CineBook Project Documentation

1. Introduction

Project Name: CineBook
Description: CineBook is an online movie ticketing system designed for a single theatre. It enables users to view movie schedules, book tickets, make payments, and receive tickets via email. Admins can manage movie listings and schedules.

2. Technologies Used

- **MEAN Stack:**
  - **MongoDB:** A NoSQL database used to store user data, movies, bookings, and other records.
  - **Express.js:** Backend framework used to build RESTful APIs.
  - **Angular:** Frontend framework for building dynamic user interfaces.
  - **Node.js:** JavaScript runtime environment for running the server-side application.
- **Additional Technologies:**
  - **TailwindCSS:** Used for styling the frontend with responsive design.
  - **Swagger:** Used for API documentation and testing.

3. Features

User Features

- **Movie Listings:** Browse recent movies with schedules.
- **Online Ticket Booking:** Book tickets for available shows.
- **Payment Integration:** Secure payment gateway for transactions.
- **Email Receipts and Tickets:** Automated email with booking confirmation and tickets.
- **Profile Management:** Users can update profiles and view past ticket bookings.

Admin Features

- **Add/Update Movie Listings:** Manage movies and schedules.
- **Booking Management:** Oversee all ticket bookings.

4. Architecture Overview

- **Frontend (Angular):** Communicates with the backend via RESTful APIs and manages user interactions.
- **Backend (Node.js with Express):** Handles business logic, processes API requests, and manages database interactions.
- **Database (MongoDB):** Stores application data, including user information, movies, schedules, and ticket bookings.
- **API Documentation (Swagger):** Provides detailed documentation of API endpoints.

5. System Requirements

- **Node.js:** v18.x or higher
- **MongoDB:** v6.x or higher
- **Angular CLI:** v16.x or higher
- **Payment Gateway Credentials:** (e.g., Stripe, PayPal API keys)
- **SMTP Configuration:** For email notifications (e.g., Gmail, SendGrid).

6. Installation Guide

1. **Clone the Repository:**

   ```
   git clone <repository_url>
   cd cinebook
   ```

2. **Backend Setup:**

   - Navigate to the backend directory.
   - Install dependencies:

     ```
     npm install
     ```

   - Set up environment variables in `.env` file.
   - Start the server:

     ```
     npm start
     ```

3. **Frontend Setup:**

   - Navigate to the frontend directory.
   - Install dependencies:

     ```
     npm install
     ```

   - Start the Angular app:

     ```
     ng serve
     ```

4. **Database Setup:**

   - Ensure MongoDB is running locally or use a cloud instance.

7. API Documentation

API Overview

The CineBook API facilitates various operations such as user authentication, movie management, schedule management, and ticket booking. The APIs are documented using Swagger, providing detailed information about each endpoint, including request parameters, request bodies, and response structures.

API Endpoints

1. Authentication

- **Register a new user**
  - **Endpoint:** `POST /auth/register`
  - **Request Body:**

    ```
    {
      "name": "John Doe",
      "email": "john.doe@example.com",
      "phone": "1234567890",
      "password": "password"
    }
    ```

  - **Response:**

    ```
    {
      "token": "jwt_token",
      "message": "User created successfully"
    }
    ```

- **Login a user**
  - **Endpoint:** `POST /auth/login`
  - **Request Body:**

    ```
    {
      "email": "john.doe@example.com",
      "password": "password"
    }
    ```

  - **Response:**

    ```
    {
      "token": "jwt_token",
      "message": "Logged in successfully"
    }
    ```

2. Movies

- **Get all movies**
  - **Endpoint:** `GET /movie`
  - **Response:** Returns a list of all movies.
- **Add a new movie**
  - **Endpoint:** `POST /movie`
  - **Request Body:**

    ```
    {
      "name": "The Shawshank Redemption",
      "duration": 142,
      "genre": "Drama",
      "rel_date": "1994-09-23",
      "cast": ["Tim Robbins", "Morgan Freeman"],
      "director": "Frank Darabont",
      "imdb_rating": 9.3,
      "poster": "https://example.com/shawshank_redemption.jpg",
      "language": "English"
    }
    ```

  - **Response:** The movie is successfully added.

3. Schedules

- **Get all schedules**
  - **Endpoint:** `GET /schedule`
  - **Response:** Returns a list of schedules.
- **Add a new schedule**
  - **Endpoint:** `POST /schedule`
  - **Request Body:**

    ```
    {
      "date": "2023-12-01",
      "start_time": "18:00",
      "end_time": "20:30",
      "movie": "movie_id",
      "screen": "screen_id"
    }
    ```

  - **Response:** The schedule is successfully added.

