValue("@MovieId", movieId);

                SqlDataAdapter adapter = new SqlDataAdapter(command);
                DataTable table = new DataTable();
                adapter.Fill(table);

                dgvShowtimes.DataSource = table;
```

```
                if (dgvShowtimes.Columns.Contains("Id"))
                    dgvShowtimes.Columns["Id"].Visible = false;

                if (dgvShowtimes.Columns.Contains("StartTime"))
                {
                    dgvShowtimes.Columns["StartTime"].HeaderText = "Час початку";
                    dgvShowtimes.Columns["StartTime"].DefaultCellStyle.Format = "dd.MM.yyyy HH:mm";
                }
                dgvShowtimes.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.Fill;
                dgvShowtimes.SelectionMode = DataGridViewSelectionMode.FullRowSelect;
                dgvShowtimes.MultiSelect = false;

                lblStatus.Text = $"Знайдено {table.Rows.Count} сеанс(ів).";
            }
            catch (Exception ex)
            {
                MessageBox.Show("Помилка при завантаженні сеансів: " + ex.Message);
            }
        }
    }
    private void LoadSeats(int showtimeId)
    {
        dgvSeats.Rows.Clear();
```

```
            dgvSeats.Columns.Clear();

            int hallId = 0;
            int totalSeats = 0;

            using (SqlConnection connection = new SqlConnection(connection-
String))
            {
                connection.Open();
                string query = "SELECT HallId FROM Showtimes WHERE Id =
@ShowtimeId";
                SqlCommand command = new SqlCommand(query, connection);
                command.Parameters.AddWithValue("@ShowtimeId", showtimeId);
                var result = command.ExecuteScalar();
                if (result != null)
                    hallId = Convert.ToInt32(result);

                query = "SELECT TotalSeats FROM Halls WHERE Id = @HallId";
                command = new SqlCommand(query, connection);
                command.Parameters.AddWithValue("@HallId", hallId);
                result = command.ExecuteScalar();
                if (result != null)
                    totalSeats = Convert.ToInt32(result);

                query = "SELECT SeatNumber FROM Bookings WHERE Show-
timeId = @ShowtimeId";
                command = new SqlCommand(query, connection);
                command.Parameters.AddWithValue("@ShowtimeId", showtimeId);
```

```
            SqlDataReader reader = command.ExecuteReader();

            var bookedSeats = new System.Collections.Generic.HashSet<int>();
            while (reader.Read())
            {
                bookedSeats.Add(Convert.ToInt32(reader["SeatNumber"]));
            }
            reader.Close();

            dgvSeats.Columns.Add("SeatNumber", "Місце");
            dgvSeats.Columns.Add("Status", "Статус");

            for (int i = 1; i <= totalSeats; i++)
            {
                string status = bookedSeats.Contains(i) ? "Зайняте" : "Вільне";
                dgvSeats.Rows.Add(i, status);
            }

            dgvSeats.AutoSizeColumnsMode = DataGridViewAutoSizeCol-
umnsMode.Fill;
            dgvSeats.SelectionMode = DataGridViewSelectionMode.FullRowSe-
lect;
            dgvSeats.MultiSelect = false;
        }
    }

    private void dgvMovies_SelectionChanged(object sender, EventArgs e)
    {
```

```
            if (dgvMovies.SelectedRows.Count > 0)
            {
                int movieId = Convert.ToInt32(dgvMovies.Select-
edRows[0].Cells["id"].Value);
                LoadShowtimes(movieId);
            }
        }

        private void dgvShowtimes_SelectionChanged(object sender, EventArgs
e)
        {
            if (dgvShowtimes.SelectedRows.Count > 0)
            {
                int showtimeId = Convert.ToInt32(dgvShowtimes.Select-
edRows[0].Cells["Id"].Value);
                LoadSeats(showtimeId);
            }
        }

        private void listBoxMovies_SelectedIndexChanged(object sender, Even-
tArgs e)
        {
            if (listBoxMovies.SelectedItem is ListBoxMovieItem selectedMovie)
            {
                LoadShowtimes(selectedMovie.Id);
            }
        }
```

