Team 8

Project Name: Book ‘em

Members:

Maryam Ataei Kachooei : PHD CS

Sankalp Vyas: MSCS

Sharvi Ghogale : MSCS

Movie Ticket Booking System Application

Abstract:

The primary goal of "Book ‘em" is to design a cutting-edge Movie Ticket Booking System, transforming the ticket purchasing process for cinemas across various locales. This system aims to streamline the user experience from movie selection to seat booking and payment, facilitating effortless interactions between cinemas and moviegoers. Our approach employs modern web technologies such as HTML, CSS, and JavaScript for front-end development, with Python and Flask managing backend interactions, all integrated with an Oracle database. This architecture ensures robust, seamless data flow and user interface responsiveness. Key accomplishments include the deployment of a highly functional user interface and the establishment of a comprehensive database schema directly connected to the UI.

Keywords:

Online Booking, Movie Ticketing, Movie Reservations, Movie Hall Seating, Movie Timings, Booking System, Customer Engagement, Ticket Sales, Event Access, Secure Payments, Entertainment Planning, Audience Experience, Movie Recommendation, Refund Processing, Movie Rating.

OVERVIEW:

"Book ‘em" is dedicated to developing a comprehensive database application that simplifies the movie ticket booking process. By leveraging a relational database, the system meticulously organizes and manages data related to movie schedules, theater capacities, user registrations, bookings, and financial transactions. This project responds to the increasing demand for digital solutions in the entertainment sector, with the online ticketing market projected to reach significant growth. Our system is designed to offer convenience and efficiency, enhancing the movie-going experience and meeting the modern consumer’s expectations for quick and easy access to entertainment.

The significance of this project stems from the increasing demand for digital solutions in the entertainment industry. As reported by Statista, the global online ticketing market size is expected to reach USD 67.99 billion by 2025, growing at a CAGR of 4.8% (Statista, 2021). This growth underscores the shift towards online platforms for event bookings, driven by convenience and efficiency. By developing a user-friendly movie ticket booking system, we aim to tap into this growing market, addressing the need for a seamless booking experience that caters to the modern consumer's lifestyle.

APPROACH and DESIGN:

Programming/Database Library to be used in our project:

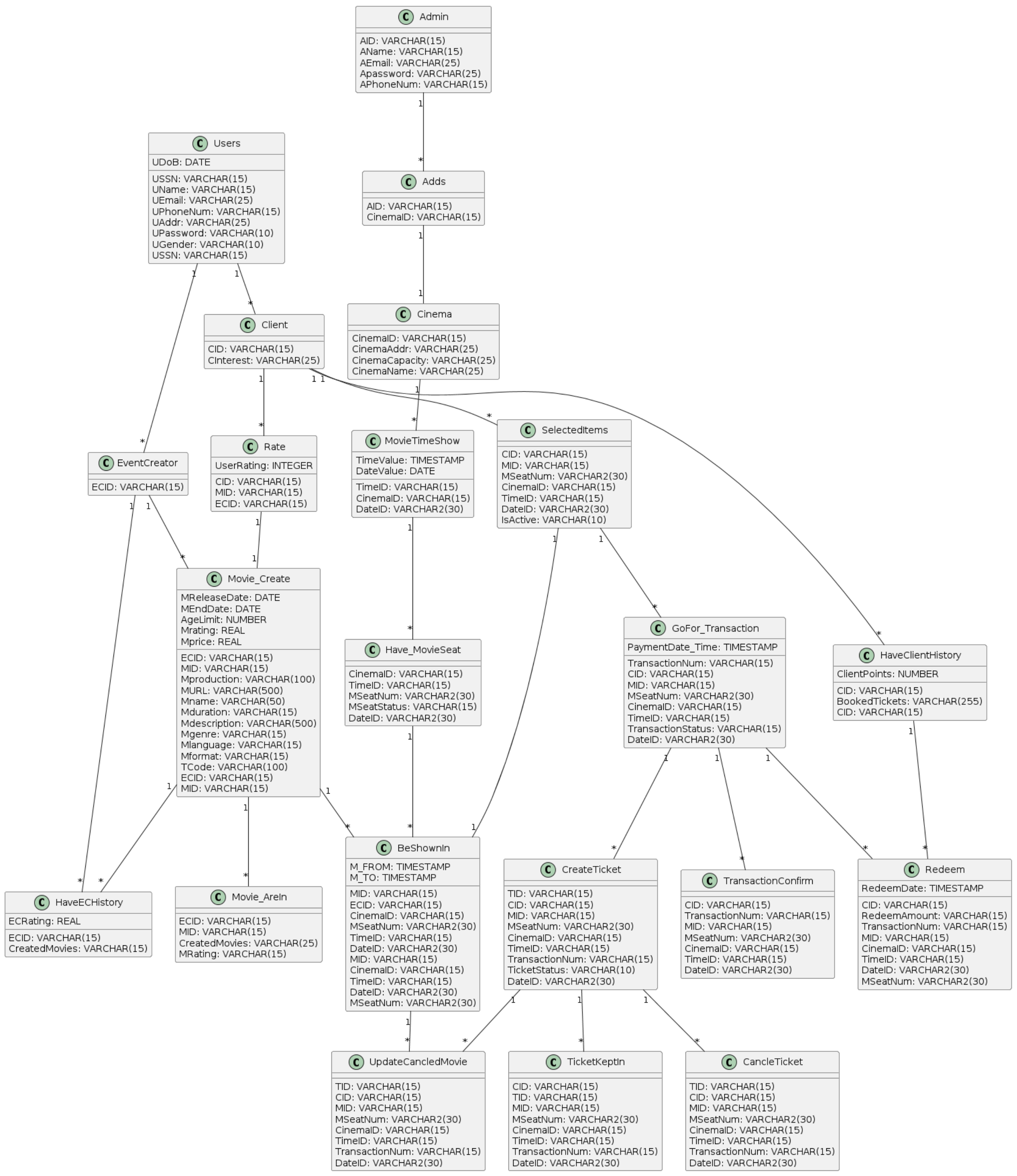
- Languages and Frameworks: HTML, CSS, JavaScript for frontend development; Python and Flask for backend operations.
- Database Management: Oracle is used to manage detailed records of movies, showtimes, user interactions, and transactions.

High Level Solution Process:

- Requirement Analysis: Gather and analyze requirements from stakeholders, including features, functionalities, and user expectations for the ticket booking system. The team will also assess State of the Art papers related to ticket booking systems and look put for different gaps and requirements that would be needed to make the ticket booking system better thereby, developing a roadmap for achieving the desired state.
- Strategy: At this level we will decide the project goals and objectives and develop a project plan.
- Data management: At this stage the team will collect required data to feed into the system, clean the data acquired and consolidate various user requirements from various sources. Doing so will establish data governance policies and procedures to ensure data accuracy, completeness, and security.
- System design and configuration: Design the architecture and system components of the ticket booking system, including database design, user interface design, and backend system design and configuration based on the organization's needs and requirements. The team will configure the system to carry out business requirements and store customer data, automate workflows, and enable reporting and analytics for business purposes.
- Implementation: In this stage, the team develops the ticket booking system software according to the design specifications, using appropriate programming languages, frameworks, and libraries.
- Testing: Conduct various tests to ensure the quality and reliability of the ticket booking system, including unit testing, integration testing, system testing, and user acceptance testing.
- Deployment: Deploy the ticket booking system software to the production environment, ensuring that it is properly configured and accessible to users. They also train end-users on how to use the system and provide ongoing support and maintenance.
- Adoption and optimization: In this phase, the project team monitors the usage and adoption of the ticket booking system and identifies areas for improvement. They also optimize the system's performance by fine-tuning processes, automating workflows, and adding new functionality as needed. Provide ongoing maintenance and support for the ticket booking system, including bug fixes, performance improvements, and updates to meet changing requirements.
- Continuous improvement: In this phase, the project team continues to refine the ticket booking system and processes to ensure that they align with the organization's evolving needs and goals. They also monitor industry trends and best practices to identify opportunities for innovation and improvement.

