

Case Study
On
The Movie Ticket Booking System

BABIPRANEETHSAI.R

INTRODUCTION

- The project aims to automate the process of movie ticket booking and management through a web-based application. The Movie Ticket Booking System provides an easy-to-use platform where users can view currently running movies, select theatres, book tickets, and make online payments.
- This system reduces manual work and improves efficiency in managing movie listings, theatre details, bookings, and payments. It allows administrators to manage movies and theatres, while users can register, log in, and book movie tickets conveniently.
- The application is developed using Spring Boot for backend processing, Thymeleaf for frontend rendering, and MySQL as the database, ensuring reliability and scalability.

ABSTRACT

Movie Ticket Booking System (MTBS) is a web-based application designed to manage movie ticket booking activities efficiently. The main objective of this project is to provide a centralized platform for users to book movie tickets online and for administrators to manage movies and theatres.

The system allows users to browse movies, select theatres, choose the number of tickets, and complete payments using Razorpay test mode. Admin users can add, update, and delete movies and theatre details.

This project is developed using Java Spring Boot, Thymeleaf, MySQL, and Razorpay API, following MVC architecture to ensure modularity and maintainability.

What we are going to build: Client Requirement

The client requires a Movie Ticket Booking System where:

- Admin can manage movies and theatres
- Users can register and log in
- Users can view available movies
- Users can book movie tickets
- Users can make online payments

The system should be simple, user-friendly, and efficient for real-time ticket booking.

What we are going to build: Some technical terms

- Proper login and registration functionality
- MVC architecture using Spring Boot
- CRUD operations for movies and theatres
- Session-based authentication
- Online payment integration using Razorpay (Test Mode)
- Exception handling and validation
- Deployment-ready backend application

What are the technologies and tools we are going use

- **Framework:** Spring Boot
- **Language:** Java 17
- **Build Tool:** Maven
- **IDE:** Spring Tool Suite (STS)
- **Database:** MySQL
- **Frontend:** HTML, CSS, Thymeleaf
- **Payment Gateway:** Razorpay (Test Mode)
- **Web Server:** Apache Tomcat

SYSTEM REQUIREMENTS

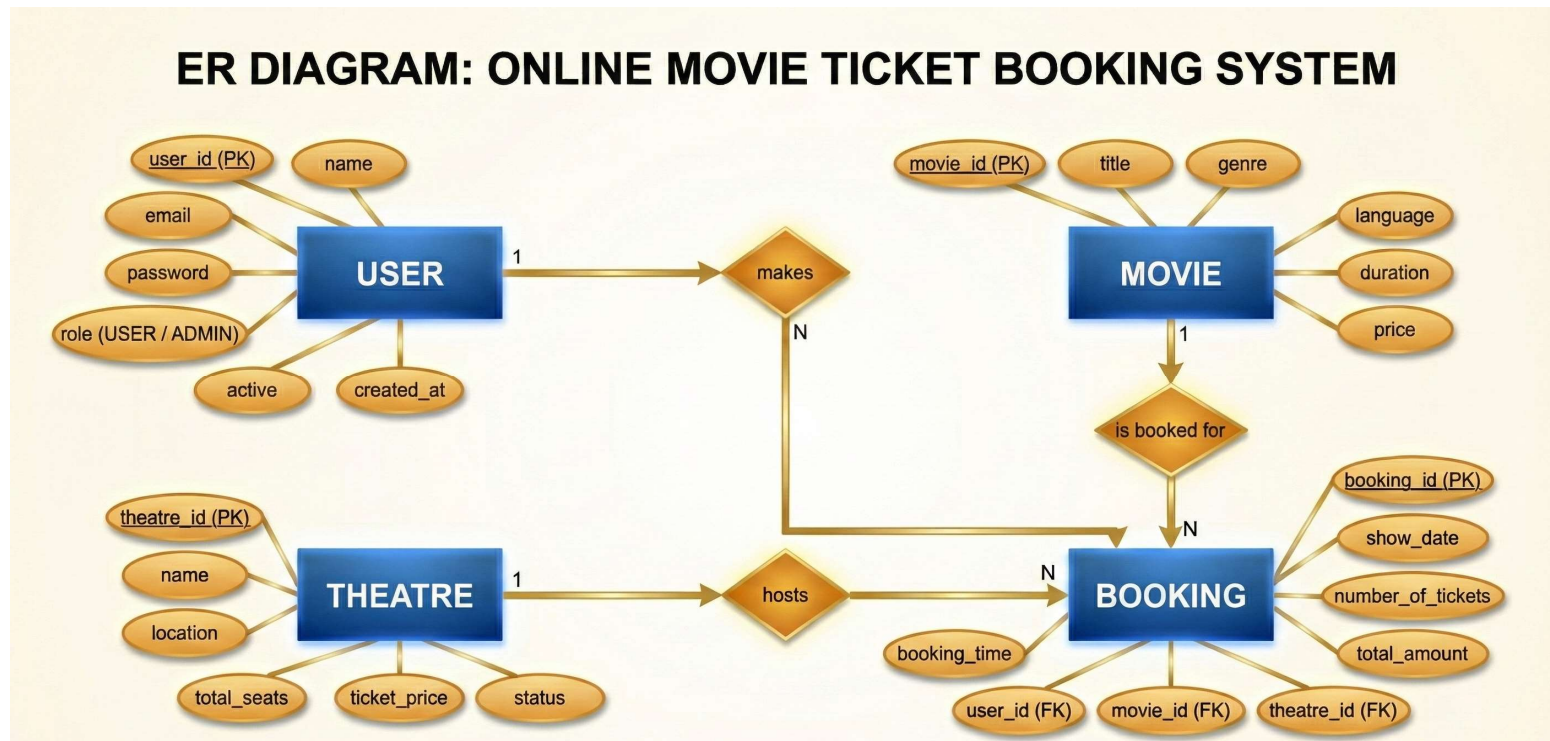
SOFTWARE REQUIREMENT :

