Step 1: Setting Up the Basic Structure

- **Goal:** Create the HTML structure to display the theater seating arrangement.
- **Tasks:**
    - Add a container (e.g., a `div`) with an ID like `seatArrangement` to display seats.
    - Style the seats using CSS (e.g., `.seat`, `.seat-available`, `.seat-booked`).
- **Practice:**
    - Create the basic layout and test how seats will look in the UI.

Step 2: Initializing Seats

- **Goal:** Dynamically generate seats based on rows and columns.
- **Tasks:**
    - Write the `initializeSeats` method to generate an array of seat objects.
    - Assign unique seat numbers (e.g., A1, B2) and set `isBooked` to `false`.
    - Test seat initialization with different row and column counts.
- **Practice:**
    - Log the `seats` array to ensure correct data structure.
    - Display a placeholder layout for the seats.

Step 3: Rendering Seats

- **Goal:** Display the seats dynamically in the DOM.
- **Tasks:**
    - Write the `renderSeats` method to create seat elements and append them to `seatArrangement`.
    - Add classes (`seat-available` or `seat-booked`) to differentiate seat states.
    - Style the seats in CSS for a grid layout.
- **Practice:**
    - Test rendering seats with various configurations (e.g., 3x5, 10x10).

Step 4: Booking a Seat

- **Goal:** Allow users to book a seat.
- **Tasks:**
    - Add a click event to each seat in `renderSeats`.
    - Check if the seat is already booked (`isBooked`).

    - If available, confirm booking with the user and update `isBooked` to `true`.
    - Re-render the seats after booking.
- **Practice:**
    - Test booking multiple seats and verify their states update dynamically.

Step 5: Handling Already Booked Seats

- **Goal:** Prevent users from booking already booked seats.
- **Tasks:**
    - Show an alert if a user clicks on a booked seat.
    - Ensure booked seats have the `seat-booked` class and are visually distinct.
- **Practice:**
    - Test clicking on booked seats and validate the error message.

Step 6: Polishing the User Interface

- **Goal:** Improve the visual design of the application.
- **Tasks:**
    - Style the seats to resemble a theater layout.
    - Use CSS grid or flexbox for the seating arrangement.
    - Add hover effects for available seats.
- **Practice:**
    - Test the UI with different screen sizes and ensure it is responsive.

Step 7: Adding Dynamic Initialization

- **Goal:** Allow users to configure rows and columns dynamically.
- **Tasks:**
    - Add an input form to accept the number of rows and columns.
    - Modify `initializeSeats` to use user input for seat generation.
    - Re-render the seats after initializing.
- **Practice:**
    - Test initializing with different inputs (e.g., 5x10, 7x8).

Bonus Steps

1. **Add Local Storage:**

   - Save the `seats` array to `localStorage` to persist data between sessions.
   - Load seat states from `localStorage` on page load.
2. **Show Booking Summary:**
   - Display a list of booked seats and their numbers.
   - Add a counter to show the total number of available and booked seats.
3. **Add Seat Price and Total Cost:**
   - Assign prices to seats (e.g., rows A-B: $10, C-D: $8).
   - Calculate the total cost dynamically as users book seats.
4. **Implement Cancellation:**
   - Allow users to cancel bookings by clicking on booked seats and confirming cancellation.
5. **Add Dark Mode:**
   - Add a toggle for switching between light and dark themes.

Practicing Each Step Work on implementing and testing each step independently. Once all steps are complete, integrate them to create a fully functional **Movie Booking System**.