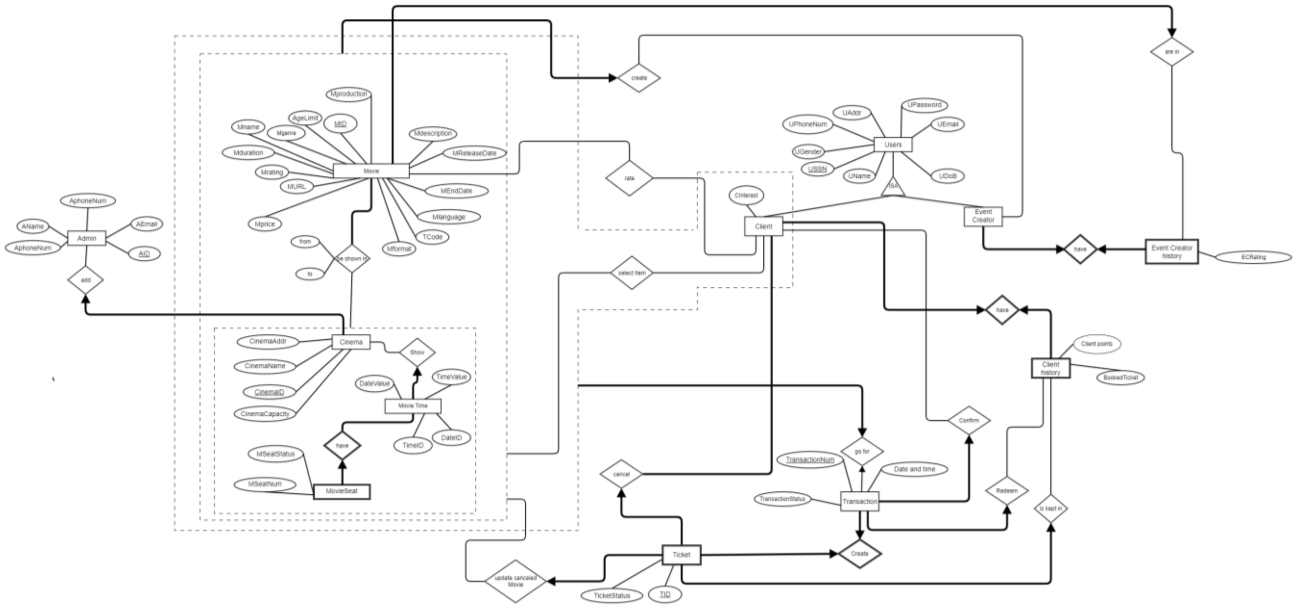
In summary, the high-level solution processes will help in creating a ticket booking system that will enable it to build better relationships with its customers and drive business growth.

UML:



Admin
AID: VARCHAR(15)
AName: VARCHAR(15)
AEmail: VARCHAR(25)
Apassword: VARCHAR(25)
APhoneNum: VARCHAR(15)
Users
UDoB: DATE
USSN: VARCHAR(15)
UName: VARCHAR(15)
UEmail: VARCHAR(25)
UPhoneNum: VARCHAR(15)
UAddr: VARCHAR(25)
UPassword: VARCHAR(10)
UGender: VARCHAR(10)
USSN: VARCHAR(15)
Adds
AID: VARCHAR(15)
CinemaID: VARCHAR(15)
Client
CID: VARCHAR(15)
CInterest: VARCHAR(25)
Cinema
CinemaID: VARCHAR(15)
CinemaAddr: VARCHAR(25)
CinemaCapacity: VARCHAR(25)
CinemaName: VARCHAR(25)
EventCreator
ECID: VARCHAR(15)
Rate
UserRating: INTEGER
CID: VARCHAR(15)
MID: VARCHAR(15)
ECID: VARCHAR(15)
MovieTimeShow
TimeValue: TIMESTAMP
DateValue: DATE
TimeID: VARCHAR(15)
CinemaID: VARCHAR(15)
DateID: VARCHAR2(30)
SelectedItems
CID: VARCHAR(15)
MID: VARCHAR(15)
MSeatNum: VARCHAR2(30)
CinemaID: VARCHAR(15)
TimeID: VARCHAR(15)
DateID: VARCHAR2(30)
IsActive: VARCHAR(10)
Movie_Create
MReleaseDate: DATE
MEndDate: DATE
AgeLimit: NUMBER
Mrating: REAL
Mprice: REAL
ECID: VARCHAR(15)
MID: VARCHAR(15)
Mproduction: VARCHAR(100)
MURL: VARCHAR(500)
Mname: VARCHAR(50)
Mduration: VARCHAR(15)
Mdescription: VARCHAR(500)
Mgenre: VARCHAR(15)
Mlanguage: VARCHAR(15)
Mformat: VARCHAR(15)
TCode: VARCHAR(100)
ECID: VARCHAR(15)
MID: VARCHAR(15)
Have_MovieSeat
CinemaID: VARCHAR(15)
TimeID: VARCHAR(15)
MSeatNum: VARCHAR2(30)
MSeatStatus: VARCHAR(15)
DateID: VARCHAR2(30)
GoFor_Transaction
PaymentDate_Time: TIMESTAMP
TransactionNum: VARCHAR(15)
CID: VARCHAR(15)
MID: VARCHAR(15)
MSeatNum: VARCHAR2(30)
CinemaID: VARCHAR(15)
TimeID: VARCHAR(15)
TransactionStatus: VARCHAR(15)
DateID: VARCHAR2(30)
HaveClientHistory
ClientPoints: NUMBER
CID: VARCHAR(15)
BookedTickets: VARCHAR(255)
CID: VARCHAR(15)
BeShownIn
M_FROM: TIMESTAMP
M_TO: TIMESTAMP
MID: VARCHAR(15)
ECID: VARCHAR(15)
CinemaID: VARCHAR(15)
MSeatNum: VARCHAR2(30)
TimeID: VARCHAR(15)
DateID: VARCHAR2(30)
MID: VARCHAR(15)
CinemaID: VARCHAR(15)
TimeID: VARCHAR(15)
DateID: VARCHAR2(30)
MSeatNum: VARCHAR2(30)
HaveECHistory
ECRating: REAL
ECID: VARCHAR(15)
CreatedMovies: VARCHAR(15)
Movie_AreIn
ECID: VARCHAR(15)
MID: VARCHAR(15)
CreatedMovies: VARCHAR(25)
MRating: VARCHAR(15)
CreateTicket
TID: VARCHAR(15)
CID: VARCHAR(15)
MID: VARCHAR(15)
MSeatNum: VARCHAR2(30)
CinemaID: VARCHAR(15)
TimeID: VARCHAR(15)
TransactionNum: VARCHAR(15)
TicketStatus: VARCHAR(10)
DateID: VARCHAR2(30)
TransactionConfirm
CID: VARCHAR(15)
TransactionNum: VARCHAR(15)
MID: VARCHAR(15)
MSeatNum: VARCHAR2(30)
CinemaID: VARCHAR(15)
TimeID: VARCHAR(15)
DateID: VARCHAR2(30)
Redeem
RedeemDate: TIMESTAMP
CID: VARCHAR(15)
RedeemAmount: VARCHAR(15)
TransactionNum: VARCHAR(15)
MID: VARCHAR(15)
CinemaID: VARCHAR(15)
TimeID: VARCHAR(15)
DateID: VARCHAR2(30)
MSeatNum: VARCHAR2(30)
UpdateCancledMovie
TID: VARCHAR(15)
CID: VARCHAR(15)
MID: VARCHAR(15)
MSeatNum: VARCHAR2(30)
CinemaID: VARCHAR(15)
TimeID: VARCHAR(15)
TransactionNum: VARCHAR(15)
DateID: VARCHAR2(30)
TicketKeptIn
CID: VARCHAR(15)
TID: VARCHAR(15)
MID: VARCHAR(15)
MSeatNum: VARCHAR2(30)
CinemaID: VARCHAR(15)
TimeID: VARCHAR(15)
TransactionNum: VARCHAR(15)
DateID: VARCHAR2(30)
CancleTicket
TID: VARCHAR(15)
CID: VARCHAR(15)
MID: VARCHAR(15)
MSeatNum: VARCHAR2(30)
CinemaID: VARCHAR(15)
TimeID: VARCHAR(15)
TransactionNum: VARCHAR(15)
DateID: VARCHAR2(30)

ERD:



User Requirements of Data to be Modeled:

The Movie Ticket Booking System must handle several complex relationships and store a wide range of data in a structured manner. The user data will consist of detailed profiles including personal contact information and preferences. Additionally, there needs to be a hierarchy within users, allowing for distinct permissions and capabilities for admins and general users. The admin users, responsible for event creation and management, should have the ability to input comprehensive movie details into the system. This includes not only basic movie information such as name, release date, show timing but also extensive details like venue capacities, pricing, production house, movie rating and unique identifiers for both movies and theatre.

In addition to movie and user data, the system will manage transactional data, tracking the entire lifecycle of a booking from initial reservation through to payment processing.

User Queries for Functional Interaction:

1. Movie Discovery:

   – "Show me upcoming Movies in [City/Location] for [Date/Next Month]."
   – "List all movies by [Movie Name] available for booking."

2. Detailed Movie Information:

   – "Provide details for the movies [Movie Name], including venue capacity and ticket pricing."
   – "Display available seating and pricing for [Movie Name] on [Date]."

3. User Account and Registration:

   – "Register a new user with the following details [Email, Password]."
   – "Log in user with [Email] and [Password]."

4. Profile Management:

- "Update account information for user [UserID] with new contact details."
- "Change password for user [UserID]."

5. Ticket Booking and Management:

- "Book [Number] tickets for [Movie Name] with seats [Seat Numbers]."
- "Cancel booking [Booking ID] and process a refund if applicable."

6. Payment and Checkout:

- "Process payment for [Total Amount] using [Payment Method] for [User ID]."
- "Calculate discount points [Client Points] based on [Total Amount] and update for [User ID]
- "Redeem [Client Points] for [Movie ID] to get a discount"

7. Cancellation and Refunds:

- "Cancel booking [Booking ID] and initiate a refund process if applicable."
- "Calculate and adjust loyalty points based on the cancellation of booking [Booking ID]"
- "Revert seat booking status to available for booked seats from booking [Booking ID]."