- ❖ **Backend:** Spring Boot, Hibernate (JPA)
- ❖ **Database Server:** MySQL 8.0
- ❖ **Web Server:** Apache Tomcat
- ❖ **IDE:** Spring Tool Suite (STS)
- ❖ **Browser:** Google Chrome

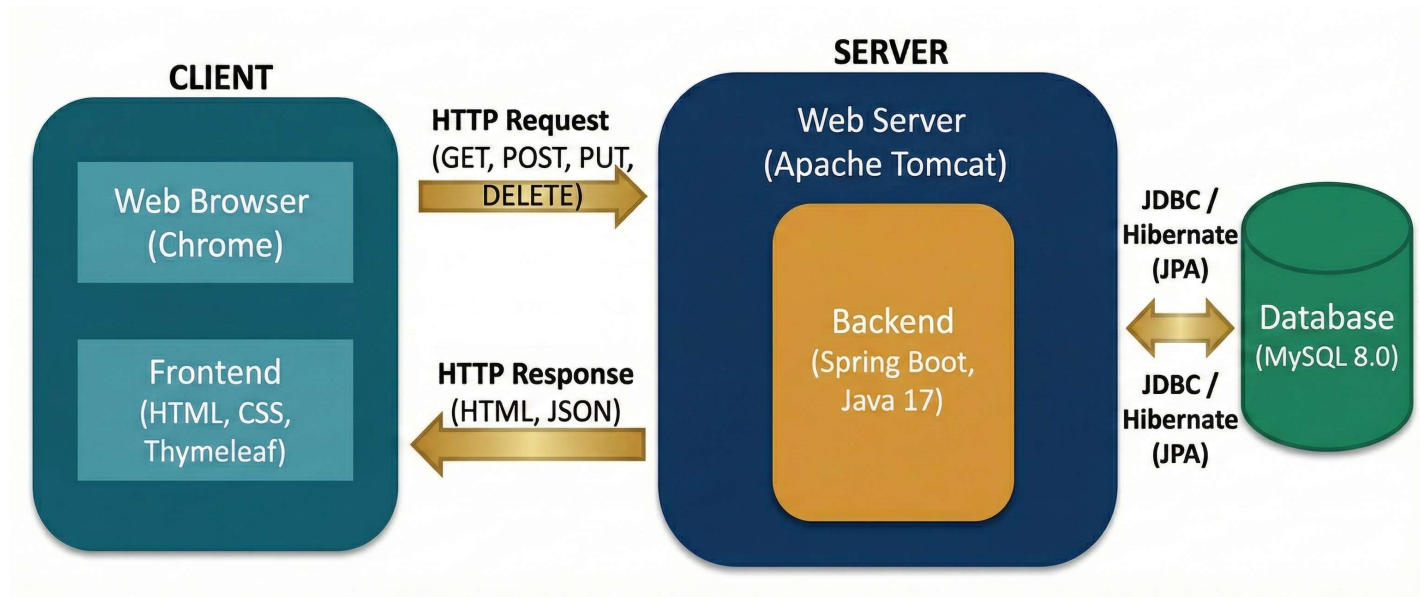
PROJECT MODULE

- ☐ Admin Module
- ☐ User Module
- ☐ Movie Module
- ☐ Theatre Module
- ☐ Booking Module
- ☐ Payment Module

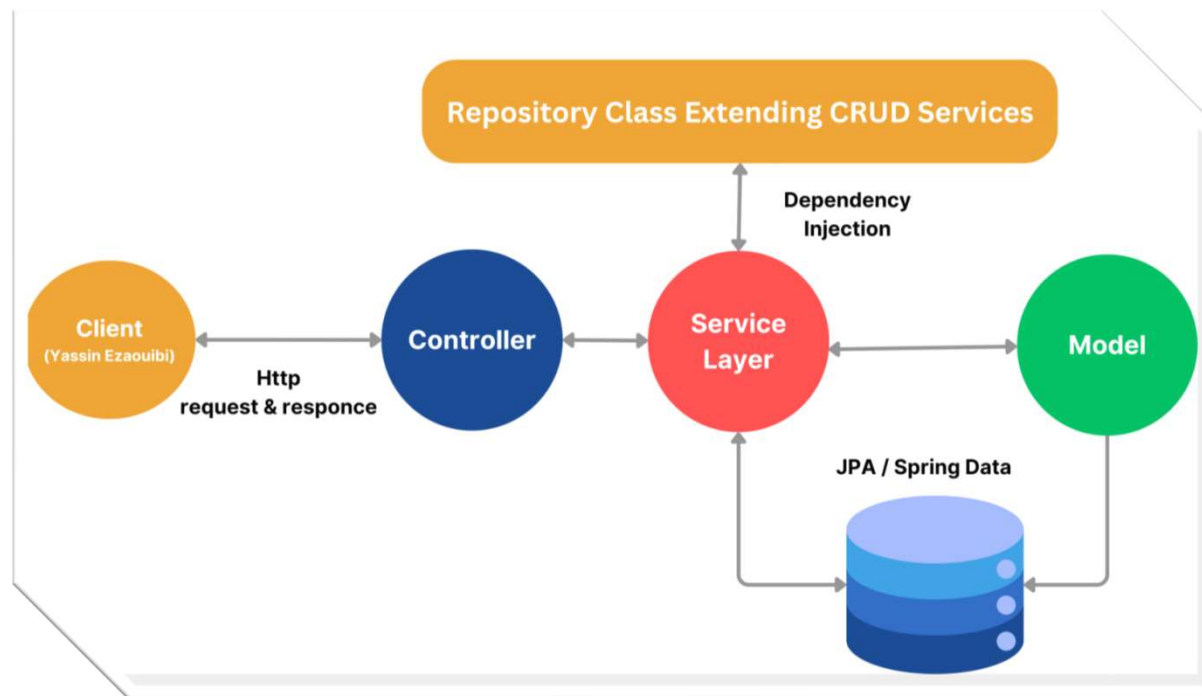
ER DIAGRAM



Client – Server Architecture



SPRING BOOT APP



ADMIN MODULE

- The Admin Module provides secure access to administrators using predefined login credentials.
- Admin users can **add new movies** by entering details such as title, genre, language, duration, and ticket price.
- The module allows admins to **update existing movie and theatre details**, ensuring accurate and up-to-date information.
- Admins can **delete movies or theatres** that are no longer available or relevant.
- The system enables admins to **view all movies and theatres** stored in the database.
- This module ensures **centralized control and management** of the booking system.
- Proper administration helps maintain data consistency and improves the overall user experience.

