Міністерство освіти і науки України
Університет митної справи та фінансів

Факультет інноваційних технологій
Кафедра комп'ютерних наук та інженерії програмного забезпечення

**Кваліфікаційна робота бакалавра**

на тему : «Розроблення сайту онлайн кінотеатру»

Виконав: студент групи ІПЗ20-2
Спеціальність 121 «Інженерія програмного забезпечення»
Пономарьов Микита Олексійович
(прізвище та ініціали)
Керівник к.ф.-м.н., доц. Мормуль М.Ф.
(науковий ступінь, вчене звання, прізвище та ініціали)
Рецензент Університет митної справи та фінансів
(місце роботи)
Доцент кафедри кібербезпеки та інформаційних технологій
(посада)
к.т.н., доцент Прокопович- Ткаченко Д.І.
(науковий ступінь, вчене звання, прізвище та ініціали)

Дніпро – 2024

# АНОТАЦІЯ

*Пономарьов М. О.* Розроблення сайту онлайн кінотеатру.

Кваліфікаційна робота на здобуття освітнього ступеня бакалавр за спеціальністю 121 «Інженерія програмного забезпечення». – Університет митної справи та фінансів, Дніпро, 2024.

Проект розробки онлайн платформи для купівлі квитків на кіно спрямований на створення зручного та ефективного сервісу для користувачів і кінотеатрів. Розглянуто процес аналізу потреб користувачів, вибір технологій та розробку функціоналу, з особливим акцентом на безпеку та захист даних.

Під час виконання цієї робити описана база даних. У системі передбачені різні ролі користувачів, що забезпечує контроль за безпекою та конфіденційністю даних.

Розроблені механізми реєстрації та авторизації користувачів керувати їх доступом до функціоналу та забезпечувати персоналізований досвід користувача. Адміністраторам та менеджерам кінотеатру надається дозволяють ідентифікувати користувачів, можливість додавати, переглядати та редагувати інформацію про фільми та кіносеанси, що дозволяє оновлювати та керувати розкладом показів та іншою інформацією про фільми. Адміністратори та менеджери мають розширені права для керування контентом та налаштування системи, тоді як звичайні користувачі мають обмежений доступ до функціоналу.

Отримані результати роботи мають важливе значення для користувачів і кінотеатрів. Для користувачів створена платформа для купівлі квитків на кіно стане зручним способом бронювання місць, що покращить їхній візит до кінотеатру. Для кінотеатрів це означає можливість оптимізувати продажі, залучати більше клієнтів та підвищувати загальний рівень обслуговування.

Ключові слова: онлайн платформа, база даних, інтерфейси, мова програмування PHP, XAMPP, MySQL.

# ABSTRACT

*Ponomarov M.A.* Development of Cinema Website.

Qualification of work to obtain a bachelor's degree in specialty 121 «Software Engineering» – University of Customs and Finance, Dnipro, 2024.

The project to develop an online ticket booking platform for cinemas aims to create a convenient and efficient service for users and cinemas. The process of analyzing user needs, selecting technologies, and developing functionality was considered, with a special focus on security and data protection.

During the execution of this work, a database was described. The system provides for various user roles, ensuring control over data security and confidentiality.

Developed mechanisms for user registration and authorization allow identifying users, managing their access to functionality, and providing a personalized user experience. Administrators and cinema managers are provided with the ability to add, view, and edit information about movies and screenings, allowing them to update and manage the screening schedule and other movie information. Administrators and managers have extended rights to manage content and system settings, while ordinary users have limited access to functionality.

The results of the work are of great importance for users and cinemas. For users, the created platform for purchasing cinema tickets will become a convenient way to reserve seats, improving their visit to the cinema. For cinemas, this means the ability to optimize sales, attract more customers, and improve overall service levels.

Keywords: online platform, database, interfaces, PHP programming language, XAMPP, MySQL.

ЗМІСТ

# ВСТУП

У сучасному цифровому світі інтернет-технології відіграють важливу роль у взаємодії людей з різноманітними сервісами та послугами. Однією з галузей, яка активно розвивається у цьому контексті, є сфера розваг та культури, зокрема кінематограф. Онлайн кінотеатри стають все більш популярними серед глядачів, що шукають зручний та доступний спосіб перегляду улюблених фільмів.

Ця робота присвячена створенню та розробці онлайн кінотеатру, який надає користувачам зручну платформу бронювання квитків. Мета проекту полягає у створенні та впровадженні інноваційного сервісу, який забезпечить високий рівень сервісу та задоволення від кінематографічного досвіду для всіх користувачів.

У рамках цієї роботи буде проведений аналіз засобів розроблення програмного забезпечення для роботи онлайн кінотеатру, визначені вимоги до системи та розглянуті основні функціональні можливості. Також буде описана база даних, ролі користувачів, механізми реєстрації та авторизації, а також інші важливі аспекти системи. Результатом цієї роботи буде розроблення функціональної та ефективної онлайн платформи.

Робота з розробки онлайн кінотеатру полягає в наступному.

У світі, де швидкість та зручність є важливими аспектами, онлайн платформи для купівлі білетів на кіно надають користувачам можливість придбати квитки з будь-якого місця і в будь-який час за допомогою Інтернету.

Онлайн бронювання квитків на кіно дозволяє уникнути черг та зайвого часу на придбання квитків на місці. Користувачі можуть легко вибрати зручний сеанс та місце з власного пристрою.

Онлайн платформи для купівлі квитків можуть надати додаткові функції, такі як вибір місця в залі, перегляд трейлерів фільмів, отримання рекомендацій щодо фільмів на основі історії перегляду, а також можливість попереднього придбання закусок та напоїв.

Для кінотеатрів робота з онлайн платформами для продажу квитків дозволяє оптимізувати процес продажу, знижує витрати на обслуговування та підвищує швидкість обробки замовлень.

Онлайн платформи забезпечують можливість збирати дані про користувачів, їхні вподобання та покупкову історію, що дозволяє кінотеатрам налаштовувати таргетовану рекламу та акції, привертаючи більше клієнтів.

Загалом, розробка сайту для купівлі білетів на кіно є актуальною, оскільки вона відповідає потребам сучасного споживача, забезпечуючи зручність, доступність та розширені можливості для всіх сторін: користувачів та кінотеатрів.

Мета проекту полягає у розробці та впровадженні високотехнологічної онлайн платформи для кінотеатру, що надає розширені можливості для користувачів у перегляді фільмів та здійсненні бронювання квитків.

Для досягнення мети розробки вебсайту онлайн кінотеатру можуть бути сформульовані наступні задачі:

1) створення інтуїтивного та привабливого інтерфейсу , який дозволить користувачам легко переглядати розклад сеансів, шукати фільми та здійснювати бронювання квитків;

2) розробка функціоналу для бронювання квитків на сеанси, включаючи вибір місць, вибір фільму, оплату та підтвердження бронювання.;

3) створення панелі адміністратора для керування вмістом сайту, включаючи додавання та редагування фільмів, оновлення розкладу сеансів та керування акціями та пропозиціями;

4) реалізація заходів безпеки для захисту особистої інформації користувачів;

5) проведення регулярних тестів для виявлення та виправлення помилок, а також вдосконалення функціональності та ефективності роботи сайту.

Об'єктом дослідження цієї роботи є онлайн платформа «Кінотеатр».

Предметом дослідження цієї роботи є процес розробки та реалізації онлайн платформи для купівлі білетів на кіно, включаючи аналіз потреб

користувачів, розробку функціональності, реалізацію безпеки та ефективність роботи системи.

Практичне значення одержаних результатів полягає в створенні функціональної та ефективної онлайн платформи для купівлі білетів на кіно. Отримані результати роботи мають важливе значення для користувачів і кінотеатрів. Для користувачів створена платформа для купівлі квитків на кіно стане зручним способом бронювання місць, що покращить їхній візит до кінотеатру. Для кінотеатрів це означає можливість оптимізувати продажі, залучати більше клієнтів та підвищувати загальний рівень обслуговування. Також, збір та аналіз даних допоможе кінотеатрам краще розуміти своїх клієнтів і підлаштовувати свою роботу під їхні потреби.

Результатами роботи є онлайн платформа «Кінотеатр».

Кваліфікаційна робота складається зі вступу, 3-х розділів, висновків, використаних джерел з 15 найменувань, 2-х додатків.

Обсяг роботи 92 сторінки, містить 8 таблиць та 32 рисунки.

# РОЗДІЛ 1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ. ПОСТАНОВКА ЗАВДАНЬ ДОСЛІДЖЕННЯ

## 1.1 Опис процесів роботи онлайн кінотеатру

Предметна область описує конкретну сферу або частину реального світу, яку ми хочемо відобразити в базі даних. Це можуть бути будь-які дані, які мають значення або інтерес для нас, незалежно від їхньої важливості. Вона може включати як ключові концепції та дані, так і менш важливі або незначущі інформаційні елементи.

Онлайн кінотеатр – це інформаційна система, яка працює в мережі Інтернет і дозволяє зберігати та обробляти різноманітні колекції електронних документів, таких як текст, зображення, аудіо та відео. Ця система дає можливість користувачам отримати доступ до фільмів та інших контентів у зручному для них форматі через глобальні мережі передачі даних.

Мета створення онлайн кінотеатру полягає в наступному:

– забезпечення користувачів можливістю обрання та покупки білетів на кіносеанс;

– забезпечення доступу до інформації, що існує лише в електронній формі;

– надання адміністраторам системи більш якісних можливостей роботи з електронними документами великих обсягів;

– інформаційне забезпечення користувачів повнотекстовими базами даних у режимі теледоступу.

Основні задачі онлайн кінотеатру – інтеграція інформаційних ресурсів і ефективна навігація в них. Інтеграція інформаційних ресурсів – це їхнє об'єднання з метою використання різної інформації зі збереженням її властивостей, особливостей представлення і можливостей її обробляти. Об'єднання ресурсів може відбуватися як фізично, так і віртуально. Але при цьому таке об'єднання повинно забезпечувати користувачу сприйняття необхідної інформації як єдиного інформаційного простору: онлайн кінотеатр

повинен забезпечити роботу з базами даних і високу ефективність інформаційних пошуків.

Ефективна навігація в онлайн кінотеатрі – це можливість користувача знаходити інформацію, яка його цікавить, в усьому доступному інформаційному просторі з найбільшою повнотою і точністю при найменших витратах зусиль. Зараз існує тисячі і навіть десятки тисяч онлайн кінотеатрів.

Створення сайту для бронювання квитків в кінотеатрі є дуже актуальним і має такі переваги:

– онлайн бронювання квитків дозволяє користувачам легко та швидко забронювати місця на сеанси з будь-якого пристрою з Інтернетом, без необхідності виходу з дому або стояння в черзі на касі;

– онлайн платформа для бронювання квитків доступна користувачам цілодобово, що дозволяє їм здійснювати покупки у будь-який зручний для них час;

– користувачі можуть переглядати розклад сеансів, обирати зручний для них час та місце у залі, а також ознайомлюватися з відгуками про фільми перед придбанням квитків;

– для кінотеатрів це забезпечує можливість ефективного управління продажами квитків, враховуючи популярність фільмів та вмісту;

– онлайн платформа дозволяє кінотеатрам впроваджувати різноманітні маркетингові стратегії, збирати дані про попит на фільми та аналізувати поведінку своїх клієнтів.

Загалом, сайт для бронювання квитків в кінотеатрі надає користувачам зручність та доступність, а кінотеатрам – ефективність у продажах та управлінні.

## 1.2 Огляд існуючого програмного забезпечення для роботи онлайн кінотеатру

Приклади деяких відомих кінотеатрів, що мають вебсайти:

1) «AMC Theatres» (рис. 1.1):

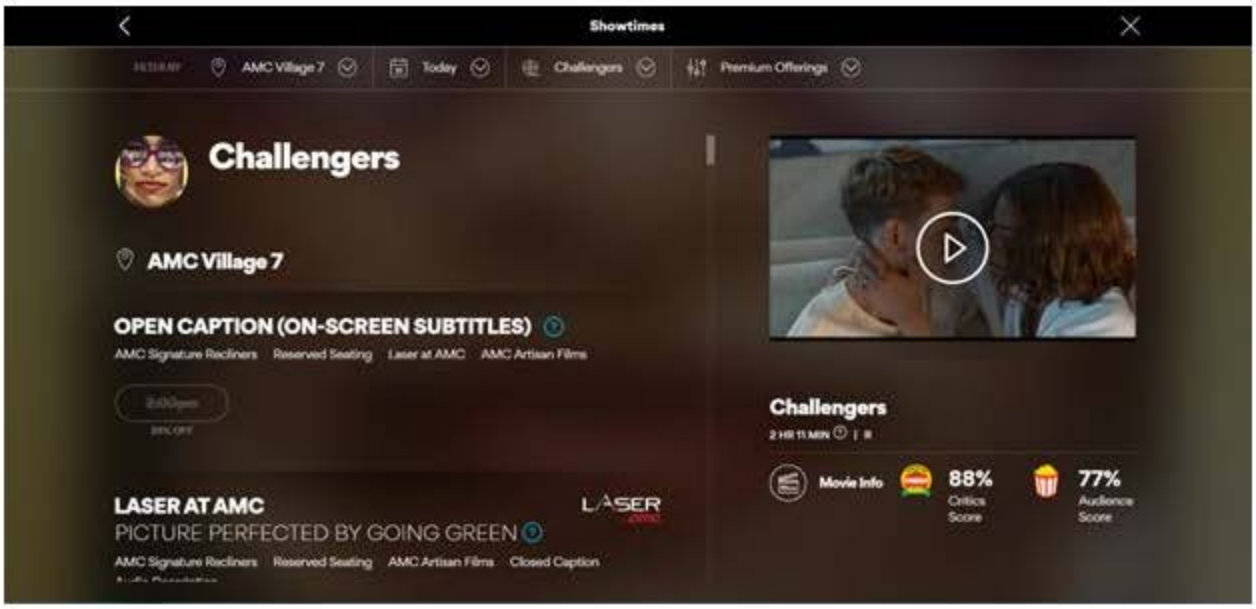


Рисунок 1.1 – Інтерфейс AMC Theatres

– розклад сеансів (користувачі можуть переглянути розклад всіх сеансів, які доступні в кінотеатрах мережі AMC, включаючи час початку фільму, назву фільму, тривалість сеансу та доступні формати);

– пошук фільмів (користувачі можуть шукати конкретні фільми за назвою, жанром або акторами, трейлери, рейтинги);

– бронювання квитків (користувачі можуть зручно бронювати квитки на сеанси прямо через вебсайт, вибрати місця у залі, формат показу, здійснити оплату онлайн);

– акції та знижки (сайт надає інформацію про поточні акції, знижки та спеціальні пропозиції);

– програма лояльності AMC Stubs (користувачі можуть отримувати знижки на квитки, безкоштовні напої, переваги для учасників);

– інформація про кінотеатри (користувачі можуть знайти інформацію про розташування, контактні дані кінотеатрів мережі AMC, а також дізнатися про особливості кожного з них);

– статті та новини (сайт надає користувачам можливість дізнатися про останні новини зі світу кіно, інтерв'ю з акторами, огляди фільмів та іншу цікаву інформацію);

– мобільний доступ (крім вебсайту, AMC також має мобільний додаток, який надає користувачам аналогічні можливості зручного бронювання квитків, доступу до іншої інформації).

2) «Regal Cinemas» має наступні можливості інтерфейсу: розклад сеансів, пошук фільмів, бронювання квитків, програма лояльності, акції та знижки, статті та новини, мобільний доступ (рис. 1.2).

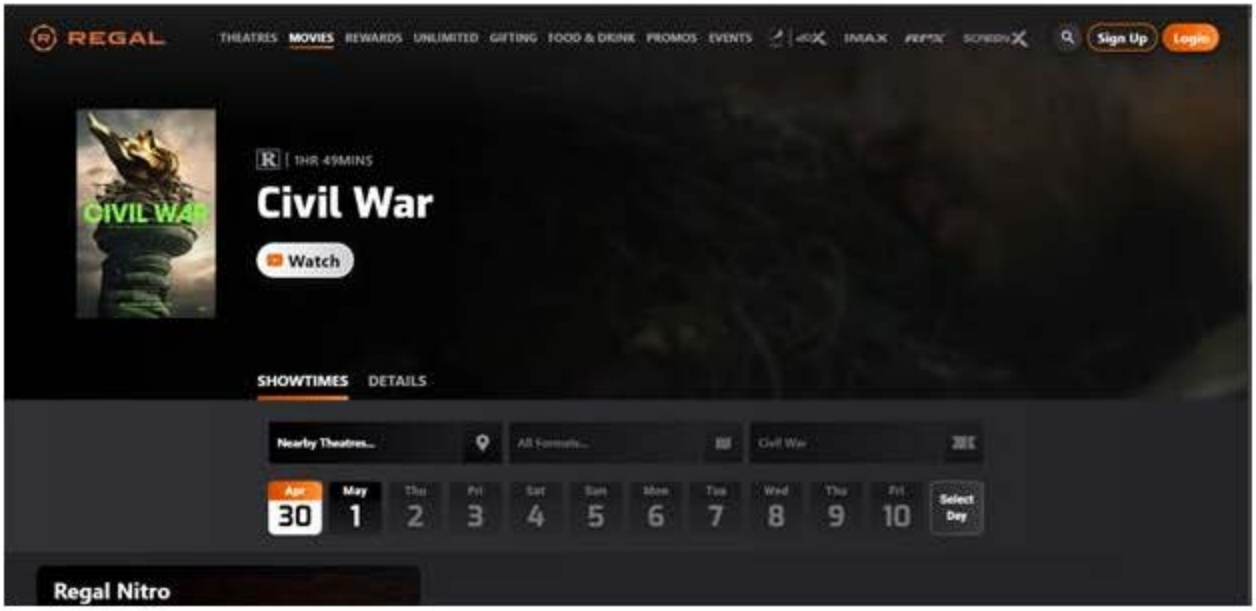


Рисунок 1.2 – Інтерфейс Regal Cinemas

3) «Cineworld» має наступні можливості інтерфейсу (рис. 1.3): розклад сеансів, пошук фільмів, бронювання квитків, програма лояльності, акції та знижки, статті та новини, мобільний доступ. А також:

– інформація про кінотеатри (користувачі можуть знайти інформацію про всі кінотеатри мережі Cineworld, включаючи адреси, контактні дані, графіки роботи, бари, кіоски зі свіжими продуктами тощо);

– функція «Мій обліковий запис» (зареєстровані користувачі можуть створити особистий обліковий запис на сайті Cineworld, де вони зможуть

зберігати свої улюблені фільми, переглядати історію бронювань, керувати підписками, отримувати індивідуалізовані пропозиції);

– комунікація з клієнтами (сайт може мати функцію зворотного зв'язку, через яку користувачі можуть надсилати запити, зауваження або скарги, а також отримувати відповіді, підтримку від представників компанії);

– онлайн-сервіси і додаткові послуги (користувачі можуть мати можливість не лише бронювати квитки на фільми, але і замовляти додаткові послуги, такі як розваги, подарункові сертифікати, абонементи та інше);

– рейтинги та відгуки користувачів (сайт може мати розділ, де користувачі можуть залишати свої відгуки, оцінки про фільми, кінотеатри, загальний кінематографічний досвід);

– міжнародні можливості (якщо мережа Cineworld має кінотеатри в різних країнах, сайт може мати функції перекладу, адаптації для міжнародних користувачів).

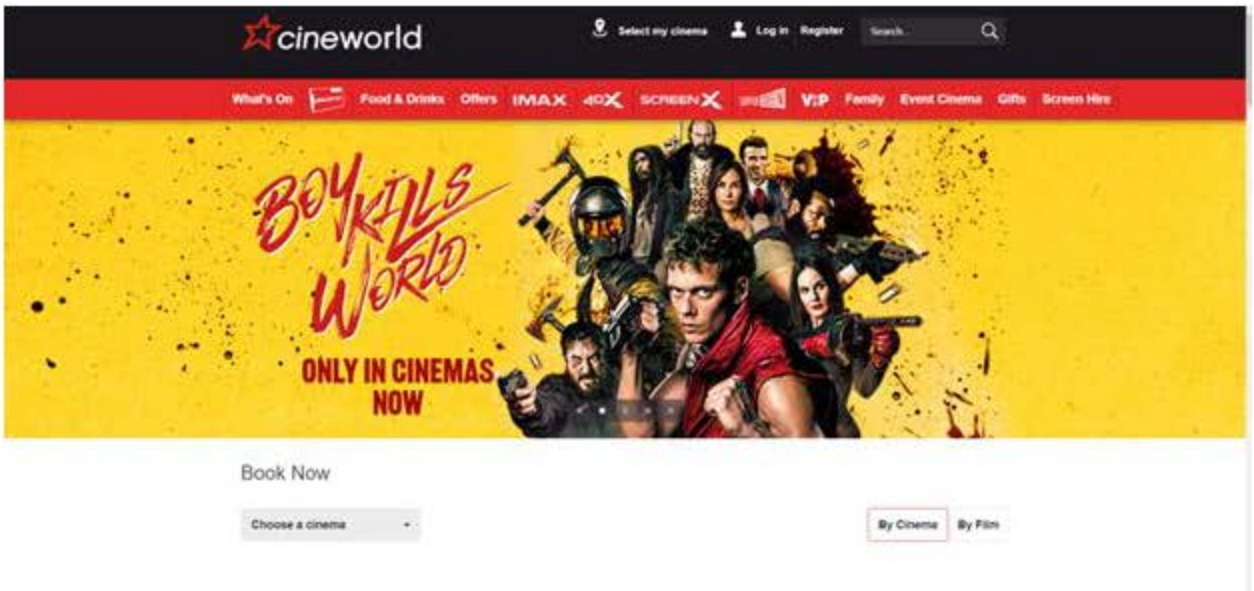


Рисунок 1.3 – Інтерфейс Cineworld

4) «ODEON Cinemas» має подібні можливості, як і у попередніх сайтів, але відрізняється від інших своїми програмами лояльності ODEON Première

Club, акціями та пропозиціями, ексклюзивними угодами та партнерством (рис. 1.4).