8. Booking History:

- "Retrieve booking history for user [UserID]."
- "Show details of the last transaction for user [UserID]."

9. Event Creation and Management (Admin):

- "Create a new event with [Event Details] in the admin panel."
- "Update event [Event ID] with new date/time [Date/Time]."

10. Venue and Seating Management:

- "Update seating configuration for venue [Venue ID]."
- "Check seat availability for [Event Name] in real-time."

Relational Schema:

Table Name	Columns	Primary Key	Foreign Keys
Admin	AID (VARCHAR), AName (VARCHAR), AEmail (VARCHAR), APassword (VARCHAR), APhoneNum (VARCHAR)	AID	
Users	USSN (VARCHAR), UName (VARCHAR), UEmail (VARCHAR), UPhoneNum (VARCHAR), UAddr (VARCHAR), UPassword (VARCHAR), UGender (VARCHAR), UDoB (DATE)	USSN	

Adds	AID (VARCHAR), CinemaID (VARCHAR)	AID, CinemaID	AID -> Admin, CinemaID -> Cinema
EventCreator	ECID (VARCHAR)	ECID	ECID -> Users
Movie_Create	ECID (VARCHAR), MID (VARCHAR), Mproduction (VARCHAR), MReleaseDate (DATE), MEndDate (DATE), MURL (VARCHAR), Mname (VARCHAR), AgeLimit (NUMBER), Mduration (VARCHAR), Mdescription (VARCHAR), Mrating (REAL), Mprice (REAL), Mgenre (VARCHAR), Mlanguage (VARCHAR), Mformat (VARCHAR), TCode (VARCHAR)	ECID, MID	ECID -> EventCreator
HaveECHistory	ECID (VARCHAR), MID (VARCHAR), ECRating (REAL)	ECID, MID	ECID, MID -> Movie_Create
Cinema	CinemaID (VARCHAR), CinemaAddr (VARCHAR), CinemaCapacity (VARCHAR), CinemaName (VARCHAR)	CinemaID	
MovieTimeShow	TimeID (VARCHAR), CinemaID (VARCHAR), TimeValue (TIMESTAMP), DateID (VARCHAR), DateValue (DATE)	CinemaID, TimeID, DateID	CinemaID -> Cinema
Have_MovieSeat	CinemaID (VARCHAR), TimeID (VARCHAR), MSeatNum (VARCHAR), MSeatStatus (VARCHAR), DateID (VARCHAR)	CinemaID, TimeID, DateID, MSeatNum	CinemaID, TimeID, DateID -> MovieTimeShow
BeShownIn	MID (VARCHAR), ECID (VARCHAR), CinemaID (VARCHAR), MSeatNum (VARCHAR), TimeID (VARCHAR), M_FROM (TIMESTAMP), M_TO (TIMESTAMP), DateID (VARCHAR)	MID, CinemaID, TimeID, DateID, MSeatNum	ECID, MID -> Movie_Create, CinemaID, TimeID, DateID, MSeatNum -> Have_MovieSeat
Movie_AreIn	ECID (VARCHAR), MID (VARCHAR), CreatedMovies	MID, ECID, CreatedMovies	ECID, MID -> Movie_Create

	(VARCHAR), MRating (VARCHAR)		
Client	CID (VARCHAR), CInterest (VARCHAR)	CID	CID -> Users
Recommend	CID (VARCHAR), MID (VARCHAR), ECID (VARCHAR)	CID, MID	CID -> Client, ECID, MID -> Movie_Create
Rate	CID (VARCHAR), MID (VARCHAR), ECID (VARCHAR), UserRating (INTEGER)	CID, MID	CID -> Client, ECID, MID -> Movie_Create
HaveClientHistory	CID (VARCHAR), BookedTickets (VARCHAR), ClientPoints (NUMBER)	CID	CID -> Client
SelectedItems	CID (VARCHAR), MID (VARCHAR), MSeatNum (VARCHAR), CinemaID (VARCHAR), TimeID (VARCHAR), DateID (VARCHAR), IsActive (VARCHAR)	CID, MID, CinemaID, TimeID, DateID, MSeatNum	CID -> Client, MID, CinemaID, TimeID, DateID, MSeatNum -> BeShownIn
GoFor_Transaction	TransactionNum (VARCHAR), CID (VARCHAR), MID (VARCHAR), MSeatNum (VARCHAR), CinemaID (VARCHAR), TimeID (VARCHAR), PaymentDate_Time (TIMESTAMP), TransactionStatus (VARCHAR), DateID (VARCHAR)	TransactionNum, CID, MID, CinemaID, TimeID, DateID, MSeatNum	CID, MID, CinemaID, TimeID, DateID, MSeatNum -> SelectedItems
TransactionConfirm	CID (VARCHAR), TransactionNum (VARCHAR), MID (VARCHAR), MSeatNum (VARCHAR), CinemaID (VARCHAR), TimeID (VARCHAR), DateID (VARCHAR)	TransactionNum, CID, MID, CinemaID, TimeID, DateID, MSeatNum	CID -> Client, TransactionNum, CID, MID, CinemaID, TimeID, DateID, MSeatNum -> GoFor_Transaction
CreateTicket	TID (VARCHAR), CID (VARCHAR), MID (VARCHAR), MSeatNum (VARCHAR), CinemaID (VARCHAR), TimeID (VARCHAR), TransactionNum (VARCHAR), TicketStatus	TID, TransactionNum, CID, MID, CinemaID, TimeID, DateID, MSeatNum	TransactionNum, CID, MID, CinemaID, TimeID, DateID, MSeatNum -> GoFor_Transaction

	(VARCHAR), DateID (VARCHAR)		
CancleTicket	TID (VARCHAR), CID (VARCHAR), MID (VARCHAR), MSeatNum (VARCHAR), CinemaID (VARCHAR), TimeID (VARCHAR), TransactionNum (VARCHAR), DateID (VARCHAR)	TID, CID, MID, CinemaID, TimeID, DateID, MSeatNum	CID -> Client, TID, TransactionNum, CID, MID, CinemaID, TimeID, DateID, MSeatNum -> CreateTicket
TicketKeptIn	CID (VARCHAR), TID (VARCHAR), MID (VARCHAR), MSeatNum (VARCHAR), CinemaID (VARCHAR), TimeID (VARCHAR), TransactionNum (VARCHAR), DateID (VARCHAR)	TID, CID, MID, CinemaID, TimeID, DateID, MSeatNum	CID -> Client, TID, TransactionNum, CID, MID, CinemaID, TimeID, DateID, MSeatNum -> CreateTicket
UpdateCancledMovie	TID (VARCHAR), CID (VARCHAR), MID (VARCHAR), MSeatNum (VARCHAR), CinemaID (VARCHAR), TimeID (VARCHAR), TransactionNum (VARCHAR), DateID (VARCHAR)	TID, MID, CinemaID, TimeID, DateID, MSeatNum	MID, CinemaID, TimeID, DateID, MSeatNum -> BeShownIn, TID, TransactionNum, CID, MID, CinemaID, TimeID, DateID, MSeatNum -> CreateTicket
Redeem	CID (VARCHAR), RedeemAmount (VARCHAR), RedeemDate (TIMESTAMP), TransactionNum (VARCHAR), MID (VARCHAR), CinemaID (VARCHAR), TimeID (VARCHAR), DateID (VARCHAR), MSeatNum (VARCHAR)	TransactionNum, CID	CID -> HaveClientHistory, TransactionNum, CID, MID, CinemaID, TimeID, DateID, MSeatNum -> GoFor_Transaction

Problems the team encountered:

The transition from a general event booking to a movie-specific ticketing system presented unique challenges, particularly in database schema refinement and integration of real-time data feeds for movie showtimes and availability. Solutions involved iterative testing and redesigning of database queries and user interface elements to improve performance and user satisfaction.

We found it difficult to create tables for our database because our ER diagram needed changes. We had to carry out brainstorming sessions before we had the ER diagram we desired. We had to change the ER diagram twice to create proper tables for our database.

Designing a User-friendly UI was also a challenge when creating the seats page for the movie/concert hall, selecting seats and carrying out the total cost of those seats parallel. But we tackled the problem buy siting different resources to make it effective.

Another problem we faced is connecting the database to the UI. Since we are using the WPI oracle database for table creation, we were unable to connect it to the UI using node.js. So, we implemented connectivity using Flask which largely streamlined the process.

VALIDATION to your approach:

Our core Functionalities of the product includes:

- The user can easily login and register to the application.
- Can view upcoming, current and recommended movies.
- User friendly UI.
- Our project gives recommendations without using an AI tool and only with the help of our database data.
- The user receives confirmation and booking details for their show after confirming the transaction.
- User can redeem points to get discounts.
- User friendly easy to understand cancellation and refund policy.
- Reverting the points and seat to original state after cancellation.
- The Event Creator can easily add upcoming movies to the application.

All these validation cases are now working.