```
        private void btnAddShowtime_Click(object sender, EventArgs e)
        {
            if (cbMoviesForShowtime.SelectedItem == null)
            {
                MessageBox.Show("Оберіть фільм для додавання сеансу.");
                return;
            }

            var selectedMovie = cbMoviesForShowtime.SelectedItem as ComboBoxItem;
            int movieId = selectedMovie.Id;
            DateTime startTime = dtpStartTime.Value;


            using (SqlConnection connection = new SqlConnection(connectionString))
            {
                try
                {
                    connection.Open();
                    string query = "INSERT INTO Showtimes (MovieId, HallId, StartTime) VALUES (@MovieId, @HallId, @StartTime)";
                    SqlCommand command = new SqlCommand(query, connection);
                    command.Parameters.AddWithValue("@MovieId", movieId);
                    command.Parameters.AddWithValue("@HallId", 1);
                    command.Parameters.AddWithValue("@StartTime", startTime);
                    command.ExecuteNonQuery();
```

```
                MessageBox.Show("Сеанс додано!");
                LoadShowtimes(movieId);
            }
            catch (Exception ex)
            {
                MessageBox.Show("Помилка при додаванні сеансу: " + ex.Mes-
sage);
            }
        }
    }
    private void btnBookSeat_Click(object sender, EventArgs e)
    {
        if (dgvShowtimes.SelectedRows.Count == 0 || dgvSeats.Select-
edRows.Count == 0)
        {
            MessageBox.Show("Оберіть сеанс і місце!");
            return;
        }

        int showtimeId = Convert.ToInt32(dgvShowtimes.Select-
edRows[0].Cells["Id"].Value);
        int seatNumber = Convert.ToInt32(dgvSeats.Select-
edRows[0].Cells["SeatNumber"].Value);
        string seatStatus = dgvSeats.SelectedRows[0].Cells["Sta-
tus"].Value.ToString();
        if (seatStatus == "Зайняте")
        {
            MessageBox.Show("Місце вже заброньоване!");
```

```
            return;
        }

        using (var bookingForm = new BookingForm())
        {
            if (bookingForm.ShowDialog() == DialogResult.OK)
            {
                string movieTitle = "";
                DateTime showtime = DateTime.Now;

                using (SqlConnection connection = new SqlConnection(connectionString))
                {
                    connection.Open();
                    string query = "SELECT m.Title, s.StartTime FROM Showtimes s JOIN Movies m ON s.MovieId = m.Id WHERE s.Id = @ShowtimeId";
                    SqlCommand command = new SqlCommand(query, connection);
                    command.Parameters.AddWithValue("@ShowtimeId", showtimeId);
                    using (var reader = command.ExecuteReader())
                    {
                        if (reader.Read())
                        {
                            movieTitle = reader["Title"].ToString();
                            showtime = Convert.ToDateTime(reader["StartTime"]);
                        }
                    }
```

```
            }

            string ticketCode = GenerateTicketCode(movieTitle, showtime, seatNumber);

            using (SqlConnection connection = new SqlConnection(connectionString))
            {
                connection.Open();
                string query = "INSERT INTO Bookings (ShowtimeId, SeatNumber, BookingTime) VALUES (@ShowtimeId, @SeatNumber, GETDATE())";
                SqlCommand command = new SqlCommand(query, connection);
                command.Parameters.AddWithValue("@ShowtimeId", showtimeId);
                command.Parameters.AddWithValue("@SeatNumber", seatNumber);
                command.ExecuteNonQuery();
            }

            using (SqlConnection connection = new SqlConnection(connectionString))
            {
                connection.Open();
                string query = "INSERT INTO Orders (Phone, CardNumber, MovieTitle, Showtime, SeatNumber, TicketCode) VALUES (@Phone, @CardNumber, @MovieTitle, @Showtime, @SeatNumber, @TicketCode)";
```

```
                    SqlCommand command = new SqlCommand(query, connection);
                    command.Parameters.AddWithValue("@Phone", bookingForm.Phone);
                    command.Parameters.AddWithValue("@CardNumber", bookingForm.CardNumber);
                    command.Parameters.AddWithValue("@MovieTitle", movieTitle);
                    command.Parameters.AddWithValue("@Showtime", showtime);
                    command.Parameters.AddWithValue("@SeatNumber", seatNumber);
                    command.Parameters.AddWithValue("@TicketCode", ticketCode);
                    command.ExecuteNonQuery();
                }

                MessageBox.Show("Бронювання успішне!\nВаш код-квиток: " + ticketCode);
                LoadSeats(showtimeId);
            }
        }
    }

    private string GenerateTicketCode(string movieTitle, DateTime showtime, int seatNumber)
    {
```

