iMovies: Movie Ticket Booking Application

INTRODUCTION

The iMovies ticket booking application transforms the traditional cinema experience by providing users with a convenient online reservation platform. From the comfort of their homes, users can browse movie selections, view available showtimes, and reserve their preferred seats. Built on a robust client-server architecture utilizing Express.js and MongoDB, the application ensures efficient data management and retrieval. Key features include secure user authentication, comprehensive booking management, and live seat availability updates, all working together to create a personalized and streamlined movie-going experience. This documentation serves as a detailed guide for setup procedures, development processes, and understanding the application's technical framework and features.

DESCRIPTION

The iMovies Ticket Booking Application revolutionizes cinema experiences by offering a streamlined online reservation system. Users enjoy an intuitive interface where they can easily navigate through movie selections, available showtimes, and seat options from any location.

Developed with Express.js and MongoDB, the application utilizes a strong client-server architecture ensuring efficient data handling and instant updates. Core functionality includes secure user authentication, comprehensive booking administration, and real-time seat availability monitoring, enhancing user convenience and personalization.

This documentation provides complete guidance covering setup requirements, development workflows, and comprehensive insights into the system's technical architecture and capabilities.

SCENARIO

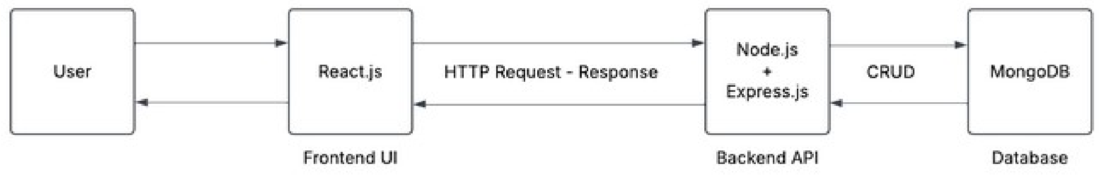
Consider Emma, an avid film enthusiast, eagerly waiting for a highly anticipated movie release. With her busy schedule, she values the convenience of securing tickets online. Upon accessing the iMovies application, Emma smoothly navigates the user-friendly interface, exploring the current movie listings and available showtimes.

After locating her desired film, Emma selects her preferred theater location and showtime. The application displays an interactive seating chart, allowing her to identify and select optimal seating for her viewing experience. With just a few clicks, Emma confirms her selection and moves to the payment process.

Using the secure payment gateway integrated within iMovies, Emma completes her transaction seamlessly, receiving immediate booking confirmation with all necessary details for her upcoming movie night. As her scheduled viewing approaches, Emma can easily access her reservation information through the application, ensuring a hassle-free experience from beginning to end.

Thanks to iMovies' intuitive design and comprehensive features, Emma enjoys a frictionless movie-going experience, making her cinema outing both memorable and stress-free.

TECHNICAL ARCHITECTURE



In this architecture diagram:

- The frontend section represents the user interface components including User Authentication, Watchlist, Movies page, Movie shows page, Profile, Seat allotment page, Bookings page, etc.
- The backend section illustrates API endpoints for Users, Favorites, Shows, Movies, Bookings, etc.
- The Database section displays collections for Users, Bookings, User Favorites, Movies, Shows, Theater information, etc.

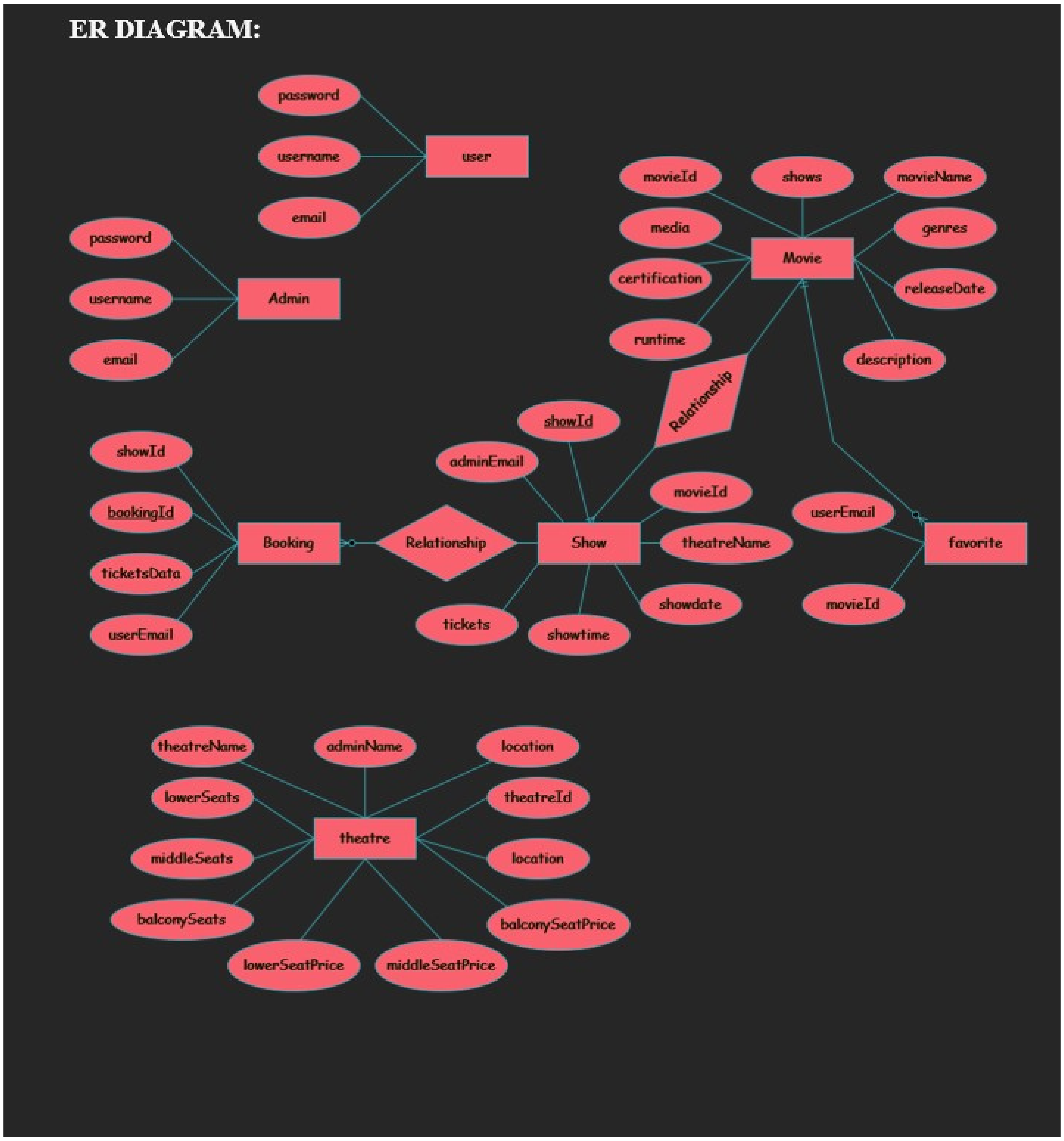
The iMovies application follows a client-server architectural model where the frontend functions as the client and the backend operates as the server. The frontend encompasses not only interface elements and presentation but also incorporates the Axios library for streamlined backend communication through RESTful APIs.