Our project timeline has now been fully realized, with both the database and the user interface (UI) effectively integrated. We have successfully implemented a user-friendly UI, and all database interactions—such as SELECT, INSERT, DELETE, and UPDATE operations—are now fully functional and connected to the UI pages. This marks the completion of all planned project components

Test Case:

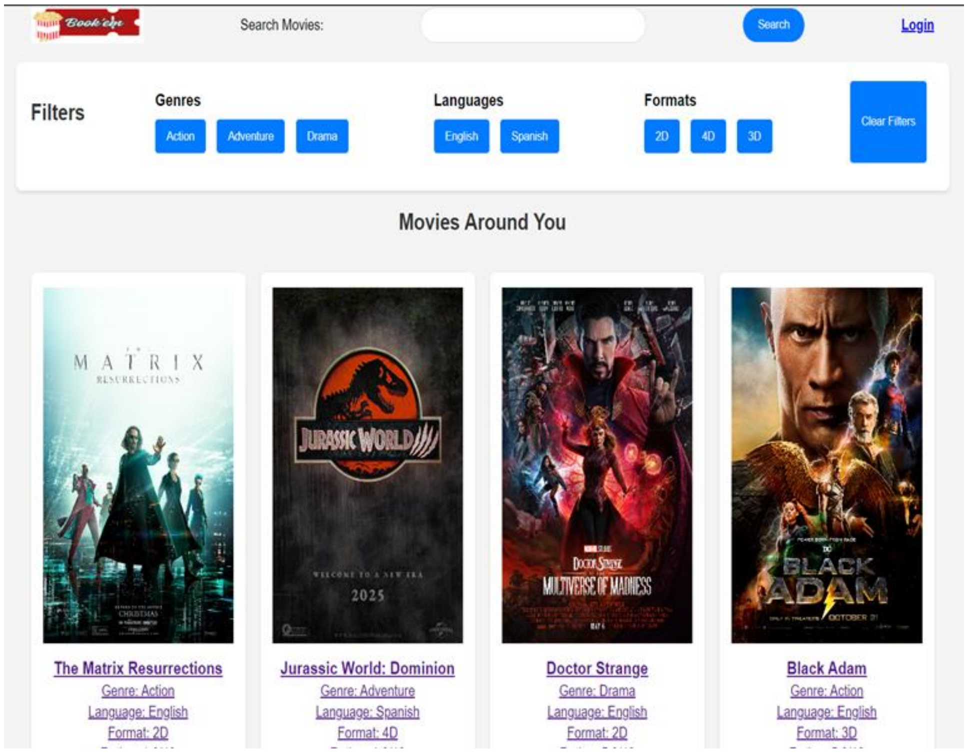
ID	Test Cases	Input Value	Expected output	Pass/Fail	Date Tested

1.	Registration and Login Success	Email: ysr@gmail.com Email: ysr@gmial.co ID: jjohn Pwd: huser@1 ID: ijohn Pwd: huser@1 ID: jjohn Pwd: huss@1	Successful Registration Invalid email ID Successful Registration/login Username/Password doesn't match Username/Password doesn't match	Pass	01/12/24
2.	Movie Display Accuracy	Date: MM/DD/YYYY, Time: hrs: min, Duration of movie: hrs, Cost of ticket, Location, Capacity, Production house, Description, Rating, Age limit	Accurate display of movies with correct details	Pass	01/15/24
3.	Seat Selection Functionality	User selects white seats User selects grey seats User picks seat green	Those white seats are available for the user to book Already booked seat Picked by the user	Pass Pass Pass	02/18/24
4.	Booking Process Validation	Name of the movie, Username, Booking ID Seat number(s), Date,	Correct display of all the booking details	Pass	01/15/24

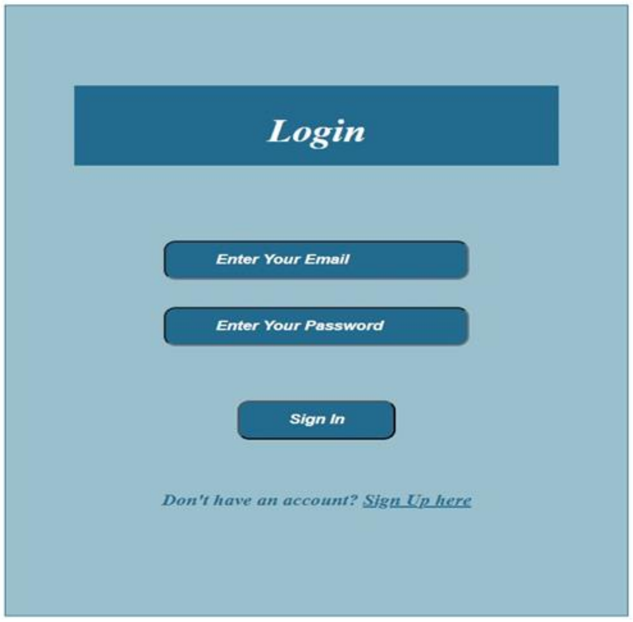
		Location			
5.	Payment Processing	User Card details, Amount to be paid, Requesting user transfer amount from their bank, Amount removed from the user's bank, Amount received to the application bank Requesting user transfer amount from their bank, Amount not received	Secure and successful transaction Transaction unsuccessful please try again	Pass	03/02/24
6.	Redeem Points for Discount	Client Points: 150, Movie ID: 3001	Discount applied for movie ID 3001 using 150 client points. Redemption failed due to insufficient points or invalid movie ID.	Pass	03/11/24
7.	Cancellation and Refund Process	Booking cancellation confirmation from user User tries to cancel the ticket after the end date	Successful cancellation and refund if applicable Sorry cannot Cancel/Refund after “date”	Pass	03/15/24

Screenshots of the UI and Database Covered:

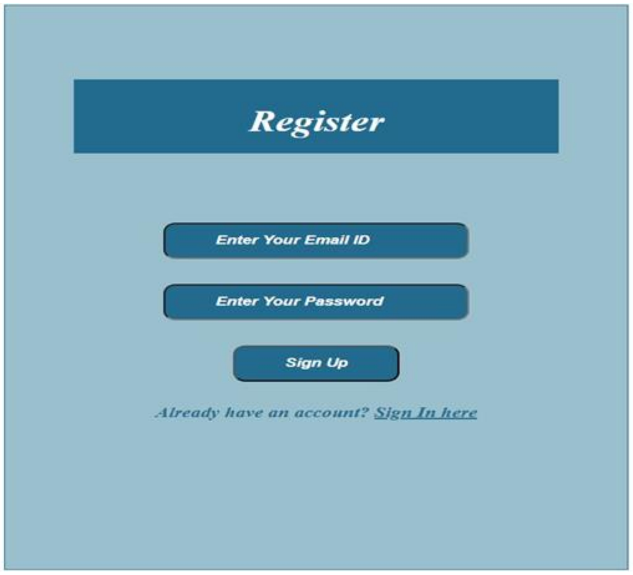
1. Front page of the application



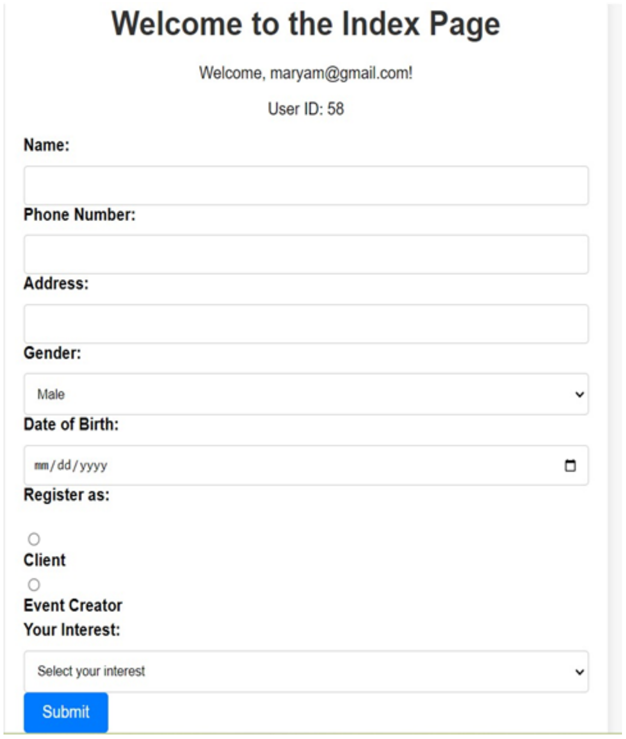
2. Login Page



3. Create Account Page



Users can add their details


Welcome to the Index Page

Welcome, maryam@gmail.com!