```
            string titlePart = movieTitle.Length >= 3 ? movieTitle.Substring(0, 3).ToUpper() : movieTitle.ToUpper();
            string datePart = showtime.ToString("ddMMyyHHmm");
            string seatPart = seatNumber.ToString("D2");
            string randomPart = new Random().Next(100, 999).ToString();
            return $"{titlePart}{datePart}{seatPart}{randomPart}";
        }
        public class ComboBoxItem
        {
            public int Id { get; set; }
            public string Title { get; set; }
            public override string ToString()
            {
                return Title;
            }
        }
        public class ListBoxMovieItem
        {
            public int Id { get; set; }
            public string Title { get; set; }
            public override string ToString()
            {
                return Title;
            }
        }
    }
}
```

Додаток В

Лістинг взаємодії форми з базою даних «Form1.cs »

```
using System;
using System.Windows.Forms;

namespace CinemaBookingApp
{
    public partial class BookingForm : Form
    {
        public string Phone { get; private set; }
        public string CardNumber { get; private set; }

        public BookingForm()
        {
            InitializeComponent();
        }

        private void InitializeComponent()
        {
            this.txtPhone = new TextBox();
            this.txtCard = new TextBox();
            this.btnOk = new Button();
            this.btnCancel = new Button();

            this.txtPhone.Location = new System.Drawing.Point(20, 20);
            this.txtPhone.Text = "Телефон";
            this.txtCard.Location = new System.Drawing.Point(20, 60);
            this.txtCard.Text = "Номер карти";
```

```
            this.btnOk.Text = "OK";
            this.btnOk.Location = new System.Drawing.Point(20, 100);
            this.btnOk.Click += BtnOk_Click;

            this.btnCancel.Text = "Відміна";
            this.btnCancel.Location = new System.Drawing.Point(100, 100);
            this.btnCancel.Click += (s, e) => this.DialogResult = DialogResult.Can-
cel;

            this.Controls.Add(this.txtPhone);
            this.Controls.Add(this.txtCard);
            this.Controls.Add(this.btnOk);
            this.Controls.Add(this.btnCancel);

            this.ClientSize = new System.Drawing.Size(220, 150);
            this.Text = "Введіть дані для бронювання";
        }

        private void BtnOk_Click(object sender, EventArgs e)
        {
            if (string.IsNullOrWhiteSpace(txtPhone.Text) || string.IsNullOr-
WhiteSpace(txtCard.Text))
            {
                MessageBox.Show("Введіть телефон та номер карти!");
                return;
            }
            Phone = txtPhone.Text.Trim();
            CardNumber = txtCard.Text.Trim();
```

```
            this.DialogResult = DialogResult.OK;
        }

        private TextBox txtPhone;
        private TextBox txtCard;
        private Button btnOk;
        private Button btnCancel;
    }
}
```

Додаток Г

Лістинг контролерів сайту

BookingsController.cs:

```
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using CinemaBookingWeb.Data;
using CinemaBookingWeb.Models;

namespace CinemaBookingWeb.Controllers
{
    public class BookingsController : Controller
    {
        private readonly CinemaDbContext _context;

        public BookingsController(CinemaDbContext context)
        {
            _context = context;
        }

        public async Task<IActionResult> SelectSeat(int showtimeId)
        {
            var showtime = await _context.Showtimes
                .Include(s => s.Hall)
                .Include(s => s.Movie)
                .FirstOrDefaultAsync(s => s.Id == showtimeId);

            if (showtime == null) return NotFound();
```

```
            var bookedSeats = await _context.Bookings
                .Where(b => b.ShowtimeId == showtimeId)
                .Select(b => b.SeatNumber)
                .ToListAsync();

            ViewBag.BookedSeats = bookedSeats;
            ViewBag.Price = 120;
            return View(showtime);
        }

        [HttpPost]
        public async Task<IActionResult> Book(int showtimeId, int seatNumber, string phone, string cardNumber)
        {
            bool isBooked = await _context.Bookings.AnyAsync(b => b.ShowtimeId == showtimeId && b.SeatNumber == seatNumber);
            if (isBooked)
            {
                TempData["Error"] = "Місце вже заброньоване!";
                return RedirectToAction("SelectSeat", new { showtimeId });
            }

            var showtime = await _context.Showtimes
                .Include(s => s.Movie)
                .FirstOrDefaultAsync(s => s.Id == showtimeId);

            if (showtime == null)
            {
```

```
            TempData["Error"] = "Сеанс не знайдено!";
            return RedirectToAction("SelectSeat", new { showtimeId });
        }

        var booking = new Booking
        {
            ShowtimeId = showtimeId,
            SeatNumber = seatNumber,
            BookingTime = System.DateTime.Now,
            UserId = 0
        };
        _context.Bookings.Add(booking);

        string ticketCode = System.Guid.NewGuid().ToString().Substring(0, 8).ToUpper();

        var order = new Order
        {
            Phone = phone,
            CardNumber = cardNumber,
            MovieTitle = showtime.Movie.Title,
            Showtime = showtime.StartTime,
            SeatNumber = seatNumber,
            TicketCode = ticketCode,
            BookingTime = System.DateTime.Now
        };
        _context.Orders.Add(order);
```