For server-side operations, the application employs Express.js frameworks to manage logic and communication processes.

Data storage and retrieval rely on MongoDB, allowing for scalable and efficient storage of user information, additional documents, and other essential data. This ensures reliable and quick access to necessary information.

Together, these frontend and backend components, combined with Express.js and MongoDB, create a comprehensive technical framework for the iMovies application. This architecture facilitates real-time communication, efficient data exchange, and seamless integration, guaranteeing a smooth and immersive experience for all users.

ER DIAGRAM



1. User Schema: Defines the structure for user information, including fields for username, email, and password. This schema enforces specific constraints such as minimum and maximum character lengths for usernames and passwords, ensuring both data integrity and security within the application.

2. Theater Schema: Outlines properties for theater entities, capturing theater name, administrator name, location, and pricing details for different seat types. It includes nested objects representing balcony, middle, and lower seating arrangements, facilitating comprehensive theater information management.

3. Show Schema: Represents individual show instances created by administrators, containing information such as show identifier, movie identifier, theater name, date, time, and ticket details. This schema enables effective organization and display of show information for user browsing and selection.

4. Movie Schema: Defines the structure for movie information, including attributes like movie name, description, genres, release date, duration, and certification rating. It also maintains an array of associated show identifiers, establishing relationships between movies and their corresponding showtimes.

5. Favorite Schema: Captures user preferences by storing their favorite movies, identified through user email and movie identifier pairs. This facilitates personalized user experiences by enabling quick bookmarking and access to preferred film selections.

6. Booking Schema: Records user reservations, containing data such as booking identifier, user email, show identifier, and comprehensive ticket information. This schema supports efficient management and tracking of user reservations, ensuring a seamless booking experience.

7. Admin Schema: Defines the structure for administrator accounts, including fields for username, email, and password. Similar to the user schema, it enforces specific constraints on username and password lengths to maintain secure access to administrative functions.

KEY FEATURES

- **User Registration and Authentication:** Enables users to create accounts, securely log in, and verify their identity to access the booking platform.
- **Movie Listings:** Presents available films with comprehensive details including title, genre, release date, and available showtimes.
- **Cinema Selection:** Offers a selection of theaters screening various movies, complete with location information.
- **Seat Selection:** Allows users to choose preferred seating from an interactive layout for their selected showtime.
- **Booking Management:** Enables users to reserve tickets for specific showtimes, indicating their desired number of seats.
- **Booking Confirmation:** Provides users with detailed booking confirmation information, including ticket details and a unique reservation identifier.
- **Booking History:** Allows users to review past and upcoming reservations, with options to cancel or modify bookings when permitted.
- **Seat Availability Tracking:** Maintains current seat availability for each showtime, updating in real-time as reservations are created or canceled.
- **Admin Dashboard:** Provides an administrative interface for managing movies, showtimes, theaters, seat availability, and user bookings.

These represent the core features, which can be customized and expanded based on specific requirements and the scale of your movie ticket booking implementation.

PRE-REQUISITES

To develop a full-stack movie ticket booking application using React JS, Node.js, and MongoDB, consider the following prerequisites:

Node.js and npm:

MongoDB:

Express.js:

React.js:

HTML, CSS, and JavaScript:

Database Connectivity:

Front-end Framework:

Version Control: Git

Development Environment:

Database Connection Guide:

ROLES AND RESPONSIBILITIES

User:

Registration: Users are responsible for establishing accounts by providing essential information including name, email address, and contact number.

Movie Search and Selection: Users can browse available films, view detailed information including title, genre, and release date, and select desired movies for booking.

Cinema Selection: Users can choose their preferred theater location from available options.

Showtime Selection: Users can select specific screening times for their chosen movie and theater.

Seat Selection: Users are responsible for choosing their preferred seating arrangements from the interactive layout for their selected showtime.

Booking and Payment: Users can complete their reservation process by booking selected seats and processing payment for tickets.

Booking Management: Users can access their reservation history, view upcoming bookings, and cancel reservations when permitted by the system.

Rating and Reviews: Users can provide ratings and submit reviews for movies they've watched, sharing their experiences with the community.

Admin:

System Management: Administrators oversee the general operation and configuration of the entire movie ticket booking platform.

Movie Management: Administrators can add, update, or remove films from the system, including details such as title, genre, and release date.

Seat Management: Administrators configure and maintain accurate seating layouts for each theater, ensuring precise seat availability information.

These roles and responsibilities may vary based on the specific requirements and scope of your movie ticket booking implementation.

USER FLOW

1. User Registration:

Users navigate to the registration page and complete the required fields including username, email address, and password.

After submitting registration information, the system validates the provided data and creates a user account if all requirements are satisfied.

Upon successful registration, users are directed to the login page to access the booking system.

2. User Login:

Users enter their credentials (username/email and password) on the login screen and submit the form.

The system verifies these credentials against the database and grants access when valid.

After successful authentication, users are directed to the homepage or personal dashboard to begin exploring movies and making reservations.

3. Browsing Movies:

Users explore available films displayed on the homepage or dedicated movie listing section.

Each movie entry includes essential details such as title, genre, release date, and promotional image, helping users make informed decisions.

By selecting a specific movie, users can access additional information including synopsis, cast details, runtime, and available showtimes.

4. Selecting Showtime and Cinema:

Users choose their desired movie, which displays corresponding showtimes and theater options.

Users select their preferred theater location and showtime based on personal convenience and availability.

The system presents an interactive seating chart for the selected showtime, allowing users to choose their preferred seats.