Рисунок 1.4 – Інтерфейс ODEON Cinemas

5) «Vue Cinemas» (рис. 1.5) має стандартні можливості, але у наявності наступні:

– вибір місць у залі (користувачі можуть вибирати конкретні місця для перегляду фільму в кінозалі під час бронювання квитків, що може бути особливо корисно для вибору місць з найкращим кутом видимості або для групових бронювань);

– електронні квитки (система иможе підтримувати електронні квитки, які можна зберегти на смартфоні або в мобільному додатку, що дозволяє уникнути необхідності друкувати квитки і забезпечує більшу зручність для користувачів);

– розсилка новин та акцій (користувачі можуть підписатися на розсилку новин та спеціальних пропозицій від Vue Cinemas, що дозволяє отримувати оновлення про нові фільми, акції та знижки безпосередньо на електронну пошту або мобільний пристрій);

– власний обліковий запис (користувачі можуть створити власний обліковий запис на сайті Vue Cinemas, де вони можуть зберігати свої особисті дані, історію бронювань, вибрані фільми та налаштування сповіщень);

– підтримка мов (сайт може підтримувати кілька мов, що дозволяє користувачам переглядати та бронювати квитки у своїй власній мові, що особливо важливо для міжнародних користувачів або для тих, хто не володіє англійською мовою);

– попереднє замовлення страв (деякі кінотеатри мережі Vue Cinemas можуть пропонувати можливість попереднього замовлення страв та напоїв на час відвідування кінозалу, що дозволяє користувачам швидко та зручно отримати свої замовлення під час перегляду фільму).

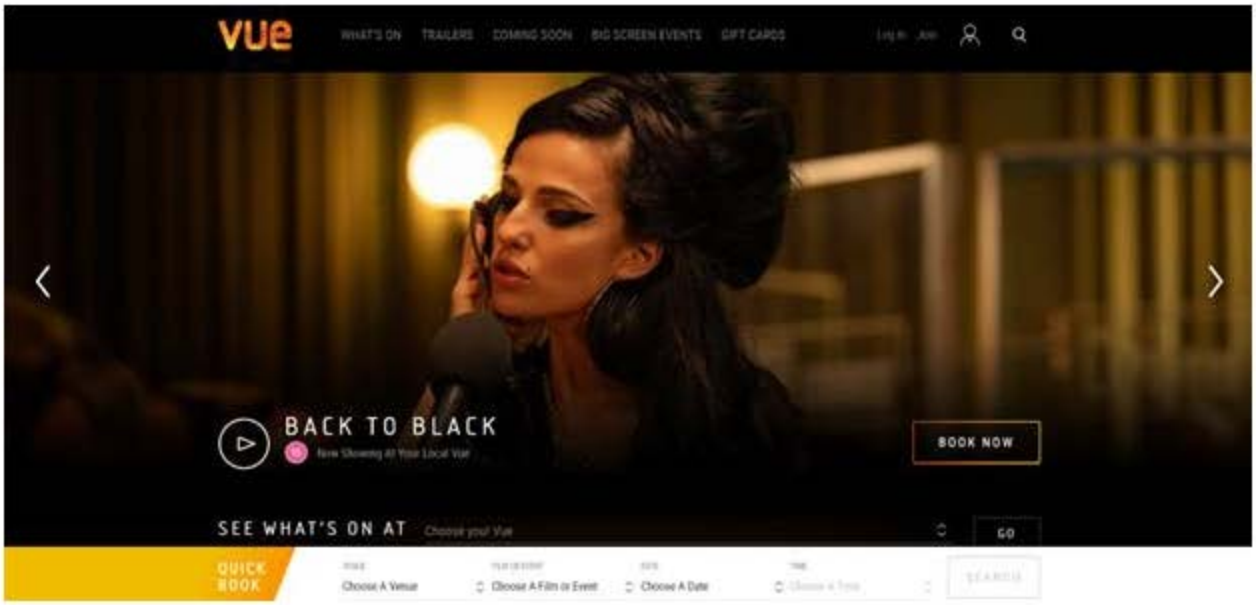


Рисунок 1.5 – Інтерфейс Vue Cinemas

б) «CGV Cinemas» Південна Корея (рис. 1.6) разом зі стандартними можливостями має також свої унікальні:

– технологічні інновації в кінотеатрах (впровадження передових технологій, що може включати великі екрани, звукові системи

останнього покоління, інтерактивні ефекти, інші технологічні інновації, які роблять кінематографічний досвід більш захопливим, емоційним);

– спеціальні заходи та події (прем'єри фільмів з участю знаменитостей, вечірки, конкурси, інші розважальні заходи, які роблять відвідування кінотеатру особливим і незабутнім);

– тематичні кінозали (кінозали, які прикрашені та оформлені в певному стилі або за мотивами популярних фільмів, що може створювати унікальну атмосферу, додавати до загального кінематографічного досвіду);

– ексклюзивні прем'єри і фільми (ексклюзивний доступ до прем'єрних показів фільмів, що може привертати кіноглядачів, які хочуть першими подивитися найочікуваніші стрічки);

– культурні та освітні ініціативи (покази фестивальних фільмів, лекції та дискусії про кіно або спеціальні програми для дітей та молоді).

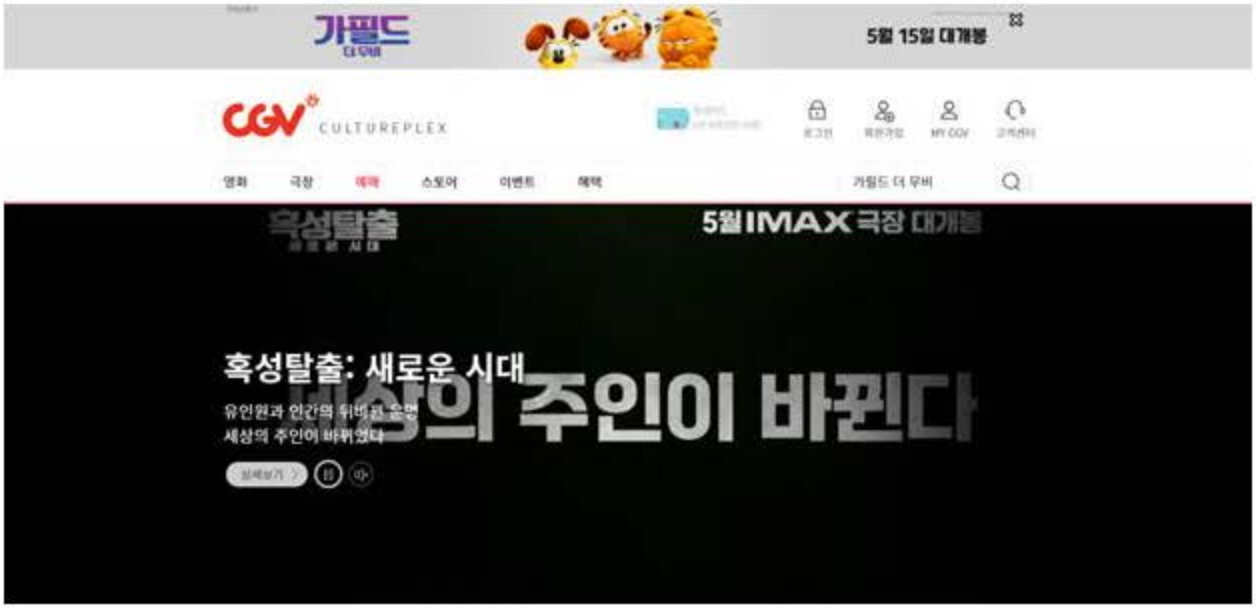


Рисунок 1.6 – Інтерфейс CGV Cinemas

7) «Planeta Kino» Україна (рис. 1.7) звичайно має стандарті можливості, проте є особливості, що нечасто трапляються:

– віртуальний тур по кінотеатру (користувачі можуть побачити внутрішнє оформлення, зони розваг, кафе, інші зручності до візиту);

– онлайн-чат або підтримка по телефону (для відповіді на запитання користувачів, вирішення проблем у реальному часі);

– попереднє замовлення або доставка їжі (кінотеатр може співпрацювати з ресторанами, кафе, щоб дозволити користувачам замовляти їжу та напої перед/після сеансу або навіть доставку їжі безпосередньо в кінозал);

– організація дитячих вечірок, подій, святкування днів народження в кінотеатрі;

– реєстрація за допомогою соціальних медіа (сайт може дозволяти користувачам реєструватися або входити на сайт за допомогою своїх облікових записів у соціальних мережах, що робить процес швидшим та зручнішим).

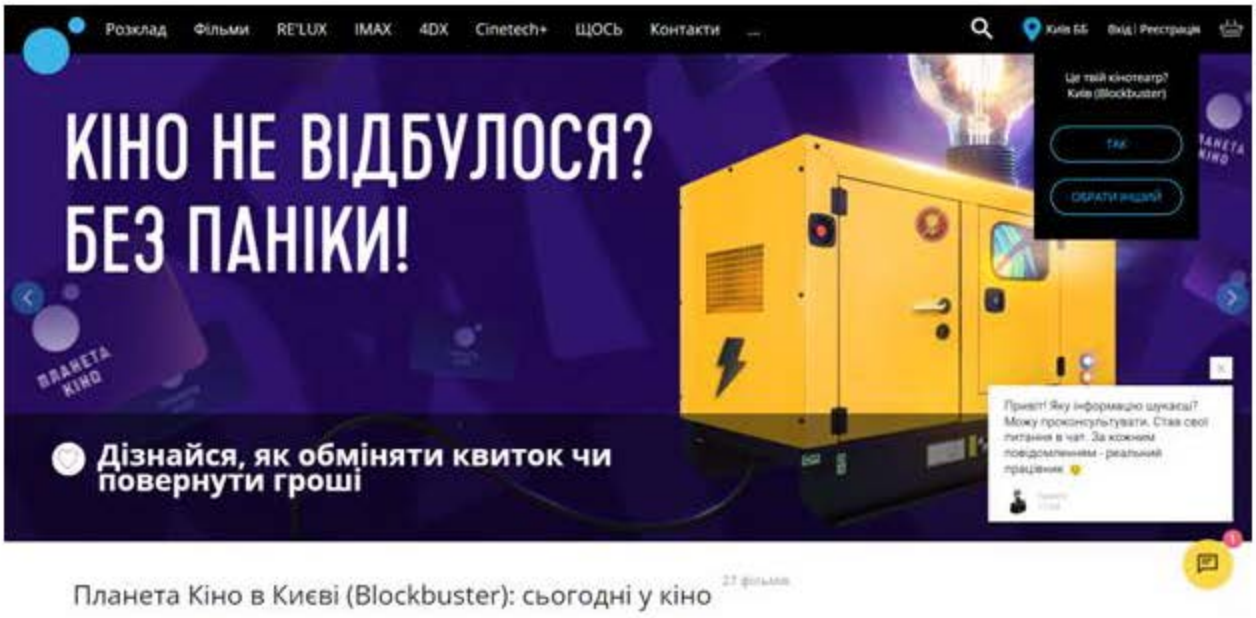


Рисунок 1.7 – Інтерфейс Planeta Kino

Як видно з описів, усі сайти мають подібний функціонал, але, звичайно, є і виняткові можливості, що можуть залежати від різних факторів. Наприклад, величини кінотеатру, країни розташування, способів комунікації, додаткових послуг, тощо.

## 1.3 Дослідження сучасних інструментів для розроблення вебсайту «Кінотеатр»

Для розроблення вебсайтів кінотеатрів можна використовувати різноманітні технології та інструменти.

Для створення клієнтської частини вебсайту (Frontend технології), де користувачі взаємодіють з інтерфейсом, можна використовувати HTML, CSS, PHP та JavaScript. Крім того, для розробки більш складних інтерфейсів та взаємодії з користувачем можна використовувати фреймворки, такі як React.js, Angular або Vue.js.

React.js – це потужна бібліотека для створення користувацьких інтерфейсів вебсайтів. React дозволяє розбити інтерфейс на невеликі, самодостатні компоненти. Це спрощує розробку, тестування та підтримку коду. React використовує віртуальний DOM для оптимізації швидкодії та ефективності оновлення сторінок. Він автоматично визначає мінімальні зміни, які необхідно внести до реального DOM для оновлення сторінки.

JSX – це розширення JavaScript, що дозволяє писати HTML-подібний код прямо в JavaScript файлі. Це полегшує роботу з UI, оскільки розмітка та логіка можуть знаходитися в одному файлі.

React ідеально підходить для створення односторінкових застосунків, які завантажуються один раз, а потім динамічно оновлюються без перезавантаження сторінки. Він надає можливість автоматичного оновлення інтерфейсу при зміні даних. Це робить реактивні інтерфейси більш динамічними та ефективними.

React має велику та активну спільноту розробників, що регулярно внесли нові функції, бібліотеки та інструменти для полегшення роботи з ним.

Angular – це потужний фреймворк для створення вебдодатків. Він пропонує модульну архітектуру, яка дозволяє розбити додаток на невеликі та самодостатні модулі. Це полегшує розробку, тестування та підтримку великих додатків.

Angular пропонує двостороннє зв'язування даних між моделлю та представленням, що дозволяє автоматично оновлювати інтерфейс користувача при зміні даних. Він має вбудовані функції для роботи з HTTP-запитами, маршрутизацією, формами, аутентифікацією, валідацією та багато іншого, що дозволяє розробникам швидко реалізувати різноманітні функціональності.

Angular поставляється з потужним набором інструментів для тестування, що дозволяє легко створювати та запускати тести для різних частин додатку.

Angular має вбудовані інструменти для оптимізації продуктивності, такі як Ahead-of-Time (AOT) компіляція, lazy loading модулів та робота з кешуванням. Angular CLI (Command Line Interface) надає широкий набір інструментів для автоматизації рутинних завдань, таких як створення компонентів, модулів, сервісів, робота з тестами тощо.

Angular розробляється та підтримується Google, що забезпечує надійність та стабільність фреймворку.

Vue.js – це прогресивний JavaScript фреймворк для створення користувацьких інтерфейсів вебдодатків. Він має простий та інтуїтивно зрозумілий синтаксис, що робить його легким для вивчення та використання, навіть для початківців. Це дозволяє швидко розпочати роботу над проектами.

Vue.js пропонує модульну структуру, яка дозволяє розбити додаток на невеликі та перевикористовувані компоненти, що полегшує розробку, тестування та підтримку коду. Він використовує реактивну архітектуру, яка автоматично оновлює інтерфейс користувача при зміні даних. Крім того, його компонентна структура сприяє реорганізації та повторному використанню коду.

Також Vue.js має вбудовані інструменти для розробки, такі як Vue DevTools, які дозволяють відслідковувати стан додатку, відлагоджувати компоненти та виявляти помилки. Розмір Vue.js досить малий, що робить його ідеальним вибором для проектів з обмеженими ресурсами. Він пропонує

простий та ефективний спосіб реалізації реактивного зв'язування даних між моделлю та представленням.

Для реалізації серверної частини вебсайту (Backend технології) та обробки запитів користувачів, можна використовувати мови програмування, такі як Python (з фреймворком Django або Flask), JavaScript (з Node.js або Express.js), PHP, Ruby або Java.

Для зберігання даних про фільми, користувачів, квитки та іншу інформацію можна використовувати різні типи баз даних, такі як MySQL, PostgreSQL, MongoDB або SQLite.

Для обробки бронювання квитків та оплати можна розробити API, яке буде взаємодіяти з платіжними системами та забезпечувати доступ до функцій бронювання квитків.

Для показу трейлерів фільмів або отримання оглядів та рейтингів фільмів можна інтегрувати сайт з зовнішніми сервісами, такими як YouTube або IMDB.

Для відстеження активності користувачів на сайті, аналізу їхнього поведінки та забезпечення кращого досвіду взаємодії можна використовувати інструменти аналітики, такі як Google Analytics або Mixpanel.

Важливо забезпечити безпеку даних користувачів та операцій з платежами шляхом використання шифрування, аутентифікації користувачів та інших заходів безпеки.

Реальний вибір технологій залежатиме від потреб бізнесу, технічних вимог та власних уподобань розробників.

## 1.4 Висновки до першого розділу. Постановка задач дослідження

У першій частині проекту було проаналізовано предметну область, описано мету, основні задачі та актуальність створення вебсайту онлайн кінотеатру.

Було визначено мету створення онлайн кінотеатру, яка може включати покращення доступності до кіно для користувачів, зручний процес бронювання квитків та збільшення прибутку кінотеатру. Також були сформульовані основні задачі, такі як розробка зручного інтерфейсу, підтримка різних пристроїв та безпека даних користувачів.

Було проведено огляд існуючого програмного забезпечення для роботи онлайн кінотеатру, що дозволило виявити переваги та недоліки різних рішень на ринку. Це допоможе визначити найкращі практики для подальшої розробки.

В рамках проекту було проведено дослідження сучасних інструментів для розробки вебсайту кінотеатру, таких як React.js, Angular, Vue.js тощо. Це дозволить обрати найбільш підходящий інструмент для реалізації задач та вимог проекту.

Отже, в першій частині проекту було зроблено значний прогрес у напрямку розробки онлайн кінотеатру, визначено стратегічні цілі та завдання, проведено аналіз існуючих рішень та вибрано інструментарій для подальшої розробки.

Зважаючи на аналіз проведений у першій частині проекту, можна зробити наступні детальні висновки.

Глибоке розуміння процесів роботи онлайн кінотеатру дозволяє зрозуміти всі етапи, які включаються у взаємодію з користувачами, від пошуку фільмів до бронювання квитків та оплати.

Аналіз існуючих рішень на ринку дозволяє виявити переваги та недоліки інших проектів, а також зрозуміти, які функціональні можливості є найбільш важливими для користувачів.

Дослідження сучасних інструментів для розроблення вебсайту дозволяє обрати той, який найкраще відповідає потребам проекту з урахуванням його масштабу, складності та доступних ресурсів.

Результати аналізу можуть показати, які проблеми потребують особливої уваги та які завдання потребують пріоритетного вирішення.

Мета проекту полягає у розробці та впровадженні високотехнологічної онлайн платформи для кінотеатру, що надає розширені можливості для користувачів у перегляді фільмів та здійсненні бронювання квитків.

Для досягнення мети розробки вебсайту онлайн кінотеатру можуть бути сформульовані наступні задачі:

1) створення інтуїтивного та привабливого інтерфейсу , який дозволить користувачам легко переглядати розклад сеансів, шукати фільми та здійснювати бронювання квитків;

2) розробка функціоналу для бронювання квитків на сеанси, включаючи вибір місць, вибір фільму, оплату та підтвердження бронювання.;

3) створення панелі адміністратора для керування вмістом сайту, включаючи додавання та редагування фільмів, оновлення розкладу сеансів та керування акціями та пропозиціями;

4) реалізація заходів безпеки для захисту особистої інформації користувачів;

5) проведення регулярних тестів для виявлення та виправлення помилок, а також вдосконалення функціональності та ефективності роботи сайту.

# РОЗДІЛ 2 АНАЛІЗ ЗАСОБІВ РОЗРОБЛЕННЯ ОНЛАЙН КІНОТЕАТРУ

## 2.1 Вибір програмних засобів для розроблення онлайн кінотеатру

Під час розробки програмного забезпечення можуть бути використані наступні програмні засоби:

1) мова програмування PHP, яка у даній роботі дасть можливість створення динамічного контенту сайту, зв'язатися із базою даних і маніпулювати даними отриманими із неї;

2) XAMPP – це безкоштовний, універсальний вебсервер, який дозволяє створювати локальний сервер на комп'ютері для розгортання вебсайтів та розвитку вебдодатків;

3) MySQL – вільна система керування реляційними базами даних.

## 2.2 Мова програмування PHP

PHP (англ. PHP: Hypertext Preprocessor – PHP: гіпертекстовий препроцесор) раніше відома як Personal Home Page Tools, це скриптова мова програмування, спеціально призначена для створення HTML-сторінок на стороні вебсервера. PHP є однією з найпопулярніших мов у сфері веброзробок, разом з Java, .NET, Perl, Python та Ruby. Більшість хостинг-провайдерів підтримують PHP. Це проект відкритого програмного забезпечення, що означає, що він є вільним для використання та розповсюдження.

