Advanced Movie Booking Application - Full Stack Project

Technology Stack:

- **Frontend**: React.js (with Redux or Context API for state management)
- **Backend**: Node.js with Express.js
- **Database**: MongoDB with Mongoose ORM
- **Authentication**: JWT + Role-Based Access Control (RBAC)
- **Optional Integrations**: Stripe/PayPal for Payments, WebSockets for real-time seat updates

---

Advanced Features:

- Role-based dashboard with fine-grained access control
- Real-time seat reservation and lock mechanism
- Enhanced booking history with filters, export
- Admin analytics (e.g., bookings/month, revenue)
- Email/SMS notification system
- Theater screen layout designer (rows, columns, seat types)
- Multi-language support (i18n)

Modules & Functionalities:

1. Admin Module:

- Manage all users: CRUD, block/unblock
- Manage theaters: approve, suspend, delete
- Global movie database: manage movie data, genres, ratings
- Analytics Dashboard
- Assign roles to new admins (Super Admin capability)

2. Theater Module:

- Theater Registration/Login
- Add/manage multiple screens (screen name, seat layout)
- Add/manage movies with showtimes and screen allocation
- View booking stats, revenue, and feedback

3. User Module:

- User registration/login
- Browse movies (filter by language, genre, rating, location)
- View theaters and available showtimes
- Real-time seat selection & ticket booking
- Payment integration

- Booking history + invoice download

DFD (Data Flow Diagram)

Level 0 (Context Diagram)

[User] <---> [Movie Booking System] <---> [Theater Owner]\ <---> [Admin]

Level 1 DFD

1. **User**

2. Register/Login -> Authentication System

3. Browse Movies -> Movie Service
4. Select Seats -> Seat Management Service
5. Book Ticket -> Booking Service
6. Make Payment -> Payment Gateway

7. View Bookings -> Booking Service

8. **Theater Owner**

9. Register/Login -> Authentication System

10. Manage Screens -> Screen Management
11. Manage Movies & Shows -> Movie/Showtime Service

12. View Revenue & Bookings -> Analytics Service

13. **Admin**

14. Manage Users -> User Service

15. Manage Theaters -> Theater Service
16. Manage Movies -> Movie Service
17. View System Analytics -> Analytics Service

Level 2 DFD (Sample - Booking Flow)

- User selects movie → system fetches shows → user selects show → system locks selected seats → user makes payment → system confirms booking → updates seat status → sends confirmation

---

MongoDB Table Structure (Collections)

1. Users

```
{
  _id: ObjectId,
  name: String,
  email: String,
  passwordHash: String,
  role: ['user', 'admin', 'theater'],
  isActive: Boolean,
  createdAt: Date
}
```

2. Theaters

```
{
  _id: ObjectId,
  ownerId: ObjectId,
  name: String,
  address: String,
  location: String,
  isApproved: Boolean,
  screens: [ObjectId],
  createdAt: Date
}
```

3. Screens

```
{
  _id: ObjectId,
  theaterId: ObjectId,
  name: String,
  seatLayout: [[{ seatNumber: String, seatType: String, isAvailable:
Boolean }]],
  createdAt: Date
}
```

4. Movies

```
{
  _id: ObjectId,
  title: String,
```

```
  genre: [String],
  language: String,
  duration: Number,
  rating: Number,
  description: String,
  posterUrl: String,
  createdAt: Date
}
```

5. Showtimes

```
{
  _id: ObjectId,
  movieId: ObjectId,
  screenId: ObjectId,
  theaterId: ObjectId,
  startTime: Date,
  price: Number,
  availableSeats: Number,
  createdAt: Date
}
```

6. Bookings

```
{
  _id: ObjectId,
  userId: ObjectId,
  showtimeId: ObjectId,
  seats: [String],
  totalPrice: Number,
  status: ['pending', 'confirmed', 'cancelled'],
  paymentId: String,
  createdAt: Date
}
```

7. Payments

```
{
  _id: ObjectId,
  userId: ObjectId,
  bookingId: ObjectId,
  amount: Number,
  status: ['success', 'failed'],
  provider: ['stripe', 'paypal'],
```

```
  transactionDetails: Object,
  createdAt: Date
}
```

Let me know if you need ER diagrams, API routes, or full frontend/backend boilerplate.