Step 1: Setting Up the Basic Structure

- **Goal:** Create the basic HTML structure for the application.
- **Tasks:**
  - Add a form with input fields for user name, email, and role.
  - Add a submit button for the form.
  - Create a container (e.g., a `div`) to display the list of users.
- **Practice:**
  - Write basic HTML with IDs like `userForm`, `userName`, `userEmail`, `userRole`, `userSubmitButton`, and `userList`.

Step 2: Initialize the Application

- **Goal:** Set up the `userManagement` object and its properties.
- **Tasks:**
  - Define the `users` array to hold user data.
  - Add boolean flags (`isEditing` and `editingEmail`) for editing state.
  - Create an initial `renderUsers` method to display "No users available" when the list is empty.
- **Practice:**
  - Test the initial rendering with an empty user list.

Step 3: Adding a User

- **Goal:** Implement the functionality to add a user.
- **Tasks:**
  - Write the `addUser` method.
  - Validate the email format using the `validateEmail` method.
  - Add the new user to the `users` array and re-render the user list.
- **Practice:**
  - Test adding users with valid and invalid email formats.
  - Ensure the list updates dynamically after adding a user.

Step 4: Displaying Users

- **Goal:** Dynamically render the user list.
- **Tasks:**

    - Update the `renderUsers` method to loop through the `users` array and display user details.
    - Include buttons for "Edit" and "Delete" for each user.
- **Practice:**
    - Test rendering with multiple users.
    - Style the user list with CSS for a better appearance.

Step 5: Editing a User

- **Goal:** Implement the functionality to edit a user.
- **Tasks:**
    - Write the `editUser` method to populate the form with the user's details.
    - Disable the email input field to prevent changes.
    - Update the form's submit button text to "Update User."
- **Practice:**
    - Test editing different users.
    - Ensure the form updates dynamically for each user.

Step 6: Updating a User

- **Goal:** Update the details of an existing user.
- **Tasks:**
    - Write the `updateUser` method to modify the user's name and role in the `users` array.
    - Re-render the user list after updating the user.
    - Reset the form after the update.
- **Practice:**
    - Test updating user details and check that changes reflect in the list.

Step 7: Deleting a User

- **Goal:** Implement the functionality to delete a user.
- **Tasks:**
    - Write the `deleteUser` method to remove the user from the `users` array.
    - Add a `deleteUserPrompt` function to get confirmation or role validation before deleting.
- **Practice:**
    - Test deleting users as "Admin" and "Subscriber."
    - Verify that only Admins can delete users.

Step 8: Resetting the Form

- **Goal:** Clear the form after adding or updating a user.
- **Tasks:**
  - Write the `resetForm` method to reset input fields and states (`isEditing` and `editingEmail`).
  - Re-enable the email input field for new users.
- **Practice:**
  - Test adding and editing users to ensure the form resets correctly each time.

Step 9: Validating User Input

- **Goal:** Ensure user input meets validation requirements.
- **Tasks:**
  - Write the `validateEmail` method to check email format.
  - Add alerts or notifications for invalid input.
- **Practice:**
  - Test adding users with invalid emails to ensure proper validation.

Step 10: Polishing the UI

- **Goal:** Improve the design and usability of the application.
- **Tasks:**
  - Style the user list and form using CSS.
  - Add hover effects for buttons.
  - Make the layout responsive for different screen sizes.
- **Practice:**
  - Test the application on mobile and desktop devices.

Bonus Steps

1. **Add Roles and Permissions:**
   - Implement role-based actions (e.g., only Admins can edit or delete users).
2. **Add Search and Filter Features:**
   - Allow users to search by name or email.
   - Filter users by role (e.g., Admin, Subscriber).

3. **Save Data Locally:**
    - Use `localStorage` to persist the user list between sessions.
4. **Add Pagination:**
    - If the user list grows, display users in pages with navigation controls.
5. **Create a Dark Mode:**
    - Add a toggle for light and dark themes.

Practicing Each Step Implement and test each step independently. Once all steps are completed, integrate them to create a fully functional User Management System.