```
                await _context.SaveChangesAsync();

                TempData["Success"] = $"Бронювання успішне! Ваш код квитка: {ticketCode}";
                return RedirectToAction("SelectSeat", new { showtimeId });
            }
        }
    }
```

MoviesController.cs:

```
using System;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using CinemaBookingWeb.Data;
using CinemaBookingWeb.Models;

namespace CinemaBookingWeb.Controllers
{
    public class MoviesController : Controller
    {
        private readonly CinemaDbContext _context;

        public MoviesController(CinemaDbContext context)
        {
            _context = context;
        }
```

```
        public async Task<IActionResult> Index(DateTime? date)
        {
            var selectedDate = date ?? DateTime.Today;

            var showtimes = await _context.Showtimes
                .Include(s => s.Movie)
                .Where(s => s.StartTime.Date == selectedDate.Date)
                .ToListAsync();

            var movies = showtimes.Select(s => s.Movie).Distinct().ToList();

            ViewBag.SelectedDate = selectedDate;
            ViewBag.Showtimes = showtimes;
            return View(movies);
        }
    }
}
```

ShowtimesController.cs:

```
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using CinemaBookingWeb.Data;
using CinemaBookingWeb.Models;

namespace CinemaBookingWeb.Controllers
```

```
{
    public class ShowtimesController : Controller
    {
        private readonly CinemaDbContext _context;

        public ShowtimesController(CinemaDbContext context)
        {
            _context = context;
        }
        public async Task<IActionResult> Index(int movieId)
        {
            var showtimes = await _context.Showtimes
                .Where(s => s.MovieId == movieId)
                .Include(s => s.Hall)
                .ToListAsync();

            ViewBag.Movie = await _context.Movies.FindAsync(movieId);
            return View(showtimes);
        }
    }
}
```

Додаток Д

Лістинг моделей таблиць бази даних

Booking.cs:

```
public class Booking { public int Id; public int UserId; public int ShowtimeId; public int SeatNumber; public DateTime BookingTime; public Showtime Showtime; }
```

Cinema.cs:

```
public class Cinema { public int Id; public string Name; public string Address; }
```

Hall.cs:

```
public class Hall { public int Id; public int CinemaId; public string Name; public int TotalSeats; public Cinema Cinema; }
```

Movie.cs:

```
public class Movie { public int Id; public string Title; public int DurationMinutes; public string Genre; public string? PosterUrl; }
```

Order.cs:

```
public class Order { public int Id; public string Phone; public string CardNumber; public string MovieTitle; public DateTime Showtime; public int SeatNumber; public string TicketCode; public DateTime BookingTime; }
```

Showtime.cs:

```
public class Showtime { public int Id; public int MovieId; public int HallId; public DateTime StartTime; public Movie Movie; public Hall Hall; }
```

Додаток Е

Лістинг взаємодії бази даних та десктопної програми

CinemaDbContext.cs:

```
using Microsoft.EntityFrameworkCore;
using CinemaBookingWeb.Models;

namespace CinemaBookingWeb.Data
{
    public class CinemaDbContext : DbContext
    {
        public CinemaDbContext(DbContextOptions<CinemaDbContext> options) : base(options) { }

        public DbSet<Movie> Movies { get; set; }
        public DbSet<Showtime> Showtimes { get; set; }
        public DbSet<Hall> Halls { get; set; }
        public DbSet<Cinema> Cinemas { get; set; }
        public DbSet<Booking> Bookings { get; set; }
        public DbSet<Order> Orders { get; set; }
    }
}
```

Додаток Ж

Лістинг підключення до бази даних

appsettings.json:

```
{
  "ConnectionStrings": {
    "DefaultConnection": "Server=...;Database=CinemaBookingSystem;Trusted_Connection=True;TrustServerCertificate=True;"
  },
  "Logging": { "LogLevel": { "Default": "Information", "Microsoft.AspNetCore": "Warning" } },
  "AllowedHosts": "*"}
```