PHP виконується на вебсервері і перетворюється у HTML-код, який потім передається на клієнтський браузер. У відміну від JavaScript, PHP-код залишається невидимим для користувача, оскільки браузер отримує готовий HTML-код. Це забезпечує підвищений рівень безпеки, але може обмежувати інтерактивність сторінок. Не заборонено використовувати PHP для генерування JavaScript-коду, який вже виконується на стороні клієнта.

PHP - це мова програмування, яку можна впроваджувати прямо в HTML-

код сторінок, що забезпечує їх правильну обробку PHP-інтерпретатором. Таким чином, PHP дозволяє змішувати динамічний контент з HTML-структурою сторінок і ефективно взаємодіяти з даними та базами даних. Це означає, що ви можете створювати вебсторінки, які взаємодіють з користувачем, генеруючи вміст на основі різних умов і даних.

Наприклад, є можливість вбудувати умовні конструкції, цикли та інші PHP-конструкції прямо в HTML-код. У прикладі, зображеному на рисунку 2.1, в залежності від значення змінної $name, створюється відповідний заголовок <h1>. Якщо значення $name дорівнює «Іван», то виводиться повідомлення «Привіт, Іван!», в іншому випадку виводиться «Вітаю, гостю!».

```
<!DOCTYPE html>
<html>
<head>
    <title>Привіт, світ!</title>
</head>
<body>

<?php
    $name = "Іван";
    if ($name == "Іван") {
        echo "<h1>Привіт, $name!</h1>";
    } else {
        echo "<h1>Вітаю, гостю!</h1>";
    }
?>

</body>
</html>
```

Рисунок 2.1 – Приклад вбудуваної умовні конструкції прямо в HTML-код

Такий підхід дозволяє створювати вебсторінки, які адаптуються до конкретних умов та даних, що є дуже важливим для створення динамічного та інтерактивного вебконтенту. PHP також дозволяє легко взаємодіяти з базами даних та іншими серверними ресурсами, що робить його потужним інструментом для розробки вебдодатків.

Велика різноманітність функцій PHP дає можливість уникати написання багаторядкових функцій, призначених для користувача. А також приємні особливості, такі як: наявність інтерфейсів до багатьох баз даних, традиційність, наявність сирцевого коду та безкоштовність, ефективність, інтерпретованість.

Іншою особливістю мови програмування PHP є її здатність до взаємодії з базами даних.

PHP надає вбудовану підтримку для різних систем управління базами даних (СУБД), таких як MySQL, PostgreSQL, SQLite та інші. Це дозволяє розробникам легко підключатися до баз даних та виконувати операції з даними.

Основні функції PHP для роботи з базами даних включають в себе [1–6]:

1) PHP надає функції для встановлення з'єднання з базою даних. Наприклад, функція mysqli_connect() встановлює з'єднання з сервером бази даних MySQL;

2) після підключення до бази даних можна виконувати різні запити SQL, такі як SELECT, INSERT, UPDATE, DELETE тощо (функції PHP, такі як mysqli_query(), дозволяють виконувати такі запити та отримувати результати);

3) після виконання запитів PHP надає можливості для обробки результатів, таких як витягування даних з запитів SELECT та обробка помилок, що виникають під час виконання запитів;

4) PHP також підтримує підготовлені запити, які дозволяють попередньо компілювати SQL-запити та використовувати їх з параметрами, що забезпечує більшу безпеку та ефективність виконання запитів;

5) після завершення роботи з базою даних, з'єднання повинно бути закрите для звільнення ресурсів сервера (функція mysqli_close() дозволяє закрити з'єднання з базою даних).

Загалом, взаємодія з базами даних є невід'ємною частиною веброзробки, і PHP забезпечує потужні та зручні інструменти для роботи з даними, що дозволяє розробникам створювати динамічні та інтерактивні вебдодатки.

Нижче наведено приклади використання PHP для роботи з базою даних MySQL.

Приклад підключення до бази даних зображено на рисунку 2.2.

```php
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$database = "dbname";

// Підключення до бази даних
$conn = new mysqli($servername, $username, $password, $database);

// Перевірка з'єднання
if ($conn->connect_error) {
    die("Помилка підключення до бази даних: " . $conn->connect_error);
} else {
    echo "Підключення успішне!";
}
?>
```

Рисунок 2.2 – Приклад підключення до бази даних MySQL

Приклад вставки даних до бази даних зображено на рисунку 2.3.

Приклад оновлення даних в базі даних зображено на рисунку 2.4.

PHP надає потужні інструменти для роботи з формами та обробки даних, що дозволяє розробникам створювати інтерактивні та функціональні вебдодатки.

Форми і обробка даних є важливою складовою веброзробки, і PHP надає потужні засоби для роботи з формами на вебсайтах.

Приклад отримання даних з форми POST-запитом і їх обробка зображено на рисунку 2.5.

```php
<?php
$name = "John";
$email = "john@example.com";

$sql = "INSERT INTO users (name, email) VALUES ('$name', '$email')";

if ($conn->query($sql) === TRUE) {
    echo "Дані успішно додано";
} else {
    echo "Помилка: " . $sql . "<br>" . $conn->error;
}
?>
```

Рисунок 2.3 – Приклад вставки даних до бази даних MySQL

```php
<?php
$sql = "UPDATE users SET email='john_new@example.com' WHERE name='John'";

if ($conn->query($sql) === TRUE) {
    echo "Дані успішно оновлено";
} else {
    echo "Помилка оновлення даних: " . $conn->error;
}
?>
```

Рисунок 2.4 – Приклад оновлення даних в базі даних MySQL

```php
<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // Отримання даних з форми
    $name = $_POST['name'];
    $email = $_POST['email'];

    // Обробка даних (наприклад, збереження в базу даних)
    // ...

    // Підтвердження відправки
    echo "Дякуємо за відправку форми, $name! Ваша електронна адреса: $email";
}
?>
```

Рисунок 2.5 – Приклад отримання даних з форми POST-запитом і їх обробка

Приклад перевірки введених даних з форми на валідність зображено на рисунку 2.6.

```php
<?php
$nameErr = $emailErr = "";
$name = $email = "";

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    if (empty($_POST["name"])) {
        $nameErr = "Поле ім'я є обов'язковим";
    } else {
        $name = test_input($_POST["name"]);
        // додаткові перевірки для імені, наприклад, чи містить тільки букви та пробі
    }

    if (empty($_POST["email"])) {
        $emailErr = "Поле електронної пошти є обов'язковим";
    } else {
        $email = test_input($_POST["email"]);
        // додаткові перевірки для електронної пошти, наприклад, чи має правильний фор
    }
}

function test_input($data) {
    $data = trim($data);
    $data = stripslashes($data);
    $data = htmlspecialchars($data);
    return $data;
}
?>
```

Рисунок 2.6 – Приклад перевірки введених даних з форми на валідність

Приклад відображення форми на вебсторінці зображено на рисунку 2.7.

HTML-форми дозволяють користувачам вводити дані, такі як текст, числа, вибори тощо, та відправляти їх на сервер для подальшої обробки. PHP може обробити дані, які надсилаються з форми, за допомогою методів POST або GET.

```
<form method="post" action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]);?>
    Ім'я: <input type="text" name="name">
    <span class="error">* <?php echo $nameErr;?></span>
    <br><br>
    Електронна пошта: <input type="text" name="email">
    <span class="error">* <?php echo $emailErr;?></span>
    <br><br>
    <input type="submit" name="submit" value="Відправити">
</form>
```

Рисунок 2.7 – Приклад відображення форми на вебсторінці

Після відправки форми дані з неї можна отримати за допомогою масивів $_POST або $_GET у залежності від методу, вказаного у формі. Наприклад, $_POST['username'] міститиме значення, яке ввів користувач у поле з ім'ям "username".

Перш ніж використовувати дані, отримані з форми, їх слід перевірити на правильність та валідність. PHP дозволяє виконувати різні перевірки, такі як перевірка на пустоту, перевірка типу даних, перевірка на коректність введення тощо.

Після отримання та перевірки даних з форми, їх можна використовувати для різних цілей, таких як збереження в базі даних, відправка електронної пошти, виведення на вебсторінку тощо. PHP надає широкі можливості для роботи з базами даних, що дозволяє легко зберігати та обробляти дані користувачів.

PHP також може виконувати перенаправлення користувача на іншу сторінку або виконувати запит на інший URL за допомогою функцій, таких як header() або curl.

Робота з файлами в PHP відкриває широкі можливості для розробки вебдодатків, які потребують роботи з різними типами файлів та зберігання даних. Відповідно, PHP надає потужні інструменти для взаємодії з файловою

системою сервера, що дозволяє розробникам створювати різноманітні та функціональні вебдодатки.

PHP дозволяє легко читати вміст файлів на сервері за допомогою функцій, таких як file_get_contents() або fopen(). Це дозволяє отримувати дані з різних джерел, таких як текстові файли, CSV файли, XML файли тощо.

Також за допомогою PHP можно записувати дані в файли на сервері за допомогою функцій, таких як file_put_contents() або fwrite(). Це корисно для зберігання інформації, введеної користувачем через вебформи, логування подій або генерації вихідних даних.

PHP дозволяє легко створювати, переіменовувати, переміщувати та видаляти каталоги на сервері за допомогою функцій, таких як mkdir(), rename(), rmdir() тощо.

PHP надає ряд функцій для роботи зі шляхами до файлів та каталогів, таких як basename(), dirname(), realpath() тощо. Це дозволяє вам легко визначити розташування файлів на сервері та працювати з ними.

За допомогою PHP можно обробляти файли, завантажені користувачем через вебформи. Це дозволяє реалізувати функціонал завантаження файлів, наприклад, для завантаження зображень, відео, документів тощо.

## 2.3 XAMPP

XAMPP – безкоштовна багатоплатформова збірка вебсервера з відкритим початковим кодом, що містить HTTP-сервер Apache, базу данних MariaDB, MySQL й інтерпретатори скриптів для мов програмування PHP та Perl, а також додаткові бібліотеки, що дозволяють запустити повноцінний вебсервер [7,8].

XAMPP – це акронім: 'X' (будь-яка з чотирьох операційних систем), 'A'pache, 'M'ySQL, 'P'HP, 'P'erl.

XAMPP працює з усіма 32-х розрядними ОС Microsoft (у Windows 98 працює тільки Apache, не працює MySQL), а також з Linux, Mac OS X і Solaris.

Програма вільно розповсюджується згідно з ліцензією GNU General Public License і є безкоштовним, зручним у роботі web-сервером, здатним обслуговувати динамічні сторінки. Кількість завантажених пакетів XAMPP у жовтні 2008 року – 775064 завантажень (33280 Гб). На сьогоднішній день xampp є однією з найкращих збірок вебсервера, за допомогою цієї збірки ви зможете швидко розгорнути на своєму комп'ютері повноцінний і швидкий вебсервер.

Для установки XAMPP необхідно завантажити всього один файл формату zip, tar або exe, а компоненти програми не вимагають настройки. Програма регулярно оновлюється, для включення до складу новітніх версій Apache / MySQL / PHP та Perl. Також XAMPP йде з безліччю інших модулів, включаючи OpenSSL та phpMyAdmin.

Для користувача інтерфейс програми настільки простий, що її називають «збіркою для ледачих» («lazy man's WAMP/LAMP installation»).

Установка XAMPP займає менше часу, ніж установка кожного компонента окремо. Цей web-сервер поширюється в повній, стандартній і мінімальній (відомої як XAMPP Lite) версіях. Всі додаткові модулі також доступні для скачування.

З додаткових можливостей можна відзначити, що сама компанія випускає пакети оновлення, які випускаються у вигляді zip, 7-zip, tar або exe, які дозволяють оновити всі компоненти з однієї версії збірки xampp на новішу.

Основні компоненти XAMPP включають в себе:

1) Apache – це вебсервер, який використовується для обробки запитів HTTP від клієнтів і відображення вебсторінок. XAMPP включає Apache в якості основного вебсервера;

2) MySQL – це система управління базами даних (СУБД), яка використовується для зберігання та управління даними. XAMPP включає сервер баз даних MySQL, що дозволяє створювати та управляти базами даних на локальному сервері;

3) PHP – це мова програмування, яка використовується для створення динамічних вебсайтів та взаємодії з вебсторінками. XAMPP містить вбудовану підтримку PHP, що дозволяє розробникам створювати вебдодатки з використанням цієї мови програмування;

4) Perl – це мова програмування, яка часто використовується для обробки текстової інформації та скриптів на вебсерверах. XAMPP включає Perl як додатковий компонент для розширення можливостей розробки.

Плюс до цих основних компонентів, XAMPP також може включати інші корисні інструменти та компоненти, такі як phpMyAdmin (інтерфейс користувача для роботи з MySQL), OpenSSL (для шифрування даних), FileZilla FTP Server (для передачі файлів через FTP) та інші.

Приклад створення нової бази даних MySQL та таблиці зображено на рисунку 2.8.

```
CREATE DATABASE mydatabase;
USE mydatabase;
CREATE TABLE users (
    id INT AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(50),
    password VARCHAR(255)
);
```

Рисунок 2.8 – Приклад створення нової бази даних MySQL та таблиці

Приклад додавання даних до таблиці зображено на рисунку 2.9.

```
INSERT INTO users (username, password) VALUES ('john_doe', 'password123');
INSERT INTO users (username, password) VALUES ('jane_doe', 'password456');
```

Рисунок 2.9 – Приклад додавання даних до таблиці

XAMPP дозволяє розробникам створювати та тестувати вебсайти та вебдодатки локально на своєму комп'ютері без необхідності підключення до реального вебсервера. Це особливо корисно для розробки та відлагодження вебдодатків перед їх розгортанням на живому сервері.

## 2.4 MySQL

MySQL – вільна система керування реляційними базами даних. MySQL був розроблений компанією «TcX» для підвищення швидкодії обробки великих баз даних. Ця система керування базами даних (СКБД) з відкритим кодом була створена як альтернатива комерційним системам. MySQL з самого початку була дуже схожою на mSQL, проте з часом вона все розширювалася і зараз MySQL – одна з найпоширеніших систем керування базами даних. Вона використовується, в першу чергу, для створення динамічних вебсторінок, оскільки має чудову підтримку з боку різноманітних мов програмування.[5]

MySQL – компактний багатониттєвий сервер баз даних. Характеризується високою швидкістю, стійкістю і простотою використання. MySQL вважається гарним рішенням для малих і середніх застосувань. Сирцеві коди сервера компілюються на багатьох платформах. Найповніше можливості сервера виявляються в UNIX-системах, де є підтримка багатонитковості, що підвищує продуктивність системи в цілому.

Можливості сервера MySQL: простота у встановленні та використанні; підтримується необмежена кількість користувачів, що одночасно працюють із БД; кількість рядків у таблицях може досягати 50 млн; висока швидкість виконання команд; наявність простої і ефективної системи безпеки.

MySQL – це потужна система управління базами даних (СУБД), яка надає розширені можливості для зберігання, організації та обробки даних. Вона використовує реляційну модель даних, що дозволяє структурувати дані у вигляді таблиць зі зв'язками між ними. Це спрощує організацію та управління даними в базі даних[1, 3, 5, 6, 9-15].

Реляційна модель даних – це структура організації даних у вигляді таблиць зі зв'язками між ними. Вона була розроблена Едгаром Коддом у 1970 році і стала стандартом для організації даних у базах даних.

Дані в реляційній моделі організовані у вигляді таблиць. Кожна таблиця складається з рядків (кортежів) та стовпців (атрибутів). Рядок таблиці представляє один запис або кортеж даних, а стовпець представляє конкретний тип даних.

Ключі в реляційній моделі використовуються для унікальної ідентифікації записів у таблицях. Ключ може бути простим (одним атрибутом) або складним (комбінацією атрибутів), і він дозволяє ефективно виконувати пошук, сортування та з'єднання записів.

У реляційній моделі дані можуть бути пов'язані за допомогою зв'язків між таблицями. Це дозволяє виконувати складні запити, що об'єднують дані з різних таблиць за певними умовами.

Нормалізація – це процес організації даних у таблицях таким чином, щоб уникнути аномалій оновлення та видалення даних. Цей процес дозволяє покращити структуру бази даних та зменшити дублювання даних.

У реляційній моделі дані можна отримувати, оновлювати, вставляти та видаляти за допомогою мови запитів SQL (Structured Query Language). SQL надає ряд операторів для виконання різних операцій з даними, що дозволяє ефективно взаємодіяти з базою даних.

Реляційна модель даних є одним з найпоширеніших та найбільш ефективних методів для організації та управління даними в базах даних. Вона надає зручний спосіб структурування та взаємодії з даними, що робить її популярним вибором для багатьох типів застосунків, від невеликих вебсайтів до складних корпоративних систем.

MySQL використовує мову запитів SQL для взаємодії з базою даних. SQL дозволяє виконувати різноманітні операції з даними, такі як вибірка, вставка, оновлення та видалення. Вона підтримує транзакції, що дозволяє

групувати кілька операцій над даними в одну логічну одиницю, що забезпечує консистентність та надійність даних, особливо у вимогливих застосунках.

Також MySQL дозволяє створювати індекси для полів таблиць, що покращує швидкодію запитів до бази даних. Крім того, він надає ряд інструментів для аналізу та оптимізації виконання запитів. Вона має розширені можливості для забезпечення безпеки даних, такі як ролі та дозволи, шифрування даних, аудит доступу та інші механізми захисту.

MySQL надає засоби для резервного копіювання та відновлення бази даних, що дозволяє захистити вашу інформацію від втрати через непередбачувані події. Вона добре масштабується як вертикально (збільшення потужності сервера), так і горизонтально (розподілення навантаження між кількома серверами), що дозволяє йому використовуватися у різних типах застосунків, від невеликих вебсайтів до великих платформ.

Загалом, MySQL є потужною та надійною СУБД, яка надає широкі можливості для зберігання та управління даними у вебдодатках. Вона є одним з найпопулярніших виборів для розробників завдяки своїй відкритості, продуктивності та надійності.

Мова запитів SQL є стандартною мовою програмування для роботи з реляційними базами даних. Вона надає ряд команд та операцій для взаємодії з даними, що зберігаються в базі даних.

Операція SELECT використовується для вибірки даних з таблиць бази даних. Вона дозволяє вказати, які стовпці потрібно вибрати, які таблиці потрібно об'єднати, а також які умови повинні бути виконані для вибору конкретних рядків.

Операції INSERT, UPDATE, DELETE використовуються для вставки нових записів в таблицю, оновлення існуючих записів або видалення записів з таблиці відповідно. Вони дозволяють змінювати дані в базі даних відповідно до вимог додатку.

Операція JOIN використовується для об'єднання даних з двох або більше таблиць на основі спільного стовпця. Це дозволяє створювати складні запити,

які використовують дані з різних таблиць.

Операції GROUP BY та HAVING використовуються для групування рядків даних за значенням певного стовпця.

Операція HAVING використовується для фільтрації груп рядків на основі певних умов.

SQL дозволяє вкладати один запит всередині іншого, що дозволяє створювати складні та потужні запити. Підзапити можуть бути використані для вибору даних, які задовольняють певним умовам, а потім використовуються в основному запиті.

SQL надає ряд вбудованих функцій агрегування, таких як SUM, AVG, COUNT, MIN, MAX, які дозволяють обчислювати загальні значення по групам даних.

Мова SQL є потужним інструментом для роботи з базами даних, який дозволяє виконувати різноманітні операції з даними, такі як вибірка, вставка, оновлення та видалення, що робить її невід'ємною частиною розробки вебдодатків та систем управління базами даних.

Індексація та оптимізація запитів в MySQL є важливими аспектами для підвищення продуктивності та ефективності роботи з базою даних.

Індекси у MySQL використовуються для швидкого доступу до даних в таблицях. Вони створюються на одному або кількох стовпцях таблиці та дозволяють базі даних швидше виконувати пошук, сортування та фільтрацію даних. Індекси дозволяють зменшити кількість рядків, які потрібно переглянути при виконанні запиту, що підвищує швидкодію операцій.

У MySQL підтримуються різні типи індексів, включаючи звичайні (B-tree), унікальні, повний текст, геопросторові та складні індекси. Кожен тип індексу має свої особливості та використовується для різних видів запитів.

Для досягнення максимальної ефективності індексів важливо правильно визначати, які стовпці потребують індексації, та обирати відповідний тип індексу для кожного стовпця. Також важливо уникати надмірного

індексування, оскільки це може призвести до збільшення обсягу дискового простору та зменшення швидкодії оновлення та вставки даних.

Оптимізація запитів включає в себе ряд технік для поліпшення продуктивності виконання SQL-запитів. Це може включати в себе використання індексів, правильне написання запитів, використання оптимізатора запитів, визначення оптимальних рівнів ізоляції транзакцій, та інші методи.

Команда EXPLAIN у MySQL дозволяє аналізувати виконання SQL-запитів та розуміти, як база даних обробляє запити. Вона надає інформацію про те, які індекси використовуються, які таблиці скануються та інші важливі параметри, що дозволяє виявити можливість оптимізації запитів.

MySQL також надає засоби для збору статистики про виконання запитів та профілювання їх виконання. Це дозволяє ідентифікувати проблемні запити та здійснювати оптимізацію для покращення продуктивності.

