Step-by-Step Guide for Building a Movie Booking System

1. Define Your Goal

Understand the core functionalities of the application:

- Dynamically initialize seats based on the specified number of rows and columns.
- Allow users to book available seats.
- Display booked and available seats with visual distinction.
- Re-render the seat arrangement dynamically after actions.

---

2. Plan the Structure

Break the application into smaller components:

1. **Data Storage**: Use an array to store seat details, including seat number and availability status.
2. **Methods**: Define functions for initializing seats, rendering the layout, and handling seat bookings.
3. **Event Handling**: Handle interactions like clicking on seats to book them.

---

3. Start Writing Code

Begin with the basics and build incrementally:

A. Set Up a Movie Booking System Object

Create a JavaScript object (`movieBookingSystem`) to manage seats and their associated actions:

- Properties:
  - `seats`: Array to store seat objects with attributes like `number` and `isBooked`.
- Methods:
  - `initializeSeats(rows, cols)`
  - `renderSeats()`

B. Implement Core Methods

Write the methods in a modular way:

- `initializeSeats`: Generate seat objects dynamically based on the specified rows and columns. Each seat should have a unique number and an availability status (`isBooked` set to `false`).
- `renderSeats`: Dynamically generate and display the seat arrangement. Use classes to visually distinguish between available and booked seats.
- Seat Click Handling: Attach an `onclick` event to each seat that:
    - Prompts the user to confirm booking.
    - Updates the `isBooked` status if the seat is available.
    - Alerts the user if the seat is already booked.
    - Re-renders the seat layout after booking.

C. Attach Event Listeners

Handle the interactions:

- **Seat Booking**: Use `onclick` on seat elements to trigger the booking process and update the seat status.

D. Test and Debug

- Test the seat initialization to ensure proper numbering and layout.
- Validate booking actions for edge cases (e.g., double booking).

---

4. Order of Implementation

Follow this sequence:

Initialize the Movie Booking System Object:

const movieBookingSystem = { seats: [] };

1.
2. **Add Core Methods to the Object**:
    - `initializeSeats(rows, cols)`
    - `renderSeats()`
3. **Create Event Handlers**:
    - Write the logic for booking seats and attach it to the seat elements.
4. **DOM Manipulation**:
    - Use `document.createElement` and `appendChild` to render the seat layout.
    - Dynamically update the DOM for seat availability status.

5. **Test the Flow**:
    - Initialize seats with various row and column values.
    - Book available seats.
    - Attempt to book already booked seats.
    - Ensure the DOM updates correctly.

---

5. Add Features Incrementally

Once the basics work, enhance the application:

- **Reset Functionality**: Allow users to reset the seat arrangement.
- **Styling**: Apply CSS classes for a theater-like seat layout.
- **Seat Categories**: Add categories like VIP, Standard, and Economy with different visual styles.
- **Validation**: Ensure seats are properly initialized and users cannot book invalid seats.

---

6. Checklist for Completion

- Seats are initialized correctly with unique numbers.
- Seats can be booked and marked as unavailable.
- Already booked seats cannot be booked again.
- The seat arrangement updates dynamically after booking actions.
- Visual distinction exists between available and booked seats.

By following this roadmap, you will systematically build the Movie Booking System with clarity and focus.