5. Seat Selection and Booking:

Users select seats from the interactive layout, with real-time availability updates guiding their selection process.

After finalizing seat choices, users proceed to the booking confirmation page.

The system calculates the total ticket cost based on selected seats and any supplementary charges, presenting a comprehensive summary for review.

6. Payment and Confirmation:

Users securely enter payment information through the integrated payment gateway.

Upon successful payment processing, the system generates booking confirmation with a unique identifier.

Users receive confirmation emails containing comprehensive booking details and instructions for accessing their tickets.

7. Viewing Booking History:

Users can access their past and upcoming reservations through their profile or dashboard.

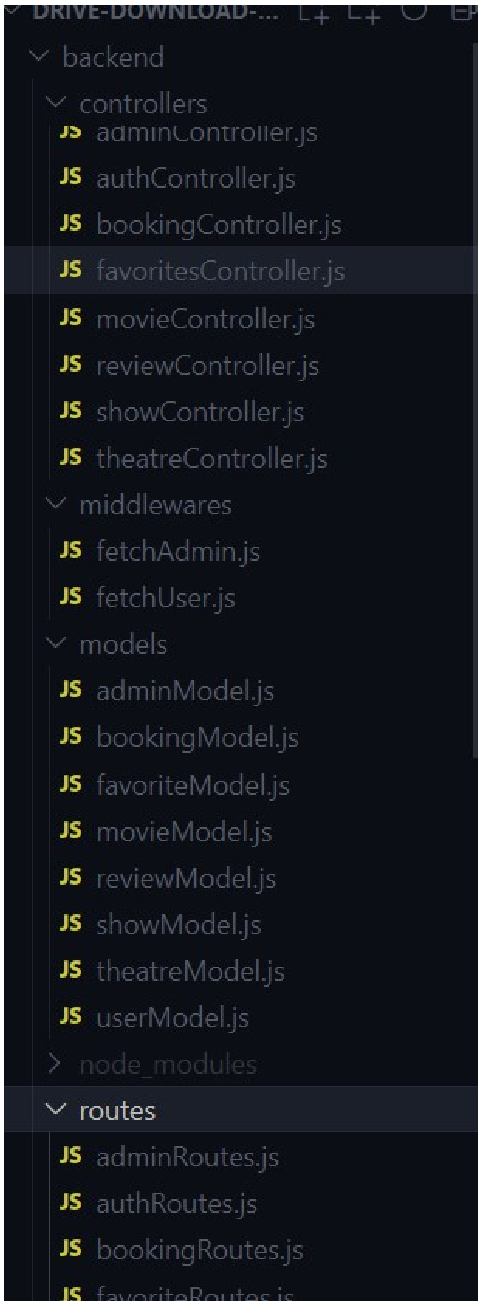
The booking history section displays comprehensive information including movie title, showtime, theater location, and reservation status.

Users can cancel or modify bookings when permitted by the system, providing flexibility and convenience.

Throughout this process, the application ensures a seamless and intuitive experience, guiding users from initial registration through ticket confirmation with minimal friction.

PROJECT STRUCTURE

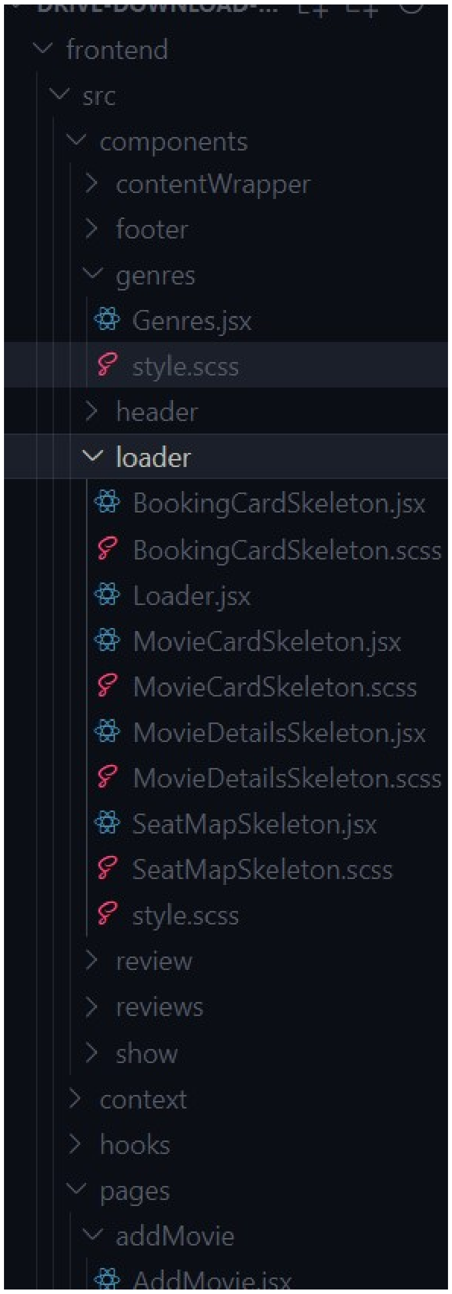
Backend Directory:



```
DRIVE-DOWNLOAD-...
  backend
    controllers
      adminController.js
      authController.js
      bookingController.js
      favoritesController.js
      movieController.js
      reviewController.js
      showController.js
      theatreController.js
    middlewares
      fetchAdmin.js
      fetchUser.js
    models
      adminModel.js
      bookingModel.js
      favoriteModel.js
      movieModel.js
      reviewModel.js
      showModel.js
      theatreModel.js
      userModel.js
    node_modules
    routes
      adminRoutes.js
      authRoutes.js
      bookingRoutes.js
      favoriteRoutes.js
```

The image above displays the backend structure showing all files and folders utilized in server-side development.

Frontend Directory:


```
DRIVE-DOWNLOAD-...
⌄ frontend
  ⌄ src
    ⌄ components
      › contentWrapper
      › footer
      ⌄ genres
          Genres.jsx
          style.scss
      › header
      ⌄ loader
          BookingCardSkeleton.jsx
          BookingCardSkeleton.scss
          Loader.jsx
          MovieCardSkeleton.jsx
          MovieCardSkeleton.scss
          MovieDetailsSkeleton.jsx
          MovieDetailsSkeleton.scss
          SeatMapSkeleton.jsx
          SeatMapSkeleton.scss
          style.scss
      › review
      › reviews
      › show
    › context
    › hooks
    ⌄ pages
      ⌄ addMovie
          AddMovie.jsx
```

The image above displays the frontend structure showing all files and folders utilized in user interface development.