USER MODULE

- The User Module allows users to **register and log in** to the system using their email and password.
- After successful login, users can **view the list of available movies** along with details such as genre, language, and duration.
- Users can **select a movie and choose a theatre** based on their preference.
- The system allows users to **enter the number of tickets** and proceed with booking.
- Users can **complete the booking using online payment** through Razorpay (test mode).
- After payment, users receive a **booking confirmation** with ticket and show details.
- This module provides a **simple and user-friendly interface** for smooth ticket booking.

HTTP REQUEST METHODS

HTTP Method	URL (localhost)	Description
GET	http://localhost:8080/login	Display user login page
POST	http://localhost:8080/login	Authenticate user credentials
GET	http://localhost:8080/user/dashboard	View list of available movies
POST	http://localhost:8080/user/book	Book movie tickets
GET	http://localhost:8080/payment	Proceed to online payment
GET	http://localhost:8080/logout	Logout from the system

ADMIN HTTP REQUEST METHODS

HTTP Method	URL (localhost)	Description
GET	http://localhost:8080/admin/login	Display admin login page
POST	http://localhost:8080/admin/login	Authenticate admin credentials
GET	http://localhost:8080/admin/dashboard	View admin dashboard
GET	http://localhost:8080/admin/movies	View all movies
POST	http://localhost:8080/admin/movies/save	Add a new movie
GET	http://localhost:8080/admin/theatres	View theatres
POST	http://localhost:8080/admin/theatres/save	Add a new theatre

URI (UNIFORM RESOURCE IDENTIFIER)

HTTP Method	URL (localhost)	Description
GET	http://localhost:8080/user/dashboard	Displays movies available for booking
GET	http://localhost:8080/user/book	Ticket booking page
POST	http://localhost:8080/user/book	Submits booking details
GET	http://localhost:8080/payment	Payment processing page
GET	http://localhost:8080/admin/movies	Manage movies (Admin)
GET	http://localhost:8080/admin/theatres	Manage theatres (Admin)

MOVIE MANAGEMENT MODULE

- The Movie Management Module is responsible for handling all movie-related operations in the system.
- Admin users can add new movies with details such as title, genre, language, duration, and ticket price.
- The module allows updating existing movie information to ensure accurate and up-to-date listings.
- Admins can also delete movies that are no longer available for booking.
- This module ensures that users always see the latest movies currently showing in theatres.

THEATRE MANAGEMENT MODULE

- The Theatre Management Module allows the admin to manage theatre details efficiently.
- Admin users can add theatres with information such as theatre name, location, total seats, and ticket price.
- The module supports updating theatre details to reflect changes in seating capacity or pricing.
- Admins can remove theatres that are no longer operational.
- This module ensures proper mapping between movies and theatres for accurate booking.

BOOKING MODULE

- The Booking Module enables users to book movie tickets seamlessly through the system.
- Users can select a movie, choose a theatre, and enter the number of tickets required.
- The system automatically calculates the total booking amount based on ticket price and quantity.
- Booking details such as booking date, show date, and status are stored securely in the database.
- This module acts as the core functionality connecting users, movies, and theatres.

PAYMENT MODULE

- The Payment Module handles online ticket payment using an external payment gateway (Razorpay – Test Mode).
- After booking confirmation, users are redirected to the payment page to complete the transaction.
- Payment processing is done securely without storing sensitive card details in the database.
- Upon successful payment, users receive booking confirmation.
- This module ensures a smooth and secure payment experience for users.

DATA DICTIONARY

TABLES OF DATABASE

The screenshot displays the MySQL Workbench interface. The top menu bar includes File, Edit, View, Query, Database, Server, Tools, Scripting, and Help. The left sidebar contains a 'Navigator' pane with sections for 'MANAGEMENT' (Server Status, Client Connections, Users and Privileges, Status and System Variables, Data Export, Data Import/Restore) and 'INSTANCE' (Startup / Shutdown, Administration, Schemas). Below this is an 'Information' pane stating 'No object selected'.