Правильна індексація та оптимізація запитів грають важливу роль у підвищенні продуктивності та ефективності роботи з базою даних у MySQL. Вони дозволяють забезпечити швидку відповідь на запити користувачів та підтримувати високу продуктивність великих обсягів даних.

MySQL має розширені можливості для забезпечення безпеки даних, такі як ролі та дозволи, шифрування даних, аудит доступу та інші механізми захисту; надає засоби для резервного копіювання та відновлення бази даних, що дозволяє захистити вашу інформацію від втрати через непередбачувані події; добре масштабується як вертикально (збільшення потужності сервера), так і горизонтально (розподілення навантаження між кількома серверами), що дозволяє йому використовуватися у різних типах застосунків, від невеликих вебсайтів до великих платформ.

Приклад шифрування даних зображено на рисунку 2.10.

Приклад аудиту доступу зображено на рисунку 2.11.

```
CREATE TABLE sensitive_data (
    id INT AUTO_INCREMENT PRIMARY KEY,
    credit_card_number VARCHAR(50),
    encrypted_data VARBINARY(255)
);

INSERT INTO sensitive_data (credit_card_number, encrypted_data)
VALUES ('1234567890123456', AES_ENCRYPT('sensitive info', 'encryption_key'));
```

Рисунок 2.10 – Приклад шифрування даних

```
CREATE TABLE audit_log (
    id INT AUTO_INCREMENT PRIMARY KEY,
    action VARCHAR(255),
    username VARCHAR(50),
    timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

INSERT INTO audit_log (action, username) VALUES ('Data accessed', 'admin');
```

Рисунок 2.11 – Приклад аудиту доступу

Приклад обмеження доступу до чутливих даних зображено на рисунку 2.12.

```
SELECT * FROM sensitive_data WHERE id = 1 AND user_role = 'admin';
```

Рисунок 2.12 – Приклад обмеження доступу до чутливих даних

Ці приклади демонструють використання різноманітних заходів безпеки даних в MySQL, включаючи створення користувачів з відповідними дозволами, шифрування чутливих даних, ведення аудиту доступу та обмеження доступу до чутливої інформації.

## 2.5 Висновок до другого розділу

У результаті аналізу засобів розроблення програмного забезпечення для роботи онлайн кінотеатру було обрано набір інструментів, що включає PHP, MySQL та XAMPP.

PHP обраний як мова програмування для реалізації логіки бізнес-логіки та взаємодії з користувачами. Він є потужним та широко використовуваним інструментом для створення динамічних вебсайтів та додатків, зручним у використанні та швидким у розробці.

MySQL обраний як система управління базами даних, оскільки вона є потужною, надійною та добре підтримується PHP. MySQL надає ефективний механізм для зберігання та управління даними про фільми, користувачів, бронювання тощо.

XAMPP виступає в ролі локального сервера для розробки та тестування програмного забезпечення. Він забезпечує легке налаштування та управління вебсервером Apache, сервером баз даних MySQL та іншими необхідними компонентами, що дозволяє зосередитися на розробці без зайвих турбот щодо налаштування середовища.

Таким чином, вибір PHP, MySQL та XAMPP є оптимальним для розроблення онлайн кінотеатру, оскільки ці інструменти допоможуть створити функціональний, надійний та ефективний вебдодаток з можливістю швидкого розгортання та розширення.

Висновки до другого розділу дають загальне уявлення про вибрані засоби розроблення програмного забезпечення для роботи онлайн кінотеатру. Вони підтверджують, що обрані засоби є потужними, функціональними та відповідають потребам онлайн сервісу, що дозволить ефективно реалізувати поставлені завдання.

# РОЗДІЛ 3 РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ОНЛАЙН КІНОТЕАТРУ

## 3.1 Опис бази даних

Проаналізувавши предметну область можна виділити наступні сутності:

- «Фільми» – зберігається інформація про фільми;
- «Жанри» – зберігається інформація про жанри фільмів;
- «Сеанси» – зберігається інформація про сеанси показу;
- «Користувачі» – зберігається інформація про користувачів;
- «Квитки» – зберігається вся інформація про купівлю білетів;

Для кожного поля таблиць (таблиці 3.1–3.8) бази даних вказується поле, його розмір, тип та, за наявності, зв'язок. Для первинних ключів вводиться заборона невизначених значень.

Таблиця 3.1

Кінозали (cinema_rooms)

| Найменування атрибутів | | Тип поля | Розмір поля | Обмеження | Зв'язок |
|---|---|---|---|---|---|
| Код залу | id | int | 11 | Not null | PK |
| Назва залу | name | varchar | 128 | Not null | - |
| Додаткова інформація | info | text | - | Null | - |

Таблиця 3.2

Жанри (genres)

| Найменування атрибутів | | Тип поля | Розмір поля | Обмеження | Зв'язок |
|---|---|---|---|---|---|
| Код жанру | id | int | 11 | Not null | PK |
| Найменування | name | varchar | 64 | Not null | - |

Таблиця 3.3

Жанри фільмів (genre_of_movies)

| Найменування атрибутів | | Тип поля | Розмір поля | Обмеження | Зв'язок |
|---|---|---|---|---|---|
| Код зв'язку | id | int | 11 | Not null | PK |
| Код фільму | movie_id | int | 11 | Not null | FK |
| Код жанру | genre_id | int | 11 | Not null | FK |

Таблиця 3.4

Фільми (movies)

| Найменування атрибутів | | Тип поля | Розмір поля | Обмеження | Зв'язок |
|---|---|---|---|---|---|
| Код фільму | id | int | 11 | Not null | PK |
| Назва | title | varchar | 128 | Not null | - |
| Рік випуску | year | year | 4 | Not null | - |
| Постер | poster | varchar | 128 | Not null | - |
| Сюжет | info | text | - | Null | - |

Таблиця 3.5

Кіносеанси (movie_sessions)

| Найменування атрибутів | | Тип поля | Розмір поля | Обмеження | Зв'язок |
|---|---|---|---|---|---|
| Код сеансу | id | int | 11 | Not null | PK |
| Код залу | cinema_room_id | int | 11 | Not null | FK |
| Код фільму | movie_id | int | 11 | Not null | FK |
| Час початку | start_time | datetime | - | Not null | - |
| VIP ціна | vip_price | double | - | Not null | - |
| Econom ціна | econom_price | double | - | Not null | - |

Таблиця 3.6

Ряди кінозалу (rows)

| Найменування атрибутів | | Тип поля | Розмір поля | Обмеження | Зв'язок |
|---|---|---|---|---|---|
| Код ряду | id | int | 11 | Not null | PK |

| Код залу | cinema_room_id | int | 11 | Not null | FK |
|---|---|---|---|---|---|
| Номер ряду | number | int | 11 | Not null | - |
| Кількість місць | count_of_seats | int | 11 | Not null | - |
| Тип | type | enum | - | Not null | - |

Таблиця 3.7

Квитки (tickets)

| Найменування атрибутів | | Тип поля | Розмір поля | Обмеження | Зв'язок |
|---|---|---|---|---|---|
| Код квитка | id | int | 11 | Not null | PK |
| Код сеансу | movie_session_id | int | 11 | Not null | FK |
| Код користувача | user_id | int | 11 | Not null | FK |
| Ряд | row | int | 11 | Not null | - |
| Місце | seat | int | 11 | Not null | - |
| Час покупки | timestamp | timestamp | - | Not null | - |

Таблиця 3.8

Користувачі (users)

| Найменування атрибутів | | Тип поля | Розмір поля | Обмеження | Зв'язок |
|---|---|---|---|---|---|
| Код користувача | id | int | 11 | Not null | PK |
| Електронна адреса | email | varchar | 128 | Not null | - |
| Логін | login | varchar | 128 | Not null | - |
| Пароль | password | varchar | 32 | Not null | - |
| Ім'я | first_name | varchar | 128 | Not null | - |
| Призвіще | last_name | varchar | 128 | Not null | - |
| По-батькові | patronymic | varchar | 128 | Null | - |
| День народження | birthday | date | - | Null | - |
| Фото | photo | varchar | 128 | Not null | - |
| Роль у системі | role | enum | - | Not null | - |

Згідно структури з табл. 3.1-3.8 можна побудувати БД. Діаграма бази

даних буде мати вигляд – рис. 3.1.

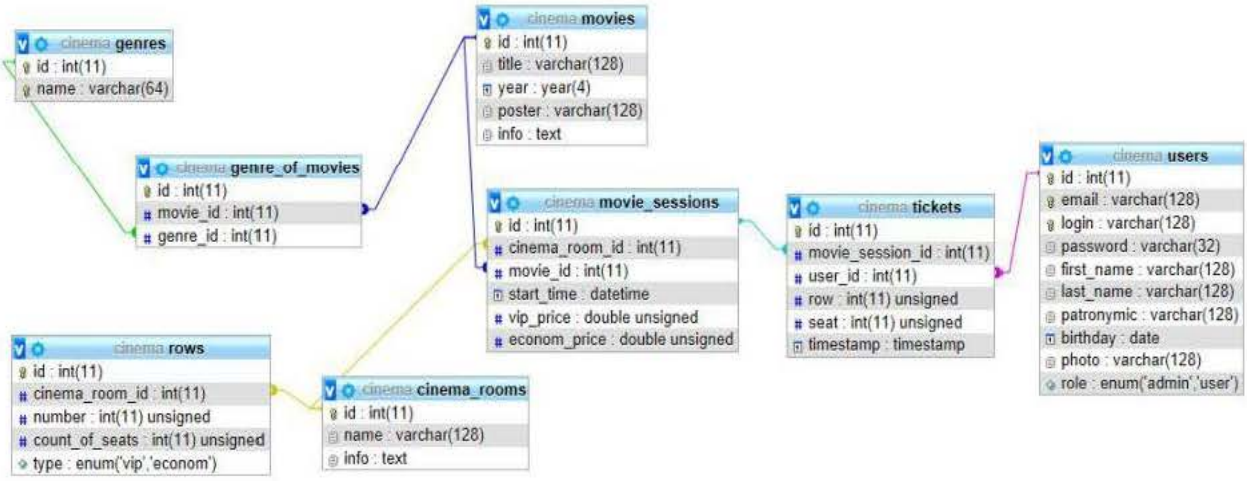


Рисунок 3.1 – Діаграма бази даних Cinema

## 3.2 Ролі користувачів

Даний динамічний web-засіб представляє 3 види користувацьких профілів:

1. Адміністратор;
2. Зареєстрований користувач;
3. Гість.

Адміністратор має повний доступ до всіх функцій сайту, роботи з БД, її заповненням, зміненням. Він може додавати, редагувати фільми, кіносеанси та власний профіль.

Зареєстрований користувач має змогу переглядати фільми, здійснювати пошук та купувати білети на необхідний кіносеанс, редагувати власний профіль.

Незареєстровані користувачі (гості) можуть лише переглядати головну сторінку, здійснювати по ній пошук (рис. 3.2) та переглядати додаткову інформацію натиснувши на необхідний фільм (рис. 3.3).

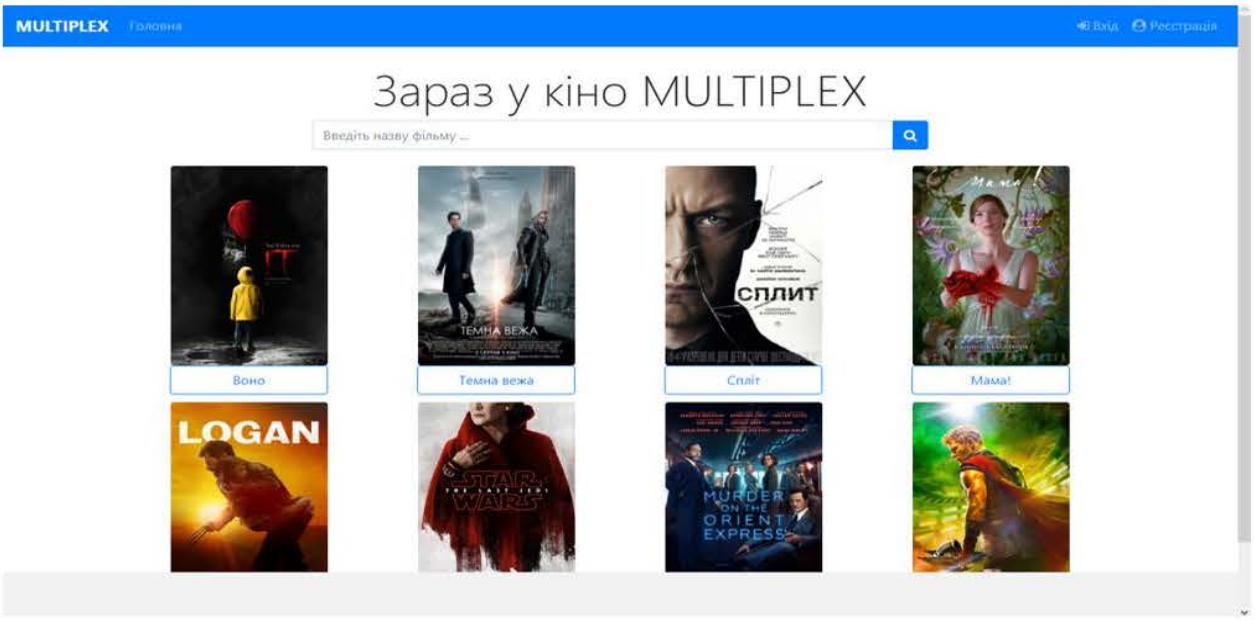


Рисунок 3.2 – Головна сторінка сайту

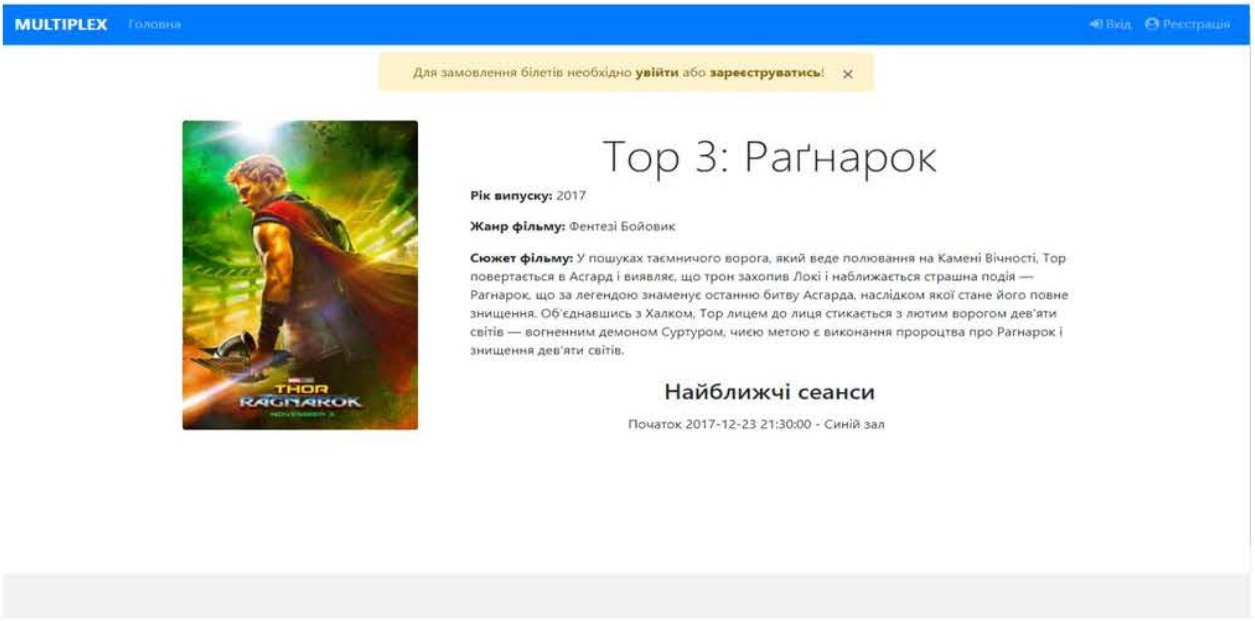


Рисунок 3.3 – Інформація про обраний фільм

## 3.3 Реєстрація та авторизація користувачів

Після того як була завантажена головна сторінка сайту. Гість сайту має змогу зареєструватися. Для реєстрації на сайті, користувач повинен ввести свої персональні дані у спеціальній формі (рис. 3.4). Користувачу після реєстрації встановлюється роль у системі – «user».

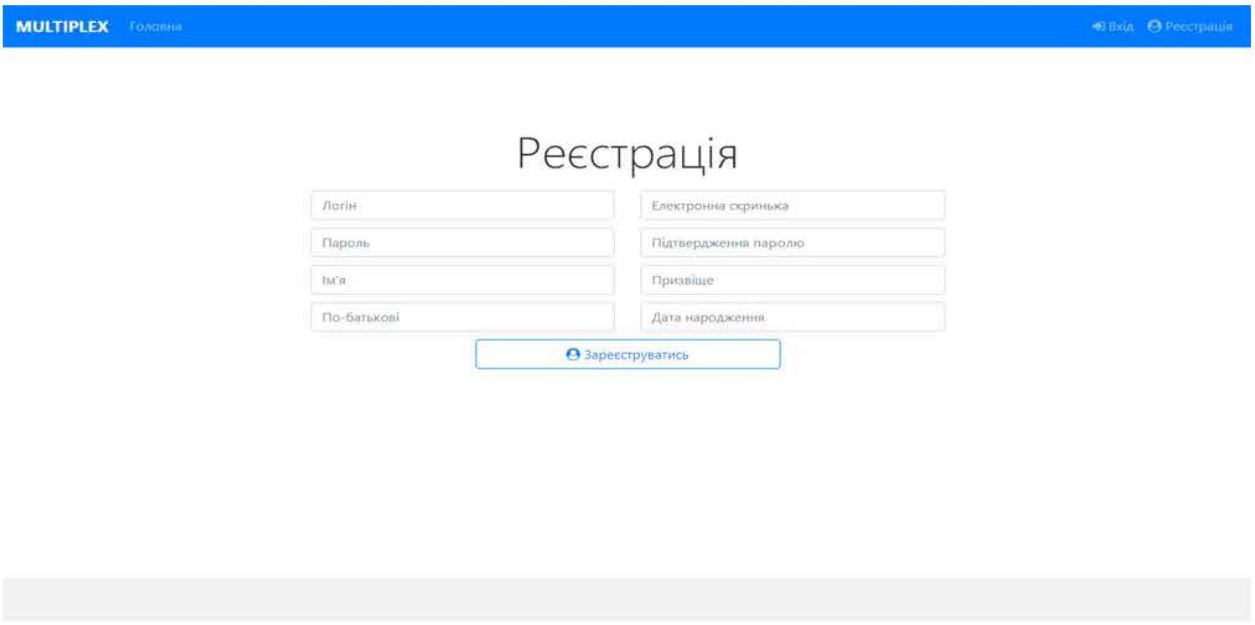


Рисунок 3.4 – Форма реєстрації користувача

Після проходження реєстрації користувач має можливість авторизуватися (рис. 3.5).

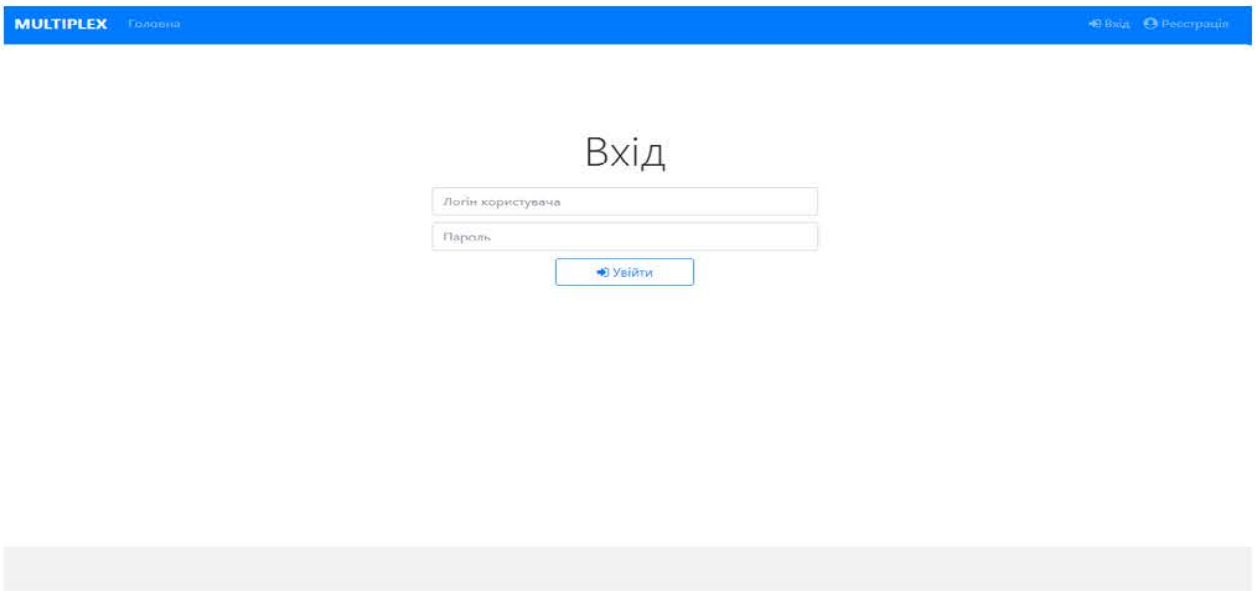


Рисунок 3.5 – Форма авторизації користувача

## 3.4 Користування контентом

Після проходження авторизації користувач потрапляє на головну сторінку, де має можливість перейти до особистого кабінету (рис. 3.6).