PROJECT FLOW

Milestone 1: Project Setup and Configuration

Folder Setup:

- Create frontend folder
- Create backend folder

Installation of Required Tools:

1. **Frontend folder requirements:**
   - React
   - Bootstrap
   - Axios
   - React-bootstrap
   - Antd
   - mdb-react-ui-kit
2. **Backend folder requirements:**
   - cors
   - bcryptjs
   - express
   - dotenv
   - mongoose
   - Nodemon
   - Jsonwebtoken
   - multer

Milestone 2: Backend Development

Setup Express Server:

- Create index.js file in the server (backend folder)
- Define port number, MongoDB connection string, and JWT key in env file
- Configure server with cors and body-parser middleware

Configure MongoDB:

- Import mongoose
- Add database connection from config.js file in the config folder

- Create models folder to store all database schemas for users and locations

Add Authentication:

- Create middleware folder containing authMiddleware.js for project authentication

1. **Initialize Project Structure:**
   - Establish backend directory for server-side code
   - Set up Node.js project with `npm init` to generate package.json file
2. **Install Required Dependencies:**
   - Express.js: `npm install express`
   - MongoDB driver: `npm install mongoose`
   - Additional packages as needed (cors, bcryptjs, dotenv, etc.)
3. **Configure Express Server:**
   - Create index.js file to initialize Express server
   - Import Express and set up server instance
   - Define middleware including cors, body-parser, and error handling
4. **Connect to MongoDB:**
   - Configure MongoDB connection using Mongoose in config.js file
   - Create models for User, Movie, and Review using Mongoose Schemas
5. **Implement User Authentication:**
   - Develop user registration and login routes with password hashing using bcryptjs
   - Generate JWT tokens for authentication and authorization
6. **Develop Application Routes:**
   - Create CRUD operations for movies, reviews, shows, bookings, theaters, etc.
   - Implement features including favorites management, ticket booking, show creation, and database entry for movies and theaters
7. **Implement Authentication Middleware:**
   - Create middleware function (fetchUser.js) for JWT verification
   - Apply middleware to protect routes requiring authentication
8. **Environment Configuration:**
   - Use dotenv for environment variable management
   - Create .env file for sensitive information storage

9. **Error Handling and Logging:**
    - Implement error handling middleware for graceful error management
    - Configure logging system to track server activities and errors

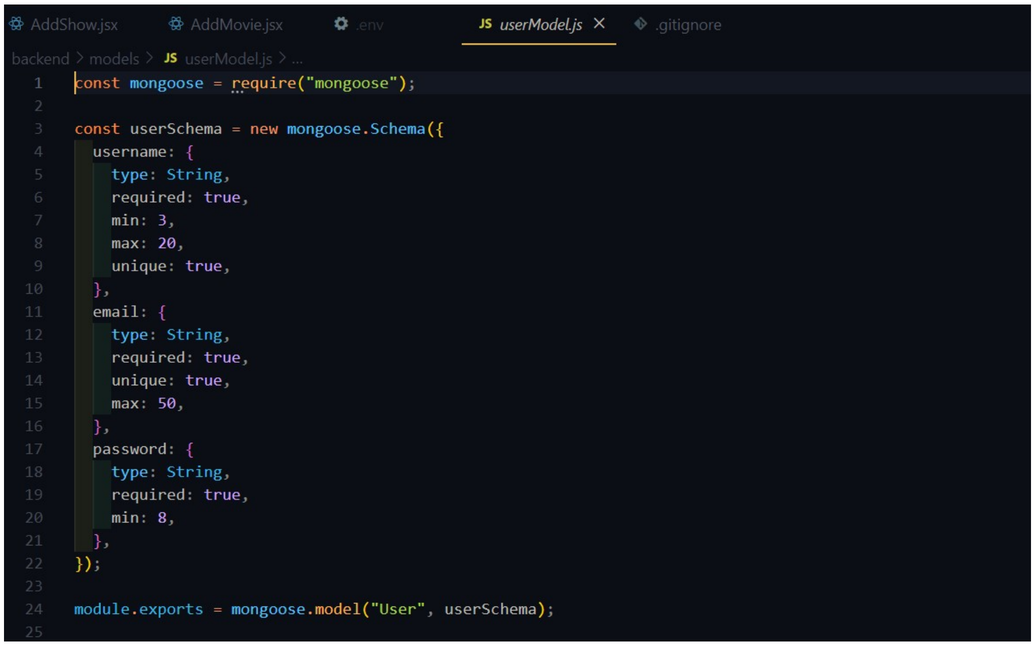
Milestone 3: Database Development

- Install Mongoose
- Establish database connection

Creating Data Schemas:

User Schema:

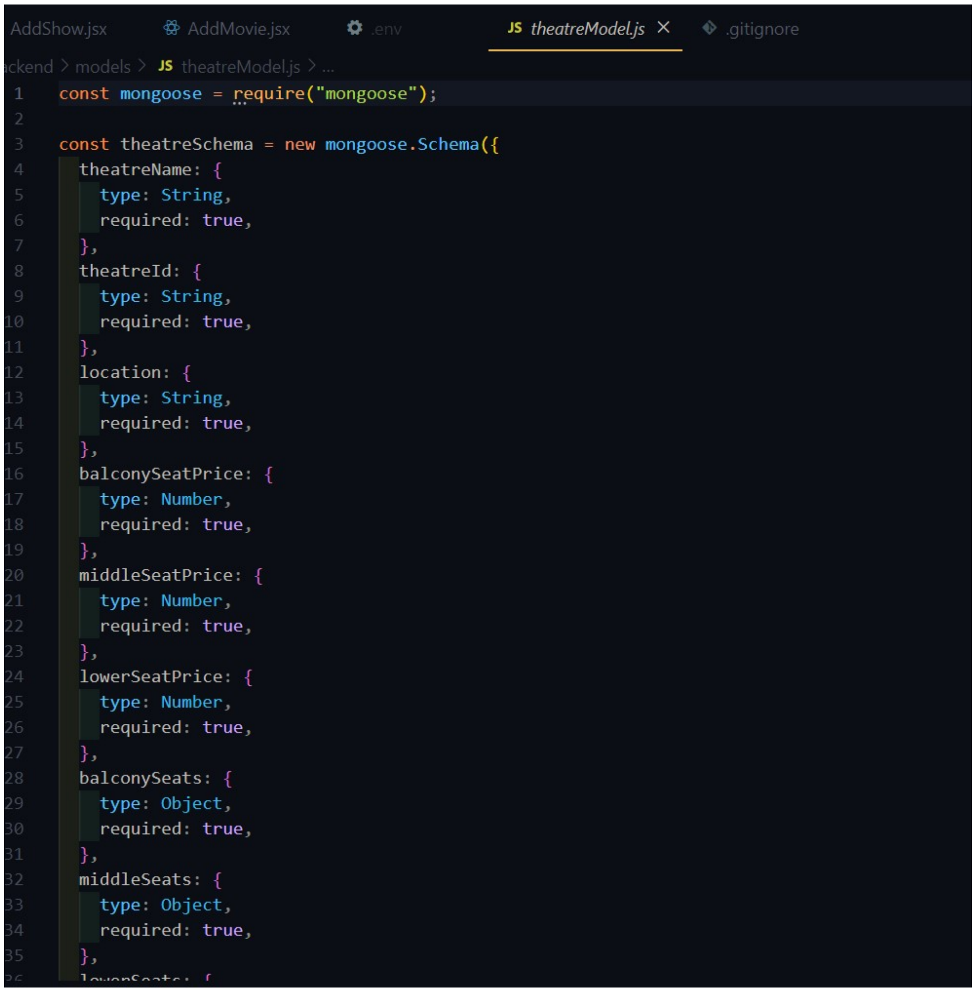
- Schema: userSchema
- Model: 'User'
- Defines fields for username, email, and password with constraints including minimum/maximum lengths and uniqueness requirements
- Ensures data integrity and security for user accounts



```
const mongoose = require("mongoose");

const userSchema = new mongoose.Schema({
  username: {
    type: String,
    required: true,
    min: 3,
    max: 20,
    unique: true,
  },
  email: {
    type: String,
    required: true,
    unique: true,
    max: 50,
  },
  password: {
    type: String,
    required: true,
    min: 8,
  },
});

module.exports = mongoose.model("User", userSchema);
```

Theater Schema:

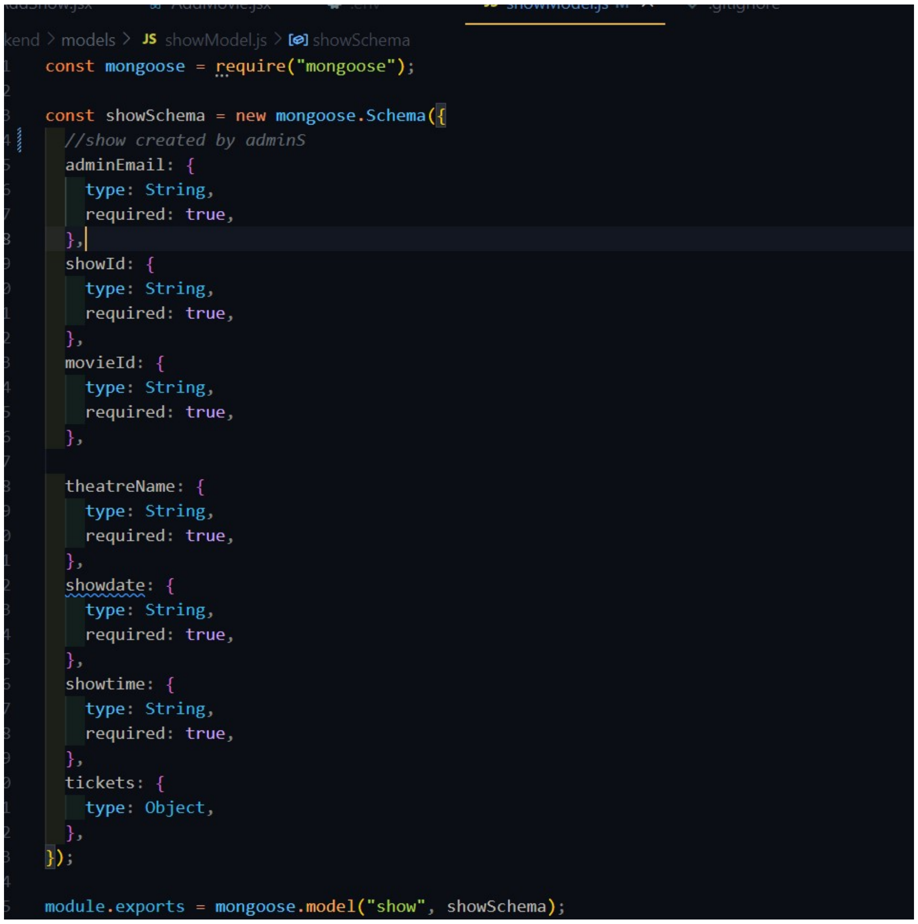
- Schema: theaterSchema
- Model: 'Theater'
- Captures theater details including names, identifiers, locations, pricing, and seating layouts
- Facilitates management of theater information including seating arrangements and pricing



```
const mongoose = require("mongoose");

const theatreSchema = new mongoose.Schema({
  theatreName: {
    type: String,
    required: true,
  },
  theatreId: {
    type: String,
    required: true,
  },
  location: {
    type: String,
    required: true,
  },
  balconySeatPrice: {
    type: Number,
    required: true,
  },
  middleSeatPrice: {
    type: Number,
    required: true,
  },
  lowerSeatPrice: {
    type: Number,
    required: true,
  },
  balconySeats: {
    type: Object,
    required: true,
  },
  middleSeats: {
    type: Object,
    required: true,
  },
```

Show Schema:

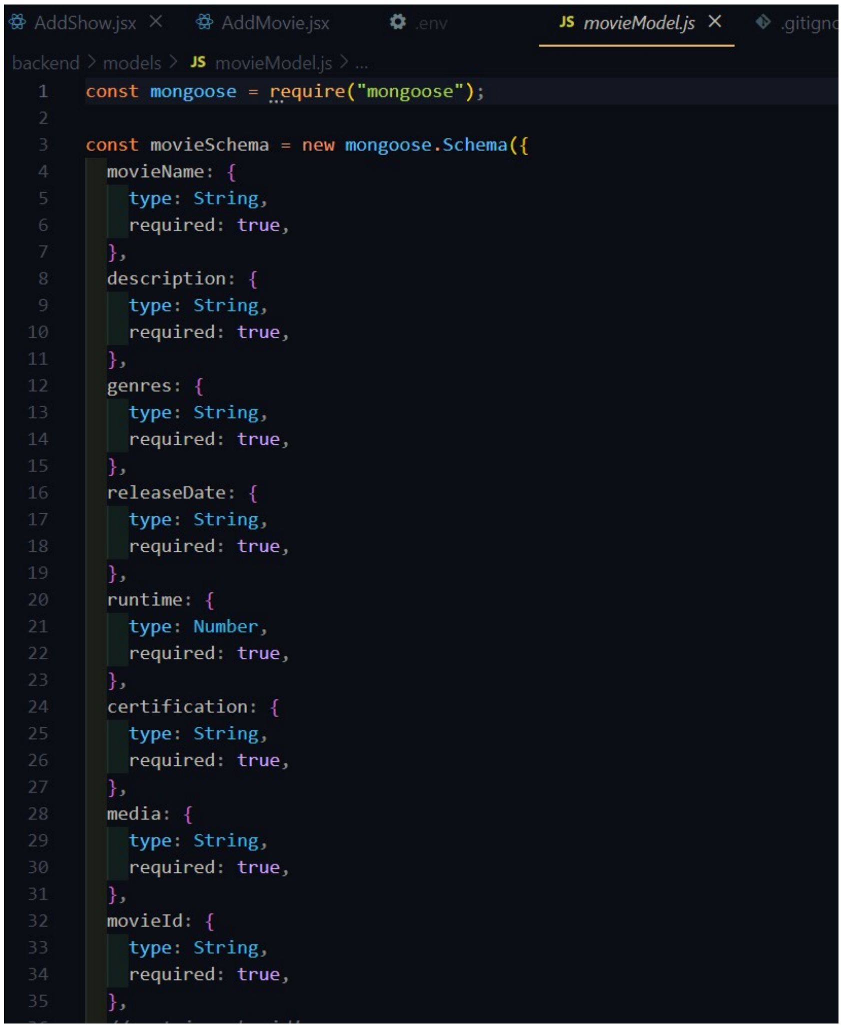
- Schema: showSchema
- Model: 'Show'
- Represents individual screenings with attributes including admin email, movie identifier, theater name, date, time, and ticket information
- Organizes show data for efficient browsing and booking

- 

```
kend > models > JS showModel.js > showSchema
const mongoose = require("mongoose");

const showSchema = new mongoose.Schema({
  //show created by adminS
  adminEmail: {
    type: String,
    required: true,
  },
  showId: {
    type: String,
    required: true,
  },
  movieId: {
    type: String,
    required: true,
  },

  theatreName: {
    type: String,
    required: true,
  },
  showdate: {
    type: String,
    required: true,
  },
  showtime: {
    type: String,
    required: true,
  },
  tickets: {
    type: Object,
  },
});

module.exports = mongoose.model("show", showSchema);
```

Movie Schema:

- Schema: movieSchema
- Model: 'Movie'
- Defines properties for movies including name, description, genres, release date, runtime, certification, format, and associated show identifiers
- Facilitates organization and retrieval of film information



```
const mongoose = require("mongoose");

const movieSchema = new mongoose.Schema({
  movieName: {
    type: String,
    required: true,
  },
  description: {
    type: String,
    required: true,
  },
  genres: {
    type: String,
    required: true,
  },
  releaseDate: {
    type: String,
    required: true,
  },
  runtime: {
    type: Number,
    required: true,
  },
  certification: {
    type: String,
    required: true,
  },
  media: {
    type: String,
    required: true,
  },
  movieId: {
    type: String,
    required: true,
  },
```

Milestone 4: Frontend Development

1. Setup React Application:

- Create React application
- Configure routing
- Install required libraries

2. Design UI Components:

- Create components
- Implement layout and styling
- Add navigation

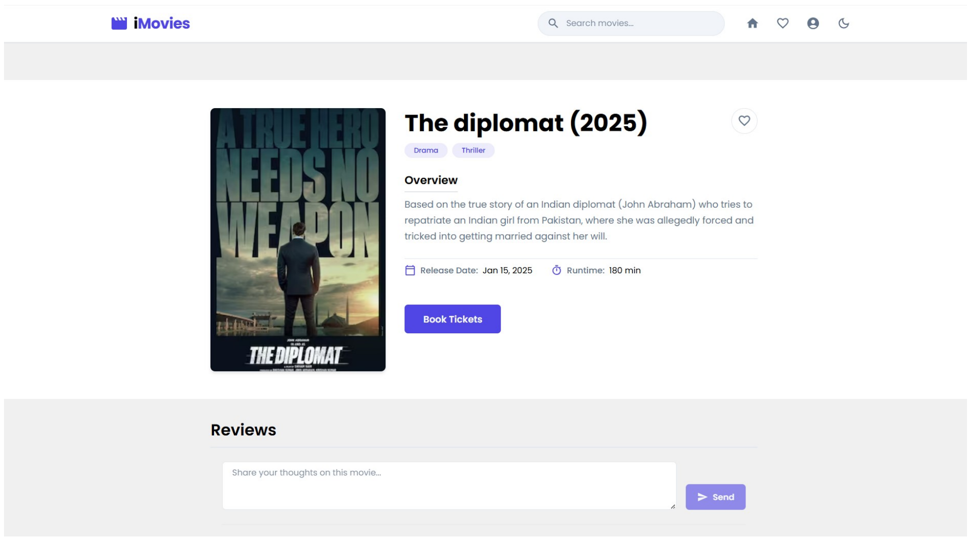
3. Implement Frontend Logic:

- Integrate with API endpoints
- Implement data binding

Milestone 5: Project Implementation

Output Screenshots

Movie Details Page



The diplomat (2025)