4. Tickets

- **Get all tickets**
  - **Endpoint:** `GET /ticket`
  - **Response:** Returns a list of tickets.
- **Book a ticket**
  - **Endpoint:** `POST /ticket`
  - **Request Body:**

```
{
  "name": "John Doe",
  "date": "2024-08-10",
  "time": "19:30",
  "movie": "movie_id",
  "schedule": "schedule_id",
  "screen": 1,
  "seat": ["A12", "A13"],
  "seatType": "VIP",
  "price": 25
}
```

  - **Response:** The ticket is successfully booked.

5. Users

- **Get all users**
  - **Endpoint:** `GET /user`
  - **Response:** Returns a list of users.

8. Detailed Architecture Overview

The CineBook application follows a modern web architecture pattern:

- **Frontend (Angular):** The Angular frontend communicates with the backend via RESTful APIs. It renders user interfaces dynamically and manages user interactions such as booking tickets and managing profiles.
- **Backend (Node.js with Express):** The backend handles business logic, processes API requests, and interacts with the MongoDB database. It manages user authentication, movie listings, ticket booking, and admin functionalities.
- **Database (MongoDB):** MongoDB stores all application data, including user information, movies, schedules, and ticket bookings. It is a NoSQL database that provides flexibility in handling unstructured data, making it suitable for scalable applications.
- **API Documentation (Swagger):** Swagger is used to document and test the API endpoints, ensuring that all functionalities are well-documented and accessible.

9. Database Schema

The CineBook application uses MongoDB to store data. Below are the key collections and their attributes:

- **Users Collection:** Stores user details such as name, email, phone, and password.
- **Movies Collection:** Stores movie information including title, genre, director, cast, release date, and ratings.
- **Schedules Collection:** Contains scheduling information including dates, start and end times, associated movies, and screens.
- **Tickets Collection:** Manages booking details like user ID, movie, schedule, seats, and ticket prices.

10. API Error Handling and Security Measures

CineBook implements comprehensive error handling to ensure smooth user experience:

- **Error Codes:** Common error codes include 400 (Bad Request), 401 (Unauthorized), 404 (Not Found), and 500 (Internal Server Error).
- **Error Messages:** Descriptive error messages guide users and developers to understand issues quickly.
- **Security Measures:**
  - **JWT Authentication:** Ensures secure access to protected routes.
  - **Input Validation:** Prevents SQL Injection, XSS attacks, and other vulnerabilities.
  - **Rate Limiting:** Protects the API from abuse by limiting the number of requests per user.

11. Deployment Guide

To deploy CineBook, follow these steps:

- **Frontend Deployment:** Deploy on Vercel or Netlify by linking the GitHub repository.
- **Backend Deployment:** Host on AWS EC2, Heroku, or DigitalOcean. Ensure the server can run Node.js.
- **Database Deployment:** Use MongoDB Atlas for cloud-based MongoDB.
- **Environment Configuration:** Set environment variables for production, including API keys, JWT secrets, and database URIs.

12. Testing Strategy

Testing ensures the stability and reliability of CineBook:

- **Unit Testing:** Tests individual components and functions using frameworks like Jest and Jasmine.
- **Integration Testing:** Verifies that the frontend, backend, and database interact correctly.
- **Manual Testing:** Comprehensive testing by team members to ensure all features work as expected.
- **API Testing:** Conducted via Swagger and Postman to validate the API endpoints.

13. Challenges Faced

Throughout the development of CineBook, the team encountered several challenges:

- **Payment Integration:** Ensuring secure and reliable payment processing was a major hurdle. We used third-party payment services like Stripe or PayPal to manage transactions securely.
- **Handling Scalability:** As the number of users grew, we had to optimize the database queries and server performance to handle increased load efficiently.
- **Cross-Browser Compatibility:** Ensuring that the application works seamlessly across different browsers and devices required thorough testing and adjustments in the frontend code.

14. Future Enhancements

The CineBook project has the potential for several future enhancements:

- **Mobile Application:** Develop a mobile app to provide a more convenient booking experience.
- **Advanced Analytics:** Integrate analytics to track user behavior, movie popularity, and booking trends.
- **Additional Payment Options:** Expand payment options to include more services and currencies.
- **User Reviews and Ratings:** Allow users to rate and review movies to help others make informed decisions.

15. Conclusion

CineBook provides a comprehensive solution for online movie ticket booking with robust features for both users and admins. It leverages modern technologies like the MEAN stack and tools like TailwindCSS and Swagger to deliver a responsive and well-documented application. Despite the challenges faced, the project successfully meets its objectives and offers a solid foundation for future growth and enhancements.