**Movie Theaters Management System**

**Requirements**

**1. Project Setup**

* **Create a New Project Directory:**

Create a new directory named <yourName>\_movieBooking (e.g., alice\_movieBooking), navigate into this directory, and initialize a Node.js project using npm init.

**2. Create a Schema and Model for Movie Theaters**

* **File:** theaterModel.js
* **Data Fields:**
  + theaterName (String): The name of the movie theater.
  + location (String): The address or location of the theater.
  + seatCapacity (Number): Total number of seats available in the theater.
  + screenType (String): Type of screen (e.g., IMAX, 3D, Standard).
  + amenities (Array): List of amenities provided (e.g., Recliner seats, Dolby Atmos).

*Example Data:* Json file

{

theaterName: "Cineplex Downtown",

location: "123 Main St, Cityville",

seatCapacity: 150,

screenType: "IMAX",

amenities: ["Recliner seats", "Dolby Atmos", "Snack Bar"]

}

**3. Create a Schema and Model for Movie Show Schedules**

* **File:** scheduleModel.js
* **Data Fields:**
  + movieName (String): Name of the movie.
  + theaterName (String): Name of the theater where the movie is shown.
  + showTime (Date): Date and time of the movie showing.
  + ticketPrice (Number): Price per ticket.
  + availableSeats (Number): Number of seats available for booking.

*Example Data:* js file

{

movieName: "Inception",

theaterName: "Cineplex Downtown",

showTime: "2025-05-15T20:00:00Z",

ticketPrice: 12.50,

availableSeats: 100

}

**4. Build APIs to Manage Bookings**

Develop RESTful APIs to handle the ticket booking process. You must implement the following endpoints:

**4.1. GET /bookings**

* **Function:** Retrieve the list of all movie ticket bookings.

**4.2. POST /bookings**

* **Function:** Add a new ticket booking.
* **Booking Data Includes:**
  + customerName (String): Name of the customer.
  + theaterName (String): The theater where the movie is shown.
  + movieName (String): The movie booked.
  + showTime (Date): The showing time.
  + numberOfTickets (Number): Number of tickets booked.
  + totalAmount (Number): Total amount to be paid.

**4.3. PUT /bookings/:bookingId**

* **Function:** Update an existing ticket booking by its booking ID.

**4.4. DELETE /bookings/:bookingId**

* **Function:** Cancel a ticket booking by its booking ID.

**4.5. Payment Calculation**

* **Requirement:**  
  The system must automatically calculate the total payment based on the number of tickets booked and the ticket price from the corresponding movie show schedule.

**5. Project Structure: Model-Controller-Routes (MVCR)**

* **Organization:**  
  Organize your project files into folders such as models, views, controllers, and routes to separate concerns:
  + **Views:** Contain file ejs call api.
  + **Models:** Contain theaterModel.js, scheduleModel.js, and bookingModel.js (create this file for booking details).
  + **Controllers:** Implement business logic for handling bookings and payment calculations.
  + **Routes:** Define API routes that call the corresponding controller functions.

Your project can authenticate with JWT/Oauth

**Submission Requirements**

* **Source Files:**

Include the package.json file and all source code files (including theaterModel.js, scheduleModel.js, bookingModel.js, controllers, routes, etc.).