User ID: 58

Name:

Phone Number:

Address:

Gender: Male

Date of Birth: mm/dd/yyyy

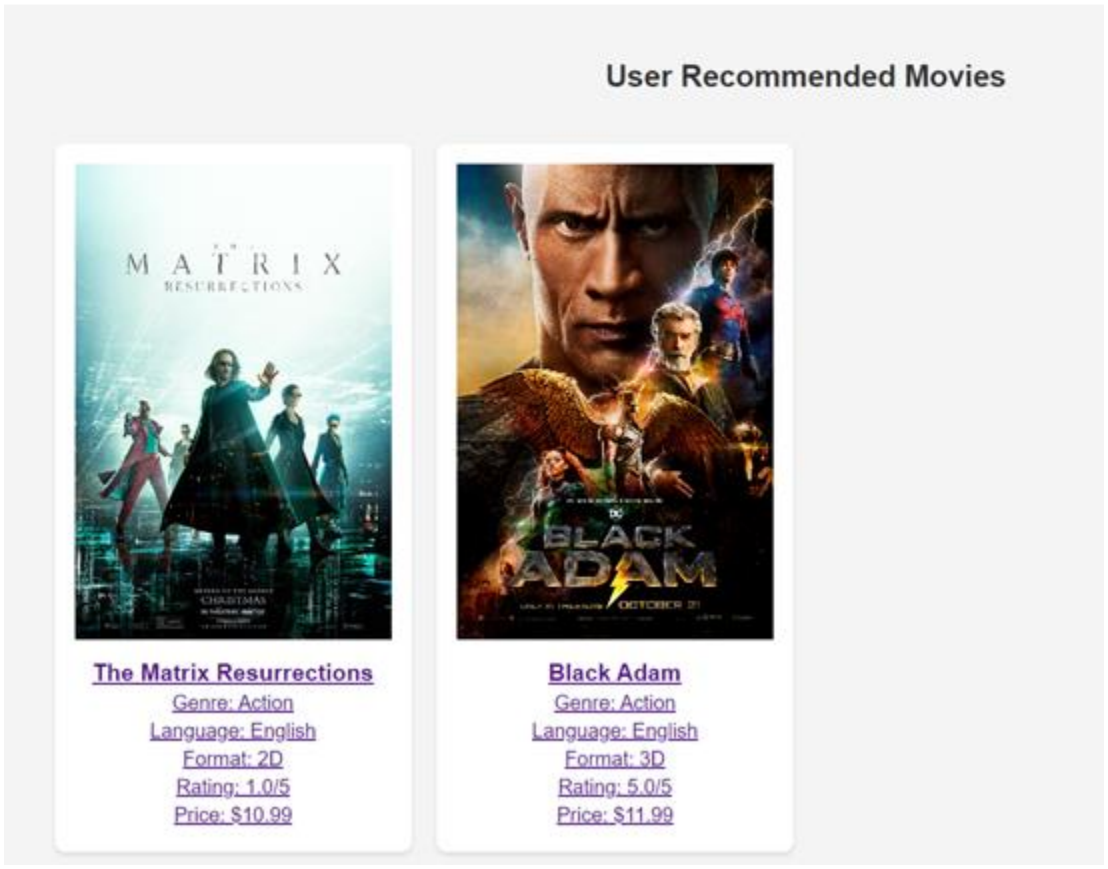
Register as:

○ Client

○ Event Creator

Your Interest: Select your interest

4. Recommended movies section for User:

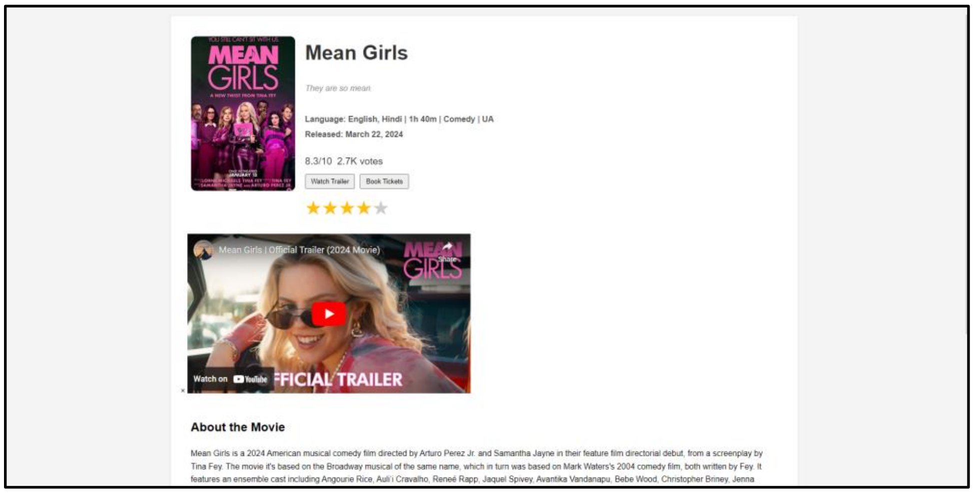


User Recommended Movies

The Matrix Resurrections
Genre: Action
Language: English
Format: 2D
Rating: 1.0/5
Price: $10.99

Black Adam
Genre: Action
Language: English
Format: 3D
Rating: 5.0/5
Price: $11.99

5. Movie details pages



Mean Girls

They are so mean

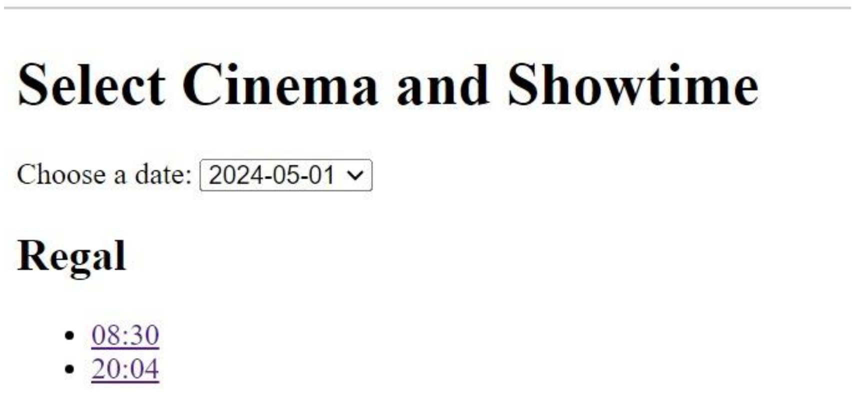
Language: English, Hindi | 1h 40m | Comedy | UA
Released: March 22, 2024

8.3/10 2.7K votes

About the Movie

Mean Girls is a 2024 American musical comedy film directed by Arturo Perez Jr. and Samantha Jayne in their feature film directorial debut, from a screenplay by Tina Fey. The movie it's based on the Broadway musical of the same name, which in turn was based on Mark Waters's 2004 comedy film, both written by Fey. It features an ensemble cast including Angourie Rice, Auli'i Cravalho, Reneé Rapp, Jaquel Spivey, Avantika Vandanapu, Bebe Wood, Christopher Briney, Jenna