Drama Thriller

Overview

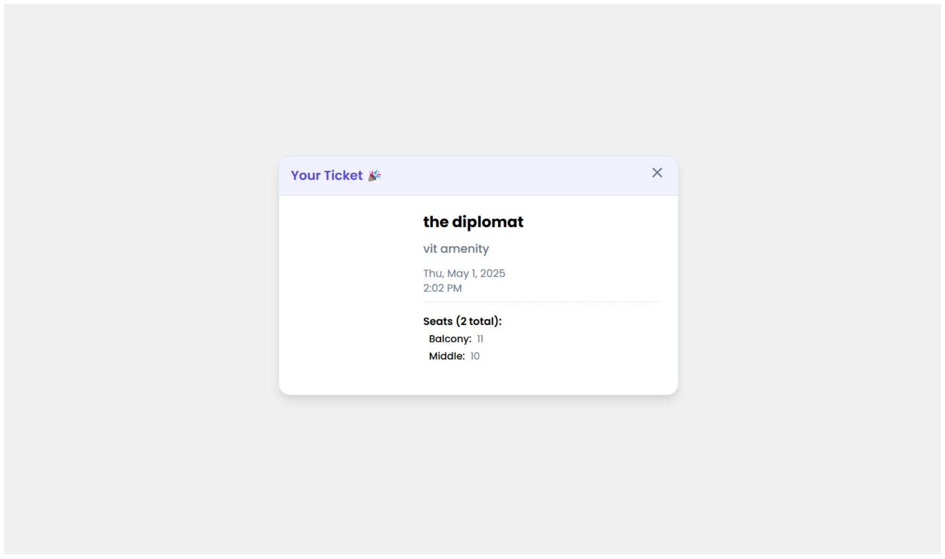
Based on the true story of an Indian diplomat (John Abraham) who tries to repatriate an Indian girl from Pakistan, where she was allegedly forced and tricked into getting married against her will.

Release Date: Jan 15, 2025 Runtime: 180 min

Reviews

Share your thoughts on this movie...

Ticket Payment Success Page



Your Ticket 🎉

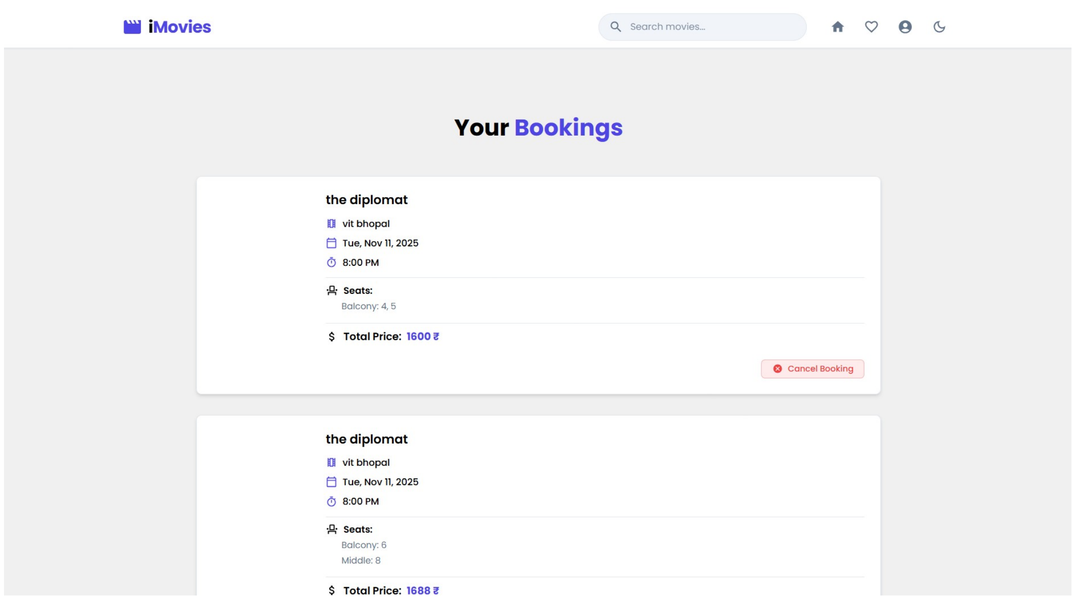
the diplomat

vit amenity

Thu, May 1, 2025
2:02 PM

Seats (2 total):
Balcony: 11
Middle: 10

Bookings Page



iMovies

Search movies...

Your Bookings

the diplomat

vit bhopal

Tue, Nov 11, 2025

8:00 PM

Seats:

Balcony: 4, 5

Total Price: 1600 ₹

Cancel Booking

the diplomat

vit bhopal

Tue, Nov 11, 2025

8:00 PM

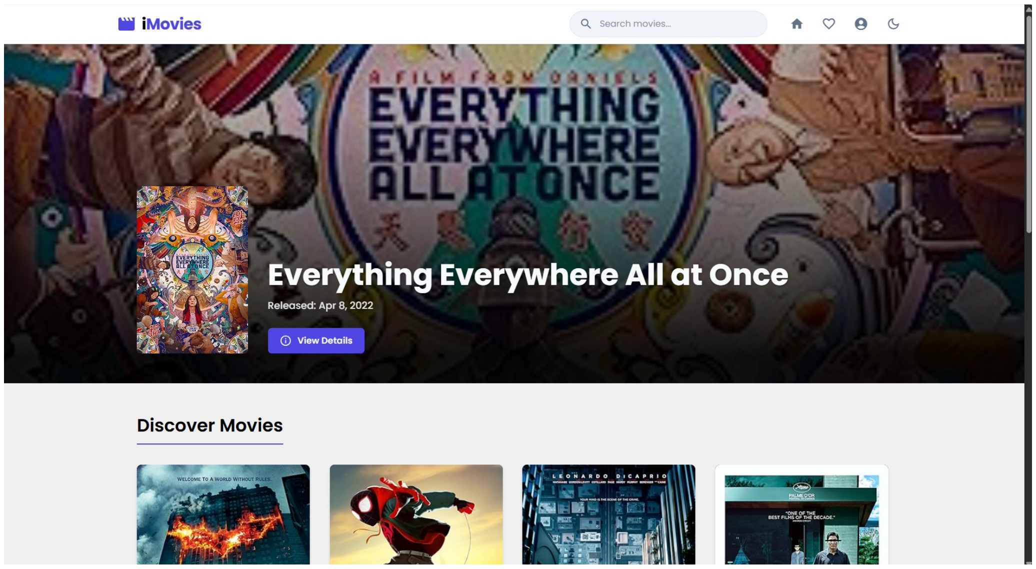
Seats:

Balcony: 6

Middle: 8

Total Price: 1688 ₹

Home Page



iMovies

Search movies...