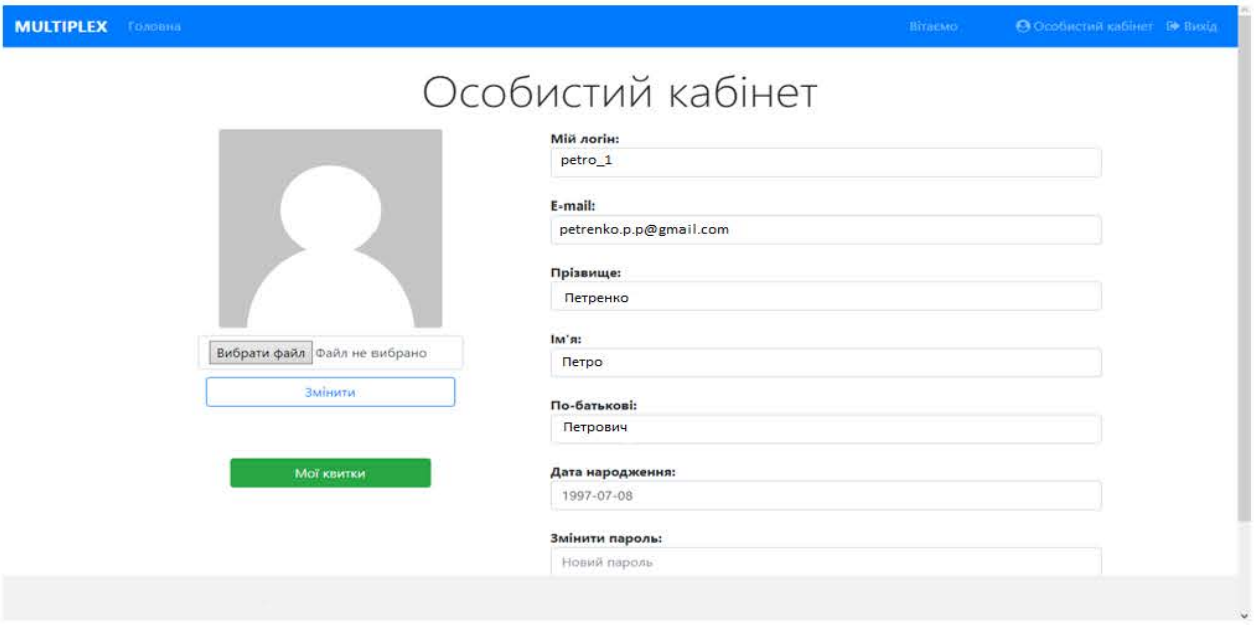


Рисунок 3.6 – Особистий кабінет

Після натискання на посилання «Мої квитки» користувач може переглянути історію придбання квитків на кіносеанси (рис. 3.7). Кожен квиток представлений у вигляді картки, на якій зображено основну інформацію – назва фільму, сеанс, вартість квитка, ряд, місце та дата придбання.

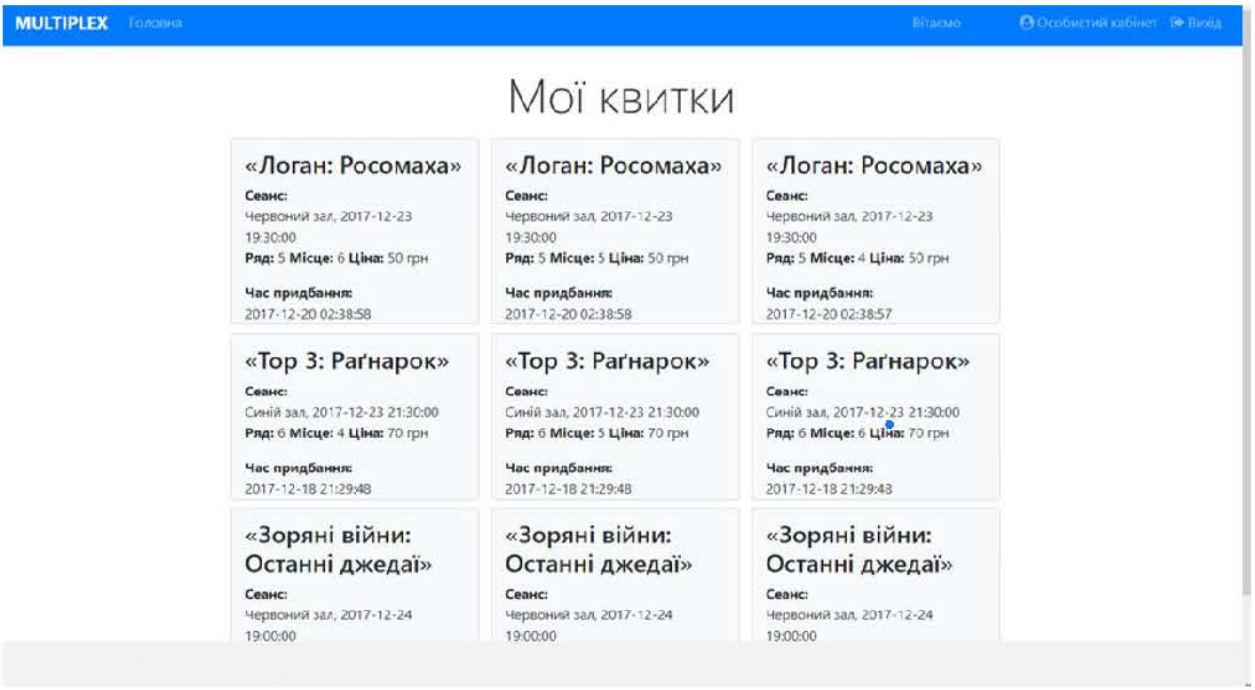


Рисунок 3.7 – Історія придбання квитків

### 3.4.1 Додавання, перегляд і редагування фільмів та кіносеансів

Для виконання дій додавання, редагування, видалення або перегляду фільмів та кіносеансів необхідно авторизуватися як адміністратор. На сторінках рис. 3.8 та рис. 3.9 є можливість додавання фільмів та кіносеансів.

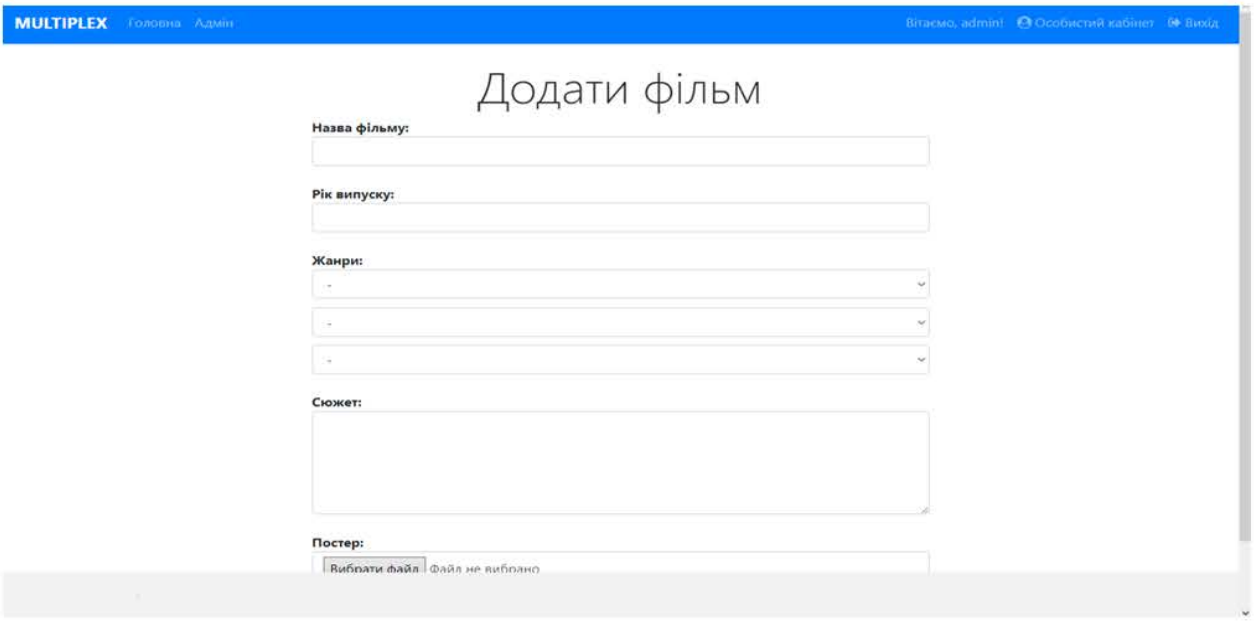


Рисунок 3.8 – Форма додання фільмів

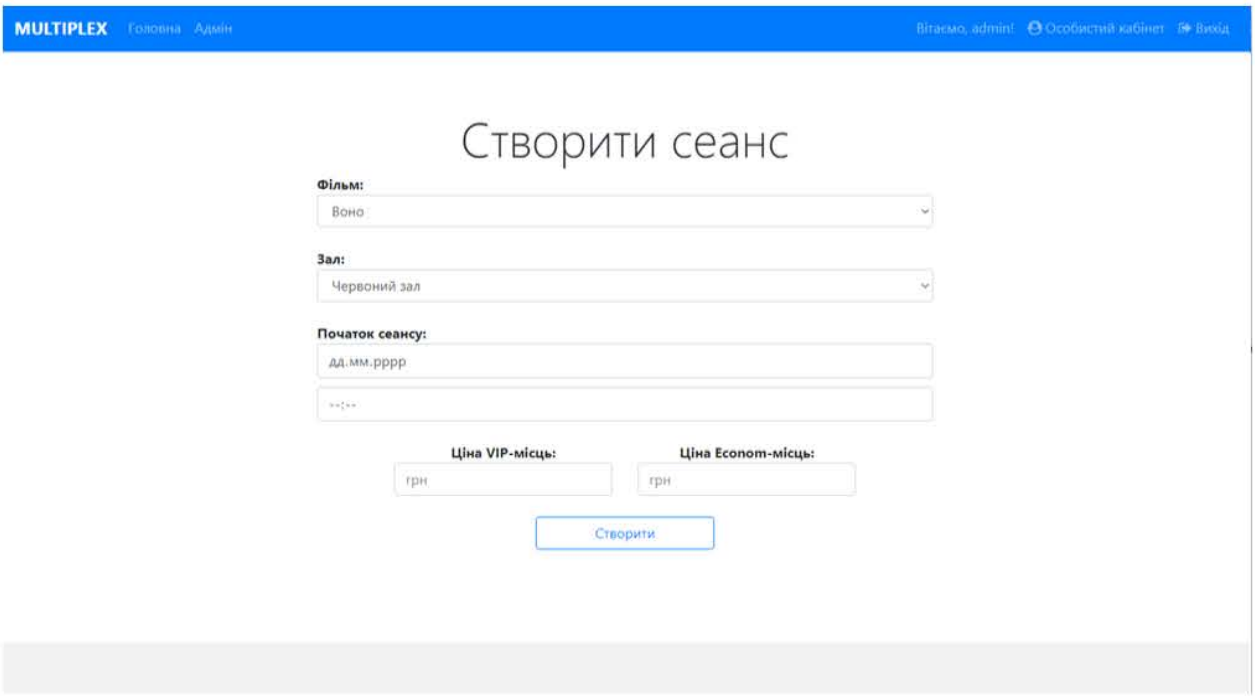


Рисунок 3.9 – Форма створення кіносеансів

Після авторизації з правами адміністратора на сторінках інформації про фільм з'являється можливість редагувати постер фільму.

### 3.4.2 Замовлення білетів

Користувачі мають можливість придбання квитків на кіносеанси, обравши бажаний сеанс на сторінці фільму, після чого відкриється сторінка обрання місць (рис. 3.10).

Користувач має можливість обрати місця у кінозалі та придбати квитки на них. Сірим відображаються зайняті місця. Після обрання місця – його колір змінюється на синій, загальна сумма покупки відображається унизу сторінки. Для купівлі необхідно натиснути на кнопку «Придбати».

Уся інформація про придбані квитки буде відображатися на сторінці «Мої квитки» (рис. 3.7).

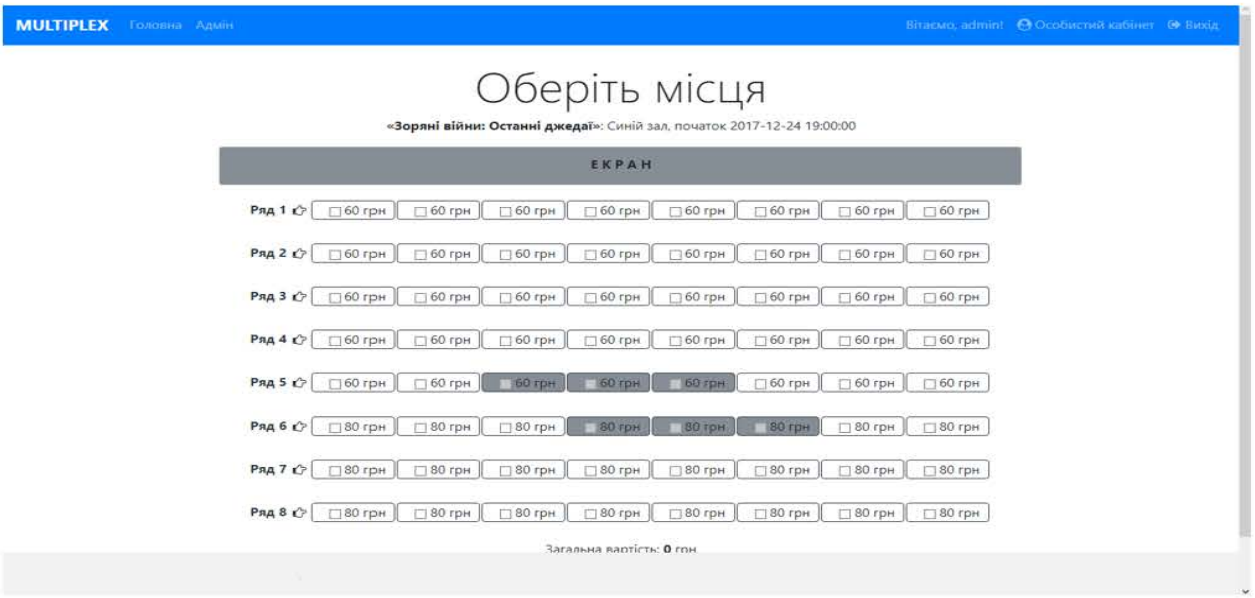


Рисунок 3.10 – Сторінка придбання квитків

## 3.5 Різниця між ролями користувачів

Різниця в користуванні контентом полягає в тому, що адміністратор має повні права на усі існуючі записи (додавання, редагування).

Зареєстрований користувач має право на перегляд додаткової інформації фільму та придбання квитків.

Незареєстрований користувач не має можливості використовувати ресурс, може тільки переглядати інформацію про фільми та переглядати можливі сеанси.

Візуально різниця показана на рис. 3.11 – 3.13.

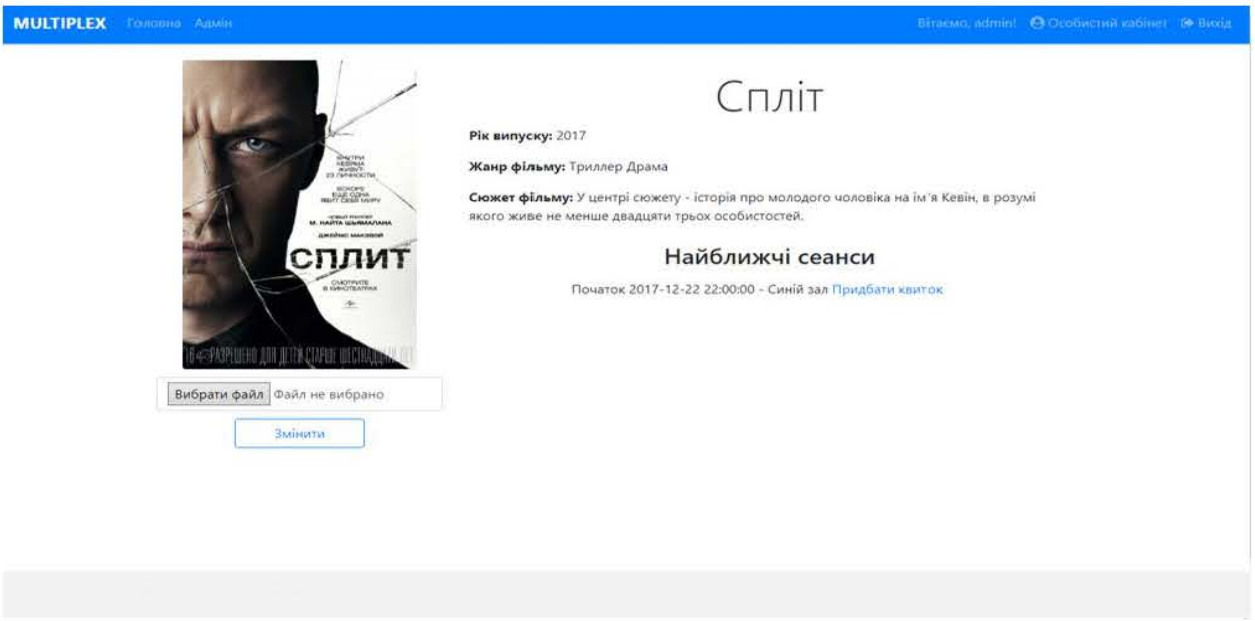


Рисунок 3.11 – Сторінка фільму для адміністраторів

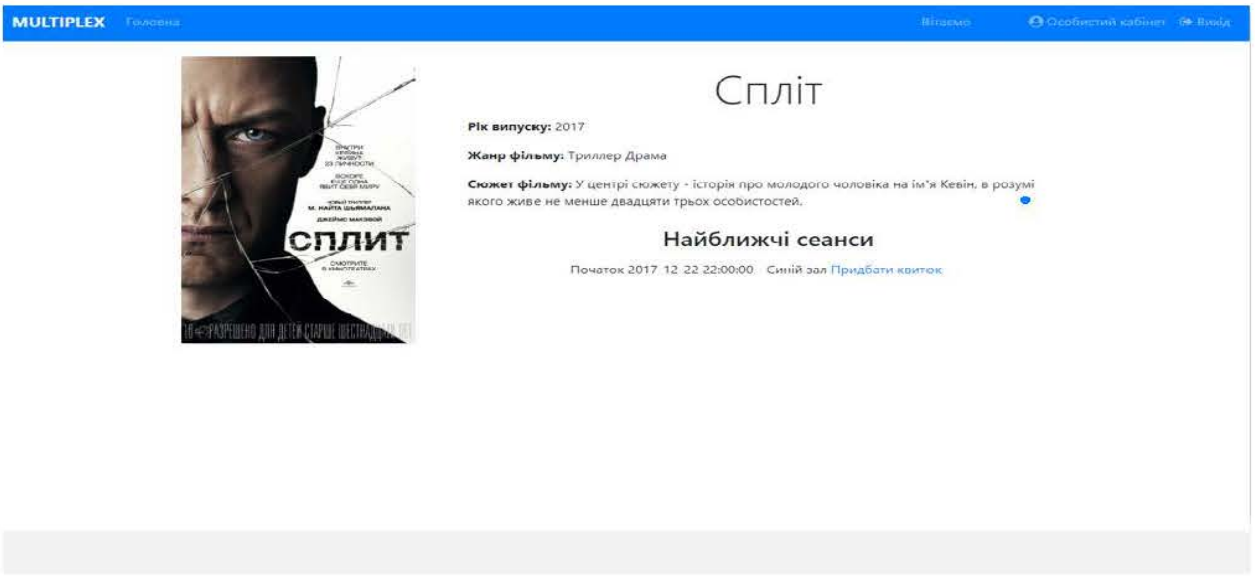


Рисунок 3.12 – Сторінка фільму для зареєстрованих користувачів

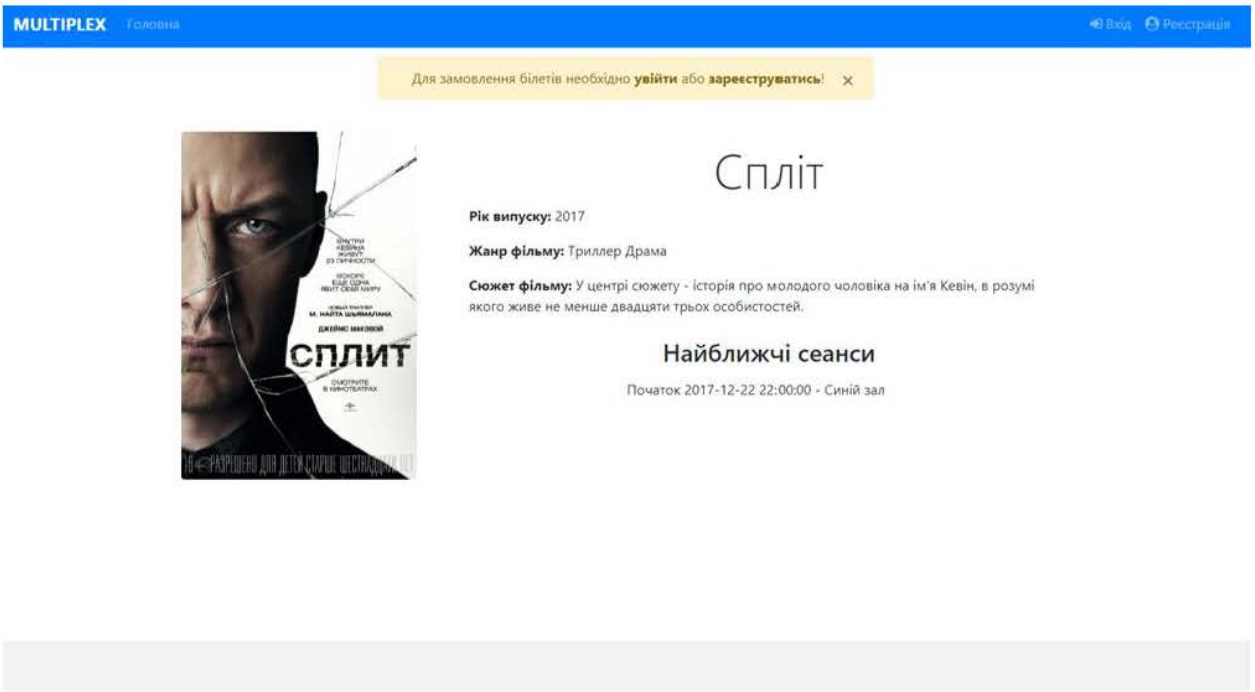


Рисунок 3.13 – Сторінка фільму для незареєстрованих користувачів

## 3.6 Висновки до третього розділу

До розділу «Розроблення програмного забезпечення для онлайн кінотеатру» можна зробити наступні висновки:

1) описана база даних є важливою складовою для роботи онлайн кінотеатру, що включає таблиці для зберігання інформації про фільми, кіносеанси, користувачів, бронювання квитків та інші важливі дані; ця структура даних дозволяє ефективно організувати і керувати інформацією в системі;

2) у системі передбачені різні ролі користувачів, такі як адміністратор, менеджер кінотеатру та звичайний користувач; кожна роль має свої права доступу до функцій системи, що забезпечує контроль за безпекою та конфіденційністю даних;

3) механізми реєстрації та авторизації користувачів забезпечують зручний та безпечний спосіб доступу до системи, що дозволяють ідентифікувати користувачів, керувати їх доступом до функціоналу та забезпечувати персоналізований досвід користувача;

4) користувачі мають можливість переглядати список фільмів, отримувати інформацію про них та обирати кіносеанси для відвідування, що забезпечує зручність та доступність для користувачів у виборі та перегляді контенту;

5) адміністраторам та менеджерам кінотеатру надається можливість додавати, переглядати та редагувати інформацію про фільми та кіносеанси, що дозволяє оновлювати та керувати розкладом показів та іншою інформацією про фільми;

6) користувачі можуть замовляти білети на кіносеанси через онлайн систему, що забезпечує зручність та доступність для користувачів у плануванні своїх візитів до кінотеатру;

7) адміністратори та менеджери мають розширені права для керування контентом та налаштування системи, тоді як звичайні користувачі мають обмежений доступ до функціоналу.

# ВИСНОВКИ

Мета проекту полягає у розробці та впровадженні високотехнологічної онлайн платформи для кінотеатру, що надає розширені можливості для користувачів у перегляді фільмів та здійсненні бронювання квитків.

Для досягнення мети розробки вебсайту онлайн кінотеатру були вирішені наступні задачі:

1) створення інтуїтивного та привабливого інтерфейсу , який дозволить користувачам легко переглядати розклад сеансів, шукати фільми та здійснювати бронювання квитків;

2) розробка функціоналу для бронювання квитків на сеанси, включаючи вибір місць, вибір фільму, оплату та підтвердження бронювання.;

3) створення панелі адміністратора для керування вмістом сайту, включаючи додавання та редагування фільмів, оновлення розкладу сеансів та керування акціями та пропозиціями;

4) реалізація заходів безпеки для захисту особистої інформації користувачів;

5) проведення регулярних тестів для виявлення та виправлення помилок, а також вдосконалення функціональності та ефективності роботи сайту.

Об'єктом дослідження цієї роботи була онлайн платформа «Кінотеатр».

Предметом дослідження цієї роботи був процес розробки та реалізації онлайн платформи для купівлі білетів на кіно, включаючи аналіз потреб користувачів, розробку функціональності, реалізацію безпеки та ефективність роботи системи.

Практичне значення одержаних результатів полягає в створенні функціональної та ефективної онлайн платформи для купівлі білетів на кіно. Отримані результати роботи мають важливе значення для користувачів і кінотеатрів. Для користувачів створена платформа для купівлі квитків на кіно стане зручним способом бронювання місць, що покращить їхній візит до кінотеатру. Для кінотеатрів це означає можливість оптимізувати продажі,

залучати більше клієнтів та підвищувати загальний рівень обслуговування. Також, збір та аналіз даних допоможе кінотеатрам краще розуміти своїх клієнтів і підлаштовувати свою роботу під їхні потреби.

У першій частині проекту було проаналізовано предметну область, описано мету, основні задачі та актуальність створення вебсайту онлайн кінотеатру.

Було визначено мету створення онлайн кінотеатру, яка може включати покращення доступності до кіно для користувачів, зручний процес бронювання квитків та збільшення прибутку кінотеатру. Також були сформульовані основні задачі, такі як розробка зручного інтерфейсу, підтримка різних пристроїв та безпека даних користувачів.

Було проведено огляд існуючого програмного забезпечення для роботи онлайн кінотеатру, що дозволило виявити переваги та недоліки різних рішень на ринку. Це допоможе визначити найкращі практики для подальшої розробки.

В рамках проекту було проведено дослідження сучасних інструментів для розробки вебсайту кінотеатру, таких як React.js, Angular, Vue.js тощо. Це дозволило обрати найбільш відповідний інструмент для реалізації задач та вимог проекту.

У результаті аналізу засобів розроблення програмного забезпечення для роботи онлайн кінотеатру було обрано набір інструментів, що включає PHP, MySQL та XAMPP.

PHP обраний як мова програмування для реалізації логіки, бізнес-логіки та взаємодії з користувачами. Він є потужним та широко використовуваним інструментом для створення динамічних вебсайтів та додатків, зручним у використанні та швидким у розробці.

MySQL обраний як система управління базами даних, оскільки вона є потужною, надійною та добре підтримується PHP. MySQL надає ефективний механізм для зберігання та управління даними про фільми, користувачів, бронювання тощо.

XAMPP виступає в ролі локального сервера для розробки та тестування програмного забезпечення. Він забезпечує легке налаштування та управління вебсервером Apache, сервером баз даних MySQL та іншими необхідними компонентами, що дозволяє зосередитися на розробці без зайвих турбот щодо налаштування середовища.

Таким чином, вибір PHP, MySQL та XAMPP є оптимальним для розроблення онлайн кінотеатру, оскільки ці інструменти допомагають створити функціональний, надійний та ефективний вебдодаток з можливістю швидкого розгортання та розширення.

Отримали загальне уявлення про вибрані засоби розроблення програмного забезпечення для роботи онлайн кінотеатру. Підтверджено, що обрані засоби є потужними, функціональними та відповідають потребам онлайн сервісу, що дозволило ефективно реалізувати поставлені завдання.

У розділі «Розроблення програмного забезпечення для онлайн кінотеатру» маємо:

1) описана база даних є важливою складовою для роботи онлайн кінотеатру, що включає таблиці для зберігання інформації про фільми, кіносеанси, користувачів, бронювання квитків та інші важливі дані; ця структура даних дозволяє ефективно організувати і керувати інформацією в системі;

2) у системі передбачені різні ролі користувачів, такі як адміністратор, менеджер кінотеатру та звичайний користувач; кожна роль має свої права доступу до функцій системи, що забезпечує контроль за безпекою та конфіденційністю даних;

3) механізми реєстрації та авторизації користувачів забезпечують зручний та безпечний спосіб доступу до системи, що дозволяють ідентифікувати користувачів, керувати їх доступом до функціоналу та забезпечувати персоналізований досвід користувача;

4) користувачі мають можливість переглядати список фільмів, отримувати інформацію про них та обирати кіносеанси для відвідування, що

забезпечує зручність та доступність для користувачів у виборі та перегляді контенту;

5) адміністраторам та менеджерам кінотеатру надається можливість додавати, переглядати та редагувати інформацію про фільми та кіносеанси, що дозволяє оновлювати та керувати розкладом показів та іншою інформацією про фільми;

6) користувачі можуть замовляти білети на кіносеанси через онлайн систему, що забезпечує зручність та доступність для користувачів у плануванні своїх візитів до кінотеатру;

7) адміністратори та менеджери мають розширені права для керування контентом та налаштування системи, тоді як звичайні користувачі мають обмежений доступ до функціоналу.

Результатами роботи є онлайн платформа «Кінотеатр».

Кваліфікаційна робота виконана у відповідності до стандарту спеціальності 121 «Інженерія програмного забезпечення» і демонструє володіння такими компетентностями як:

– здатність ідентифікувати, класифікувати та формулювати вимоги до програмного забезпечення;

– здатність брати участь у проектуванні програмного забезпечення, включаючи проведення моделювання (формальний опис) його структури, поведінки та процесів функціонування;

– здатність розробляти архітектури, модулі та компоненти програмних систем;

– володіння знаннями про інформаційні моделі даних, здатність створювати програмне забезпечення для зберігання, видобування та опрацювання даних;

– здатність обґрунтовано обирати та освоювати інструментарій з розробки та супроводження програмного забезпечення;

– здатність до алгоритмічного та логічного мислення тощо.

Серед програмних результатів, визначених стандартом, кваліфікаційна робота реалізовує наступні:

– аналізувати, цілеспрямовано шукати і вибирати необхідні для вирішення професійних завдань інформаційно-довідникові ресурси і знання з урахуванням сучасних досягнень науки і техніки;

– сміти розробляти людино-машинний інтерфейс;

– знати та вміти використовувати методи та засоби збору, формулювання та аналізу вимог до програмного забезпечення;

– проводити передпроектне обстеження предметної області, системний аналіз об'єкта проектування;

– вибирати вихідні дані для проектування, керуючись формальними методами опису вимог та моделювання;

– мотивовано обирати мови програмування та технології розробки для розв’язання завдань створення і супроводження програмного забезпечення;

– знати та вміти застосовувати інформаційні технології обробки, зберігання та передачі даних;

– вміти документувати та презентувати результати розробки програмного забезпечення тощо.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Welling, L., & Thomson, L. PHP and MySQL Web Development. Addison-Wesley, 2016.

2. PHP Documentation. URL: https://www.php.net/.

3. Julie C. Meloni. PHP, MySQL & JavaScript All in One, Sams Teach Yourself. 6th Edition, 2019.

4. Josh Lockhart, Kris Jordan, Phil Sturgeon. PHP: The Right Way, 2016.

5. Janet Valade , John W. Gosney. PHP & MySQL Web Development All-in-One Desk Reference For Dummies (For Dummies Series), 2008.

6. Mike McGrath. Php Mysql In Easy Steps, 2018.

7. XAMPP Documentation URL https://www.apachefriends.org/.

8. John Henderson. Understanding XAMPP, For Newbies!, 2013.

9. MySQL Documentation. URL https://www.mysql.com/.

10. Mitchell, R. SQL: The Ultimate Beginner's Guide to Learn SQL Programming Step by Step. Independently published, 2019.

11. G. V. Reilly. SQL Queries for Mere Mortals: A Hands-On Guide to Data Manipulation in SQL. Addison-Wesley, 2018.

12. W3School. The SQL LIKE Operator – [Електронний ресурс] – Режим доступу: http://www.w3schools.com/sql/sql_like.asp.

13. Karthik Appigatla. MySQL 8 Cookbook: Over 150 recipes for high-performance database querying and administration, 2014.

14. Luc Perkins, Eric Redmond, Jim R. Wilson. Seven Databases in Seven Weeks: A Guide to Modern Databases and the NoSQL Movement, 2018.

15. Russell J.T. Dyer. Learning MySQL and MariaDB, 2015.

# ДОДАТОК А

## Лістинг створення БД

```
CREATE DATABASE `cinema`;
SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
SET AUTOCOMMIT = 0;
START TRANSACTION;
SET time_zone = "+00:00";
CREATE TABLE `cinema_rooms` (
  `id` int(11) NOT NULL,
  `name` varchar(128) NOT NULL,
  `info` text
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
CREATE TABLE `genres` (
  `id` int(11) NOT NULL,
  `name` varchar(64) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
CREATE TABLE `genre_of_movies` (
  `id` int(11) NOT NULL,
  `movie_id` int(11) NOT NULL,
  `genre_id` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
CREATE TABLE `movies` (
  `id` int(11) NOT NULL,
  `title` varchar(128) NOT NULL,
  `year` year(4) NOT NULL,
  `poster` varchar(128) NOT NULL DEFAULT 'default.png',
  `info` text
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
CREATE TABLE `movie_sessions` (
  `id` int(11) NOT NULL,
  `cinema_room_id` int(11) NOT NULL,
  `movie_id` int(11) NOT NULL,
  `start_time` datetime NOT NULL,
  `vip_price` double UNSIGNED NOT NULL DEFAULT '70',
  `econom_price` double UNSIGNED NOT NULL DEFAULT '50'
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
CREATE TABLE `rows` (
  `id` int(11) NOT NULL,
  `cinema_room_id` int(11) NOT NULL,
  `number` int(11) UNSIGNED NOT NULL,
  `count_of_seats` int(11) UNSIGNED NOT NULL DEFAULT '8',
  `type` enum('vip','econom') NOT NULL DEFAULT 'econom'
```

```
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
CREATE TABLE `tickets` (
  `id` int(11) NOT NULL,
  `movie_session_id` int(11) NOT NULL,
  `user_id` int(11) NOT NULL,
  `row` int(11) UNSIGNED NOT NULL,
  `seat` int(11) UNSIGNED NOT NULL,
  `timestamp` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
CREATE TABLE `users` (
  `id` int(11) NOT NULL,
  `email` varchar(128) NOT NULL,
  `login` varchar(128) NOT NULL,
  `password` varchar(32) NOT NULL,
  `first_name` varchar(128) NOT NULL,
  `last_name` varchar(128) NOT NULL,
  `patronymic` varchar(128) DEFAULT NULL,
  `birthday` date DEFAULT NULL,
  `photo` varchar(128) NOT NULL DEFAULT 'default.png',
  `role` enum('admin','user') NOT NULL DEFAULT 'user'
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
INSERT INTO `users` (`id`, `email`, `login`, `password`, `first_name`,
`last_name`, `patronymic`, `birthday`, `photo`, `role`) VALUES
(1, 'admin@admin.ua', 'admin', '21232f297a57a5a743894a0e4a801fc3',
'Admin_Name', 'Admin_Surname', 'Admin_Patronymic', 'Date_Of_Birth',
'default.png', 'admin'),
(2, 'user@user.ua', 'martin', '925d7518fc597af0e43f5606f9a51512', 'Мартин',
'Мартинов', NULL, '1997-08-28', 'default.png', 'user'),
(3, 'petrenko.p.p@gmail.com', 'petro', '481f693417e9a74e783caea72063b606',
'Петро', 'Петренко', 'Петрович', '1997-07-08', 'default.png', 'user'),
(4, 'vasil@vasil.ua', 'vasil', 'dd677267f076e036dd1a3a36949375d3', 'Василенко',
'Василь', 'Васильович', '1996-03-09', 'default.png', 'user');
ALTER TABLE `cinema_rooms`
  ADD PRIMARY KEY (`id`);
ALTER TABLE `genres`
  ADD PRIMARY KEY (`id`),
  ADD UNIQUE KEY `name` (`name`);
ALTER TABLE `genre_of_movies`
  ADD PRIMARY KEY (`id`),
  ADD KEY `foreign_key_movie` (`movie_id`),
  ADD KEY `foreign_key_genre` (`genre_id`);
ALTER TABLE `movies`
  ADD PRIMARY KEY (`id`);
ALTER TABLE `movie_sessions`
```

```
  ADD PRIMARY KEY (`id`),
  ADD KEY `foreign_key_cinema_room` (`cinema_room_id`),
  ADD KEY `foreign_key_movie` (`movie_id`);
ALTER TABLE `rows`
  ADD PRIMARY KEY (`id`),
  ADD KEY `cinema_room_id` (`cinema_room_id`);
ALTER TABLE `tickets`
  ADD PRIMARY KEY (`id`),
  ADD KEY `movie_session` (`movie_session_id`),
  ADD KEY `user` (`user_id`);
ALTER TABLE `users`
  ADD PRIMARY KEY (`id`),
  ADD UNIQUE KEY `login` (`login`),
  ADD UNIQUE KEY `email` (`email`);
ALTER TABLE `cinema_rooms`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT,
AUTO_INCREMENT=3;
ALTER TABLE `genres`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT,
AUTO_INCREMENT=16;
ALTER TABLE `genre_of_movies`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT,
AUTO_INCREMENT=20;
ALTER TABLE `movies`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT,
AUTO_INCREMENT=9;
ALTER TABLE `movie_sessions`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT,
AUTO_INCREMENT=11;
ALTER TABLE `rows`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT,
AUTO_INCREMENT=17;
ALTER TABLE `tickets`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT,
AUTO_INCREMENT=35;
ALTER TABLE `users`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT,
AUTO_INCREMENT=5;
ALTER TABLE `genre_of_movies`
  ADD CONSTRAINT `foreign_key_genre` FOREIGN KEY (`genre_id`)
REFERENCES `genres` (`id`) ON DELETE CASCADE ON UPDATE
CASCADE,
```

```
  ADD CONSTRAINT `foreign_key_movie` FOREIGN KEY (`movie_id`)
REFERENCES `movies` (`id`) ON DELETE CASCADE ON UPDATE
CASCADE;
ALTER TABLE `movie_sessions`
  ADD CONSTRAINT `movie_sessions_ibfk_1` FOREIGN KEY
(`cinema_room_id`) REFERENCES `cinema_rooms` (`id`) ON DELETE
CASCADE ON UPDATE CASCADE,
  ADD CONSTRAINT `movie_sessions_ibfk_2` FOREIGN KEY (`movie_id`)
REFERENCES `movies` (`id`) ON DELETE CASCADE ON UPDATE
CASCADE;
ALTER TABLE `rows`
  ADD CONSTRAINT `rows_ibfk_1` FOREIGN KEY (`cinema_room_id`)
REFERENCES `cinema_rooms` (`id`) ON DELETE CASCADE ON UPDATE
CASCADE;
ALTER TABLE `tickets`
  ADD CONSTRAINT `tickets_ibfk_1` FOREIGN KEY (`user_id`)
REFERENCES `users` (`id`) ON DELETE CASCADE ON UPDATE CASCADE,
  ADD CONSTRAINT `tickets_ibfk_2` FOREIGN KEY (`movie_session_id`)
REFERENCES `movie_sessions` (`id`) ON DELETE CASCADE ON UPDATE
CASCADE;
COMMIT;
```

# ДОДАТОК Б

## Лістинг створення вебсторінок

### Лістинг файлу sign_in.php

```
<?php
$title = "Вхід до акаунту | MULTIPLEX";
function content()
{
   if(isset($_SESSION['user']))
   {
      include "templates/404_page.php";
   }
   else
   {
      include "templates/sign_in_page.php";
   }
}
include "templates/layouts/main_layout.php";
```

### Лістинг файлу account.php

```
<?php
$title = "Особистий кабінет | MULTIPLEX";
function content()
{
   if(isset($_SESSION['user']))
   {
      include "templates/account_page.php";
   }
   else
   {
      include "templates/404_page.php";
   }
}
include "templates/layouts/main_layout.php";
```

### Лістинг файлу add_movie.php

```
<?php
$title = "Додання кінофільму | MULTIPLEX";
function content()
{
   if(isset($_SESSION['user']) && $_SESSION['user']['role'] == 'admin')
```

```
    {
        include "templates/add_movie_page.php";
    }
    else
    {
        include "templates/404_page.php";
    }
}
include "templates/layouts/main_layout.php";
```

Лістинг файлу add_movie_session.php

```
<?php
$title = "Створення сеансу | MULTIPLEX";
function content()
{
    if(isset($_SESSION['user']) && $_SESSION['user']['role'] == 'admin')
    {
        include "templates/add_movie_session_page.php";
    }
    else
    {
        include "templates/404_page.php";
    }
} include "templates/layouts/main_layout.php";
```

Лістинг файлу admin.php

```
<?php
$title = "Адмін панель | MULTIPLEX";
function content()
{
    if(isset($_SESSION['user']) && $_SESSION['user']['role'] == 'admin')
    {
        include "templates/admin_page.php";
    }
    else
    {
        include "templates/404_page.php";
    }
}
include "templates/layouts/main_layout.php";
```