6. Page to select the cinema and timings to book seats

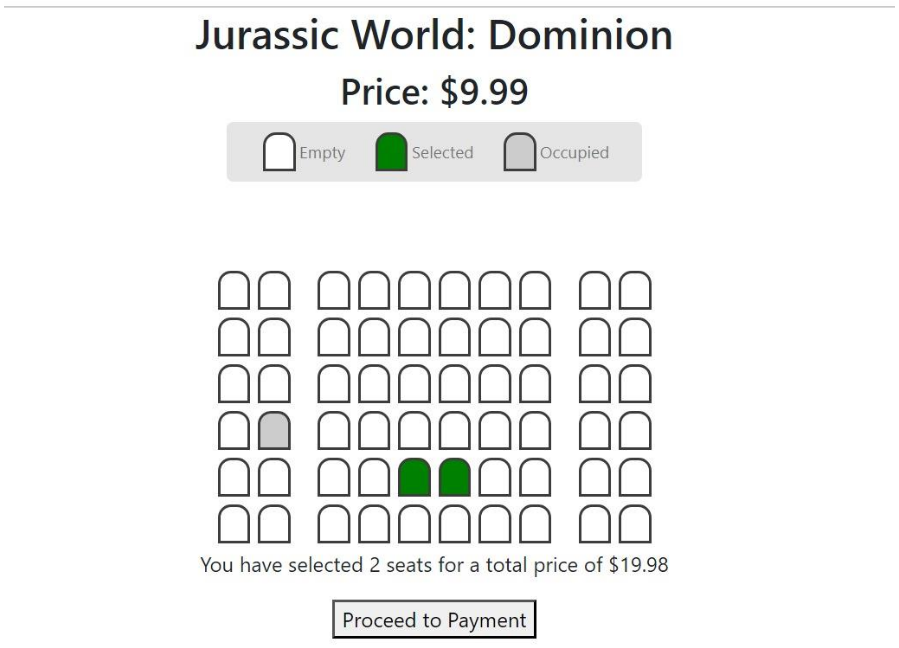

Select Cinema and Showtime

Choose a date: 2024-05-01

Regal

- 08:30
- 20:04

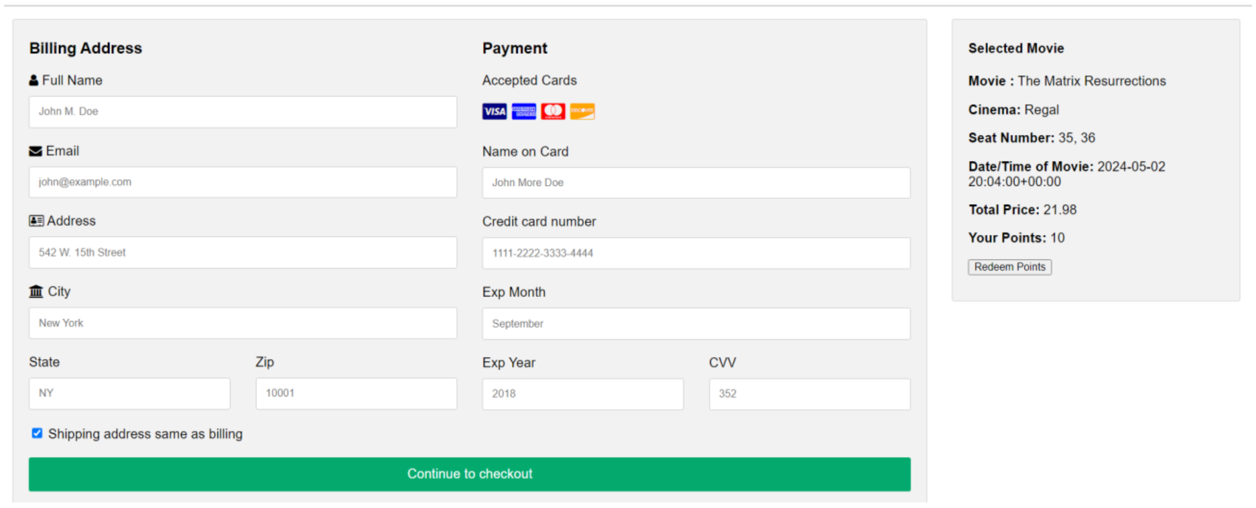
7. Hall View for seat booking page


Jurassic World: Dominion

Price: $9.99

You have selected 2 seats for a total price of $19.98

8. Payment Page



Billing Address

Full Name: John M. Doe

Email: john@example.com

Address: 542 W. 15th Street

City: New York

State: NY

Zip: 10001

☑ Shipping address same as billing

Payment

Accepted Cards

Name on Card: John More Doe

Credit card number: 1111-2222-3333-4444

Exp Month: September

Exp Year: 2018

CVV: 352

Selected Movie

Movie : The Matrix Resurrections

Cinema: Regal

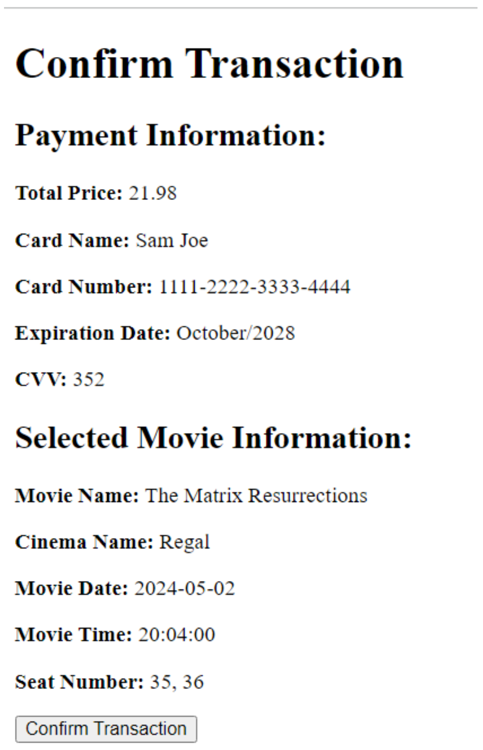
Seat Number: 35, 36

Date/Time of Movie: 2024-05-02 20:04:00+00:00

Total Price: 21.98

Your Points: 10

9. Ticket Confirmation page and Confiticket page



Confirm Transaction

Payment Information:

Total Price: 21.98

Card Name: Sam Joe

Card Number: 1111-2222-3333-4444

Expiration Date: October/2028

CVV: 352

Selected Movie Information:

Movie Name: The Matrix Resurrections

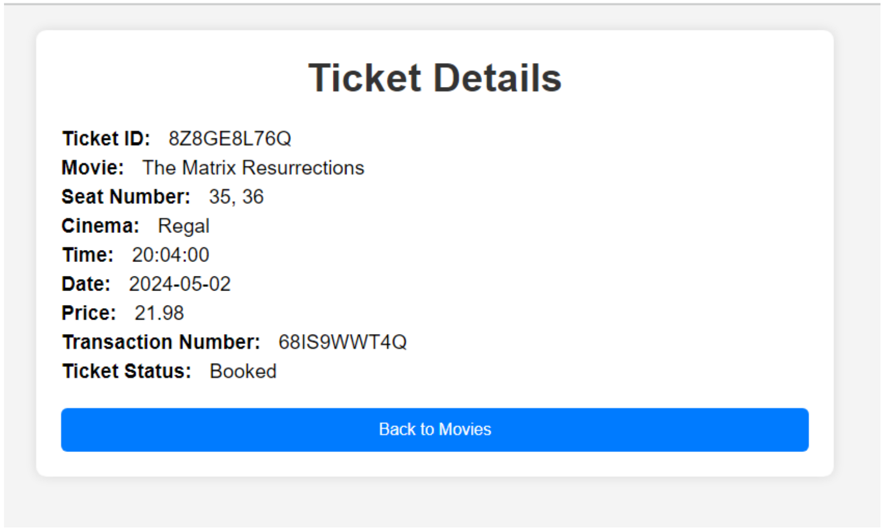
Cinema Name: Regal

Movie Date: 2024-05-02

Movie Time: 20:04:00

Seat Number: 35, 36

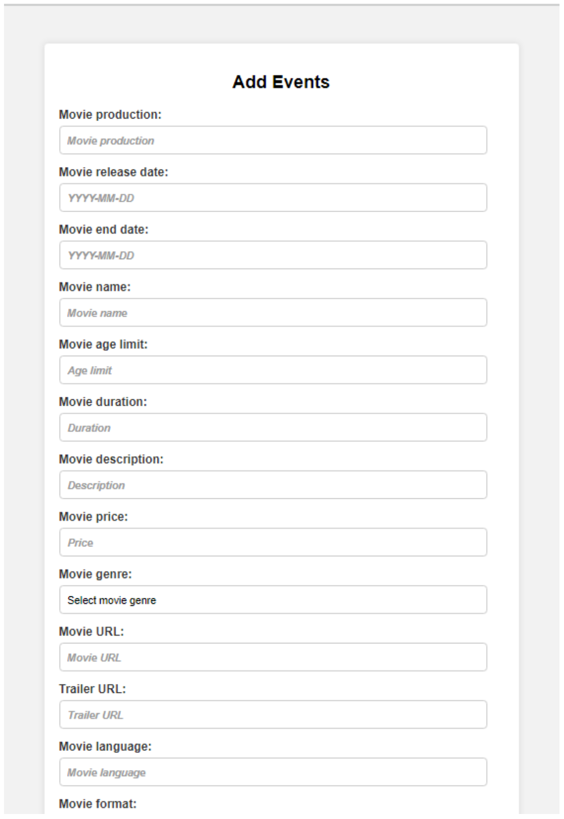
10. Ticket Page



Ticket Details

Ticket ID: 8Z8GE8L76Q
Movie: The Matrix Resurrections
Seat Number: 35, 36
Cinema: Regal
Time: 20:04:00
Date: 2024-05-02
Price: 21.98
Transaction Number: 68IS9WWT4Q
Ticket Status: Booked

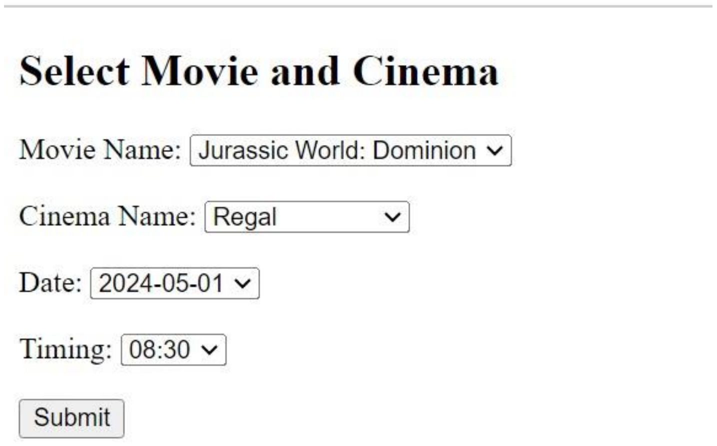
11. Add new upcoming movies on the application – done by Event creator



Add Events

Movie production:	Movie production
Movie release date:	YYYY-MM-DD
Movie end date:	YYYY-MM-DD
Movie name:	Movie name
Movie age limit:	Age limit
Movie duration:	Duration
Movie description:	Description
Movie price:	Price
Movie genre:	Select movie genre
Movie URL:	Movie URL
Trailer URL:	Trailer URL
Movie language:	Movie language
Movie format:	