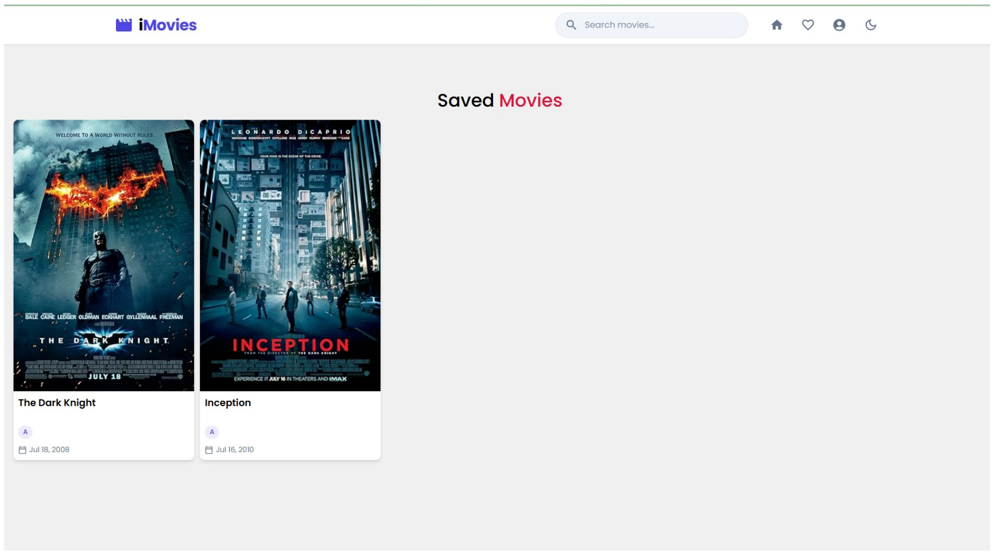
Everything Everywhere All at Once

Released: Apr 8, 2022

View Details

Discover Movies

Saved Movies Page



Saved Movies

The Dark Knight

A

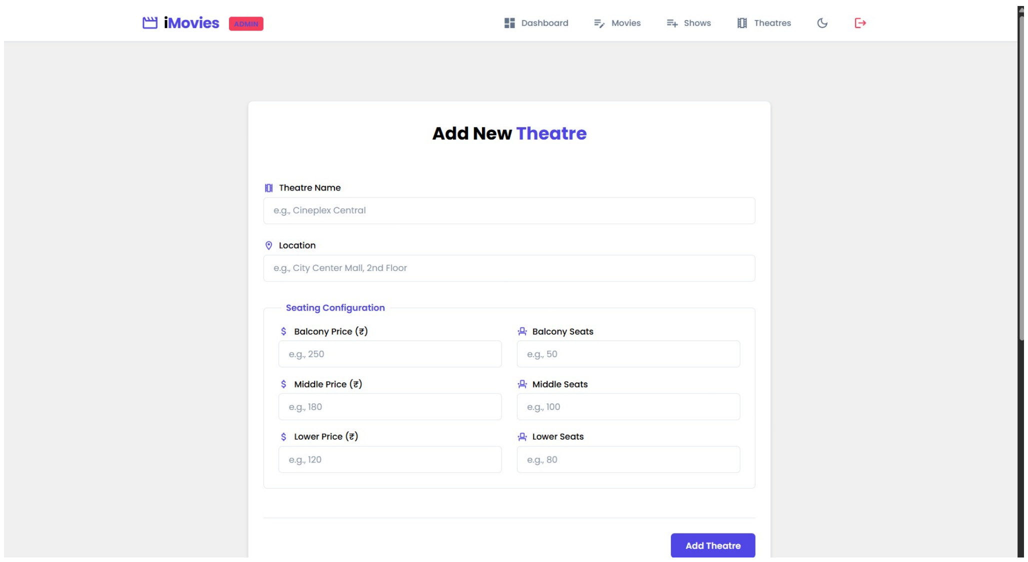
Jul 18, 2008

Inception

A

Jul 16, 2010

Add Theater Page



Add New Theatre

Theatre Name

e.g., Cineplex Central

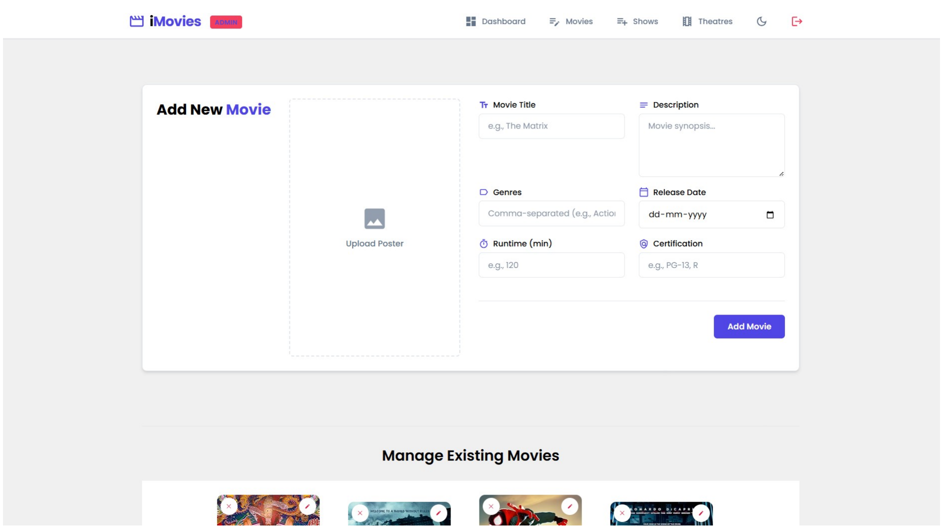
Location

e.g., City Center Mall, 2nd Floor

Seating Configuration

Balcony Price (₹)	Balcony Seats
e.g., 250	e.g., 50
Middle Price (₹)	Middle Seats
e.g., 180	e.g., 100
Lower Price (₹)	Lower Seats
e.g., 120	e.g., 80

Edit Movie Details Page



Conclusion

In summary, the iMovies project represents a significant advancement in the cinema industry, delivering a seamless and personalized movie-going experience. Through its robust client-server architecture, intuitive user interfaces, and efficient data management systems, iMovies ensures both convenience and satisfaction for users. By integrating key features including secure user authentication, comprehensive booking management, and real-time seat availability tracking, the platform establishes new standards for online movie ticket reservation systems. Its adaptable design and scalability make it appropriate for diverse applications, promising to transform how audiences engage with cinema experiences. Overall, iMovies demonstrates the potential of technology to enhance and revolutionize traditional experiences in the entertainment industry.

Documentation Link:
https://drive.google.com/drive/folders/1hpEk99liQJICb29s1a5Z1rLzmwWxXLlh?usp=drive_link

Code Link: https://github.com/nmnjain/iMovies