Лістинг файлу buy_ticket.php

```
<?php
session_start();
```

```
if(!isset($_SESSION['user']) || !isset($_GET['movie']) || !isset($_GET['session']))
{
   $title = "Помилка!  |  MULTIPLEX";
}
else
{
   $title = "Покупка білету  |  MULTIPLEX";
}
session_abort();
function content()
{
   if(!isset($_SESSION['user']) || !isset($_GET['movie']) ||
!isset($_GET['session']))
   {
      include "templates/404_page.php";
   }
   else
   {
      include "templates/buy_ticket_page.php";
   }
}
include "templates/layouts/main_layout.php";
```

Лістинг файлу index.php

```
<?php
$title = "Головна  |  MULTIPLEX";
function content()
{
   include "templates/main_page.php";
}
include "templates/layouts/main_layout.php";
```

Лістинг файлу movie.php

```
<?php
if(!isset($_GET['id']))
{
   $title = "Помилка!  |  MULTIPLEX";
}
else
{
   include "database/database_connection.php";
```

```
    $id = $_GET['id'];
    $sql = "SELECT * FROM movies WHERE id = '$id'";
    foreach ($connection->query($sql) as $movie){
        $title_movie = $movie['title'];
        $title = "$title_movie | MULTIPLEX";
    }
}
function content()
{
    if(!isset($_GET['id']))
    {
        include "templates/404_page.php";
    }
    else
    {
        include "templates/movie_page.php";
    }
}
include "templates/layouts/main_layout.php";
```

Лістинг файлу my_tickets.php

```
<?php
$title = "Мої квитки | MULTIPLEX";
function content()
{
    if(isset($_SESSION['user']))
    {
        include "templates/my_tickets_page.php";
    }
    else
    {
        include "templates/404_page.php";
    }
}
include "templates/layouts/main_layout.php";
```

Лістинг файлу registration.php

```
<?php
$title = "Реєстрація | MULTIPLEX";
function content()
{
    if(isset($_SESSION['user']))
    {
```

```
        include "templates/404_page.php";
    }
    else
    {
        include "templates/registration_page.php";
    }
}
include "templates/layouts/main_layout.php";
```

Лістинг файлу database_connection.php

```
<?php
$server = "127.0.0.1";
$username = "root";
$password = "123456789";
$database = "cinema";
try
{
    $connection = new
PDO("mysql:host=$server;dbname=$database;charset=UTF8", $username,
$password);
    $connection->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);
}
catch(PDOException $e)
{
    echo "Виникла помилка: " . $e->getMessage();
}
```

Лістинг файлу add_movie_session_action.php

```
<?php
include "../database/database_connection.php";
$cinema_room_id = $_POST['cinema_room'];
$movie_id = $_POST['movie'];
$start_time = $_POST['date'] . ' ' . $_POST['time'] . ':00';
$vip = $_POST['vip'];
$econom = $_POST['econom'];
$sql = "INSERT INTO movie_sessions (cinema_room_id, movie_id, start_time,
vip_price, econom_price) VALUES ('$cinema_room_id', '$movie_id',
'$start_time', '$vip', '$econom');";
$connection->query($sql);
```

```
header('Location: ../index.php?add_movie_session=true');
```

Лістинг файлу buy_ticket_action.php

```
<?php
include "../database/database_connection.php";
session_start();
$movie_id = $_GET['movie'];
$user_id = $_SESSION['user']['id'];
$session_id = $_GET['session'];
if ($_POST)
{
   for ($i = 1; $i <= 8; $i++)
   {
      for ($j = 1; $j <= 8; $j++)
      {
         if (isset($_POST["$i" . "_" . "$j"]))
         {
            $sql = "INSERT INTO tickets (movie_session_id, user_id, row, seat)
VALUES ('$session_id', '$user_id', '$i', '$j');";
            $connection->query($sql);
         }
      }
   }
   header("Location: ../index.php?buy_success=true");
}
else
{
   header("Location:
../buy_ticket.php?movie=$movie_id&session=$session_id&buy_fail=true");
}
```

Лістинг файлу registration_action.php

```
<?php
include "../database/database_connection.php";
session_start();
$login = htmlspecialchars($_POST['login']);
$sql = "SELECT * FROM users WHERE login = '$login'";
if($connection->query($sql)->rowCount() != 0)
{
   $_SESSION['post'] = $_POST;
   header('Location: ../registration.php?fail_login=true');
}
else
```

```
{
   $email = htmlspecialchars($_POST['email']);
   $sql = "SELECT * FROM users WHERE email = '$email'";
   if($connection->query($sql)->rowCount() != 0)
   {
      $_SESSION['post'] = $_POST;
      header('Location: ../registration.php?fail_email=true');
   }
   else
   {
      $password1 = md5($_POST['password1']);
      $password2 = md5($_POST['password2']);
      if($password1 != $password2)
      {
         $_SESSION['post'] = $_POST;
         header('Location: ../registration.php?fail_password=true');
      }
      else
      {
         $first_name = htmlspecialchars($_POST['first_name']);
         $last_name = htmlspecialchars($_POST['last_name']);
         $patronymic = htmlspecialchars($_POST['patronymic']);
         $birthday = $_POST['birthday'];
         $sql = "INSERT INTO users (login, email, password, first_name,
last_name, patronymic, birthday) VALUES ('$login', '$email', '$password1',
'$first_name', '$last_name', '$patronymic', '$birthday');";
         $connection->query($sql);
         $sql = "SELECT * FROM users WHERE login = '$login' AND password =
'$password1' LIMIT 1;";
         foreach ($connection->query($sql) as $user)
         {
            $_SESSION['user'] = $user;
         }
         header('Location: ../index.php?registration_success=true');
      }
   }
}
```

Лістинг файлу sign_in_action.php

```
<?php
include "../database/database_connection.php";
$login = htmlspecialchars($_POST['login'], ENT_QUOTES);
$password = md5($_POST['password']);
```

```
$sql = "SELECT * FROM users WHERE login = '$login' AND password =
'$password' LIMIT 1;";
foreach ($connection->query($sql) as $user)
{
   session_start();
   $_SESSION['user'] = $user;
}
if (isset($_SESSION['user']))
{
   header('Location: ../index.php');
}
else
{
   header('Location: ../sign_in.php?fail=true');
}
```

Лістинг файлу sign_out_action.php

```
<?php
session_start();
session_destroy();
header('Location: ../index.php');
```

Лістинг файлу update_pass.php

```
<?php
session_start();
include "../database/database_connection.php";
$id = $_SESSION['user']['id'];
$pass1 = $_POST['password1'];
$pass2 = $_POST['password2'];
if ($pass1 == $pass2)
{
   $hash = md5($pass1);
   $sql = "UPDATE users SET password = '$hash' WHERE id = '$id'";
   $connection->query($sql);
   header("Location: ../account.php?pass_success=true");
}
else
   header("Location: ../account.php?pass_fail=true");
```

Лістинг файлу update_photo_action.php

```
<?php
session_start();
```

```
include "../database/database_connection.php";
$login = $_SESSION['user']['login'];
$type = explode('.', $_FILES['photo']['name'])[1];
$upload_file = "../img/photo/$login.$type";
if ($type == 'jpg' || $type == 'jpeg' || $type == 'png' || $type == 'gif')
{
   if ($_FILES['photo']['size'] <= 5 * 1024 * 1024)
   {
      copy($_FILES['photo']['tmp_name'], $upload_file);
      $sql = "UPDATE users SET photo = '$login.$type' WHERE login = '$login'";
      $connection->query($sql);
      $_SESSION['user']['photo'] = "$login.$type";
      header("Location: ../account.php");
   }
   else
   {
      header("Location: ../account.php?fail_size_photo=true");
   }
}
else
{
   header("Location: ../account.php?fail_type_photo=true");
}
```

Лістинг файлу update_poster_action.php

```
<?php
session_start();
include "../database/database_connection.php";
$id = $_GET['id'];
$type = explode('.', $_FILES['poster']['name'])[1];
$upload_file = "../img/posters/$id.$type";
if ($type == 'jpg' || $type == 'jpeg' || $type == 'png' || $type == 'gif')
{
   if ($_FILES['poster']['size'] <= 6 * 1024 * 1024)
   {
      $sql_unlink = "SELECT poster FROM movies WHERE id = '$id'";
      foreach ($connection->query($sql_unlink) as $poster)
      {
         if ($poster['poster'] != 'default.png')
         {
            $p = $poster['poster'];
            unlink("../img/posters/$p");
         }
      }
```

```
        copy($_FILES['poster']['tmp_name'], $upload_file);
        $sql = "UPDATE movies SET poster = '$id.$type' WHERE id = '$id'";
        $connection->query($sql);
        header("Location: ../movie.php?id=$id");
    }
    else
    {
        header("Location: ../movie.php?id=$id&fail_size_photo=true");
    }
}
else
{
    header("Location: ../movie.php?id=$id&fail_type_photo=true");
}
```

Лістинг файлу main_layout.php

```
<!doctype html>
<html lang="ua">
    <head>
        <meta name="viewport" content="width=device-width, user-scalable=no,
initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0">
        <meta http-equiv="X-UA-Compatible" content="ie=edge">
        <link href="css/bootstrap.min.css" rel="stylesheet" type="text/css"/>
        <link href="css/font-awesome.min.css" rel="stylesheet" type="text/css"/>
        <link href="daterangepicker/daterangepicker.css" rel="stylesheet"
type="text/css"/>
        <link href="css/mystyles.css" rel="stylesheet" type="text/css"/>
        <title><?= $title; ?></title>
        <?php
            session_start();
            header("Content-Type: text/html; charset=utf-8");
        ?>
    </head>
    <body>
        <nav class="navbar navbar-expand-sm navbar-dark bg-primary">
            <button class="navbar-toggler navbar-toggler-right" type="button" data-
toggle="collapse" data-target="#navbarNav" aria-controls="navbarNav" aria-
expanded="false" aria-label="Toggle navigation">
                <span class="navbar-toggler-icon"></span>
            </button>
            <a class="navbar-brand family" href="#"><b>MULTIPLEX</b></a>
            <div class="collapse navbar-collapse" id="navbarNav">
                <ul class="navbar-nav mr-auto">
                    <li class="nav-item">
```

```
                <a class="nav-link" href="index.php">Головна</a>
            </li>
            <?php
                if(isset($_SESSION['user']) && $_SESSION['user']['role'] ==
'admin')
                {
                    echo "<li class='nav-item'>
                            <a class='nav-link' href='admin.php'>Адмін</a>
                        </li>";
                }
            ?>
        </ul>
        <ul class="navbar-nav">
            <?php
                if(isset($_SESSION['user']))
                {
                    $login = $_SESSION['user']['login'];
                    echo "<li class='nav-item'>
                            <a class='nav-link' href='#'>Вітаємо, $login!</a>
                        </li>
                        <li class='nav-item'>
                            <a class='nav-link' href='account.php'><span class='fa fa-
user-circle'></span> Особистий кабінет</a>
                        </li>
                        <li class='nav-item'>
                            <a class='nav-link'
href='actions/sign_out_action.php'><span class='fa fa-sign-out'></span>
Вихід</a>
                        </li>";
                }
                else
                {
                    echo "<li class='nav-item'>
                            <a class='nav-link' href='sign_in.php'><span class='fa fa-
sign-in'></span> Вхід</a>
                        </li>
                        <li class='nav-item'>
                            <a class='nav-link' href='registration.php'><span class='fa
fa-user-circle'></span> Реєстрація</a>
                        </li>";
                }
            ?>
        </ul>
    </div>
```

```
        </nav>
        <div class="container">
            <?= content(); ?>
        </div>
        <footer class="footer">
            <div class="container">
                <span class="text-muted"><span class='fa fa-copyright'></span>
Admin_Surname Admin_Name, 2024 рік.</span>
            </div>
        </footer>
        <script src="js/popper.min.js" type="text/javascript"></script>
        <script src="js/jquery-3.2.1.min.js" type="text/javascript"></script>
        <script src="js/bootstrap.min.js" type="text/javascript"></script>
        <script src="js/myscripts.js" type="text/javascript"></script>
        <script src="daterangepicker/moment.js" type="text/javascript"></script>
        <script src="daterangepicker/daterangepicker.js"
type="text/javascript"></script>
        <script src="js/mydatepicker.js" type="text/javascript"></script>
    </body>
</html>
```

Лістинг файлу 404_page.php

```
<div class="my-container">
    <h1 class="display-3 text-danger">404</h1>
    <h2>Сторінку не знайдено!</h2>
</div>
```

Лістинг файлу account_page.php

```
<?php
if(isset($_GET["fail_size_photo"]) || isset($_GET["fail_type_photo"]))
{
    ?>
    <div class="row justify-content-center">
        <div class="alert alert-warning alert-dismissable col-8 my-container"
align="center">
            <button type="button" class="close" data-
dismiss="alert">&times;</button>
            <b>Фото не завантажене!</b> Неправильний тип фото, або розмір
перевищує допустимий.
        </div>
    </div>
    <?php
}
```

```
if(isset($_GET["pass_success"]))
{
   ?>
   <div class="row justify-content-center">
      <div class="alert alert-success alert-dismissable col-6 my-container"
align="center">
         <button type="button" class="close" data-
dismiss="alert">&times;</button>
         <b>Пароль успішно змінено!</b>
      </div>
   </div>
   <?php
}
if(isset($_GET["pass_fail"]))
{
   ?>
   <div class="row justify-content-center">
      <div class="alert alert-warning alert-dismissable col-6 my-container"
align="center">
         <button type="button" class="close" data-
dismiss="alert">&times;</button>
         <b>Паролі не співпадають!</b>
      </div>
   </div>
   <?php
}
?>
<div class="row justify-content-center my-container my-padding">
   <h1 class="display-4">Особистий кабінет</h1>
</div>
<div class="row my-container">
   <div class="justify-content-center col-md-5 col-12" align="center">
      <div>
         <img class="rounded" src="img/photo/<?= $_SESSION['user']['photo'] ?>"
height="256px" width="256px">
                  <form enctype="multipart/form-data"
action="actions/update_photo_action.php" method="post">
                     <div align="center" class="col-9 my-container">
                        <input required class="form-control" type="file"
name="photo">
                     </div>
                     <div class="row justify-content-center">
                        <div align="center" class="col-8 my-container">
```

```
                                        <button class="btn btn-outline-primary btn-
block" type="submit">Змінити</button>
                                    </div>
                                </div>
                            </form>
                        </div>
                <div class="col-7" style="padding-top: 15%">
                    <a class="btn btn-success btn-block" href="my_tickets.php">Мої
квитки</a>
                </div>
            </div>
            <div class="col-md-7 col-12 container">
                <div>
                    <div>
                        <b>Мій логін:</b>
                    </div>
                    <div class="form-control">
                        <?= $_SESSION['user']['login']; ?>
                    </div>
                </div>
                <br>
                <div>
                    <div>
                        <b>E-mail:</b>
                    </div>
                    <div class="form-control">
                        <?= $_SESSION['user']['email']; ?>
                    </div>
                </div>
                <br>
                <div>
                    <div>
                        <b>Прізвище:</b>
                    </div>
                    <div class="form-control">
                        <?= $_SESSION['user']['last_name']; ?>
                    </div>
                </div>
                <br>
                <div>
                    <div>
                        <b>Ім'я:</b>
                    </div>
                    <div class="form-control">
```

```
            <?= $_SESSION['user']['first_name']; ?>
          </div>
        </div>
        <br>
        <div>
          <div>
            <b>По-батькові:</b>
          </div>
          <div class="form-control">
            <?= $_SESSION['user']['patronymic'] ?
$_SESSION['user']['patronymic'] : "-"; ?>
          </div>
        </div>
        <br>
        <div>
          <div>
            <b>Дата народження:</b>
          </div>
          <div class="form-control">
            <?= $_SESSION['user']['birthday']; ?>
          </div>
        </div>
        <br>
        <div>
          <b>Змінити пароль:</b>
        </div>
        <form action="actions/update_pass.php" method="post">
          <input class="form-control" required type="password" name="password1"
id="password1" minlength="4" maxlength="32" placeholder="Новий пароль">
          <input class="form-control my-container" required type="password"
name="password2" id="password2" minlength="4" maxlength="32"
placeholder="Підтвердження паролю">
          <div class="col-6 offset-3 justify-content-center">
            <button type="submit" class="btn btn-outline-primary btn-block my-
container">Змінити</button>
          </div>
        </form>
        <br>
      </div>
</div>
```

Лістинг файлу add_movie_page.php

```
<div class="row justify-content-center my-container my-padding">
  <h1 class="display-4">Додати фільм</h1>
</div>
```

```
<?php
include "database/database_connection.php";
$sql_genres = "SELECT id, name FROM genres";
if ($_POST)
{
   $title = $_POST['title'];
   $year = $_POST['year'];
   $info = $_POST['info'];
   $insert = "INSERT INTO movies (title, year, info) VALUES ('$title', '$year',
'$info');";
   $connection->query($insert);
   $select = "SELECT id FROM movies WHERE title = '$title' AND year = '$year'
AND info = '$info'";
   $id = null;
   foreach ($connection->query($select) as $movie)
      $id = $movie['id'];
   if ($_FILES['poster']['size'])
   {
      $type = explode('.', $_FILES['poster']['name'])[1];
      $upload_file = "img/posters/$id.$type";

      if ($type == 'jpg' || $type == 'jpeg' || $type == 'png' || $type == 'gif')
      {
         if ($_FILES['poster']['size'] <= 6 * 1024 * 1024)
         {
            copy($_FILES['poster']['tmp_name'], $upload_file);

            $update = "UPDATE movies SET poster = '$id.$type' WHERE id =
'$id'";
            $connection->query($update);
         }
      }
   }
   $genre1 = $_POST['genre1'];
   if ($genre1 != 'null')
   {
      $insert1 = "INSERT INTO genre_of_movies (movie_id, genre_id) VALUES
('$id', '$genre1');";
      $connection->query($insert1);
   }
   $genre2 = $_POST['genre2'];
   if ($genre2 != 'null' && $genre2 != $genre1)
   {
```

```
        $insert2 = "INSERT INTO genre_of_movies (movie_id, genre_id) VALUES
('$id', '$genre2');";
        $connection->query($insert2);
    }
    $genre3 = $_POST['genre3'];
    if ($genre3 != 'null' && $genre3 != $genre2 && $genre3 != $genre1)
    {
        $insert3 = "INSERT INTO genre_of_movies (movie_id, genre_id) VALUES
('$id', '$genre3');";
        $connection->query($insert3);
    }
    header("Location: index.php?add_movie=true");
}
?>
<div class="col-8 offset-2 justify-content-center">
    <form enctype="multipart/form-data" action="add_movie.php" method="post">
        <b>Назва фільму:</b><br>
        <input class="form-control" required name="title" value="<?= $_POST ?
$_POST['title'] : ""; ?>"><br>
        <b>Рік випуску:</b><br>
        <input class="form-control" required type="number" min="1900"
max="2100" name="year" value="<?= $_POST ? $_POST['year'] : ""; ?>"><br>
        <b>Жанри:</b><br>
        <select class="form-control" name="genre1">
            <option value="null">-</option>
            <?php foreach ($connection->query($sql_genres) as $genre) { ?>
                <option value="<?= $genre['id']; ?>"><?= $genre['name']; ?></option>
            <?php } ?>
        </select>
        <select class="form-control my-container" name="genre2">
            <option value="null">-</option>
            <?php foreach ($connection->query($sql_genres) as $genre) { ?>
                <option value="<?= $genre['id']; ?>"><?= $genre['name']; ?></option>
            <?php } ?>
        </select>
        <select class="form-control my-container" name="genre3">
            <option value="null">-</option>
            <?php foreach ($connection->query($sql_genres) as $genre) { ?>
                <option value="<?= $genre['id']; ?>"><?= $genre['name']; ?></option>
            <?php } ?>
        </select><br>
        <b>Сюжет:</b><br>
        <textarea class="form-control" required name="info" rows="5"><?= $_POST
? $_POST['info'] : ""; ?></textarea><br>
```

```
        <b>Постер:</b><br>
        <div align="center">
            <input class="form-control" type="file" name="poster">
        </div><br>
        <div class="col-4 offset-4" align="center">
            <button class="btn btn-outline-primary btn-block"
type="submit">Додати</button>
        </div>
    </form>
</div>
```