The main workspace is titled 'Query 1' and contains the following SQL script:

```
1 • CREATE DATABASE movie_booking_db;
2 • USE movie_booking_db;
3 • show tables;
4
5
```

Below the query editor, the 'Result Grid' is visible, showing the output of the 'show tables;' command. The results are listed under the heading 'Tables_in_movie_booking_db':

Tables_in_movie_booking_db
booking
bookings
movie
movies
payments
theatres
users

The right sidebar contains 'SQLAdditions' and 'My Snippets' panes. The bottom status bar shows 'Result 1' and options for 'Read Only', 'Context Help', and 'Snippets'.

USER TABLE

The screenshot shows a database management interface with a sidebar on the left, a central query editor, and a right sidebar. The sidebar on the left has sections for 'MANAGEMENT' (Server Status, Client Connections, Users and Privileges, Status and System Variables, Data Export, Data Import/Restore), 'INSTANCE' (Startup / Shutdown, Administration, Schemas), and 'Information' (No object selected). The central query editor shows a query: `select * from movie_booking_db.users;`. The right sidebar shows 'SQLAdditions' and 'My Snippets'. Below the query editor is a 'Result Grid' showing the results of the query. The grid has columns: id, active, created_at, email, name, password, and role. There are two rows of data. The first row has id 1, active 1, created_at 2025-12-13 22:46:23.000000, email rajavarapubabipraneethsai99@gmail.com, name BABIPRANEETHSAI RAJAVARAPU, password 1234, and role. The second row has id 2, active 1, created_at 2025-12-13 22:56:55.000000, email babi@gmail.com, name Babipraneethsai, password 1234, and role. Below the grid are buttons for 'Result Grid', 'Form Editor', and 'Apply'. The bottom status bar shows 'users 2 x', 'Apply', 'Context Help', and 'Snippets'.

Navigator: Query 1 SQL File 3* x

MANAGEMENT

- Server Status
- Client Connections
- Users and Privileges
- Status and System Variables
- Data Export
- Data Import/Restore

INSTANCE

- Startup / Shutdown
- Administration
- Schemas

Information

No object selected

Query 1

```
select * from movie_booking_db.users;
```

Limit to 50000 rows

Result Grid

id	active	created_at	email	name	password	role
1	1	2025-12-13 22:46:23.000000	rajavarapubabipraneethsai99@gmail.com	BABIPRANEETHSAI RAJAVARAPU	1234	
2	1	2025-12-13 22:56:55.000000	babi@gmail.com	Babipraneethsai	1234	
NULL	NULL	NULL	NULL	NULL	NULL	NULL

users 2 x

SQLAdditions

My Snippets

Result Grid

Form Editor

Apply

Context Help

Snippets

MOVIE TABLE

Navigation

MANAGEMENT

- Server Status
- Client Connections
- Users and Privileges
- Status and System Variables
- Data Export
- Data Import/Restore

INSTANCE

- Startup / Shutdown

Administration

Schemas

Information

No object selected

Query 1

SQL File 3*

Limit to 50000 rows

1

•

select

*

from

movies;

Result Grid

Filter Rows:

Edit:

Export/Import:

Wrap Cell Content:

	id	description	duration	genre	language	poster_url	rating	title
▶	1	prabhas	175	Action / Drama	Telugu	NULL	0	Salaar
	2	Thalapathy	165	Action / Thriller	Tamil	NULL	0	Leo
	3	PraBOSS	185	Sci-Fi / Mythology	Telugu	NULL	0	Kalki 2898 AD
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

movies 4

Result Grid

Form Editor

Apply

THEARE TABLE

Navigator: Query 1 SQL File 3*

MANAGEMENT

- Server Status
- Client Connections
- Users and Privileges
- Status and System Variables
- Data Export
- Data Import/Restore

INSTANCE

- Startup / Shutdown

Administration Schemas

Information

No object selected

1 • `select * from theatres;`

Limit to 50000 rows

Result Grid


	id	location	name	ticket_price	total_seats
▶	1	Vijayawada	INOX Multiplex	250	200
	2	Guntur	PVR Cinemas	220	180
	3	Amaravati	Asian Cinemas	200	150
	4	Hyderabad	Cinepolis	280	220
	5	Chennai	AGS cinemas	200	100
*	NULL	NULL	NULL	NULL	NULL