12. The event creator adds movies to the desired cinema and the desired time


Select Movie and Cinema

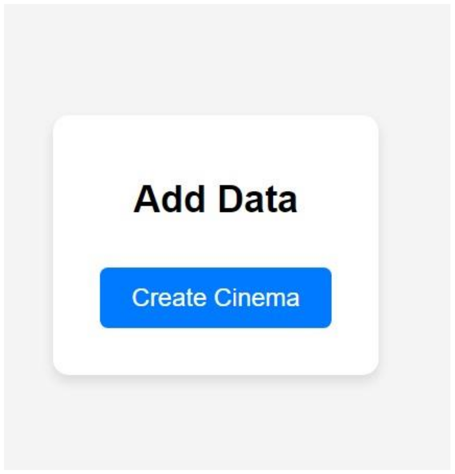
Movie Name: Jurassic World: Dominion

Cinema Name: Regal

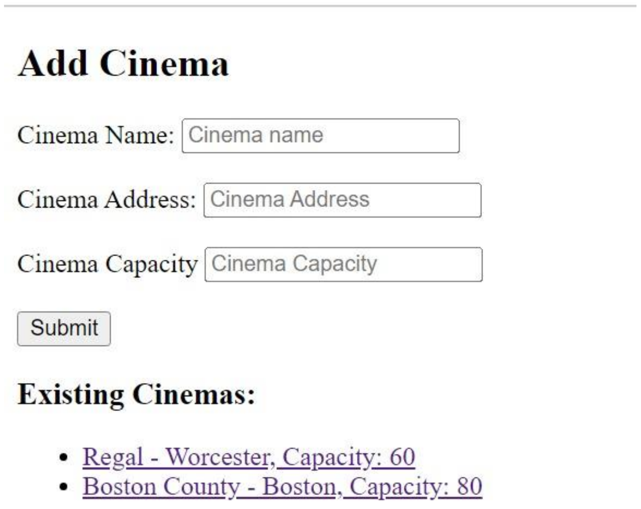
Date: 2024-05-01

Timing: 08:30

13. Admin task



14. Admin add new cinemas


Add Cinema

Cinema Name: Cinema name

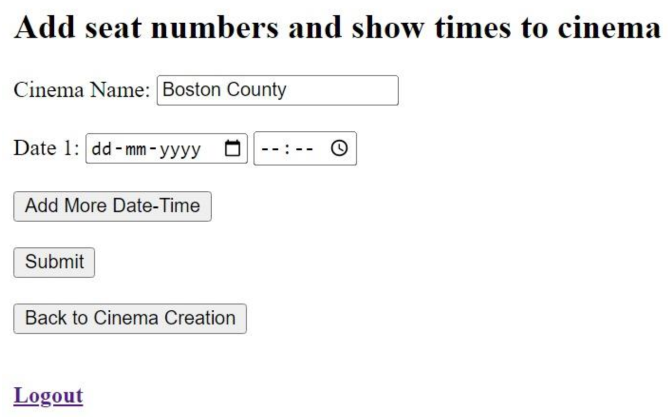
Cinema Address: Cinema Address

Cinema Capacity Cinema Capacity

Existing Cinemas:

- Regal - Worcester, Capacity: 60
- Boston County - Boston, Capacity: 80

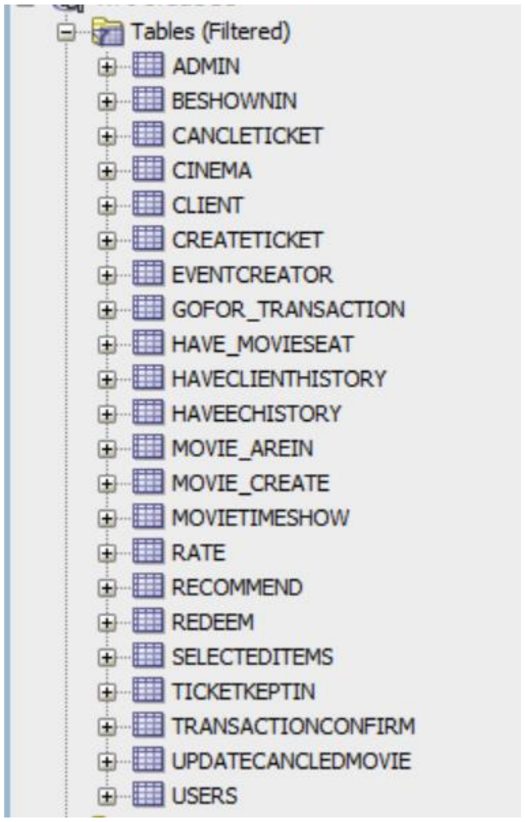
15. Admin adds dates and timings to the cinema


Add seat numbers and show times to cinema

Cinema Name: Boston County

Date 1: dd-mm-yyyy --:--

16. Database Tables:


Tables (Filtered)

ADMIN
BESHOWNIN
CANCLETICKET
CINEMA
CLIENT
CREATETICKET
EVENTCREATOR
GOFOR_TRANSACTION
HAVE_MOVIESEAT
HAVECLIENTHISTORY
HAVEECHISTORY
MOVIE_AREIN
MOVIE_CREATE
MOVIETIMESHOW
RATE
RECOMMEND
REDEEM
SELECTEDITEMS
TICKETKEPTIN
TRANSACTIONCONFIRM
UPDATECANCLEDMOVIE
USERS

LESSONS LEARNED:

Throughout our group project, we navigated several complex challenges and acquired substantial practical skills in database and application development. Our initial challenge was creating an efficient

Entity-Relationship (ER) diagram, which is pivotal for database integrity. This process required multiple revisions to ensure our database tables were properly structured to support our application's requirements. We also gained experience in integrating user interface pages with our back-end systems using Flask, which was a new tool for many team members. Additionally, we learned to implement AJAX for real-time data interactions, particularly in the dynamic seating selection features of our system. This project significantly improved our collaborative and technical skills, highlighting the importance of adaptability and continuous learning in software development.

CONCLUSIONS:

Our project's main goal was to deliver a comprehensive and user-friendly movie ticket booking system, which we achieved by completing all planned functionalities. This included user registration, movie searching, ticket booking, and admin event management, all interfaced through a well-designed and responsive front-end. We adhered closely to our project timeline and successfully integrated the database with the user interface, allowing for seamless data management and operations. However, some advanced features like real-time analytics for movie popularity and advanced personalized recommendations were not completed. These features were part of our initial ambitious goals but remained outside the scope of this semester's work due to time constraints.

FUTURE WORK:

For future enhancement of our system, we propose several initiatives. First, implementing personalized movie recommendations using machine learning could significantly enhance user engagement and satisfaction. Additionally, integrating a dynamic pricing model would optimize revenue management for theaters by adjusting prices based on real-time demand. Another critical area for development is the improvement of the system's scalability and security features to support a larger user base and protect sensitive user data effectively. Finally, gathering and incorporating user feedback will be essential to refine functionality and user interface design, ensuring the system remains aligned with user needs and preferences. These steps will be vital in evolving our movie ticket booking system into a more robust and user-centered platform.

Task Table:

Admin Side		Status	User Side		Status
Task ID	Task Description		Task ID	Task Description	
1	Create a table to store movies with columns for event ID, name, description, date, time, duration, capacity, limitation, and category.	Done	1	Design a user interface for account registration and login.	Done

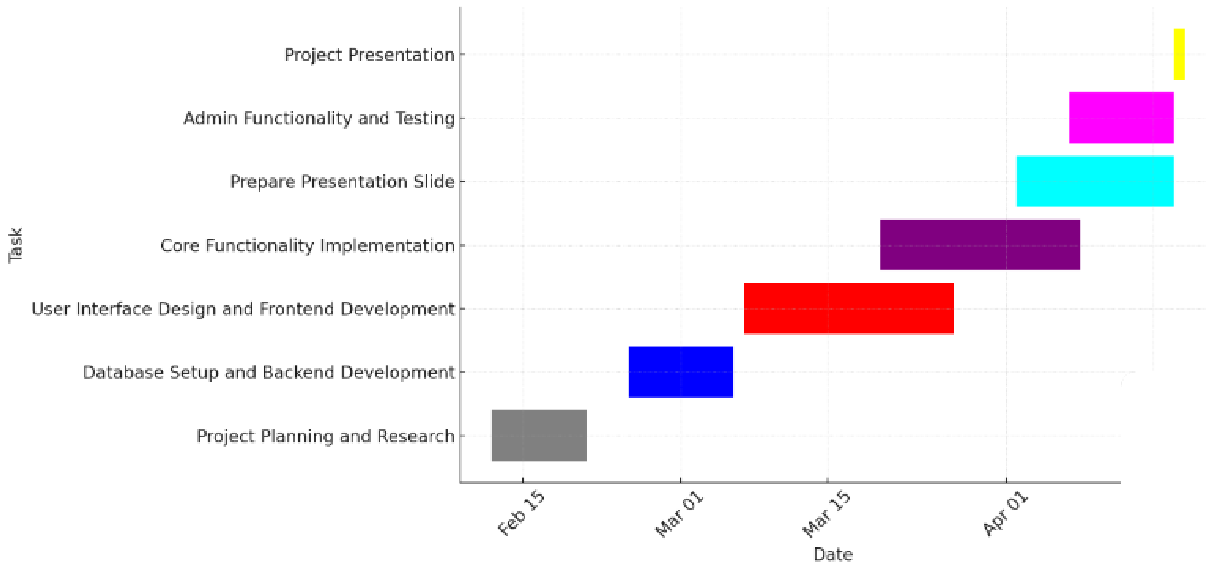
2	Develop a table for user details including		2	Create a user dashboard for users to manage their profile, preferences, and bookings.	Done
3	user ID, name, email, phone number, age, interests, and account details.	Done	3	Develop functionality for users to search and view movies based on upcoming dates, current dates and interests.	Done
4	Design a table for ratings with fields for event ID, user ID, and rating value.	Done	4	Implement a feature for users to book tickets for movies.	Done
5	Create a table for payment transactions with columns for transaction ID, user ID, event ID, amount, and timestamp.	Done	5	Develop a system for users to view and provide ratings for watched movies	Done
7	Establish relationships between tables for movies, users, ratings, and payments.	Done	6	Develop a system for users to view their booking history and transactions.	Done
8	Implement functionality to select, add, edit, and delete movies from the database.	Done	7	Calculate movie rating based on the ratings given by all the users for a movie.	Done
			8	Create an interface for users to view trailer before booking.	Done
			9	Create a reward system, where users can redeem points to get discounts.	Done

