Full Stack Development with MERN

Project Documentation format

1. Introduction

Project Title: iMovies – Online Movie Ticket Booking System

Team Members:

- Bhaskar Rai – Worked on Backend and Video Demonstration
- Dhruv Rai – Worked on Frontend and Database Connectivity
- Rishabh Kumar Singh – Worked on UI/UX design and User/Admin Authentication
- Srayash Kumar – Worked on Project Documentation and Database Connectivity

2. Project Overview

Purpose: iMovies is a full-stack movie ticket booking system built using the MERN stack. It enables users to browse trending movies, explore categorized films, view offers, and book movie tickets seamlessly online. It supports separate authentication for users and admins.

Features:

- JWT-based authentication and authorization
- Admin and user registration/login
- Movie listing with trending carousel
- Categorized movie sections (Action, Horror, Romantic, Sci-Fi)
- Offers section for promotional deals
- Favorite movies tracking and recommendations
- Secure, token-based access for sensitive operations

3. Architecture

Frontend (React): Built using React with Axios for API calls and Context API for global state. Features include component-based structure, reusable UI, SCSS styling, and dynamic rendering based on API data.
Backend (Node.js + Express.js): RESTful API with Express routing. Handles authentication (with JWT and bcrypt), CRUD operations for movies, user management, and admin roles.
Database (MongoDB): MongoDB stores users, admins, and movie data. Mongoose is used to define schemas, including user favorites and admin-created movie entries.

4. Setup Instructions

Prerequisites:

- Node.js (v18+)
- MongoDB (local or Atlas)
- Git
- npm

Installation:

```
git clone https://github.com/bhaskar-rai1/iMovies.git
cd iMovies
```

Create .env files in both /client and /server folders:

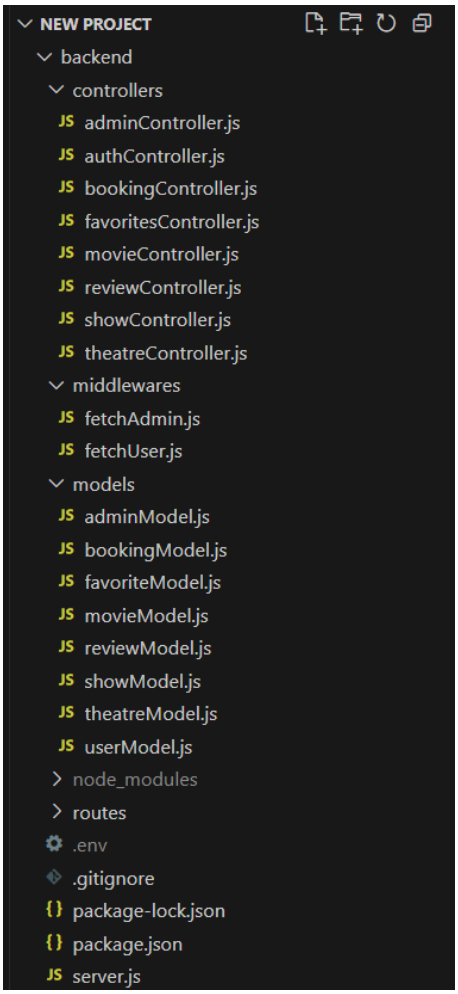
```
MONGO_URI=<Your MongoDB URI>
JWT_SECRET=<Your Secret>
```

Install dependencies:

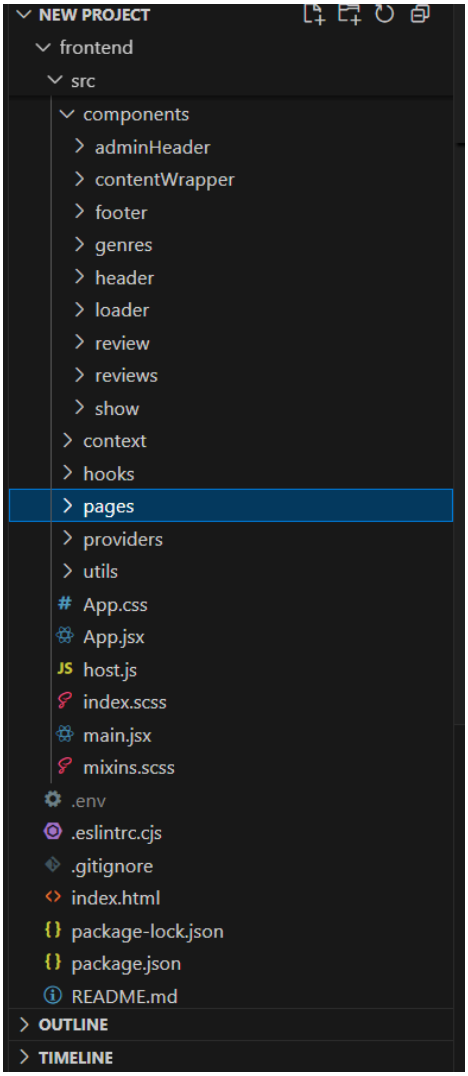
```
cd client
npm install
cd ../server
npm install
```

5. Folder Structure

Backend Folder Structure



Frontend Folder Structure



6. Running the Application

Provide commands to start the frontend and backend servers locally.

Frontend:

```
cd New Project
cd frontend
npm run dev
```

Backend:

```
cd New Project
cd frontend
npm run dev
```

7. API Documentation

Endpoint	Method	Description	Auth Required
`/api/auth/register`	POST	Register new user	No
`/api/auth/login`	POST	Login user	No
`/api/admin/register`	POST	Register new admin	No
`/api/admin/login`	POST	Login admin	No
`/api/movie/getmovies`	GET	Get all movies	Yes (token)
`/api/user/favorites`	GET	Get user favorite movies	Yes
`/api/user/favorites/add`	POST	Add to favorite movies	Yes

Example response

```
{
  "success": true,
  "data": {
    "movies": [
      {
        "_id": "...",
        "title": "Inception",
        "category": "SciFi"
      }
    ]
  }
}
```

8. Authentication

Method: JWT (JSON Web Tokens) stored in cookies.

Process:

- Upon login or registration, token is issued.
- Axios attaches token with protected requests.

- Backend verifies the token and grants access based on user role (admin/user).

Libraries Used: `jsonwebtoken`, `bcrypt`, `cookie-parser`

9. User Interface

The website has a **home screen** that engages users with a dynamic Trending Movies carousel, showcasing the latest titles in an **interactive scrollable format.** Complementing this is a minimal and **modern-themed** Login/Register UI, designed for a smooth onboarding experience. To enhance user engagement, an Offers section is featured, highlighting exclusive discounts and promotional deals for movie bookings.
Users can also explore movies **based on genre** via the Categories grid, which organizes content into Action, Horror, Romantic, and SciFi. Additionally, the project is built with **scalability** in mind, with an Admin Panel planned as a future enhancement to allow for content management and analytics capabilities.

15. Testing

Tools Used: Manual testing via Postman and browser.
Strategy:

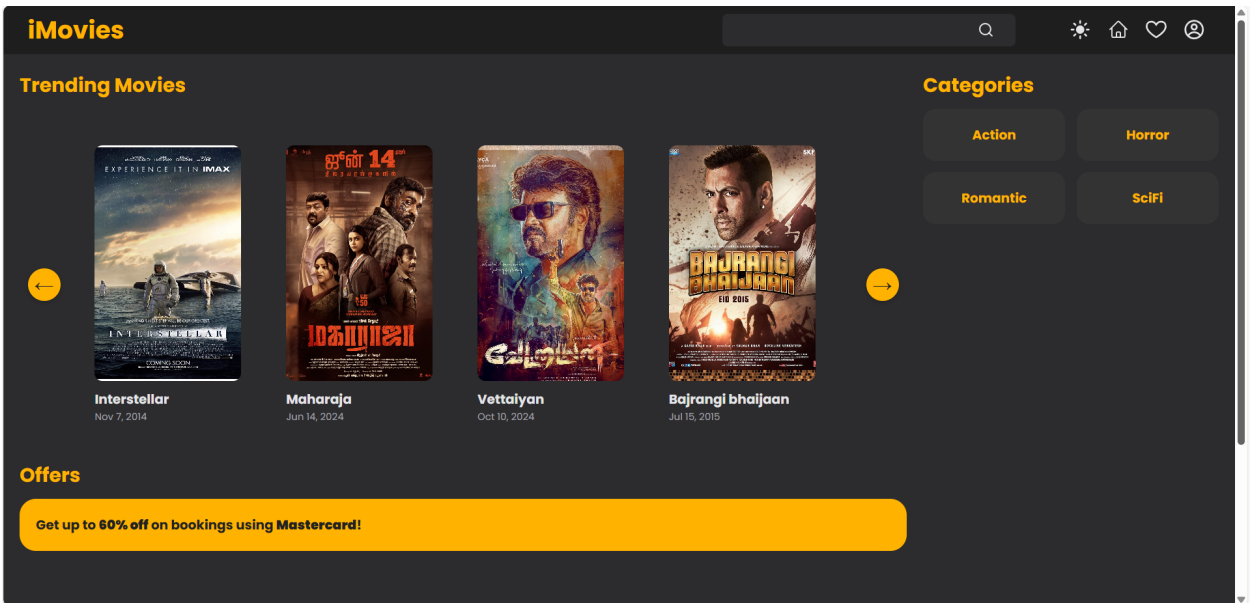
- All endpoints were tested for valid and invalid cases.
- JWT validation and cookie handling were verified.
- UI tested across desktop browsers.

16. Screenshots or Demo

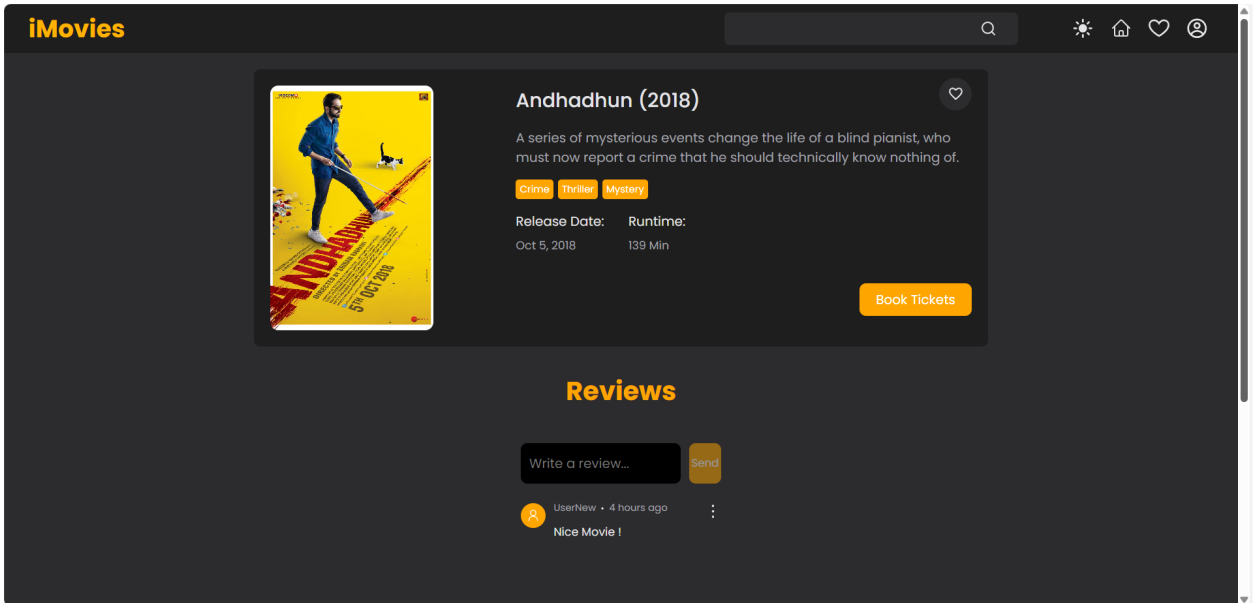
- GitHub Repo: [iMovies GitHub]
- iMovies Demo Video: [Project Demo Video]
- Project Documentation Link: [Project Documents Google Drive]