Filter Rows: Edit: Export/Import: Wrap Cell Content:


Result Grid

Form Editor

USER DASHBOARD

 **Movie Ticket Booking System**

Logout

 **Now Showing**

Book Tickets Online

Salaar
Genre: Action / Drama
Language: Telugu
Duration: 175 mins
[Book Now](#)

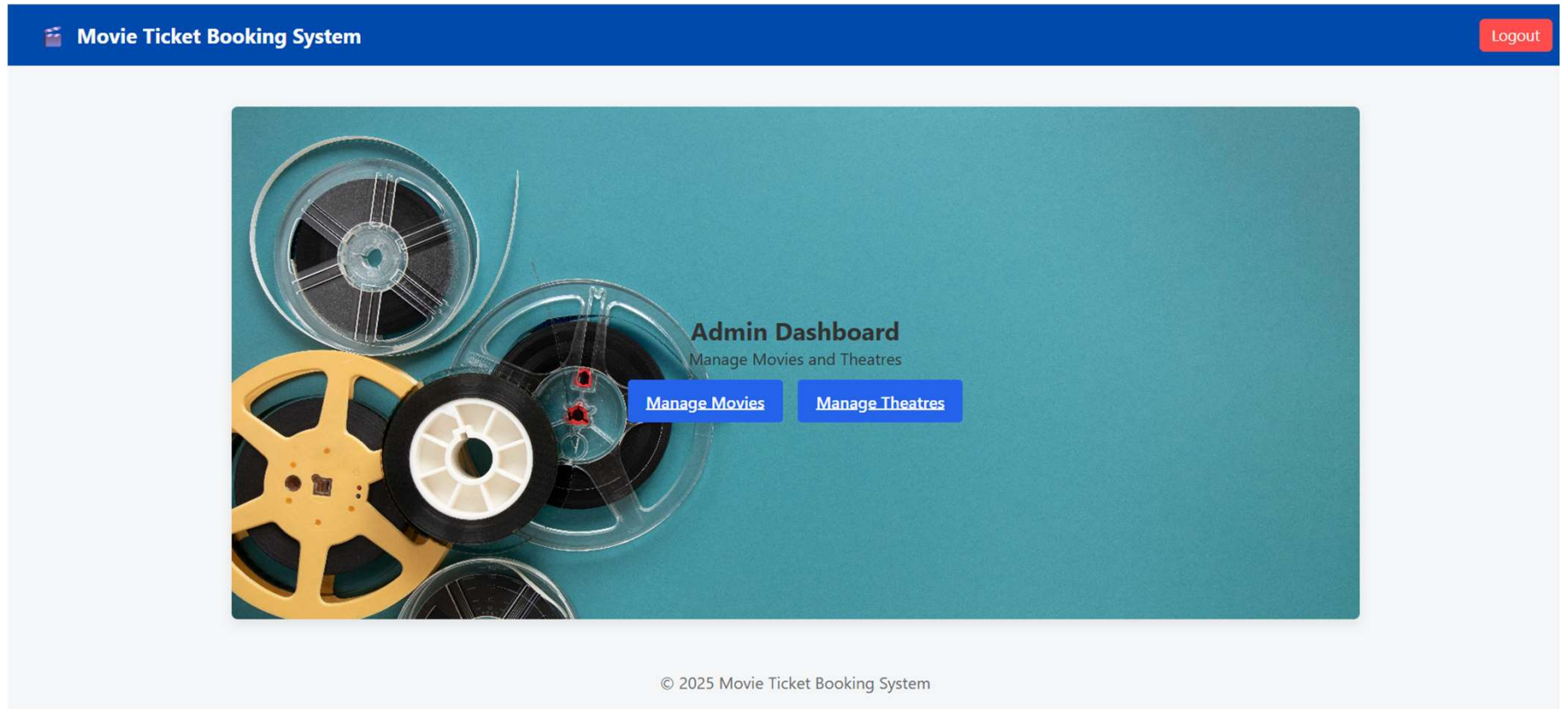
Leo
Genre: Action / Thriller
Language: Tamil
Duration: 165 mins
[Book Now](#)

Kalki 2898 AD
Genre: Sci-Fi / Mythology
Language: Telugu
Duration: 180 mins
[Book Now](#)