PLAN AND SCHEDULE FOR THE PROJECT :

Timeline:

Task	**Start Date**	**Due Date**
Project Planning and Research	2024-02-18	2024-03-02
Database Setup and Backend Development	2024-03-03	2024-03-16

User Interface Design and Frontend Development	2024-03-17	2024-03-30
Core Functionality Implementation	2024-03-31	2024-04-13
Prepare Presentation Slide	2024-04-14	2024-04-16
Admin Functionality and Testing	2024-04-17	2024-04-30
Project Presentation	2024-05-01	2024-05-01



Final Deliverables:

The final deliverable of the project is a comprehensive and operational database management system, meticulously documented, adept at managing movies, user registrations, bookings, ratings, payments, and other pertinent features.

The success of the project will be evaluated based on the following criteria:

1. Database Management System:
    - A robust and well-designed database schema that efficiently stores and retrieves data.
    - Implementation of backend logic for handling movies, user registrations, ratings, payments, and other core functionalities.
2. User Interface:
    - User interfaces for both the admin and user sides that are responsive, and user-friendly.
    - Effective presentation of information to users.
3. Functionalities:

- Successful implementation of key features such as event creation, user registration, event booking, rating movies.

4. Security:
   - Authentication and authorization mechanisms to ensure secure access to admin and user functionalities.
5. Scalability and Performance:
   - The system should handle a reasonable number of concurrent users and data without significant degradation in performance.
   - Scalability considerations for potential future growth in terms of movies, users, and transactions.

Evaluation Methods:

1. Functional testing:
   - Develop and execute comprehensive test cases covering all aspects of the system.
2. Performance testing:
   - Evaluate the application's response time, scalability, and stability under various workloads.
3. User feedback:
   - Collect feedback from users during the testing phase to identify usability issues or areas for improvement.
4. Security testing:
   - Perform security testing to ensure the application is secure against potential threats.

By employing a combination of testing methodologies, user feedback, code reviews, and performance evaluations, the project's success and performance can be effectively assessed. Continuous improvement based on feedback and thorough evaluation will contribute to a successful and well-performing final deliverable.

Ensuring optimal performance is crucial for user satisfaction and system reliability. Here are some performance standards and strategies to meet them:

1. Response Time:
   - **Standard**: Aim for fast response times to enhance user experience.
   - **Target**: Define specific response time thresholds based on user expectations. For example:
     - **Page Load Time**: Under 2 seconds for critical pages.
     - **API Response Time**: Under 200 milliseconds for common endpoints.
   - **Testing Approach**:
     - Use tools like **JMeter**, **Gatling**, or **Locust** to simulate concurrent users and measure response times.
     - Monitor server-side metrics (CPU, memory, network) during load testing.
     - Optimize database queries, minimize external API calls, and use caching.
2. Scalability:
   - **Standard**: Ensure the system can handle increased load without performance degradation.
   - **Target**:
     - **Vertical Scalability**: Scale up resources (CPU, RAM) on a single server.
     - **Horizontal Scalability**: Add more servers to distribute load.
   - **Testing Approach**:

        - Conduct **load testing** with gradually increasing user loads.
        - Monitor system behavior (response times, resource utilization) as load increases.
        - Implement auto-scaling based on predefined thresholds (e.g., CPU utilization).
3. Stability:
    - **Standard**: The system should remain stable under varying conditions.
    - **Target**:
        - **Error Handling**: Properly handle exceptions, avoid crashes, and gracefully recover.
        - **Memory Leaks**: No memory leaks over extended usage.
        - **Resource Management**: Avoid resource exhaustion (e.g., file descriptors, database connections).
    - **Testing Approach**:
        - Perform **stress testing** with extreme loads to identify bottlenecks and stability issues.
        - Monitor system logs for errors, exceptions, and memory leaks.
        - Use tools like **Valgrind** or **New Relic** for memory profiling.
4. Load Testing Strategies:
    - **Ramp-Up Testing**: Gradually increase the load to observe system behavior.
    - **Peak Load Testing**: Test at expected peak usage (e.g., during ticket sales for movies).
    - **Steady-State Testing**: Sustain a constant load for an extended period.
    - **Spike Testing**: Introduce sudden spikes in traffic to assess system responsiveness.
5. Performance Metrics:
    - **Throughput**: Transactions processed per second.
    - **Concurrency**: Maximum concurrent users.
    - **Error Rate**: Percentage of failed requests.
    - **Latency**: Time taken for a request to complete.
    - **Resource Utilization**: CPU, memory, disk I/O.
6. Monitoring and Profiling:
    - Implement real-time monitoring tools (e.g., **Prometheus**, **Grafana**).
    - Profile code to identify bottlenecks (e.g., slow database queries, inefficient algorithms).
    - Set up alerts for abnormal behavior (e.g., sudden spikes in error rate).
7. Continuous Improvement:
    - Regularly review performance metrics.
    - Optimize code, database queries, and infrastructure.
    - Address bottlenecks promptly.

Background material and References:

1. Randle, T. (2020, November 6). *How to build a Dashboard*. Geckoboard blog. https://www.geckoboard.com/blog/how-to-build-a-dashboard/
2. Merrell Sheehan_Currently. (2019, October 22). *How to add payment processing to a website*. Electronic Merchant Systems. https://www.emscorporate.com/news/how-to-add-payment-processing-to-website

3. Admin. (2024, January 16). *How to optimize & improve database performance*. Buchanan Technologies. https://www.buchanan.com/improve-database-performance/
4. S, R. A. (2024, February 6). *Best backend languages for 2024: Everything you need to know: Simplilearn*. Simplilearn.com. https://www.simplilearn.com/tutorials/programming-tutorial/backend-languages
5. *Frameworks for responsive web design: Everything you need to know*. webdew. (2023, December 6). https://www.webdew.com/blog/framework-for-responsive-web-design
6. *Database design best practices*. Lucidchart. (2021, January 13). https://www.lucidchart.com/blog/database-design-best-practices
7. Azmi, P. A. A. S. U., & Ibrahim, N. (2021). UTHM Students' Event Management System. Applied Information Technology And Computer Science, 2(2), 697-716.
8. Baker, S. (2024, January 19). *How to do payment gateway integration in PHP, Java and C# in 2024*. Financesonline.com. https://financesonline.com/how-to-do-payment-gateway-integration-in-php-java-and-c/
9. What is web application security? | web security | cloudflare. (n.d.). https://www.cloudflare.com/en-gb/learning/security/what-is-web-application-security/
10. Fitzgerald, A. (2024, February 7). *User interface (UI) design: What is it? the beginner's guide*. HubSpot Blog. https://blog.hubspot.com/website/ui-design
11. *Event tickets - worldwide: Statista market forecast*. Statista. (n.d.). https://www.statista.com/outlook/dmo/eservices/event-tickets/worldwide

From the above list, the main concepts are given below:

- **Dashboard Design:**
  - Offers insights into creating effective admin dashboards, essential for visualizing and monitoring key metrics related to movies, user engagement, and transactions.
- **Payment Processing Integration:**
  - Provides guidance on integrating secure and efficient payment systems into the database, ensuring a seamless user experience during ticket booking.
- **Database Performance Optimization:**
  - Offers strategies for optimizing database performance, crucial for handling large volumes of data and transactions efficiently.
- **Best Backend Languages for 2024:**
  - Helps in selecting a suitable backend language for server-side logic development, ensuring scalability and high performance.
- **Frameworks for Responsive Web Design:**
  - Essential for creating a responsive user interface that adapts to various devices, ensuring a consistent and user-friendly experience.
- **Database Design Best Practices:**
  - Provides guidance on structuring database tables and relationships, ensuring data integrity, efficient queries, and scalability.
- **Uthm Students' Event Management System:**

    - Offers real-world insights into event management systems, providing ideas for features, functionalities, and potential challenges in your project.
- **Payment Gateway Integration:**
    - Guides on payment gateway integration in PHP, Java, and C#, offering practical insights into incorporating secure payment mechanisms into the system.
- **Web Application Security:**
    - Explains web application security, a crucial aspect for safeguarding user data, preventing unauthorized access, and protecting against potential cyber threats.
- **User Interface (UI) Design:**
    - A beginner's guide to UI design, providing insights into creating user-friendly interfaces for account registration, login, and overall user interactions.