Лістинг файлу add_movie_session_page.php

```
<div class="row justify-content-center my-container" style="padding-top: 5%">
    <h1 class="display-4">Створити сеанс</h1>
</div>
<?php
include "database/database_connection.php";
$sql_movies = "SELECT id, title FROM movies";
$sql_cinema_rooms = "SELECT id, name FROM cinema_rooms";
?>
<div class="col-8 offset-2 justify-content-center">
    <form action="actions/add_movie_session_action.php" method="post">
        <b>Фільм:</b><br>
        <select class="form-control" name="movie">
            <?php foreach ($connection->query($sql_movies) as $movie) { ?>
                <option value="<?= $movie['id']; ?>"><?= $movie['title']; ?></option>
            <?php } ?>
        </select><br>
        <b>Зал:</b><br>
        <select class="form-control" name="cinema_room">
            <?php foreach ($connection->query($sql_cinema_rooms) as
$cinema_room) { ?>
                <option value="<?= $cinema_room['id']; ?>"><?=
$cinema_room['name']; ?></option>
            <?php } ?>
        </select><br>
        <b>Початок сеансу:</b><br>
        <input required class="form-control" type="date" name="date">
        <input required class="form-control my-container" type="time"
name="time"><br>
        <div class="row col-10 offset-1" align="center">
            <div class="col-6" align="center">
                <b>Ціна VIP-місць:</b><br>
```

```
            <input class="form-control" required name="vip" type="number"
min="1" max="1000" placeholder="грн">
          </div>
          <div class="col-6" align="center">
            <b>Ціна Econom-місць:</b><br>
            <input class="form-control" required name="econom" type="number"
min="1" max="1000" placeholder="грн">
          </div>
        </div>
        <br>
        <div class="col-4 offset-4" align="center">
          <button class="btn btn-outline-primary btn-block"
type="submit">Створити</button>
        </div>
    </form>
</div>
```

Лістинг файлу admin_page.php

```
<div class="row justify-content-center my-container" style="padding-top: 15%">
    <h1 class="display-4">Панель Адміністратора</h1>
</div>
<div class="row justify-content-center my-container" align="center">
    <div class="col-5">
        <a class="btn btn-outline-primary btn-block"
href="add_movie_session.php"><span class="fa fa-plus-circle"></span>
Створити сеанс</a>
    </div>
    <div class="col-5">
        <a class="btn btn-outline-primary btn-block" href="add_movie.php"><span
class="fa fa-plus-circle"></span> Додати кінофільм</a>
    </div>
</div>
```

Лістинг файлу buy_ticket_page.php

```
<?php
include "database/database_connection.php";
$movie_id = $_GET['movie'];
$session_id = $_GET['session'];
$movie_name = null;
$sql_movie_data = "SELECT title FROM movies WHERE id = '$movie_id';";
foreach ($connection->query($sql_movie_data) as $row)
{
    $movie_name =  $row['title'];
}
$session_time = null;
```

```
$sql_session_time_data = "SELECT start_time FROM movie_sessions WHERE id
= '$session_id';";
foreach ($connection->query($sql_session_time_data) as $row)
{
   $session_time =  $row['start_time'];
}
$session_room_name = null;
$sql_session_room_data = "SELECT cinema_rooms.name as name FROM
movie_sessions JOIN cinema_rooms ON cinema_rooms.id =
movie_sessions.cinema_room_id WHERE movie_sessions.id = '$session_id';";
foreach ($connection->query($sql_session_room_data) as $row)
{
   $session_room_name =  $row['name'];
}
if(isset($_GET['buy_fail']))
   {
      ?>
      <div class="row justify-content-center">
         <div class="alert alert-warning alert-dismissable col-6 my-container"
align="center">
            <button type="button" class="close" data-
dismiss="alert">&times;</button>
            <b>Необхідно обрати місця!</b>
         </div>
      </div>
      <?php
   }
?>
<div class="row justify-content-center my-container my-padding">
   <h1 class="display-4">Оберіть місця</h1>
</div>
<div align="center">
   <p><b>«<?= $movie_name; ?>»</b>: <?= $session_room_name; ?>, початок
<?= $session_time; ?></p>
</div>
<div align="center">
   <div class="col-10 border border-light rounded bg-secondary my-container my-
padding" align="center">
      <p class="t"><b>E K P A H</b></p>
   </div>
</div>
<div class="row justify-content-center my-padding">
   <form action="actions/buy_ticket_action.php?movie=<?= $movie_id;
?>&session=<?= $session_id; ?>" method="post">
```

```
<?php
    $sql = "SELECT * FROM movie_sessions WHERE id = '$session_id'";
    foreach ($connection->query($sql) as $session)
    {
        for ($i = 1; $i <= 8; $i++)
        {
            $row_price = null;
            $sql = "SELECT cinema_rooms.name as name, cinema_rooms.id as id
FROM movie_sessions JOIN cinema_rooms ON cinema_rooms.id =
movie_sessions.cinema_room_id WHERE movie_sessions.id = '$session_id';";
            foreach ($connection->query($sql) as $room)
            {
                $room_id = $room['id'];
                $sql = "SELECT * FROM rows WHERE cinema_room_id =
'$room_id' and number = $i";
                foreach ($connection->query($sql) as $row)
                {
                    $row_price =  $session[$row['type'] . "_price"];
                }
            }
            echo "<div class='row justify-content-center my-padding'>";
            echo "<text class='text-dark' style='margin-right: 7px;'><b>Ряд
$i</b></text><span class='fa fa-hand-o-right' style='margin-top: 5px; margin-
right: 5px'></span>";
            for ($j = 1; $j <= 8; $j++)
            {
                $sql_seat = "SELECT * FROM tickets WHERE movie_session_id
= '$session_id' and row = '$i' and seat = '$j'";
                if ($connection->query($sql_seat)->rowCount())
                {
                    ?>
                    <div class="justify-content-center border border-dark rounded
bg-secondary" style="margin-right: 2px; padding-right: 10px">
                        <label class="form-check-label">
                            <input class="btn btn-outline-primary" disabled
type="checkbox" name="<?= "$i"."_"."$j"; ?>"> <?= $row_price; ?> грн
                        </label>
                    </div>
                    <?php
                }
                else
                {
                    ?>
```

```
                    <div class="justify-content-center border border-dark rounded"
id="div<?= "$i"."_"."$j"; ?>" style="margin-right: 2px; padding-right: 10px">
                        <label class="form-check-label">
                            <input class="btn btn-outline-primary" type="checkbox"
name="<?= "$i"."_"."$j"; ?>" onchange="sum_price('<?= "$i"."_"."$j"; ?>' ,'<?=
$row_price; ?>');"> <?= $row_price; ?> грн
                        </label>
                    </div>
                    <?php
                }
            }
            echo "</div><br>";
        }
    }
    ?>
    <div class="col-12 my-conptainer" align="center">
        Загальна вартість: <b id="sum_price">0</b> грн</span>
    </div>
    <div class="col-12 my-container" align="center">
        <button class="btn btn-outline-primary" type="submit" onclick="return
confirm('Ви впевнені?')">Придбати</button>
    </div>
</form>
</div>
```

Лістинг файлу main_page.php

```
<?php
include "database/database_connection.php";
$title_part = isset($_POST['title_part']) ? $_POST['title_part'] : "";
if(isset($_GET["registration_success"]))
{
?>
<div class="row justify-content-center">
    <div class="alert alert-success alert-dismissable col-6 my-container"
align="center">
        <button type="button" class="close" data-
dismiss="alert">&times;</button>
        Реєстрація пройшла усішно! Вітаємо, <b><?=
$_SESSION['user']['first_name']; ?></b>!
    </div>
</div>
<?php
}
if(isset($_GET["add_movie_session"]))
```

```
{
  ?>
  <div class="row justify-content-center">
    <div class="alert alert-success alert-dismissable col-6 my-container"
align="center">
      <button type="button" class="close" data-
dismiss="alert">&times;</button>
      <b>Сеанс успішно створено!</b>
    </div>
  </div>
  <?php
}
if(isset($_GET["add_movie"]))
{
  ?>
  <div class="row justify-content-center">
    <div class="alert alert-success alert-dismissable col-6 my-container"
align="center">
      <button type="button" class="close" data-
dismiss="alert">&times;</button>
      <b>Кінофільм успішно додано!</b>
    </div>
  </div>
  <?php
}
if(isset($_GET["buy_success"]))
{
  ?>
    <div class="row justify-content-center">
      <div class="alert alert-success alert-dismissable col-7 my-container"
align="center">
        <button type="button" class="close" data-
dismiss="alert">&times;</button>
        <b>Білети успішно придбані!</b> Знайти їх можливо у
"Особистому кабінеті".
      </div>
    </div>
  <?php
}
?>
<div class="row justify-content-center my-container my-padding">
  <h1 class="display-4">Зараз у кіно MULTIPLEX</h1>
</div>
<div class="col-8 offset-2">
```

```
   <form class="form form-vertical" action="index.php" method="post"
enctype="multipart/form-data">
      <div class="input-group">
         <input type="text" name="title_part" class="form-control"
placeholder="Введіть назву фільму ..." value="<?= $title_part; ?>">
         <div class="input-group-btn">
            <button class="btn btn-primary" type="submit">
               <i class="fa fa-search"></i>
            </button>
         </div>
      </div>
   </form>
</div>
<div class="row my-padding">
<?php
$now = date("Y-m-d H:i:s");
$movies = "SELECT * FROM movies WHERE title LIKE '%$title_part%';";
foreach ($connection->query($movies) as $movie)
{
   $movie_id = $movie['id'];
   $sql_movie_sessions = "SELECT * FROM movie_sessions WHERE start_time
>= '$now' and movie_id = '$movie_id';";
   if ($connection->query($sql_movie_sessions)->rowCount())
   {
      ?>
         <div class="col-md-3 col-12 my-container" align="center" title="<?=
$movie['title']; ?>">
            <div class="col-12">
               <a href="movie.php?id=<?= $movie_id; ?>">
                  <img class="rounded" src="img/posters/<?= $movie['poster']; ?>"
height="260px" width="181px">
               </a>
            </div>
            <div class="col-10">
               <a class="btn btn-block btn-outline-primary" align="center"
href="movie.php?id=<?= $movie_id; ?>"><?= strlen($movie['title']) <= 40 ?
$movie['title'] : substr($movie['title'], 0, 39); ?></a>
            </div>
         </div>
      <?php
   }
}
?>
</div>
```

Лістинг файлу movie_page.php

```
<?php
if(isset($_GET["fail_size_photo"]) || isset($_GET["fail_type_photo"]))
{
   ?>
   <div class="row justify-content-center">
      <div class="alert alert-warning alert-dismissable col-8 my-container"
align="center">
         <button type="button" class="close" data-
dismiss="alert">&times;</button>
         <b>Фото не завантажене!</b> Неправильний тип фото, або розмір
перевищує допустимий.
      </div>
   </div>
   <?php
}
?>
<?php
   include "database/database_connection.php";
   $this_movie = null;
   $id = $_GET['id'];
   $sql = "SELECT * FROM movies WHERE id = '$id';";
   foreach ($connection->query($sql) as $movie)
   {
      $this_movie = $movie;
   }
   if(!isset($_SESSION["user"]))
   {
      ?>
      <div class="row justify-content-center">
         <div class="alert alert-warning alert-dismissable col-6 my-container"
align="center">
            <button type="button" class="close" data-
dismiss="alert">&times;</button>
            Для замовлення білетів необхідно <b>увійти</b> або
<b>зареєструватись</b>!
         </div>
      </div>
      <?php
   }
?>
<div class="row my-padding my-container container">
   <div class="col-md-4 col-12" align="center">
```

```
        <img class="rounded" src="img/posters/<?= $this_movie['poster']; ?>"
height="400px" width="270px">
        <?php if (isset($_SESSION['user']) && $_SESSION['user']['role'] == 'admin')
{ ?>
        <div>
          <form enctype="multipart/form-data"
action="actions/update_poster_action.php?id=<?= $_GET['id']; ?>"
method="post">
            <div align="center" class="my-container">
              <input required class="form-control" type="file" name="poster">
            </div>
            <div class="row justify-content-center">
              <div align="center" class="col-6 my-container">
                <button class="btn btn-outline-primary btn-block"
type="submit">Змінити</button>
              </div>
            </div>
          </form>
        </div>
        <?php }?>
      </div>
      <div class="col-md-8 col-12">
        <div class="row justify-content-center my-container">
          <h1 class="display-4"><?= $this_movie['title']; ?></h1>
        </div>
        <p><b>Рік випуску:</b> <?= $this_movie['year']; ?></p>
        <p><b>Жанр фільму:</b>
          <?php
            $genres = "SELECT name FROM genre_of_movies JOIN genres ON
genres.id = genre_of_movies.genre_id WHERE movie_id = '$id';";
            foreach ($connection->query($genres) as $genre){
              echo $genre['name'] . ' ';
            }
          ?>
        </p>
        <p><b>Сюжет фільму:</b> <?= $this_movie['info']; ?></p>
        <div class="row justify-content-center my-container my-padding">
          <h3>Найближчі сеанси</h3>
        </div>
        <div class="row justify-content-center my-padding container"
align="center">
          <?php
          $now = date("Y-m-d H:i:s");
```

```
        $sql_movie_sessions = "SELECT * FROM movie_sessions WHERE
start_time >= '$now' and movie_id = '$id' ORDER BY start_time;";
        foreach ($connection->query($sql_movie_sessions) as $session)
        {
          $session_id = $session['id'];
          $sql_room = "SELECT cinema_rooms.name as name FROM
movie_sessions JOIN cinema_rooms ON cinema_rooms.id =
movie_sessions.cinema_room_id WHERE movie_sessions.id = '$session_id';";
          foreach ($connection->query($sql_room) as $room)
          {
            echo "<p>";
            ?>
            Початок <?= $session['start_time']; ?> - <?= $room['name']; ?>
            <?php
            if (isset($_SESSION["user"]))
            {
              ?>
              <a href="buy_ticket.php?movie=<?= $id; ?>&session=<?=
$session_id; ?>">Придбати квиток</a>
              <?php
            }
            echo "</p>";
          }
        }
        ?>
      </div>
    </div>
</div>
```

Лістинг файлу my_tickets_page.php

```
<?php
include "database/database_connection.php";
$user_id = $_SESSION['user']['id'];
$sql = "SELECT tickets.row as row, tickets.seat as seat, tickets.timestamp as
purchase_date, cinema_rooms.name as cinema_rooms_name, movies.title as
movie, movie_sessions.start_time as start, movie_sessions.vip_price as vip,
movie_sessions.econom_price as econom FROM tickets JOIN movie_sessions ON
movie_sessions.id = tickets.movie_session_id JOIN movies ON movies.id =
movie_sessions.movie_id JOIN cinema_rooms ON cinema_rooms.id =
movie_sessions.cinema_room_id WHERE tickets.user_id = '$user_id' ORDER BY
tickets.timestamp DESC;";
?>
<div class="row justify-content-center my-container my-padding">
  <h1 class="display-4">Мої квитки</h1>
</div>
```

```
<div class="row justify-content-center">
<?php
foreach ($connection->query($sql) as $ticket)
{
   ?>
   <div class="card col-3 my-container my-padding bg-light">
      <h3>«<?= $ticket['movie']; ?>»</h3>
      <b>Сеанс:</b>
      <?= $ticket['cinema_rooms_name']; ?>, <?= $ticket['start']; ?>
      <p><b>Ряд:</b> <?= $ticket['row']; ?> <b>Місце:</b> <?= $ticket['seat'];
?> <b>Ціна:</b> <?= $ticket['row'] >= 6 ? $ticket['vip'] : $ticket['econom']; ?>
грн</p>
      <b>Час придбання:</b>
      <?= $ticket['purchase_date']; ?>
   </div>
   <p style="color: white;">m</p>
   <?php
}?>
</div>
```

Лістинг файлу registration_page.php

```
<form action="actions/registration_action.php" method="post">
   <div class="row justify-content-center my-container" style="padding-top: 8%">
      <h1 class="display-4">Реєстрація</h1>
   </div>
   <div class="row">
      <div class="col-md-4 offset-md-2 my-container">
         <input class="form-control" required type="text" name="login" id="login"
minlength="4" maxlength="128" placeholder="Логін" pattern="^[a-zA-Z]+$"
value="<?= isset($_SESSION['post']) ? $_SESSION['post']['login'] : '' ; ?>">
      </div>
      <div class="col-md-4 my-container">
         <input class="form-control" required type="email" name="email"
id="email" minlength="5" maxlength="128" placeholder="Електронна скринька"
value="<?= isset($_SESSION['post']) ? $_SESSION['post']['email'] : '' ; ?>">
      </div>
   </div>
   <div class="row">
      <div class="col-md-4 offset-md-2 my-container">
         <input class="form-control" required type="password" name="password1"
id="password1" minlength="4" maxlength="32" placeholder="Пароль">
      </div>
      <div class="col-md-4 my-container">
```

```
        <input class="form-control" required type="password" name="password2"
id="password2" minlength="4" maxlength="32" placeholder="Підтвердження
паролю">
      </div>
    </div>
    <div class="row">
      <div class="col-md-4 offset-md-2 my-container">
        <input class="form-control" required type="text" name="first_name"
id="first_name" maxlength="128" placeholder="Ім'я" value="<?=
isset($_SESSION['post']) ? $_SESSION['post']['first_name'] : '' ; ?>">
      </div>
      <div class="col-md-4 my-container">
        <input class="form-control" required type="text" name="last_name"
id="last_name" maxlength="128" placeholder="Призвіще" value="<?=
isset($_SESSION['post']) ? $_SESSION['post']['last_name'] : '' ; ?>">
      </div>
    </div>
    <div class="row">
      <div class="col-md-4 offset-md-2 my-container">
        <input class="form-control" type="text" name="patronymic"
id="patronymic" maxlength="128" placeholder="По-батькові" value="<?=
isset($_SESSION['post']) ? $_SESSION['post']['patronymic'] : '' ; ?>">
      </div>
      <div class="col-md-4 my-container">
        <input class="form-control" required type="text" name="birthday"
id="birthday" placeholder="Дата народження" pattern="([12]\d{3}-(0[1-9]|1[0-
2])-(0[1-9]|[12]\d|3[01]))" value="<?= isset($_SESSION['post']) ?
$_SESSION['post']['birthday'] : '' ; ?>">
      </div>
    </div>
    <?php
    if(isset($_GET['fail_login']))
    {
      echo "<div class='row justify-content-center my-container'>
            <div class='col-12 col-md-4' align='center'>
              <span class='text-danger'>Введений логін вже зайнято!</span>
            </div>
          </div>";
    }
    elseif(isset($_GET['fail_email']))
    {
      echo "<div class='row justify-content-center my-container'>
            <div class='col-12 col-md-4' align='center'>
              <span class='text-danger'>Введений e-mail вже зайнято!</span>
```

```
            </div>
        </div>";
    }
    elseif(isset($_GET['fail_password']))
    {
        echo "<div class='row justify-content-center my-container'>
                <div class='col-12 col-md-4' align='center'>
                    <span class='text-danger'>Паролі не співпадають!</span>
                </div>
            </div>";
    }
    session_destroy();
    ?>
    <div class="row justify-content-center">
        <div class="col-md-4 col-xs-12 my-container">
            <button class="btn btn-outline-primary btn-block" type="submit" ><span
class="fa fa-user-circle"></span> Зареєструватись</button>
        </div>
    </div>
</form>
```

Лістинг файлу sign_in_page.php

```
<form action="actions/sign_in_action.php" method="post">
    <div class="row justify-content-center my-container" style="padding-top: 9%">
        <h1 class="display-4">Вхід</h1>
    </div>
    <div class="row justify-content-center">
        <div class="col-12 col-md-5 my-container">
            <input class="form-control" required type="text" name="login" id="login"
minlength="4" maxlength="128" placeholder="Логін користувача">
        </div>
    </div>
    <div class="row justify-content-center">
        <div class="col-12 col-md-5 my-container">
            <input class="form-control" required type="password" name="password"
id="password" minlength="4" maxlength="32" placeholder="Пароль">
        </div>
    </div>
    <?php if(isset($_GET['fail']))
    {
        echo "<div class='row justify-content-center my-container'>
                <div class='col-12 col-md-4' align='center'>
                    <span class='text-danger'>Логін або пароль - невірні! Перевірте
дані.</span>
```

```
            </div>
          </div>";
    } ?>
    <div class="row justify-content-center">
        <div class="col-6 col-md-2 my-container">
            <button class="btn btn-outline-primary btn-block" type="submit" ><span
class="fa fa-sign-in"></span> Увійти</button>
        </div>
    </div>
</form>
```

Лістинг файлу mydatepicker.js

```
$(function() {
    $('#birthday').daterangepicker({
        singleDatePicker: true,
        autoUpdateInput: false,
        showDropdowns: true,
        locale: {
            cancelLabel: 'Clear'
        }
    });
    $('#birthday').on('apply.daterangepicker', function(ev, picker) {
        $(this).val(picker.startDate.format('YYYY-MM-DD'));
    });
});
```

Лістинг файлу myscripts.js

```
function sum_price(id, price){
    var label = document.getElementById("sum_price");
    var div = document.getElementById("div" + id);
    var checkbox = document.getElementsByName(id)[0];
    console.log(label);
    console.log(checkbox);
    if (checkbox.checked){
        label.textContent = parseInt(label.textContent) + parseInt(price);
        div.style.backgroundColor = "#007bff";
    }
    else {
        label.textContent = parseInt(label.textContent) - parseInt(price);
        div.style.backgroundColor = "white";
    }
}
```

Лістинг файлу mystyles.css

```
body {
    padding-bottom: 80px;
}
.my-container {
    margin-top: 10px;
}
.footer {
    position: fixed;
    bottom: 0;
    width: 100%;
    height: 60px;
    line-height: 60px;
    background-color: #f2f2f2;
}
.family {
    font-family: 'Segoe UI';
}
.my-padding {
    padding-top: 1%
}
```