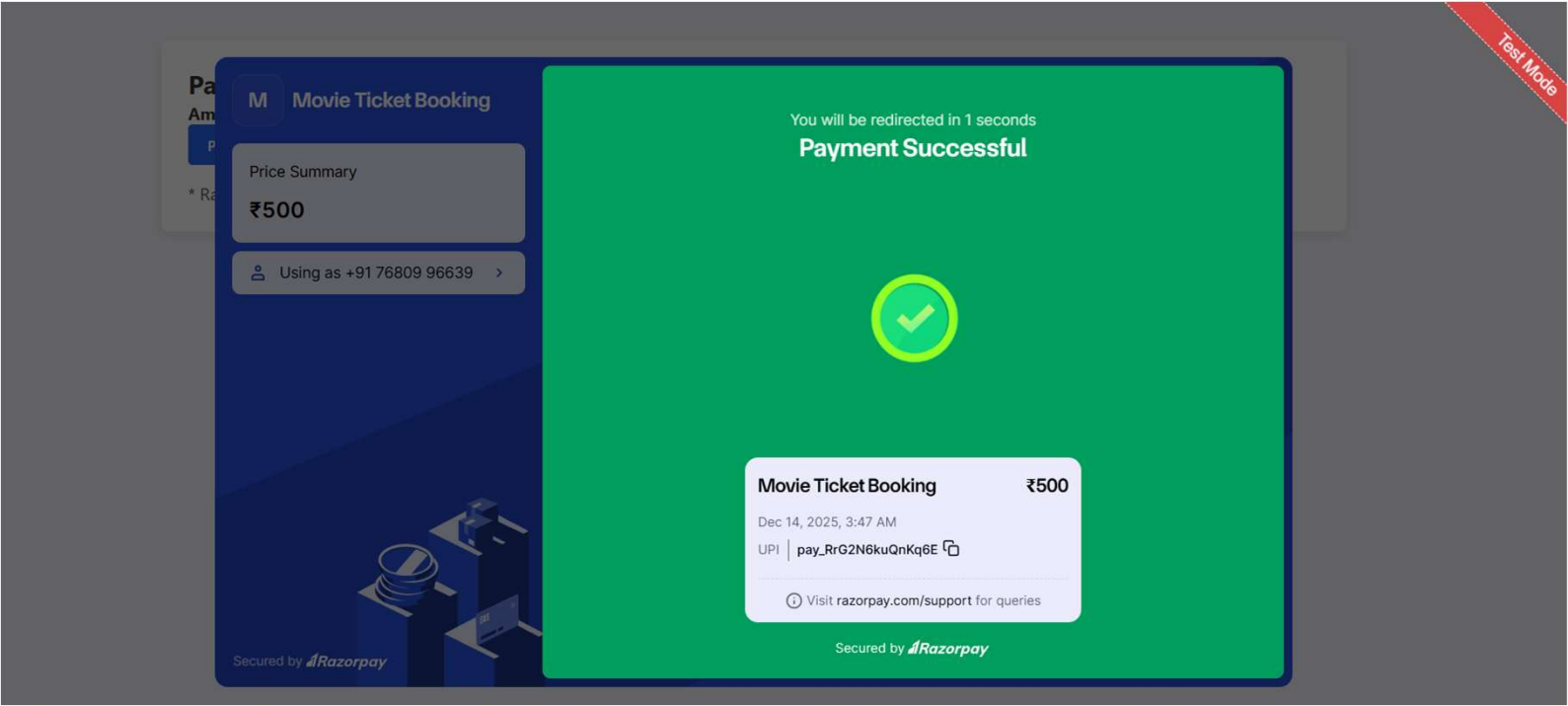
Hi Nanna
Genre: Romance / Family
Language: Telugu
Duration: 155 mins
[Book Now](#)

dude
Genre: love
Language: Tamil
Duration: 180 mins
[Book Now](#)

ADMIN DASHBOARD



PAYMENT STATUS



PROJECT VIDEO EXPLANATION LINK

https://drive.google.com/drive/folders/1xGE8BVyDyjDkQDyz6LXs4MbX1_6V5QbK?usp=sharing