Sample Screenshots:

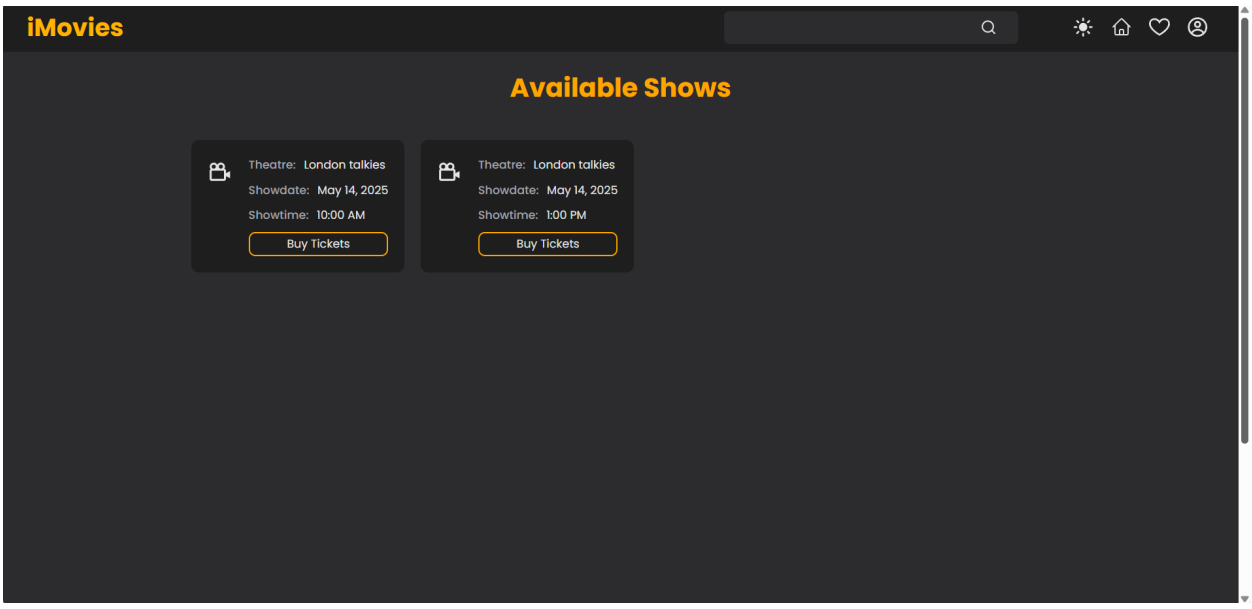
User home page



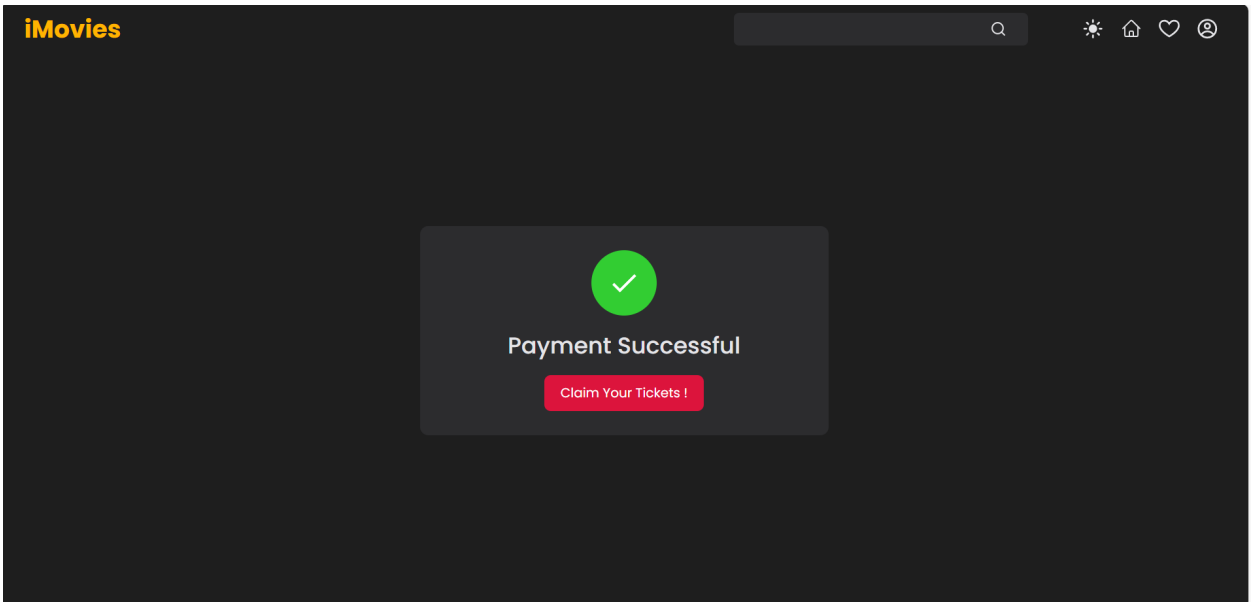
Movie details page



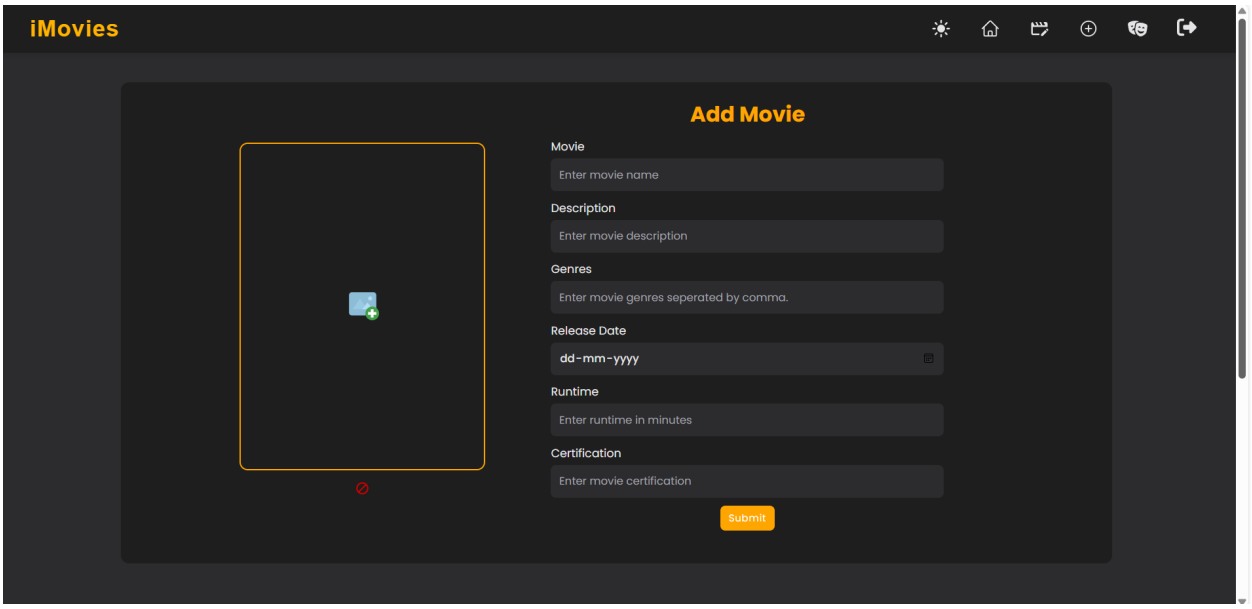
Available shows page



Payment successful page



Admin add movies page



17. Known Issues

The current **Admin Dashboard** serves its fundamental purpose but remains **basic in design** and could benefit from further **UI enhancements** to improve usability and visual appeal. Additionally, the application **does not yet include a payment gateway,** which limits its real-world transactional capabilities.

18. Future Enhancements

- Admin panel to manage movies and view analytics
- Payment gateway integration (e.g., Razorpay/Stripe)
- Improved recommendations using ML models
- Unit testing and CI/CD pipelines

Project Documentation Link: [Project Documents Google Drive]